Επίλυση – Resolution

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

Η Μέθοδος της Επίλυσης στον Κατηγορηματικό Προγραμματισμό

Η Μέθοδος της Επίλυσης στον Λογικό Προγραμματισμό

Υπολογισμός και Λογική

- Το όνειρο του Leibniz:
  - μια καθολική μαθηματική γλώσσα μέσω της οποίας να είναι δυνατή η διατύπωση κάθε ανθρώπινης γνώσης και κανόνες που μπορούν να εφαρμοστούν μηχανιστικά για την παραγωγή κάθε λογικής αλήθειας και σχέσης. "Calculemus" : Ας υπολογίσουμε!

- Το πρόγραμμα του David Hilbert
  - Η επινόηση ενός τυπικού συστήματος που θα επιτρέπει την απόδειξη κάθε μαθηματικής αλήθειας μέσω μιας μηχανικής διαδικασίας

- Alonzo Church και Alan Turing:
  - Υπάρχουν προβλήματα τα οποία κανένας αλγόριθμος δεν μπορεί να επιλύσει. Επομένως δεν μπορεί να υπάρχει κάποιος αλγόριθμος ο οποίος να μπορεί να παραγάγει κάθε μαθηματική αλήθεια.

Leibniz, 1646–1716

Hilbert, 1862-1943

Turing, 1912-1954

Church 1903-1995

Resolution – Επίλυση

- Ο Κατηγορηματικός Λογισμός είναι μη διαγνώσιμος.
- Εντούτοις, υπάρχει αλγοριθμική μέθοδος μέσω της οποίας μπορούμε να δείξουμε ότι μια πρόταση του ΚΛ είναι μη ικανοποιήσιμη, η:
- Επίλυση
  - Κατάλληλη για χρήση από υπολογιστή (μόνο 1 κανόνας, 0 αξιώματα)
  - Δυνατόν να μην τερματίζει σε κάποια δεδομένα
- J. Alan Robinson 1965
  - Αυτοματοποιημένη απόδειξη θεωρημάτων (theorem proving)
- Λογικός Προγραμματισμός: Robert Kowalski
- Prolog: Alain Colmerauer 1973
- Prolog compiler: David Warren, 1997
- Constraint Logic Programming: Jaffar and Lassez, 1987

Διαδικασία Επίλυσης στον Κατηγορηματικό Λογισμό

•Απαραίτητη προεπεξεργασία αφορά στη μετατροπή των προτάσεων υπό μελέτη σε προτασιακή μορφή. Για κάθε πρόταση αυτό επιτυγχάνεται ως εξής:

- Μετατροπή πρότασης σε Κανονική Μορφή Prenex (KMP) - συζευκτική κανονική μορφή με όλους τους ποσοδείκτες στην αρχή της πρότασης
- Απαλοιφή των υπαρξιακών ποσοδεικτών μέσω της μεθόδου του Skolem
- Διαγραφή των καθολικών ποσοδεικτών
- Εξαγωγή των προτασιακών συνόλων

Αλγόριθμος Μετατροπή σε ΚΜΡ

- Βήμα 1: Απαλοιφή συνεπαγωγών μέσω της ισοδυναμίας

  $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$

- Βήμα 2: Αφαίρεση διπλών αρνήσεων και μετακίνηση αρνήσεων στο επίπεδο των ατομικών προτάσεων μέσω των ισοδυναμιών

  $\neg\neg\phi \equiv \phi$

  $\neg(\phi \vee \psi) \equiv \neg\, \phi \ \wedge \neg\psi$ $\qquad$ $\neg(\phi \wedge \psi) \equiv \neg\, \phi \ \vee \neg\psi$

  $\neg \forall x\, \phi \ \equiv \exists x\, \neg\phi$ $\qquad$ $\neg\, \exists x\, \phi \ \equiv \forall x\, \neg\phi$

- Βήμα 3: Μετονομασία δεσμευμένων μεταβλητών έτσι ώστε να μην υπάρχει το ίδιο όνομα μεταβλητής σε διαφορετικούς ποσοδείκτες.

- Βήμα 4: Μεταφορά όλων των ποσοδεικτών στα αριστερά (σε συνδυασμός με αντιμεταθετική ιδιότητα):

  $\forall x\, \varphi \wedge \psi \equiv \forall x\, (\varphi \wedge \psi)$ $\qquad$ $\forall x\, \varphi \vee \psi \equiv \forall x\, (\varphi \vee \psi)$

  $\exists x\, \varphi \wedge \psi \equiv \exists\, x\, (\varphi \wedge \psi)$ $\qquad$ $\exists x\, \varphi \vee \psi \equiv \exists x\, (\varphi \vee \psi)$

  (Σημείωση: εφόσον το x δεν εμφανίζεται ελεύθερο στην πρόταση ψ.)

- Βήμα 5: Εισαγωγή συζεύξεων από διαζεύξεις

  $\phi \vee (\psi_1 \wedge \psi_2) \equiv (\phi \vee \psi_1) \ \wedge (\phi \vee \psi_2)$ $\qquad$ $(\psi_1 \wedge \psi_2) \vee \phi \equiv (\psi_1 \vee \phi) \ \wedge (\psi_2 \vee \phi\ )$

Παράδειγμα

Μετάτρεψε την πρόταση σε KMP

$$\forall x\,(A(x) \rightarrow B(x)) \rightarrow \exists x\, Q(x)$$

1(a) $\neg\,(\forall x\,(A(x) \rightarrow B(x))\,) \vee \exists x\, Q(x)$

1(b) $\neg\,(\forall x\,(\neg A(x) \vee\ B(x))\,) \vee \exists x\, Q(x)$

2(a) $\exists x\,(\neg(\neg A(x) \vee B(x)))\ \vee \exists x\, Q(x)$

2(b) $\exists x\,(\neg\neg A(x) \wedge \neg\, B(x))\ \vee \exists x\, Q(x)$

2(c) $\exists x\,(A(x) \wedge \neg\, B(x))\ \vee \exists x\, Q(x)$

3 $\exists x\,(A(x) \wedge \neg\, B(x))\ \vee \exists y\, Q(y)$

4(a) $\exists x\,[(A(x) \wedge \neg\, B(x)))\ \vee \exists y\, Q(y)]$

4(b) $\exists x\,\exists y\,[(A(x) \wedge \neg\, B(x)))\ \vee Q(y)]$

5 $\exists x\,\exists y\,[(A(x) \vee Q(y)) \wedge (\neg\, B(x) \vee Q(y))]$

Μέθοδος του Skolem – Skolemization

- Thoralf Skolem
- Στόχος: Απαλοιφή των υπαρξιακών ποσοδεικτών
- Πετυχαίνεται με την εισαγωγή καινούριων σταθερών και συναρτήσεων στην πρόταση υπό μελέτη.
- Μέθοδος:
  - Για κάθε $\exists x$ που προηγείται όποιων $\forall y$: αντικατάσταση της μεταβλητής x από μια καινούρια σταθερά, π.χ.

    Η πρόταση $\exists x \forall y\ A(x,y)$ μετατρέπεται στην $\forall y\ A(c,y)$
  - Για κάθε $\exists x$ που έπεται των ποσοδεικτών $\forall y_1 \ldots \forall y_n$: αντικατάσταση της μεταβλητής x από μια καινούρια συνάρτηση f με παραμέτρους τις $y_1 \ldots \forall y_n$ π.χ.

    Η πρόταση $\forall x \exists y \forall z \exists w\ [A(x,y,z,w) \wedge B(y,w)]$

    μετατρέπεται στην $\forall x \forall z[A(x,f_1(x),z,f_2(x,z)) \wedge B(f_1(x),f_2(x,z))]$

Ορθότητα Μεθόδου του Skolem

- Βασική Ιδέα:
  - Η πρόταση $\exists x \forall y\ A(x,y)$ προβλέπει την ύπαρξη κάποιου αντικειμένου x που συνδέεται με κάθε αντικείμενο y μέσω της σχέσης A. Δεν ξέρουμε την ακριβή τιμή του x όμως μπορούμε να χρησιμοποιήσουμε ένα σύμβολο σταθεράς για να το συμβολίσουμε. Έτσι η πρότασή μας μεταφράζεται σε $\forall y\ A(c,y)$.
  - Η πρόταση $\forall y \exists x\ A(x,y)$ προβλέπει για κάθε αντικείμενο y την ύπαρξη κάποιου αντικειμένου x για το οποίο ισχύει A(x,y). Η συγκεκριμένη τιμή του x εξαρτάται από το y. Έτσι μπορούμε να υποθέσουμε την ύπαρξη μιας συνάρτησης που για κάθε αντικείμενο y μας δίνει το αντικείμενο x με το οποίο συνδέεται το y. Έτσι η πρότασή μας μεταφράζεται σε $\forall y\ A(f(y),y)$.

Θεώρημα: Έστω πρόταση ϕ και ψ η πρόταση μετά από την εφαρμογή της μεθόδου Skolem. Τότε η πρόταση ϕ είναι ικανοποιήσιμη αν και μόνο αν η πρόταση ψ είναι ικανοποιήσιμη.

Παράδειγμα

Να μετατρέψετε την πιο κάτω πρόταση σε προτασιακή μορφή.

$$\forall x \exists y(P(y) \rightarrow \neg Q(x,y))$$

1.	$\forall x \exists y(\neg P(y) \vee \neg Q(x,y))$	Μετατροπή σε ΚΜΡ
2.	$\forall x(\neg P(f(x)) \vee \neg Q(x,f(x)))$	Απαλοιφή υπαρξιακών ποσοδεικτών
3.	$\neg P(f(x)) \vee \neg Q(x,f(x)))$	Διαγραφή καθολικών ποσοδεικτών
4.	$\{\{\neg P(f(x)),\neg Q(x,f(x))\}\}$	Κατασκευή προτασιακού συνόλου

Ενοποίηση

- Ενοποίηση είναι μια διαδικασία μέσω της οποίας εξετάζουμε αν τα στοιχεία ενός προτασιακού συνόλου μπορούν να γίνουν συντακτικά ταυτόσημα με τη χρήση μιας αντικατάστασης.

- Μια αντικατάσταση σ ονομάζεται ενοποιήτρια αντικατάσταση του προτασιακού συνόλου $\{L_1, L_2, \ldots, L_n\}$ αν είναι τέτοια ώστε $L_1\sigma = L_2\sigma = \ldots = L_n\sigma$. Το σύνολο καλείται ενοποιήσιμο.

- Γενικότερη ενοποιήτρια ενός προτασιακού συνόλου S ονομάζεται εκείνη η ενοποιήτρια σ για την οποία ισχύει ότι κάθε άλλη ενοποιήτρια θ του συνόλου μπορεί να γραφτεί ως επέκτασης της σ. Δηλαδή, υπάρχει αντικατάσταση σ' τέτοια ώστε $\theta = \sigma\sigma'$.

Ενοποίηση

- Οποιοδήποτε ενοποιήσιμο σύνολο S έχει μια γενικότερη ενοποιήτρια.

- Κανόνες ενοποίησης όρων:
  - Μια σταθερά c είναι ενοποιήσιμη με τον όρο t αν είτε t=c είτε t=x όπου x οποιαδήποτε μεταβλητή. Δεν είναι ενοποιήσιμη με συναρτήσεις.
  - Μια μεταβλητή είναι ενοποιήσιμη με οποιοδήποτε όρο εκτός από συναρτήσεις που περιέχουν τη μεταβλητή.
  - Μια συνάρτηση είναι ενοποιήσιμη με μια άλλη συνάρτηση εφόσον οι δύο έχουν το ίδιο σύμβολο συνάρτησης και ενοποιήσιμους όρους.

Παράδειγμα

- Το σύνολο {P(x,c), P(b,c)} είναι ενοποιήσιμο με ενοποιήτρια αντικατάσταση την {b/x}.
- Το σύνολο {P(x,a), P(b,c)} δεν είναι ενοποιήσιμο.
- Το σύνολο {P(f(x),y), P(a,w)} δεν είναι ενοποιήσιμο.
- Το σύνολο {P(f(x),y), P(f(a),w)} είναι ενοποιήσιμο με ενοποιήτρια αντικατάσταση την {a/x, w/y}. Άλλη δυνατή ενοποιήτρια αντικατάσταση είναι η {a/x, b/y, b/w}. Η {a/x, w/y} είναι όμως η γενικότερη ενοποιήτρια {a/x, b/y, b/w} = {a/x, w/y} {b/w}.
- Το σύνολο {P(f(x)), P(x)} δεν είναι ενοποιήσιμο

Αλγόριθμος Ενοποίησης

Δεδομένο Εισόδου: Δύο όροι L_1 και L_2

Δεδομένο Εξόδου: ενοποιήτρια σ των όρων, αν υπάρχει.

1. $T_1 = L_1$; $T_2 = L_2$; $\sigma_0 = \{\}$; $i = 0$;
2. Αν τα $T_1 = T_2$ τότε τερμάτισε και επέστρεψε την σ_i ως τη γενικότερη ενοποιήτρια
3. Διαφορετικά, εντόπισε το αριστερότερο σημείο στο οποίο διαφέρουν τα T_1 και T_2. Αν τα δύο σημεία είναι μια μεταβλητή v_i και ένας όρος t_i ο οποίος δεν περιέχει τη μεταβλητή v_i τότε θέσε

   $\sigma_{i+1} = \sigma_i\{t_i/v_i\}$, $T_1 = T_1\{t_i/v_i\}$,

   και $T_2 = T_2\{t_i/v_i\}$
4. Διαφορετικά, τερμάτισε και ανάφερε ότι οι όροι δεν είναι ενοποιήσιμοι.
5. Θέσε $i = i + 1$ και επέστρεψε στο βήμα 2.

Παράδειγμα

Να εφαρμόσετε τον Αλγόριθμο Ενοποίησης στο πιο κάτω ζεύγος όρων:

$L_1 = f(x,g(x))$ $\quad$ $L_2 = f(h(y), g(h(z))$

1. $T_1 = f(x,g(x))$, $T_2 = f(h(y), g(h(z))$, $\sigma_0 = \{\}$, i=0
2. $\sigma_1 = \sigma_0 \{h(y)/x\}$, $T_1 = f(h(y),g(h(y)))$, $T_2 = f(h(y), g(h(z))$
3. $\sigma_2 = \sigma_1 \{y/z\}$, $T_1 = f(h(y),g(h(y)))$, $T_2 = f(h(y), g(h(y))$
4. Επέστρεψε τη $\sigma_2 = \{h(y)/x, y/z\}$ ως τη γενικότερη ενοποιήτρια των δύο όρων.

Επιλύουσες στον Κατηγορηματικό Λογισμό

- Στον Προτασιακό Λογισμό:
  - Ονομάζουμε δύο στοιχεία L και L' συμπληρωματικά αν L'= ¬L ή L= ¬L'
  - Έστω προτασιακά σύνολα C_1 και C_2 και συμπληρωματικά στοιχεία L_1 και L_2 τέτοια ώστε $L_1 \in C_1$ και $L_2 \in C_2$. Τότε το προτασιακό σύνολο $(C_1 - \{L_1\}) \cup (C_2 - \{L_2\})$ ονομάζεται επιλύουσα των C_1 και C_2 .
- Στον Κατηγορηματικό Λογισμό:
  - Ονομάζουμε δύο στοιχεία L και ¬L' συμπληρωματικά αν το σύνολο {L, L'} είναι ενοποιήσιμο
  - Έστω προτασιακά σύνολα C_1 και C_2 και συμπληρωματικά στοιχεία L_1 και L_2 τέτοια ώστε $L_1 \in C_1$ και $L_2 \in C_2$ με επιλύουσα σ. Τότε το προτασιακό σύνολο $(C_1\sigma - \{L_1\sigma\}) \cup (C_2\sigma - \{L_2\sigma\})$ ονομάζεται επιλύουσα των C_1 και C_2 .

Διαδικασία Επίλυσης για ΚΛ

- Αν το D είναι επιλύουσα των C_1 και C_2 τότε $C_1 \wedge C_2 \models D$.
- Η Διαδικασία Επίλυσης του ΚΛ εφαρμόζεται ακριβώς όπως και στον Προτασιακό Λογισμό (Διαφάνεια 4-12) με τη διαφορά ότι η επιλύουσα δύο προτασιακών συνόλων επιλέγεται μέσω της ενοποίησης τους.
- Θεώρημα Ορθότητας (Robinson): Αν ο όρος $\perp$ ληφθεί από ένα προτασιακό σύνολο κατά τη διαδικασία επίλυσης, τότε το σύνολο αυτό είναι μη ικανοποιήσιμο.
- Θεώρημα Πληρότητας: Αν ένα προτασιακό σύνολο είναι μη ικανοποιήσιμο τότε ο όρος $\perp$ μπορεί να ληφθεί από τη διαδικασία της επίλυσης.

Παράδειγμα

1. Όλοι οι σκύλοι γαβγίζουν τα βράδια
2. Όποιος έχει γάτα δεν έχει ποντίκια.
3. Άνθρωποι που έχουν ελαφρύ ύπνο δεν έχουν ζώα που γαβγίζουν τις νύκτες.
4. Ο Γιάννης έχει γάτα ή σκύλο.

5. Συμπέρασμα: Αν ο Γιάννης ελαφρύ ύπνο τότε δεν έχει ποντίκια.

- Ισχύει το συμπέρασμα;

Παράδειγμα – Βήμα 1

- Βήμα 1: Γράψε τις προτάσεις σε γλώσσα κατηγορηματικού λογισμού.

1. Όλοι οι σκύλοι γαβγίζουν τα βράδια

   $\forall x\ (\Sigma K(x) \rightarrow \Gamma\Tau B(x))$

2. Όποιος έχει γάτα δεν έχει ποντίκια.

   $\forall x\ \forall y\ (\Epsilon\Chi\Epsilon\Iota\ (x,y) \wedge \Gamma A(y) \rightarrow \neg\ \exists z\ (\Epsilon\Chi\Epsilon\Iota(x,z) \wedge \Pi O\ (z)))$

3. Άνθρωποι που έχουν ελαφρύ ύπνο δεν έχουν ζώα που γαβγίζουν τις νύκτες.

   $\forall x\ (\Epsilon Y(x) \rightarrow \neg\ \exists y\ (\Epsilon\Chi\Epsilon\Iota(x,y) \wedge \Gamma\Tau B(y)))$

4. Ο Γιάννης έχει γάτα ή σκύλο.

   $\exists x\ (\Epsilon\Chi\Epsilon\Iota\ (\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma,x) \wedge (\Gamma A(x) \vee \Sigma K(x)))$

5. Συμπέρασμα: Αν ο Γιάννης ελαφρύ ύπνο τότε δεν έχει ποντίκια.

   $\Epsilon Y(\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma) \rightarrow \neg\ \exists z\ (\Epsilon\Chi\Epsilon\Iota(\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma,z) \wedge \Pi O(z))$

Παράδειγμα – Βήμα 1

- Υποθέτουμε την άρνηση του συλλογισμού που αντιστοιχεί σε

$$
\begin{aligned}
&[\forall x\ (\Sigma K(x) \rightarrow \Gamma TB(x)) \\
&\quad \wedge \\
&\quad \forall x\ \forall y\ (EXEI\ (x,y) \wedge \Gamma A(y) \rightarrow \neg\ \exists z\ (EXEI(x,z) \wedge \Pi O\ (z))) \\
&\quad \wedge \\
&\quad \forall x\ (EY(x) \rightarrow \neg\ \exists y\ (EXEI(x,y) \wedge \Gamma TB(y))) \\
&\quad \wedge \\
&\quad \exists x\ (EXEI\ (\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma,x) \wedge (\Gamma A(x) \vee \Sigma K(x)))] \\
&\quad \wedge \\
&\quad \neg\ (EY(\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma) \rightarrow \neg\ \exists z\ (EXEI(\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma,z) \wedge \Pi O(z))\ )
\end{aligned}
$$

Παράδειγμα – Βήμα 2

- Μετατροπή σε Κανονική Μορφή Prenex:

1. $\forall x\ (\Sigma K(x) \rightarrow \Gamma TB(x)) = \forall x\ (\neg\ \Sigma K(x) \vee \Gamma TB(x)\ )$

2. $\forall x\ \forall y\ (EXEI\ (x,y) \wedge \Gamma A(y) \rightarrow \neg\ \exists z\ (EXEI(x,z) \wedge \Pi O\ (z)))$
   $= \forall x\ \forall y\ (EXEI\ (x,y) \wedge \Gamma A\ (y) \rightarrow \forall z\ \neg\ (EXEI(x,z) \wedge \Pi O\ (z)))$
   $= \forall x\ \forall y\ \forall z\ (\neg\ (EXEI\ (x,y) \wedge \Gamma A(y)) \vee \neg\ (EXEI(x,z) \wedge \Pi O\ (z)))$
   $= \forall x\ \forall y\ \forall z\ (\neg\ EXEI(x,y) \vee \neg\ \Gamma A(y) \vee \neg\ EXEI(x,z) \vee \neg\ \Pi O(z))$

3. $\forall x\ (EY(x) \rightarrow \neg\ \exists y\ (EXEI\ (x,y) \wedge \Gamma TB(y)))$
   $= \forall x\ (EY(x) \rightarrow \forall\ y\ \neg\ (EXEI(x,y) \wedge \Gamma TB(y)))$
   $= \forall x\ \forall y\ (EY(x) \rightarrow \neg\ EXEI(x,y) \vee \neg\ \Gamma TB(y))$
   $= \forall x\ \forall y\ (\neg EY\ (x) \vee \neg\ EXEI(x,y) \vee \neg\ \Gamma TB(y))$

Παράδειγμα – Βήμα 2

4. $\exists x\ (\text{ΕΧΕΙ}\ (\text{Γιάννης},x) \wedge (\text{ΓΑ}(x) \vee \text{ΣΚ}(x)))$

5. $\neg\ [\text{ΕΥ}(\text{Γιάννης}) \rightarrow \neg\ \exists z\ (\text{ΕΧΕΙ}(\text{Γιάννης},z) \wedge \text{ΠΟ}(z))]$
   $= \neg\ [\neg\ \text{ΕΥ}\ (\text{Γιάννης}) \vee \neg\ \exists z\ (\text{ΕΧΕΙ}(\text{Γιάννης}, z) \wedge \text{ΠΟ}(z))]$
   $= \text{ΕΥ}(\text{Γιάννης}) \wedge \exists z\ (\text{ΕΧΕΙ}(\text{Γιάννης}, z) \wedge \text{ΠΟ}(z)))$
   $= \exists z\ (\text{ΕΥ}(\text{Γιάννης}) \wedge \text{ΕΧΕΙ}\ (\text{Γιάννης},z) \wedge \text{ΠΟ}(z)\ )$

Επομένως η πρόταση προς επίλυση είναι ισοδύναμη με:

$\forall x\ (\neg\ \text{ΣΚ}(x) \vee \text{ΓΤΒ}(x)\ )$

$\wedge\ \forall x\ \forall y\ \forall z\ (\neg\ \text{ΕΧΕΙ}(x,y) \vee \neg\ \text{ΓΑ}(y) \vee \neg\ \text{ΕΧΕΙ}(x,z) \vee \neg\ \text{ΠΟ}(z))$

$\wedge\ \forall x\ \forall y\ (\neg\text{ΕΥ}\ (x) \vee \neg\ \text{ΕΧΕΙ}(x,y) \vee \neg\ \text{ΓΤΒ}(y))$

$\wedge\ \exists x\ (\text{ΕΧΕΙ}\ (\text{Γιάννης},x) \wedge (\text{ΓΑ}(x) \vee \text{ΣΚ}(x)))$

$\wedge\ \exists z\ (\text{ΕΥ}(\text{Γιάννης}) \wedge \text{ΕΧΕΙ}\ (\text{Γιάννης},z) \wedge \text{ΠΟ}(z)\ )$

Παράδειγμα – Βήμα 2

Μεταφέρουμε τους ποσοδείκτες προς τα έξω μετονομάζοντας τις μεταβλητές όπου χρειάζεται

$\forall x_1 (\neg \Sigma K(x_1) \vee \Gamma TB(x_1))$

$\wedge \forall x_2 \forall y_2 \forall z_2 (\neg EXEI(x_2,y_2) \vee \neg \Gamma A(y_2) \vee \neg EXEI(x_2,z_2) \vee \neg \Pi O(z_2))$

$\wedge \forall x_3 \forall y_3 (\neg EY (x_3) \vee \neg EXEI(x_3,y_3) \vee \neg \Gamma TB(y_3))$

$\wedge \exists x_4 (EXEI (\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma,x_4) \wedge (\Gamma A(x_4) \vee \Sigma K(x_4)))$

$\wedge \exists z_5 (EY(\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma) \wedge EXEI (\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma,z_5) \wedge \Pi O(z_5))$

$=$

$\exists x_4 \exists z_5 \forall x_1 \forall x_2 \forall y_2 \forall z_2 \forall x_3 \forall y_3 (\neg \Sigma K(x_1) \vee \Gamma TB(x_1))$

$\wedge (\neg EXEI(x_2,y_2) \vee \neg \Gamma A(y_2) \vee \neg EXEI(x_2,z_2) \vee \neg \Pi O(z_2))$

$\wedge (\neg EY (x_3) \vee \neg EXEI(x_3,y_3) \vee \neg \Gamma TB(y_3))$

$\wedge (EXEI (\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma,x_4) \wedge (\Gamma A(x_4) \vee \Sigma K(x_4)))$

$\wedge (EY(\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma) \wedge EXEI (\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma,z_5) \wedge \Pi O(z_5))$

Παράδειγμα – Βήμα 3

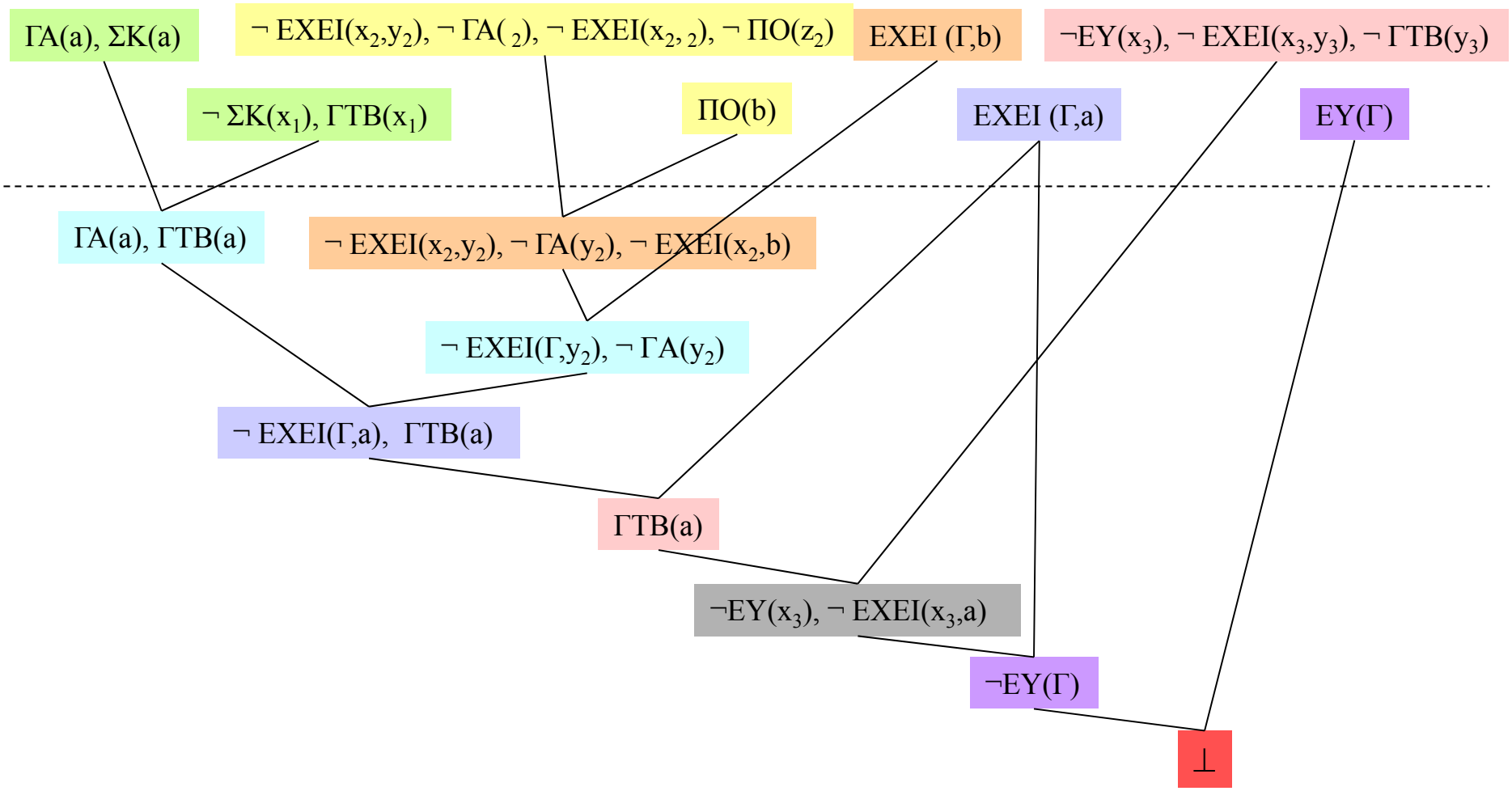
- Εφάρμοσε τη μέθοδο του Skolem στις κανονικές μορφές Prenex και κατασκεύασε την προτασιακή μορφή των προτάσεων:

$\{\neg\, \Sigma\Kappa(x_1) \vee \Gamma\Tau\Beta(x_1),$
$\neg\, \Epsilon\Chi\Epsilon\Iota(x_2,y_2) \vee \neg\, \Gamma\Alpha(y_2) \vee \neg\, \Epsilon\Chi\Epsilon\Iota(x_2,z_2) \vee \neg\, \Pi\Omicron(z_2),$
$\neg\Epsilon\Upsilon(x_3) \vee \neg\, \Epsilon\Chi\Epsilon\Iota(x_3,y_3) \vee \neg\, \Gamma\Tau\Beta(y_3),$
$\Epsilon\Chi\Epsilon\Iota\,(\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma,a) \wedge (\Gamma\Alpha(a) \vee \Sigma\Kappa(a)),$
$\Epsilon\Upsilon(\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma) \wedge \Epsilon\Chi\Epsilon\Iota\,(\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma,b) \wedge \Pi\Omicron(b)\ \}$

$\{\{\neg\, \Sigma\Kappa(x_1), \Gamma\Tau\Beta(x_1)\},$
$\{\neg\, \Epsilon\Chi\Epsilon\Iota(x_2,y_2), \neg\, \Gamma\Alpha(y_2), \neg\, \Epsilon\Chi\Epsilon\Iota(x_2,z_2), \neg\, \Pi\Omicron(z_2)\},$
$\{\neg\Epsilon\Upsilon(x_3), \neg\, \Epsilon\Chi\Epsilon\Iota(x_3,y_3), \neg\, \Gamma\Tau\Beta(y_3)\},$
$\{\Epsilon\Chi\Epsilon\Iota\,(\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma,\alpha)\},\{(\Gamma\Alpha(\alpha), \Sigma\Kappa(\alpha)\},$
$\{\Epsilon\Upsilon(\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma)\},\ \{\Epsilon\Chi\Epsilon\Iota\,(\Gamma\iota\acute{\alpha}\nu\nu\eta\varsigma,b)\},\ \{\Pi\Omicron(b)\}\}$

Παράδειγμα – Βήμα 4

- Εφάρμοσε τη μέθοδο της επίλυσης μέσω ενοποίησης.



Λογικός Προγραμματισμός

- Εξαγωγή συμπερασμάτων από ένα σύνολο λογικών εκφράσεων.
- Χρησιμοποιεί μια περιορισμένη γλώσσα λογικής η οποία επιτρέπει την αποδοτική απόδειξη θεωρημάτων μέσω της μεθόδου της επίλυσης.
- Η λογική αυτή περιέχει προτάσεις της μορφής:

  $\forall x_1 \ldots \forall x_k \, (B_1 \wedge \ldots \wedge B_n \rightarrow A)$

  οι οποίες μπορούν να γραφτούν ως διαζεύξεις με μόνο ένα θετικό όρο

  $\forall x_1 \ldots \forall x_k \, (\neg B_1 \vee \ldots \vee \neg B_n \vee A)$

  και έχουν προτασιακή μορφή $\{\neg B_1, \ldots, \neg B_n , A\}$
- Μια τέτοια πρόταση στον λογικό προγραμματισμό γράφεται ως

  $\forall x_1 \ldots \forall x_k \, (A \leftarrow B_1, \ldots, B_n)$

  και μπορεί να ερμηνευθεί ως η διαδικασία: *για να υπολογίσεις το A υπολόγισε τα* $B_1, \ldots, B_n$.

Παράδειγμα

- Αξιώματα για συμβολοσειρές
  1. $\forall x\ substr(x,x)$
  2. $\forall x \forall y \forall z\ (substr(x,z) \leftarrow (suffix(y,z) \wedge substr(x,y))$
  3. $\forall x \forall y\ suffix(x,yx)$
  4. $\forall x \forall y \forall z\ (substr(x,z) \leftarrow (prefix(y,z) \wedge substr(x,y))$
  5. $\forall x \forall y\ prefix(x,xy)$

  Μπορούν να τύχουν ερμηνείας ως:

  1. Κάθε συμβολοσειρά είναι υποσυμβολοσειρά του εαυτού της
  2. Για να αποφασίσουμε αν το x είναι υποσυμβολοσειρά του z τότε εντοπίζουμε συμβολοσειρά y τέτοια ώστε η y να είναι επίθημα της z και η x υποσυμβολοσειρά της y.
  3. Η x είναι επίθημα της yx
  4. Για να αποφασίσουμε αν το x είναι υποσυμβολοσειρά του z τότε εντοπίζουμε συμβολοσειρά y τέτοια ώστε η y να είναι πρόθημα της z και η x υποσυμβολοσειρά της y.
  5. Η x είναι πρόθημα της xy

Βασική Ιδέα

- Πιθανά ερωτήματα (στόχοι):
  - substr(ααβ,γβαβααγβααββ) ;
  - $\exists$x (substr(x,αβγααγ) $\wedge$ suffix (δαβ, x)) ;

- Βασική Ιδέα Διαδικασίας Επίλυσης στον ΛΠ
  - Θεώρησε την άρνηση του στόχου,
    π.χ. $\neg$ $\exists$x (substr(x,αβγααγ) $\wedge$ suffix (δαβ, x))
  - Μετάτρεψε σε προτασιακή μορφή:
    {{$\neg$substr(x,αβγααγ) , $\neg$suffix (δαβ, x)}}
  - Εφάρμοσε επίλυση στο σύνολο των αξιωμάτων που απαρτίζουν το πρόγραμμα και την προτασιακή μορφή του ερωτήματος.

Ροή Προγράμματος

- Μη ντετερμινισμός: Δυνατόν να υπάρχουν πολλά αξιώματα τα οποία μπορούν να συνδεθούν με τον στόχο του προγράμματος. Ποιο από αυτά θα επιλεχθεί σε κάθε βήμα;
- Ροή Προγράμματος:
  - Στον διαδικαστικό προγραμματισμό η ροή του προγράμματος καθορίζεται πλήρως από τον προγραμματιστή μέσω της σειράς παράταξης των εντολών που το απαρτίζουν.
  - Στον Λογικό Προγραμματισμό ο προγραμματιστής γράφει δηλώσεις που συλλαμβάνουν τη σχέση που έχουν οι οντότητες του προγράμματος. Εναπόκειται στον compiler να επιλύσει τον μη ντετερμινισμό που εμπεριέχεται στο πρόγραμμα.
- Κανόνας Υπολογισμού: Επιλογή στοιχείου της πρότασης-στόχος στο οποίο θα εφαρμοστεί επίλυση.
- Κανόνας Εύρεσης: Επιλογή "αξιώματος" στο οποίο να εφαρμοστεί επίλυση έναντι του στοιχείου της πρόταση-στόχος που έχουμε επιλέξει.

Ορισμοί

- Σε μια πρόταση $A \leftarrow B_1,\ldots,B_n$ (όρος Horn) ονομάζουμε
  - Τον θετικό όρο A κεφαλή (head) και
  - Τους αρνητικούς όρους $B_1,\ldots,B_n$ κορμό (body).
- Αν n = 0, τότε δεν υπάρχει κορμός και η πρόταση έχει τη μορφή $A \leftarrow$ και ονομάζεται γεγονός (fact).
- Αν η πρόταση δεν έχει κεφαλή, $\leftarrow B_1,\ldots,B_n$, ονομάζεται στόχος (goal).
- Ένα σύνολο από όρους Horn που δεν αποτελούν στόχους και των οποίων η κεφαλή χρησιμοποιεί το ίδιο κατηγορηματικό σύμβολο ονομάζεται διαδικασία (procedure).
- Μία διαδικασία η οποία δεν περιέχει μεταβλητές ονομάζεται βάση δεδομένων (database).
- Ένα σύνολο από διαδικασίες ονομάζεται λογικό πρόγραμμα.

Παράδειγμα

- Το πιο κάτω πρόγραμμα περιέχει δύο διαδικασίες, μία από τις οποίες αποτελεί βάση δεδομένων.

1. $q(x,y) \leftarrow p(x,y)$
2. $q(x,y) \leftarrow p(x,z), q(z,y)$

3. $p(b,a) \leftarrow$
4. $p(c,a) \leftarrow$
5. $p(d,b) \leftarrow$
6. $p(e,b) \leftarrow$
7. $p(f,b)$
8. $p(h,g)$
9. $p(i,h)$
10. $p(j,h)$

Αντικαταστάσεις ορθής απάντησης

- Έστω P ένα πρόγραμμα και G ένας στόχος. Μία αντικατάσταση θ ονομάζεται αντικατάσταση ορθής απάντησης αν $P \models \forall(\neg G\theta)$ όπου ο ποσοδείκτης $\forall$ συμβολίζει την εφαρμογή του καθολικού ποσοδείκτη σε όλες τις μεταβλητές που εμφανίζονται ελεύθερες στην G.

- Παράδειγμα:

  Έστω P το σύνολο των αξιωμάτων της αριθμητικής.

  Έστω G η πρόταση $\neg(6+y=13)$. $\neg G$ είναι η πρόταση $6+y=13$ και αντικατάσταση ορθής απάντησης για τη G αποτελεί η $\theta = \{7/y\}$.

  Έστω G η πρόταση $\neg(x = y+13)$. Αντικατάσταση ορθής απάντησης για τη G αποτελεί η $\theta = \{x - 13/y\}$.

Παράδειγμα

ancestor(X,Y) ← parent(X,Y)

ancestor(X,Y) ← ancestor(Z,Y), parent(X,Z)

parent(bob,alan) ←	parent(fred,dave)
parent(catherine,alan) ←	parent(harry,george)
parent(dave,bob)	parent(ida,harry)
parent(ellen,bob)	parent(john,harry)

Ο στόχος

← ancestor(Y, bob), ancestor(bob,Z)

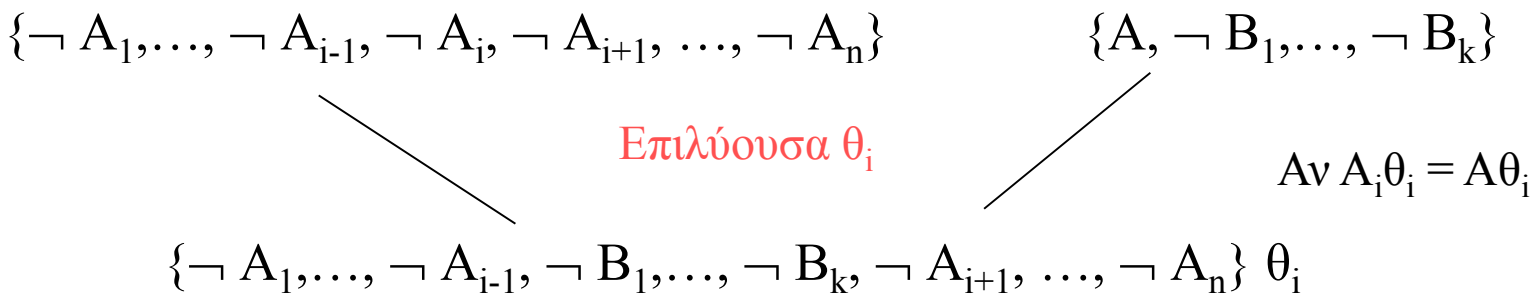
θα επιφέρει το αποτέλεσμα true και θα επιστρέψει την αντικατάσταση ορθής απάντησης Y = dave , Z = alan

Επίλυση SLD

- Η απόδειξη ενός στόχου γίνεται με τη χρήση του κανόνα SLD-Επίλυση ο οποίος αποτελεί ειδική περίπτωση της μεθόδου της επίλυσης εφόσον αναφέρεται σε ζεύγη προτάσεων που περιέχουν μόνο ένα θετικό όρο.

$$\frac{\leftarrow A_1,\ldots,A_{i-1}, A_i, A_{i+1}, \ldots, A_n, \quad A \leftarrow B_1,\ldots,B_k, A_i\theta_i = A\theta_i}{\leftarrow \ (A_1,\ldots,A_{i-1}, B_1,\ldots,B_k, A_{i+1}, \ldots, A_n)\ \theta_i}$$

- Αιτιολόγηση Κανόνα:



Παράδειγμα

- Θεωρήστε το πρόγραμμα P από τη διαφάνεια 5-7. Θέλουμε να αποδείξουμε ότι η πρόταση $\exists y \exists z\ (q(y,b) \wedge q(b,z))$ προκύπτει ως λογικό επακόλουθο του προγράμματος.

- Τότε, αφού $\neg\ (\exists y \exists z\ (q(y,b) \wedge q(b,z))) = \forall y \forall z\ \neg(q(y,b) \wedge q(b,z)) = \forall y\ \forall z\ (\neg q(y,b) \vee \neg\ q(b,z)) = \forall y\ \forall z\ (q(y,b) \wedge q(b,z) \rightarrow \text{False})$ ο στόχος της απόδειξης είναι ο

  $\leftarrow q(y,b), q(b,z)$

- Έχουμε ότι:
  - Από το (1) και το q(y,b), $\leftarrow p(y,b), q(b,z)$
  - Από το (5) και το p(y,b), $\leftarrow q(b,z)$ μέσω της αντικατάστασης {d/y}
  - Από το (1) και το q(b,z), $\leftarrow p(b,z)$
  - Και τέλος, από το (3) και το p(b,z), $\perp$ μέσω της αντικατάστασης {a/z}

- Μπορούμε να επιβεβαιώσουμε ότι πράγματι η {d/y, a/z} είναι αντικατάσταση ορθής απάντησης της πρότασης.

Ορθότητα και Πληρότητα

- Θεώρημα (Ορθότητα): Έστω P ένα σύνολο προτάσεων προγράμματος, και G ένας στόχος. Έστω η ύπαρξη μιας SLD-διάψευσης του G. Αν $\theta = \theta_1\theta_2\ldots\theta_n$ η ακολουθία των ενοποιητών που χρησιμοποιήθηκαν στη διάψευση και σ ο περιορισμός του θ στις μεταβλητές του G, τότε σ είναι μια αντικατάσταση ορθής απάντησης για τον στόχο G.

- Θεώρημα (Πληρότητα): Έστω P ένα σύνολο προτάσεων προγράμματος, και G ένας στόχος. Έστω σ μια αντικατάσταση ορθής απάντησης για τον στόχο G. Τότε υπάρχει μια SLD-διάψευση του G από το P τέτοια ώστε σ είναι ο περιορισμός της ακολουθίας των ενοποιητών που χρησιμοποιήθηκαν στη διάψευση $\theta = \theta_1\theta_2\ldots\theta_n$ στις μεταβλητές του G.

SLD-Επίλυση και μη-ντετερμινισμος (1)

- Θεωρήστε ξανά το πρόγραμμα P από τη διαφάνεια 5-7, την πρόταση $\exists y \exists z\ (q(y,b) \wedge q(b,z))$ και τον σχετικό στόχο $\leftarrow q(y,b), q(b,z)$
- Έχουμε επίσης ότι:
  - Από το (1) και το q(y,b), $\leftarrow p(y,b), q(b,z)$
  - Από το (6) και το p(e,b), $\leftarrow q(b,z)$ μέσω της αντικατάστασης {e/y}
  - Από το (1) και το q(b,z), $\leftarrow p(b,z)$
  - Και τέλος, από το (3) και το p(b,z), $\perp$ μέσω της αντικατάστασης {a/z}
- Επομένως και η {e/y, a/z} είναι αντικατάσταση ορθής απάντησης της πρότασης: μπορεί ο ίδιος στόχος να διαθέτει περισσότερες από μια αντικαταστάσεις ορθής απάντησης.

SLD-Επίλυση και μη-ντετερμινισμος (2)

- Αν, για το ίδιο πρόγραμμα, χρησιμοποιήσουμε ως κανόνα υπολογισμού τον:

  Να διαλέγεις πάντα το τελευταίο στοιχείο του στόχου.

  και ως κανόνα εύρεσης τον:

  Να διαλέγεις πάντα το τελευταίο συμβατό αξίωμα του προγράμματος.

  τότε θα είχαμε:

  ← q(y,b), q(b,z)
  ← q(y,b), p(b,z'), q(z',z)
  ← q(y,b), p(b,z'), p(z,z''), q(z'',z')
  …

- Αυτό δείχνει ότι υπάρχουν μέθοδοι υπολογισμού οι οποίοι δεν μας οδηγούν στην κατασκευή διάψευσης ακόμη και αν υπάρχουν διαψεύσεις.

SLD-Επίλυση και μη-ντετερμινισμος (3)

- Αν, για το ίδιο πρόγραμμα, χρησιμοποιήσουμε ως κανόνα υπολογισμού τον:

  Να διαλέγεις πάντα το πρώτο στοιχείο του στόχου

  τότε θα είχαμε:

← q(y,b), q(b,z)	
← p(y,z'), q(z',b), q(b,z)	(Από το (2))
← q(b,b), q(b,z)	(Από το (5) μέσω της αντικατάστασης {d/y,b/z'})
← p(b,b), q(b,z)	(Από το (1))

- Από τώρα και στο εξής, κανένας όρος δεν ενοποιείται με τον p(b,b). Επομένως, ενώ υπάρχει διάψευση στο πρόγραμμα, η εκτέλεση τερματίζει χωρίς να την εντοπίσει.

Δένδρα SLD

- Η διαδικασία της Επίλυσης μπορεί να αναπαρασταθεί μέσω δένδρων τα οποία έχουν την πρόταση υπό μελέτη ως ρίζα και ακμές ανάμεσα σε δύο κόμβους υπάρχουν αν υπάρχει εφαρμογή του κανόνα SLD-επίλυση που οδηγά από τον κόμβο-πατέρα προς τον κόμβο παιδί.

- Μια κατά πλάτος διερεύνηση σε ένα δένδρο SLD οδηγεί πάντα σε εύρεση διάψευσης αν υπάρχει ενώ η κατά βάθος διερεύνηση δεν συνοδεύεται από τέτοιες εγγυήσεις.

- Εντούτοις, το μέγεθος του δένδρο αυξάνεται εκθετικά από το ένα επίπεδο του δένδρο στο επόμενο και έτσι αποτελεί δαπανηρή διαδικασία.

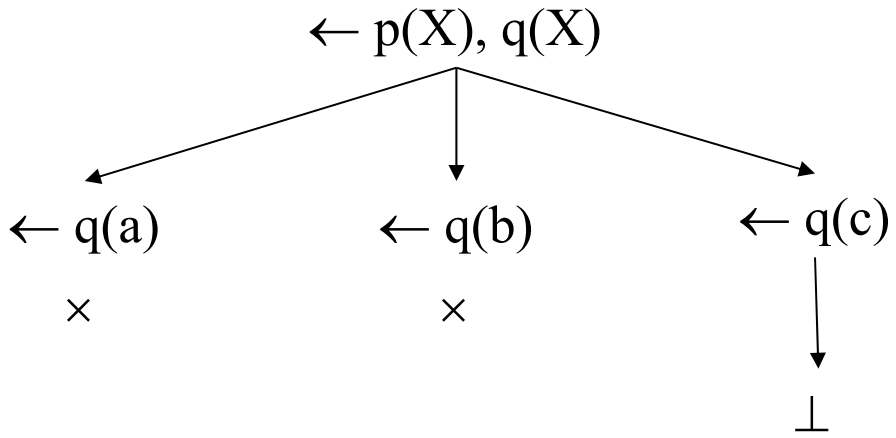
Παράδειγμα

- Θεωρήστε το πρόγραμμα με τα 4 γεγονότα και τον στόχο που εμφανίζονται πιο κάτω:

  p(a) p(b) p(c) q(c)

  ← p(X), q(X)

- Το SLD δένδρο του προγράμματος είναι το:



Εργασία

(α) Θεωρήστε το πιο κάτω πρόγραμμα λογικού προγραμματισμού και χρησιμοποιήστε τη μέθοδο της SLD επίλυσης για να φθάσετε σε διάψευση του στόχου.

add(X, 0, X)

add(X, succ(Y), succ(Z)) ← add(X,Y,Z)

← add(succ(succ(0)), succ(succ(0)), U)

(β) Θεωρήστε το πιο κάτω πρόγραμμα λογικού προγραμματισμού και επιδείξτε τις δυνατές εκτελέσεις του μέσω ενός SLD-δένδρου.

add(X, 0, X)

add(X, succ(Y), succ(Z)) ← add(X,Y,Z)

← add(U, V, succ(succ(succ(0))))

Prolog

- PROgramming in LOGic
- Prolog: Alain Colmerauer 1973, R. Kowaksi Αρχή Επίλυσης
- Prolog compiler: David Warren, 1977
- Δηλωτική Γλώσσα: διάκριση μεταξύ λογικής και ελέγχου
- Εξίσωση Kowalski

  Πρόγραμμα = Λογική + Έλεγχος
- Κανόνας Υπολογισμού Prolog:
  - Επέλεξε το αριστερότερο στοιχείο από τον στόχο
- Κανόνας Εύρεσης Prolog:
  - Επέλεξε τον πρώτο κατάλληλο κανόνα διασχίζοντας το πρόγραμμα από την αρχή και προχωρώντας προς το τέλος.

Prolog – Οντότητες

- Ένα πρόγραμμα στην Prolog αφορά οντότητες του προβλήματος με το οποίο ασχολείται, τις ιδιότητές τους και τις σχέσεις τους. Μπορεί να περιλαμβάνει
  - Γεγονότα: παριστάνουν ιδιότητες και σχέσεις που έχουν αντικείμενα του προβλήματος μεταξύ τους – τα δεδομένα του προβλήματος
  - Κανόνες: παριστάνουν γενικευμένες σχέσεις που συνδέουν τα αντικείμενα του προβλήματος – οι συλλογισμοί μέσω των οποίων μπορούμε να καταλήξουμε σε καινούρια συμπεράσματα/γεγονότα
  - Ερωτήματα: παριστάνουν τα ζητούμενα από το πρόγραμμα
  - Σχόλια
- Ο συμβολισμός της Prolog διαφέρει από αυτόν που χρησιμοποιούμε μέχρι τώρα:
  - Οι μεταβλητές ξεκινούν με κεφαλαία γράμματα
  - Τα σύμβολα και οι σταθερές ξεκινούν με μικρά γράμματα
  - Το σύμβολο :- χρησιμοποιείται αντί του ←

Παράδειγμα

ancestor(X,Y) :- parent(X,Y)

ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y)

parent(bob,alan)

p(catherine,alan)

p(dave,bob)

p(ellen,bob)

p(fred,dave)

p(harry,george)

p(ida,harry)

p(john,harry)

Ο στόχος

:- ancestor(Y, bob), ancestor(bob,Z)

Θα επιφέρει το αποτέλεσμα true και θα επιστρέψει την αντικατάσταση ορθής απάντησης Y = dave , Z = alan

Εκτέλεση Προγράμματος

- Η αναζήτηση διάψευσης για την επίτευξη ενός στόχου μέσω των κανόνων υπολογισμού και εύρεσης της Prolog πιθανόν να οδηγήσει σε μη τερματισμό ακόμη και αν υπάρχει λύση.

- Επαφίεται στον προγραμματιστή να διατάξει τους κανόνες του προγράμματος με τέτοιο τρόπο έτσι ώστε να αποφευχθεί ο μη τερματισμός.

- Αποδοτική μέθοδος εύρεσης όρων προς ενοποίηση.

- Περιορισμένος αριθμός δομών δεδομένων προς διατήρηση

- Ανάγκη για οπισθοδρόμηση

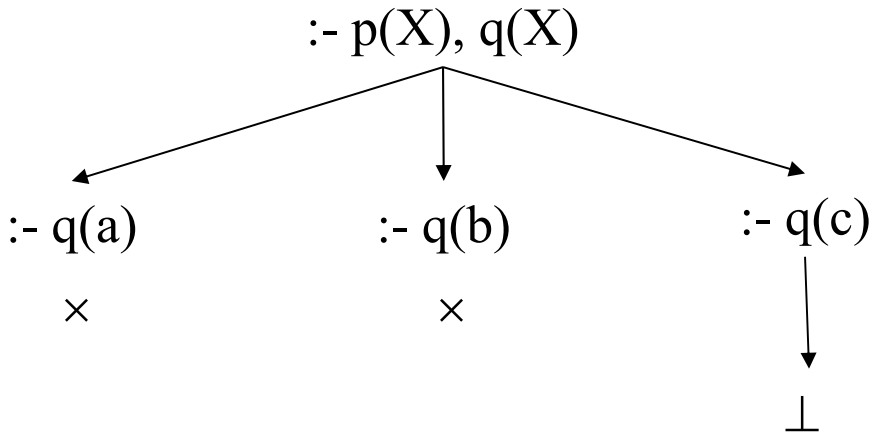
Παράδειγμα

- Θεωρήστε το πρόγραμμα με τα 4 γεγονότα και τον στόχο που εμφανίζονται πιο κάτω:

  p(a) p(b) p(c) q(c)

  :- p(X), q(X)

- Το δένδρο που θα κτιστεί κατά τη διαδικασία επίλυσης είναι το



Μη λογικά Κατηγορήματα

- Η Prolog περιέχει μη-λογικά κατηγορήματα τα οποία διευρύνουν την εκφραστικότητα και αποδοτικότητα της γλώσσας όπως:
  - Κατηγορήματα για Input/output: get, put
  - Αριθμητικές Εκφράσεις και σχετικές διαδικασίες υπολογισμού

    Result is Expression

- Παράδειγμα

```
selling_price(Item, Price) :- list_price(Item, List),
                    discount_percent(Item, Discount),
                    Price is List - List*Discount/100
```

Η τιμή της έκφρασης List – List*Discount/100 θα τύχει υπολογισμού και θα ενοποιηθεί με τη μεταβλητή Price

- *Από τη στιγμή που μια μεταβλητή πάρει κάποια τιμή μέσω μιας εντολής is οποιαδήποτε επιπρόσθετη προσπάθεια να ενοποιηθεί με μια άλλη τιμή θα οδηγήσει σε λάθος.*

Cuts

- Μια αποκοπή (cut) αποτελεί μια σήμανση για διακοπή της μεθόδου της επίλυσης σε συγκεκριμένο σημείο.
- Θεωρήστε το πιο κάτω πρόγραμμα για υπολογισμό παραγοντικών αριθμών (fact) και έλεγχο κατά πόσο ο N-ιοστός παραγοντικός αριθμός είναι άρτιος .

```
fact(0,1).
fact(N,F) :-
   N1 is N-1,
   fact(N1,F1),
   F is N*F1.

check(N):- fact(N,F),
           even(F).
```

- Έστω το ερώτημα check(0). Αφού fact(0,1) και όχι even(1), η διαδικασία της επίλυσης θα οπισθοδρομήσει και θα επιχειρήσει να χρησιμοποιήσει τον δεύτερο κανόνα fact. Και η κλήση fact(-1,1) θα ξεκινήσει ένα ατέρμονο υπολογισμό.

Cuts

- Αυτό μπορεί να αποφευχθεί μέσω μιας αποκοπής.
- Γράφουμε

```
fact(0,1):-! .
```

απαγορεύοντας οπισθοδρόμηση μετά από αυτό το σημείο στη διαδικασία.

- Οι αποκοπές επεμβαίνουν στη φιλοσοφία του Λογικού Προγραμματισμού και είναι καλύτερο να αποφεύγονται. Στο παράδειγμα θα μπορούσαμε να γράψουμε

```
fact(N,F) :-
        N > 0,
        N1 is N-1,
        fact(N1,F1),
        F is N*F1 .
```