ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση της γλώσσας Λογικού Προγραμματισμού PrefLog

Λυδία Χ. Ζογμπή

Επιβλέποντες: **Παναγιώτης Ροντογιάννης,** Καθηγητής
Άγγελος Χαραλαμπίδης, Μεταδιδάκτορας
Αντώνιος Τρουμπούκης, Υποψήφιος Διδάκτωρ

ΑΘΗΝΑ

ΝΟΕΜΒΡΙΟΣ 2016

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση της γλώσσας Λογικού Προγραμματισμού PrefLog

Λυδία Χ. Ζογμπή
ΑΜ: 1115201100073

Επιβλέποντες: **Παναγιώτης Ροντογιάννης,** Καθηγητής
Άγγελος Χαραλαμπίδης, Μεταδιδάκτορας
Αντώνιος Τρουμπούκης, Υποψήφιος Διδάκτωρ

ΠΕΡΙΛΗΨΗ

Οι γλώσσες Λογικού Προγραμματισμού στηρίζονται στη λογική πρώτης τάξης (first order logic) και αποτελούνται από ένα σύνολο λογικών κανόνων που εκφράζουν το ζητούμενο του προβλήματος. Ακόμα και χωρίς συναρτησιακά σύμβολα, οι λογικοί κανόνες έχουν τη δυνατότητα να εκφράσουν υπολογισμούς που δεν εκφράζονται στις συμβατικές γλώσσες χειρισμού δεδομένων. Παρόλα αυτά όμως, δεν ήταν εφικτή η έκφραση προτιμήσεων, καθώς για κάθε πρόγραμμα, δύο είναι οι πιθανές καταστάσεις για οποιοδήποτε ερώτημα τεθεί. Σύμφωνα με τις προδιαγραφές του εκάστοτε προγράμματος, κάτι μπορεί να ισχύει και να είναι αληθές ή να μην ισχύει και να είναι ψευδές.
Στα πλαίσια αυτής της πτυχιακής εργασίας, θα παρουσιαστεί η υλοποίηση μιας νέας γλώσσας Λογικού Προγραμματισμού η οποία θα δίνει στο χρήστη τη δυνατότητα να δηλώσει προτιμήσεις με τη χρήση ειδικών τελεστών. Αυτό επιτυγχάνεται μέσω ενός μη πεπερασμένου συνόλου από τιμές αληθείας. Η υλοποίηση αυτή έχει στηριχθεί στην ήδη υπάρχον σύστημα IRIS Reasoner και με τις κατάλληλες δομές δεδομένων και αλλαγές επιτυγχάνεται η επιπλέον εκφραστικότητα που επιθυμούμε. Στόχος είναι η νέα γλώσσα να είναι όσο το δυνατόν πιο αποδοτική στην εκτέλεση των προγραμμάτων, αλλά ταυτόχρονα και εύκολη για κάθε χρήστη να τη χρησιμοποιήσει.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Επιστήμη υπολογιστών**,** Λογικός Προγραμματισμός, Γλώσσες προγραμματισμού

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Λογική πρώτης τάξης, Προτιμήσεις, Λογικοί κανόνες

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διαικπεραίωση της παρούσας πτυχιακής εργασίας θα ήθελα να ευχαριστήσω τους επιβλέποντες, καθηγητή Παναγιώτη Ροντογιάννη, υποψήφιο διδάκτορα Αντώνιο Τρουμπούκη και μεταδιδάκτορα Άγγελο Χαραλαμπίδη για συνεργασία τους και όλη τη βοήθεια που προσέφεραν.
Έπειτα, θα ήθελα να ευχαριστήσω την οικογένεια μου για τη συνεχή στήριξη και συμπαράσταση όλα τα χρόνια των σπουδών μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εργασία αποτελεί μέρος των υποχρεώσεων για την λήψη πτυχίου στο Τμήμα Πληροφορικής και Τηλεπικοινωνιών του Εθνικού Καποδιστριακού Πανεπιστημίου Αθηνών. Η εργασία εκπονήθηκε κατά τη διάρκεια του ακαδημαϊκού έτους 2015-2016 υπό την επίβλεψη του καθηγητή Ροντογιάννη Παναγιώτη, του υποψήφιου διδάκτωρα Αντώνιου Τρουμπούκη και του μεταδιδάκτορα Άγγελου Χαραλαμπίδη.

1. ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο παρουσιάζεται αρχικά το θέμα και ο στόχος της παρούσας πτυχιακής εργασίας. Έπειτα γίνεται μια περιγραφή της διάρθρωσης της εργασίας στα επόμενα κεφάλαια.

1.1 Αντικείμενο εργασίας

Το αντικείμενο αυτής της πτυχιακής εργασίας είναι να πραγματοποιηθεί η υλοποίηση της γλώσσας PrefLog [1]. Η γλώσσα προγραμματισμού PrefLog είναι μια γλώσσα Λογικού Προγραμματισμού και είναι γλώσσα πρώτης τάξης. Η ιδιαιτερότητα της σε σχέση με άλλες παρόμοιες γλώσσες είναι ότι παρέχει τη δυνατότητα να εκφραστούν προτιμήσεις, με τη χρήση ειδικών τελεστών. Αυτό επιτυγχάνεται μέσω ενός μη πεπερασμένου συνόλου από τιμές αληθείας, από το οποίο αποδίδεται η κατάλληλη σε κάθε πρόγραμμα. Μια ήδη υπάρχουσα υλοποίηση με παρόμοια σύνταξη, αλλά χωρίς την επιπλέον εκφραστικότητα, είναι της IRIS Reasoner. Στην υλοποίηση του συστήματος IRIS θα στηριχθεί η υλοποίηση της PrefLog, που παρουσιάζεται στην παρούσα πτυχιακή εργασία. Σκοπός είναι να μπορέσει ο χρήστης να αξιοποιήσει τις πρόσθετες δυνατότητες που παρέχει αυτή η γλώσσα με όσο το δυνατόν πιο εύκολο τρόπο. Επιπλέον, είναι επιθυμητό η παρούσα υλοποίηση να είναι όσο πιο αποδοτική γίνεται και η πρόσθετη επεξεργασία των προγραμμάτων, που απαιτείται για την εξαγωγή αποτελέσματος, να μην επιβαρρύνει αισθητά το χρόνο εκτέλεσης.

1.2 Διάρθρωση εργασίας

Στο κεφάλαιο 1 γίνεται μια εισαγωγική παρουσίαση της εργασίας και το τι αυτή πραγματεύεται.

Στο κεφάλαιο 2 γίνεται μια αναφορά στο απαραίτητο θεωρητικό υπόβαθρο, για να αποσαφηνιστούν οι έννοιες και οι όροι που θα χρησιμοποιηθούν στα επόμενα κεφάλαια. Επίσης, παρουσιάζεται το σύστημα IRIS Reasoner.

Το κεφάλαιο 3 είναι μια περιγραφή της σημασιολογίας και των ιδιοτήτων της γλώσσας PrefLog.

Το κεφάλαιο 4 αποτελεί μια περιγραφή του τρόπου που επιτεύχθηκε η υλοποίηση της γλώσσας PrefLog και τι αλλαγές χρειάστηκε να γίνουν στο σύστημα IRIS.

Το κεφάλαιο 5 παρέχει συγκρίσεις και μετρήσεις μεταξύ των δύο διαφορετικών αλγορίθμων αποτίμησης που υλοποιήθηκαν, όσον αφορά το συνολικό χρόνο εκτέλεσης.

Τέλος, στο κεφάλαιο 6 παρουσιάζονται κάποιες προτάσεις μελλοντικών επεκτάσεων που θα ενισχύσουν ακόμα περισσότερο την εκφραστικότητα της γλώσσας.

2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Ο Λογικός Προγραμματισμός είναι ένα θεμελιώδες στυλ προγραμματισμού βασισμένο στη Λογική. Ένα πρόγραμμα γραμμένο σε μια γλώσσα λογικού προγραμματισμού αποτελείται από ένα σύνολο από προτάσεις, που εκφράζουν γεγονότα και κανόνες σχετικά με το εκάστοτε πρόβλημα. Οι δύο κυριότερες ομάδες γλώσσεων Λογικού Προγραμματισμού συμπεριλαμβάνουν τις Prolog και Datalog.

Στο κεφάλαιο αυτό παρουσιάζονται κάποιες βασικές έννοιες του Λογικού Προγραμματισμού και λεπτομέρειες σχετικά με τη σημασιολογία και τις ιδιότητες της Datalog.

2.1 Συντακτικό Datalog

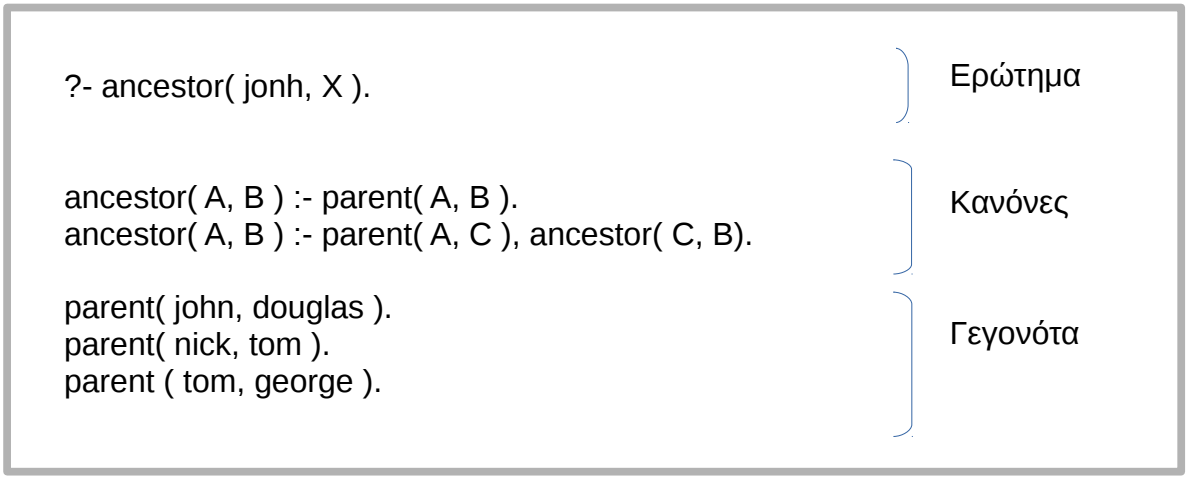
Το θεωρητικό υπόβαθρο του Λογικού Προγραμματισμού είναι η λογική πρώτης τάξης, όπου ορίζονται τα σύνολα των κατηγορημάτων, των συναρτησιακών συμβόλων και των μεταβλητών. Στην περίπτωση της Datalog δεν υποστηρίζονται τα συναρτησιακά σύμβολα.

Οι μεταβλητές μπορεί να είναι ένα κεφαλαίο γράμμα ή μια συμβολοσειρά που ξεκινάει με κεφαλαίο γράμμα. Ως *αναγνωριστικό* ορίζεται μια ακολουθία χαρακτήρων που ξεκινάει με πεζό λατινικό χαρακτήρα και δεν περιέχει κενό ή κάποιον από τους ειδικούς χαρακτήρες (, `, ',), =, :, ., ~, ?, '', % και ένα αναγνωριστικό ή μια συμβολοσειρά αποτελούν ενα *κατηγόρημα*. Κάθε κατηγόρημα χαρακτηρίζεται από το *βαθμό* ή *τάξη* του, δηλαδή έναν ακέραιο αριθμό που δηλώνει το πλήθος των παραμέτρων που δέχεται. Για παράδειγμα, το point(X, Y) έχει βαθμό 2, εφόσον κάθε σημείο καθορίζεται από δύο συντεταγμένες. Κατηγορήματα με βαθμό 0 ονομάζονται και *σταθερές*.

Μια σταθερά ή μια μεταβλητή συνθέτουν έναν *όρο*. Ένας όρος που δεν περιέχει μεταβλητές θεωρείται πλήρως αποτιμημένος.
Ένα κατηγόρημα μηδενικού βαθμού ή ένα κατηγόρημα θετικού βαθμού, του οποίου οι παράμετροι είναι όροι, ονομάζεται *ατομικός τύπος* ή *άτομο*.

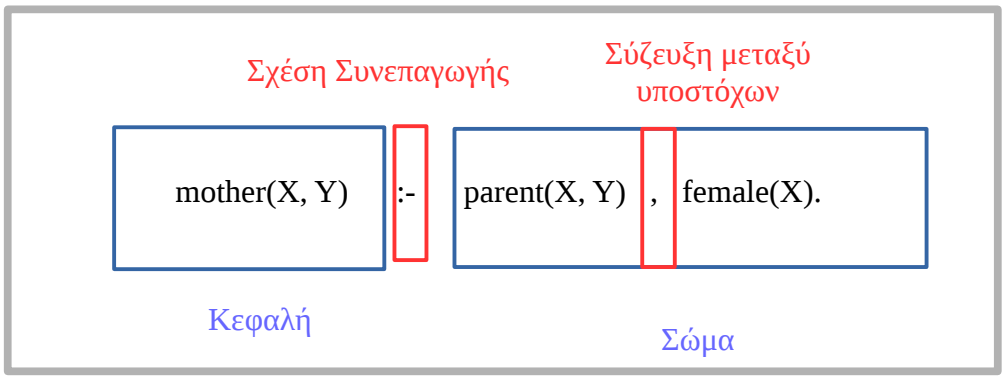
Ένα άτομο ή η άρνηση ενός ατόμου αποτελούν έναν άλλο δομικό τύπο που ονομάζεται *λεκτικό*. Τα λεκτικά μαζί με τις μεταβλητές που περιλαμβάνουν διαμορφώνουν μια πολύ σημαντική κατηγορία τύπων, τις *προτάσεις*.
Οι προτάσεις χωρίζονται σε πλήρως αποτιμημένα *γεγονότα*, σε λογικούς *κανόνες* και *ερωτήματα*. Τα γεγονότα αποθηκεύονται σε άμεση σχεσιακή βάση(EDB) και οι κανόνες σε έμμεση σχεσιακή βάση (IDB). Οι κανόνες αποτελούνται από το *σώμα* και την *κεφαλή*. Η κεφαλή είναι ένα άτομο και το σώμα αποτελείται από ένα λεκτικό ή τη σύζευξη περισσοτέρων λεκτικών.
Τα γεγονότα σε συνδυασμό με τους κανόνες απαρτίζουν ένα πρόγραμμα.

Για παράδειγμα, θεωρείστε το ακόλουθο πρόγραμμα υλοποιημένο σε Datalog:



Εικόνα 1: Πρόγραμμα Datalog

Η ανατομία ενός κανόνα μπορεί να αναλυθεί ως εξής,

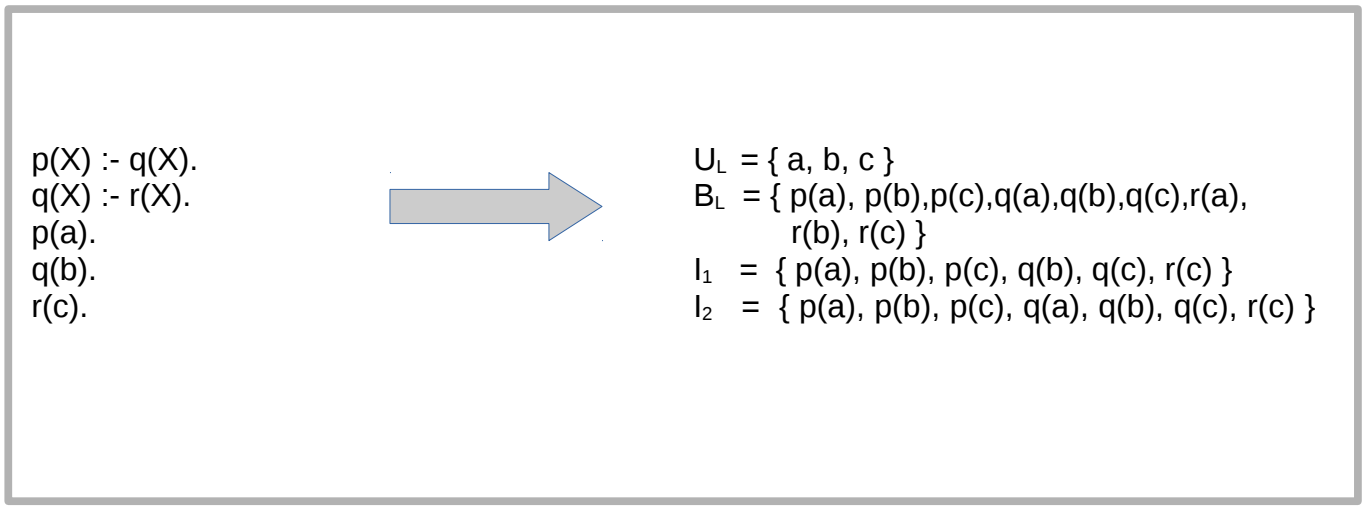


Εικόνα 2: Ανατομία κανόνα

2.2 Herbrand Βάση και Herbrand Σύμπαν

Το σύνολο των πλήρως αποτιμημένων όρων που προκύπτουν από τα συναρτησιακά σύμβολα και τις σταθερές μιας γλώσσας L πρώτης τάξης, ονομάζεται Herbrand σύμπαν (Herbrand Universe) και συμβολίζεται ως U_L. Αν δεν υπάρχει κάποια σταθερά, διαλέγουμε μία αυθαίρετα για να μην είναι κενό το U_L . Όλα τα αποτιμημένα άτομα που προκύπτουν από τα κατηγορήματα του προβλήματος και θέτοντας ως ορίσματά τους όρους από το Herbrand σύμπαν, συνθέτουν τη Herbrand βάση (Herbrand Base), B_L . Κάθε υποσύνολο της βάσης Herbrand ονομάζεται Herbrand ερμηνεία (Herbrand Interpretation). Μια ερμηνεία αποτελείται από όλα εκείνα τα αποτιμημένα άτομα που ισχύουν σε μια δεδομένη κατάσταση, επομένως είναι μια περιγραφή αυτής της κατάστασης. Το σύνολο όλων των ερμηνειών για ένα πρόγραμμα P είναι $2^{BL.}$ Εάν μια ερμηνεία επαληθεύει όλους τους κανόνες ενός προγράμματος, τότε αυτή η ερμηνεία αποτελεί μοντέλο του προγράμματος αυτού και η τομή όλων των μοντέλων είναι το ελάχιστο μοντέλο Herbrand (Least Herbrand Model), M_p. [2]

Ένα απλό παραδείγματα εύρεσης του σύμπαντος, της βάσης και κάποιων ερμηνειών είναι τα ακόλουθα:



Εικόνα 3: Παράδειγμα Herbrand βάσης και σύμπαντος

Παρατηρούμε ότι και οι δύο ερμηνείες είναι μοντέλα του προγράμματος και η I_2 συμπίπτει με το M_p, είναι δηλαδή το ελάχιστο μοντέλο Herbrand.

2.3 Από κάτω προς τα πάνω αποτίμηση

Βασική αρχή του από-κάτω-προς-τα-πάνω υπολογισμού είναι ότι ξεκινάει από τα ήδη γνωστά γεγονότα και παράγει καινούρια μέχρι να υπολογιστεί το ελάχιστο μοντέλο του προγράμματος. Υπάρχουν δύο αλγόριθμοι που χρησιμοποιούνται για αυτού του είδους την αποτίμηση, ο αφελής και ο ημι-αφελής.

2.3.1 Σημασιολογία Σταθερού Σημείου

Η σημασιολογία σταθερού σημείου (Fixpoint Semantics) παρέχει έναν απλό κι αποδοτικό τρόπο υπολογισμού του ελάχιστου μοντέλου ενός προγράμματος [4]. Ιδιαίτερης σημασίας είναι ο τελεστής T_P , που αναπαριστά μια απεικόνιση από ερμηνείες σε ερμηνείες ($T_P : I_P \rightarrow I_P$) και έχει ελάχιστο σταθερό σημείο, αν είναι συνεχής και οι ερμηνείες είναι πλήρες πλέγμα. Η απεικόνιση T_P έχει σταθερό σημείο α, όταν $T_P(α) = α$. Ο εν λόγω τελεστής περιέχει όλα τα λογικά επακόλουθα που προκύπτουν από το πρόγραμμα P. Το ελάχιστο σταθερό σημείο του T_P, συμβολίζεται ως lfp(T_P), και αποτιμάται αν εφαρμόσουμε διαδοχικά τον τελεστή T_P στα παραγόμενα γεγονότα (όσο εξακολουθούν να παράγονται καινούρια), ξεκινώντας από το βασικό στοιχείο του πλέγματος των ερμηνειών, δηλαδή το κενό σύνολο (∅). Το ελάχιστο μοντέλο Herbrand ισούται με το ελάχιστο σταθερό σημείο του T_P τελεστή, δηλαδή ισχύει M_P=lfp(T_P). Ακολουθεί μια ενδεικτική αναπαράσταση της σημασιολογίας, για το παρακάτω πρόγραμμα.

```
p(X):- q(X).
q(X):- r(X).
p(a).
q(b).
r(c).
```

Βήμα 1: παίρνουμε τα γεγόνοτα
- $T_P(∅) = \{ p(a), q(b), r(c) \}$

Βήμα 2: $T_P(T_P(∅)) = T_P(∅)$ και ό,τι μπορούμε να συμπεράνουμε από τα γεγόνοτα
- $T_P(T_P(∅)) = T_P\{ p(a), q(b), r(c) \} = \{ p(a), q(b), r(c), p(b), q(c) \}$

Βήμα 3: $T_P(T_P(T_P(∅))) = T_P(T_P(∅))$ και ό,τι μπορούμε να συμπεράνουμε από το $T_P(T_P(∅))$
- $T_P(T_P(T_P(∅))) = T_P\{ p(a), q(b), r(c), p(b), q(c) \} = \{p(a), q(b), r(c), p(b), q(c), p(c) \}$

Βήμα 4: $T_P(T_P(T_P(T_P(∅)))) = T_P(T_P(T_P(∅)))$ και ό,τι μπορούμε να συμπεράνουμε από το $T_P(T_P(T_P(∅)))$
- $T_P(T_P(T_P(T_P(∅)))) = T_P\{ p(a), q(b), r(c), p(b), q(c) \} = \{p(a), q(b), r(c), p(b), q(c), p(c) \}$

Βήμα 5: Δεν παράγεται τίποτα νέο => fixpoint
- $T_P(T_P(T_P(T_P(T_P(∅))))) = T_P\{ p(a), q(b), r(c), p(b), q(c) \} = \{p(a), p(b), p(c), q(b), q(c), r(c) \}$
$= T_P(T_P(T_P(T_P(∅))))$.

Εικόνα 4: Σημασιολογία σταθερού σημείου

2.3.2 Αφελής αποτίμηση για Datalog

Κατά τη διαδικασία της αφελούς αποτίμησης εφαρμόζονται σε κάθε βήμα όλα τα γεγονότα σε όλους τους κανόνες και όσο συνεχίζεται να παράγονται νέα γεγονότα επαναλαμβάνεται αυτή η διαδικασία. Ο υπολογισμός ξεκινάει με την ερμηνεία Herbrand του κενού, ∅, και εφαρμόζει διαδοχικά τον τελεστή T_P σε αυτήν. Το μειονέκτημά της είναι ότι υπολογίζει ξανά γεγονότα τα οποία έχουν ήδη αποτιμηθεί και είναι γνωστά.

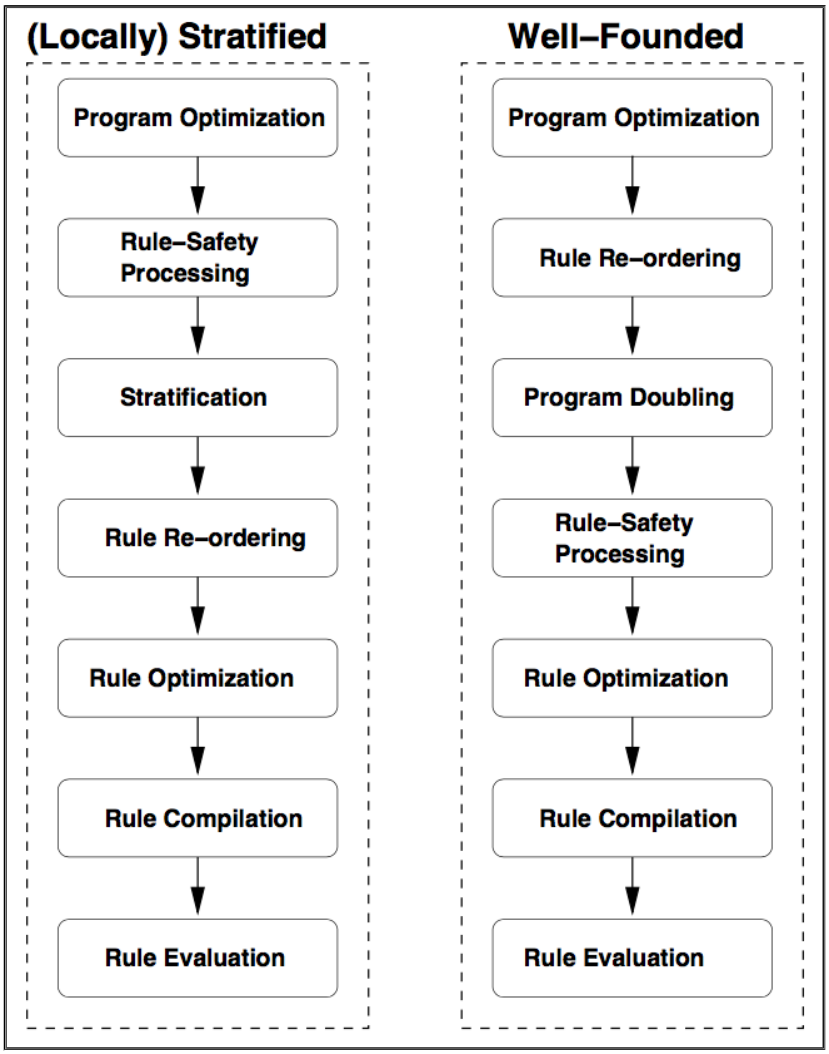
2.3.3 Ημι-αφελής Αποτίμηση

Κατά την ημι-αφελή αποτίμηση η διαφορά είναι ότι δεν εφαρμόζονται όλα τα γεγονότα σε όλους τους κανόνες διαρκώς, αλλά σε κάθε βήμα εφαρμόζονται μόνο τα γεγονότα που παράχθηκαν στο προηγούμενο βήμα. Αρχικά, ξεκινάει ο υπολογισμός με τα γεγονότα του προβλήματος. Έπειτα, παράγονται νέα από τους κανόνες όταν τα εφαρμόσουμε σε αυτούς. Στη συνέχεια, τα αρχικά γεγονότα δε θα ξαναχρησιμοποιηθούν.

2.4 IRIS Reasoner

Η IRIS Reasoner είναι μια μηχανή ανοιχτού λογισμκού για Datalog, όπου παρέχει επιπλέον XML Schema τύπο δεδομένων, ενσωματωμένα κατηγορήματα, συναρτησιακά σύμβολα και Well-founded τεχνική για αποτίμηση της άρνησης [3]. Η γλώσσα IRIS έχει υλοποιηθεί ώστε να είναι όσο το δυνατόν πιο δομοστοιχειωτή γίνεται, με σκοπό μελλοντικά να είναι εφικτή η προσθήκη πρόσθετων στρατηγικών αποτίμησης. Οι στρατηγικές αποτίμησης κανόνων που υποστηρίζει αυτή τη στιγμή είναι για από-πάνω-προς-τα-κάτω (top-down) αποτίμηση οι SLDNF και OLDT, και για από-κάτω-προς-τα-πάνω (bottom-up) η αφελής αποτίμηση (Naive) και η ημι-αφελής (Semi-naive) [3]. Στα πλαίσια της παρούσας πτυχιακής εργασίας θα αναλυθούν μόνο οι στρατηγικές για από-κάτω-προς-τα-πάνω αποτίμηση. Στην IRIS οι μεταβλητές ξεχωρίζουν διότι γράφονται πάντα μετά από ένα αγγλικό ερωτηματικό, δηλαδή για παράδειγμα για τη μεταβλητή x έχουμε “?x“. Δε χρειάζεται να είναι μόνο ένα γράμμα ή να ξεκινάει απαραίτητα με κεφαλαίο, όπως προϋποθέτουν άλλες γλώσσες. Αυτή η προσέγγιση έχει διατηρηθεί και στην υλοποίηση της PrefLog.

2.4.1 Φάσεις επεξεργασίας προγραμμάτων στο σύστημα IRIS



Εικόνα 5: Φάσεις επεξεργασίας στην IRIS

Στην Εικόνα 5 απεικονίζονται οι δύο βασικές στρατηγικές αποτίμησης του συστήματος IRIS [3]. Η πρώτη στρατηγική χρησιμοποιεί την τακτική της στρωματοποίησης (stratification), που χρειάζεται για τη σωστή αποτίμηση όταν υπάρχει άρνηση στο σώμα των κανόνων. Η δεύτερη προσσέγγιση επιλύει αυτό το πρόβλημα με την τεχνική του διπλασιασμού του προγράμματος (program doubling), αντί για στρωματοποίηση. Με αυτό τον τρόπο, δημιουργείται μια εκδοχή του προβλήματος για τους κανόνες με άρνηση και μια γι'αυτούς χωρίς άρνηση.

Στη φάση της βελτιστοποίησης του προγράμματος (program optimization) εφαρμόζονται διάφορες τεχνικές για αποδοτικότερη και γρηγορότερη αποτίμηση. Μια από αυτές ονομάζεται Magic Sets και αναδιαμορφώνει το σύνολο των κανόνων σύμφωνα με το εκάστοτε ερώτημα, ώστε να υπολογιστούν μόνο οι πλειάδες που ενδέχεται να το ικανοποιήσουν.

Κατά την επεξεργασία ασφαλείας των κανόνων (rule safety processing) ελέγχεται αν οι πιθανές τιμές για όλες τις μεταβλητές είναι περιορισμένες. Αυτό ικανοποιείται αν η μεταβλητή εμφανίζεται τουλάχιστον μια φορά σε ένα θετικό λεκτικό στο σώμα του κανόνα, αν εξισώνεται με κάποια σταθερά ή αν εξισώνεται με άλλη μεταβλητή που είναι γνωστό ότι πληροί τις προϋποθέσεις.

Στην επανατοποθέτηση των κανόνων (rule reordering) δίνεται προτεραιότητα στην αποτίμηση κανόνων που ενδέχεται να συμβάλλουν στον υπολογισμό άλλων κανόνων. Έπειτα, κατά τη βελτιστοποίηση των κανόνων τακτικές όπως η αντικατάσταση μεταβλητών με σταθερές στο σώμα του κανόνα, αν οι συγκεκριμένες μεταβλητές έχουν εξισωθεί ή αποτιμηθεί ως τις αντίστοιχες σταθερές, ή η αφαίρεση διπλότυπων λεκτικών στον ίδιο κανόνα, εξασφαλίζουν αποδοτικότερους υπολογισμούς.

Στη συνέχεια, κατά τη μεταγλώττιση ενός κανόνα (rule compilation) υπολογίζονται εκ των προτέρων όλες οι πιθανές πληροφορίες που είναι απαραίτητες για την αποτίμηση του κανόνα. Ο αρχικός κανόνας εισόδου μετατρέπεται στο μεταγλωττισμένο κανόνα, που μπορεί να επεξεργαστεί γρήγορα από τον κατάλληλο εκτιμητή (evaluator). Ο μεταγλωττισμένος κανόνας αποτελείται από επιμέρους οντότητες, ανάλογα με τη μορφή του αρχικού κανόνα. Αυτές οι οντότητες έχουν όλες τις απαραίτητες πληροφορίες και με βάση αυτές πραγματοποιείται η αποτίμηση στο επόμενο στάδιο.

Τέλος, στη φάση της αποτίμησης υπάρχουν διαθέσιμοι 2 εκτιμητές, ο αφελής (naive evaluator) και ο ημι-αφελής (semi-naive evaluator). Καθένας από αυτούς χρησιμοποιεί τον αντίστοιχο αλγόριθμο ώστε να εφαρμόζει γνωστά γεγονότα στο σύνολο των κανόνων με σκοπό να παραχθούν νεα.

3. Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PrefLog

Η γλώσσας πρώτης τάξης PrefLog παρέχει τη δυνατότητα αναπαράστασης προτιμήσεων, μέσω ενός μη πεπερασμένου συνόλου από τιμές αληθείας, που μπορεί να έχει ως έξοδο ένα πρόγραμμα. Το σύνολο αυτό μπορεί να αναπαρασταθεί ως εξής, $F_0 < F_1 < F_2 < \cdots < 0 < \cdots < T_2 < T_1 < T_0$, όπου η ενδιάμεση τιμή 0, χρησιμοποιείται όταν ο υπολογισμός του ελάχιστου μοντέλου Herbrand απαιτεί άπειρο πλήθος βημάτων. Η PrefLog παρέχει επίσης ένα σύνολο από τελεστές που διευκολύνουν την έκφραση των προτιμήσεων, κατά τη σύνταξη των κανόνων του προγράμματος. Η υλοποίηση της στηρίχθηκε στην υλοποίηση της IRIS Reasoner, αλλά με τις κατάλληλες αλλαγές, ώστε να μπορεί να αξιοποιηθεί η επιπλέον εκφραστικότητα της PrefLog. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι Java.

3.1 Γραμματική της γλώσσας PrefLog

Τα βασικά δομικά χαρακτηριστικά των δύο γλωσσών, IRIS και PrefLog, είναι σε μεγάλο βαθμό ίδια. Η βασική διαφορά που παρουσιάζεται είναι ότι η PrefLog παρέχει επιπλέον τελεστές για την έκφραση των προτιμήσεων. Επομένως, η γραμματική της έχει μεταβληθεί ώστε να μπορεί να τους υποστηρίξει. Οι νέοι τελεστές είναι οι “and”, “or”, “opt”, “alt” και “ε”.

Αναλυτικότερα, στη γλώσσα IRIS το σώμα των κανόνων απαρτίζεται από ένα λεκτικό ή τη σύζευξη περισσότερων, η οποία εκφράζεται με κόμμα “,”. Στην PrefLog οι τελεστές που ενώνουν τα λεκτικά στο σώμα ενός κανόνα είναι αυτοί της σύζευξης και της διάζευξης, οι οποίοι εκφράζονται μέσω των and και or, αντίστοιχα. Οι τελεστές opt^n, alt^n και ε^n χρησιμοποιούνται για να διαμορφώσουν τις προτιμήσεις, όπου είναι επιθυμητό, και γι’ αυτό ονομάζονται τελεστές προτίμησης (preferential operators). Όποτε είναι επιθυμητή η ύπαρξη θετικού εκθέτη, $n > 0$, αυτό υποδηλώνει τη γραφή n φορών στη σειρά του εκάστοτε τελεστή, όπως για παράδειγμα για alt^3 A θα έχουμε alt alt alt A. Επιπλέον, ο τελεστής and μπορεί να ακολουθείται από τον opt^n ή τον ε^n και ο τελεστής or από τον alt^n ή τον ε^n.

Επομένως, το σώμα των κανόνων μπορεί να αποτελείται από:

- Ένα λεκτικό, το οποίο μπορεί να έπεται ή όχι του τελεστή ε^n, όπως p(?x) ή ε p(?x).
- Από τη σύζευξη ή τη διάζευξη δύο ή περισσότερων λεκτικών, όπου ο τελεστής and ενδέχεται να ακολουθείται από τον opt^n και ο or από τον alt^n. Για παράδειγμα,

  p(?x) and opt opt q(?x) or z(?x).

Ακολουθεί ένα ολοκληρωμένο παράδειγμα προγράμματος σε PrefLog για αποσαφήνιση όλων των παραπάνω.

```
flight(?f):- to("athens", "boston", ?f) and has_stopover(?f) and opt carrier(?f, "aegean").
has_stopover(?f) :- stopover(?f, "rome") or alt stopover(?f, "london") .

to("athens", "boston", “fl1”).
to("athens", "boston", ”fl2”).
to("athens", "boston", ”fl3”).
stopover(“fl1”, "rome").
stopover(“fl2”, "london").
stopover(“fl3”, "rome").
carrier(“fl1”, "british").
carrier(“fl2”, "aegean").
carrier(“fl3”, "aegean").
```

Εικόνα 6: Πρόγραμμα PrefLog

Η έξοδος του παραπάνω προγράμματος, με βάση τις ιδιότητες των τελεστών, είναι η ακόλουθη.

```
Query:  ?- flight(?X).
Variables:  ?X

(“fl2”) = T1
(“fl1”) = F1
(“fl3”) = T0
```

Εικόνα 7: Έξοδος προγράμματος PrefLog

3.1.2 Επεξεργασία των κανόνων του προγράμματος

Για να είναι συμβατή η μορφή των προγραμμάτων PrefLog με τις προδιαγραφές της IRIS, προγράμματα με πιο σύνθετους κανόνες θα χρειαστούν κάποια επεξεργασία και μετατροπή. Ο κυριότερος λόγος που είναι επιθυμητή αυτή η συμβατότητα είναι για να μπορέσουν να αξιοποιηθούν αναλλοίωτες οι φάσεις επεξεργασίας της IRIS που δεν επηρεάζονται από την επιπλέον εκφραστικότητα της PrefLog. Εφόσον στην IRIS υπάρχει σύζευξη των λεκτικών στο σώμα ενός κανόνα, τότε θα πρέπει οποιοδήποτε πρόγραμμα PrefLog περιέχει τους νέους τελεστές να μετατραπεί σε πρόγραμμα με κανόνες που να αποτελούνται μόνο από and και ε. Επιπλέον, αυτή η μετατροπή βοηθάει στην απλοποίηση των υπολογισμών, αφού όλα τα προγράμματα έχουν την ίδια μορφή κατά τη διάρκεια της αποτίμησης.

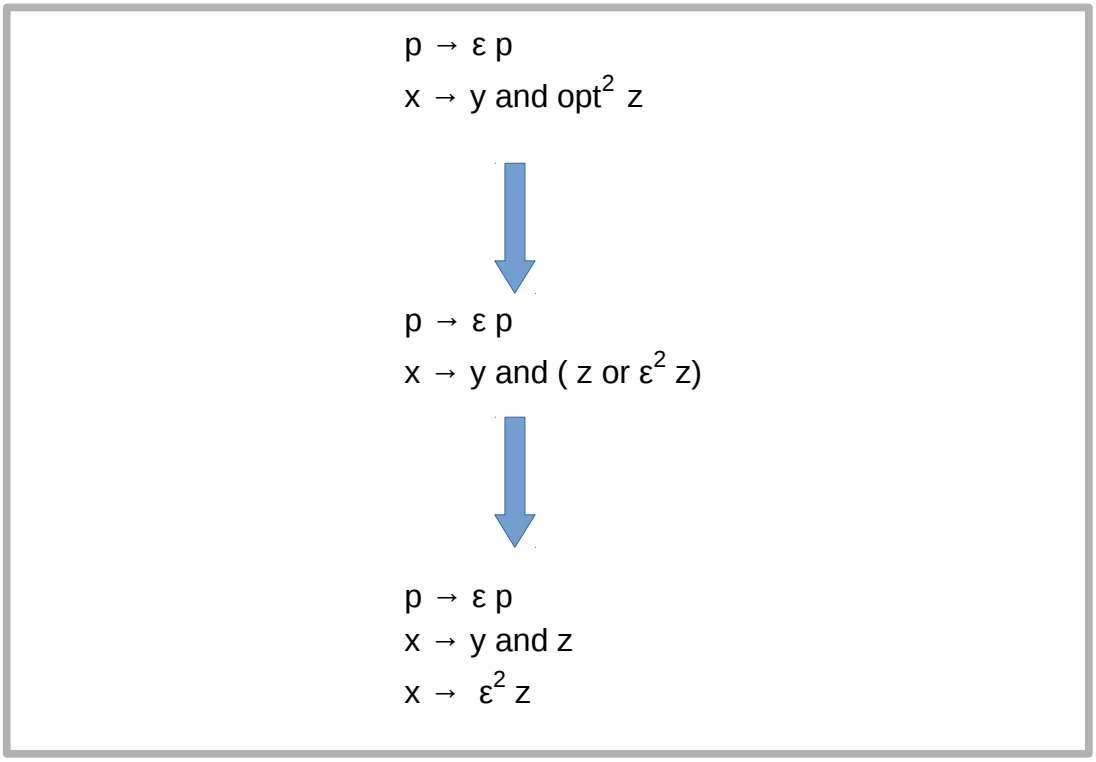
Με τη βοήθεια του τελεστή ε μπορούμε να απλοποιήσουμε τους τελεστές opt^n και alt^n. Η λογική ισοδυναμία που ισχύει για καθένα από αυτούς είναι η εξής,

$$opt^n\ A \equiv (A \text{ or } \varepsilon^n\ A)$$

$$alt^n\ A \equiv (A \text{ and } \varepsilon^n\ A)$$

Εικόνα 8: Λογικές ισοδυναμίες τελεστών opt και alt

Αυτή η μετατροπή γίνεται στον parser της γλώσσας PrefLog με βάση τις λογικές ισοδυναμίες για τους τελεστές opt και alt, που περιγράφηκαν παραπάνω. Επιπλέον, όταν δύο υποστόχοι ενώνονται με τον τελεστή or, τότε κατά τη μετατροπή του κανόνα δημιουργείται νέος κανόνας με το ίδιο όνομα κατηγορήματος και μοιράζονται οι υποστόχοι στους δύο, πλέον, κανόνες.



Εικόνα 9: Μετατροπή κανόνα

Μετά το πέρας της επεξεργασίας, παρατηρούμε ότι έχει προστεθεί ένας επιπλέον κανόνας με σκοπό να εξαλειφθεί ο τελεστής διάζευξης.

3.2 Αλγόριθμοι αποτίμησης κανόνων

Οι αλγόριθμοι αποτίμησης του ελάχιστου μοντέλου Herbrand ενός προγράμματος στηρίζονται στη λογική της από-κάτω-προς-τα-πάνω στρατηγικής. Υλοποιήθηκαν οι αλγόριθμοι για αφελή και ημι-αφελή αποτίμηση. Το γεγονός ότι το πεδίο τιμών αληθείας είναι μη πεπερασμένο καθιστά απαραίτητη τη διαμόρφωση του αλγορίθμου αποτίμησης ενός προγράμματος PrefLog [1], ώστε να εξασφαλίζεται η ολοκλήρωση του μετά από πεπερασμένο αριθμό βημάτων.

```
1:  procedure TERMINATINGEVALUATION(P)
2:      n := 0
3:      S := B_P
4:      repeat
5:          I(A) := F_n, for all A ∈ S
6:          repeat
7:              I' := I
8:              I(A) := max{I(B) : (A ← B) ∈ P}, ∀A ∈ B_P
9:          until I'(A) = I(A), ∀A ∈ B_P s.t. ord(I(A)) = n
10:         S' := {A : A ∈ B_P, ord(I(A)) = n}
11:         if S' = ∅ then
12:             I(A) := 0, for all A ∈ S
13:             S := ∅
14:         else
15:             S := S − S'
16:         end if
17:         n := n + 1
18:     until S = ∅
19:     return I
20: end procedure
```

Εικόνα 10: Αφελής αλγόριθμος αποτίμησης

Στον παραπάνω αλγόριθμο, η διαδικασία ξεκινάει αποδίδοντας σε όλα τα στοιχεία της βάσης Herbrand την τιμή αληθείας F_0. Έπειτα, εφαρμόζεται διαδοχικά ο τελεστής T_P στην ερμηνεία αυτή, μέχρι να παραχθούν και να σταθεροποιηθούν όλα τα άτομα με τιμή αλήθειας T_0 ή F_0. Στην συνέχεια, κρατώντας αυτά τα άτομα αναλλοίωτα, επαναρχικοποιούνται όλα τα υπόλοιπα στοιχεία της βάσης σε F_1 και επαναλαμβάνεται η ίδια διαδικασία μέχρι να έχει αποτιμηθεί πλήρως η βάση Herbrand. Κατ' αυτόν τον τρόπο παράγεται το ελάχιστο μοντέλο M_P ενός προγράμματος. Ο τερματισμός του υπολογισμού του M_P εξασφαλίζεται ελέγχοντας αν στο κ-οστό βήμα δεν παράχθηκε κάποια τιμή με δείκτη κ. Τότε, σε όσα στοιχεία έχουν περισσέψει στη βάση ανατίθεται η ενδιάμεση τιμή 0.
Στην πράξη, για λόγους απόδοσης δε χρειάζεται να υπολογιστεί εκ των προτέρων η βάση Herbrand για να υπολογίσουμε το ελάχιστο μοντέλο, M_P . Με από-κάτω-προς-τα-πάνω στρατηγικές υπολογισμού παράγονται όλα τα γεγονότα με την τιμή αληθείας τους και ότι δεν είναι άμεσο επακόλουθο των κανόνων του προγράμματος δεν αποθηκεύεται στη βάση και υπονοείται ότι αποτιμάται σε F_0.

3.3 Ανίχνευση κενών

Η έννοια του κενού όσον αφορά τις τιμές αληθείας, σχετίζεται με την περίπτωση να χρειάζεται μη πεπερασμένος αριθμός βημάτων κατά την αποτίμηση για να αποδοθεί συγκεκριμένη τιμή αληθείας σε κάποιο άτομο. Στην παρούσα υλοποίηση κενό για κάποιο κατηγόρημα μπορεί να προκύψει αν η τιμή αληθείας των γεγονότων που παράγονται γι' αυτό δε σταθεροποιείται και συνεχώς αυξάνεται. Για παράδειγμα, πρώτα παράγονται κάποια άτομα με σειρά (order) τιμής αληθείας ίση με μηδεν (F_0 ή T_0). Έπειτα, θα παραχθούν κάποια με σειρά ίση με ένα και ενδέχεται ορισμένα από αυτά που παράχθηκαν προηγουμένως να αυξήσουν την τιμή αληθείας τους, ενώ άλλα θα παραμείνουν σταθεροποιημένα. Όταν όλα όσα παράχθηκαν στο προηγούμενο επίπεδο αυξάνουν την τιμή αληθείας τους και δεν παραμένει κανένα σταθερό, για το συγκεκριμένο κατηγόρημα, τότε θεωρούμε ότι έχει ανιχνευθεί κενό για το κατηγόρημα αυτό.

Κατά την εκτύπωση της εξόδου ενός προγράμματος, στην περίπτωση της τιμής αληθείας μηδέν, εμφανίζεται μόνο το όνομα του κατηγορήματος, χωρίς καμία πλειάδα, ίσο με μηδέν. Πιο αναλυτικά, για το πρόγραμμα

```
?- p(?x).

p(?x):- ε p(?x).

s('a').
s('b').
s('c').
s('d').
```

η έξοδος εκτυπώνεται ως [p = 0], αντί για [p('a') = 0, p('b') = 0, p('c') = 0, p('d') = 0].
Η μορφή αυτή υπονοεί ότι όλα τα άτομα του κατηγορήματος p αποτιμώνται σε μηδέν.

Αυτό συμβαίνει διότι, όπως προαναφέρθηκε, δεν υπολογίζεται ολόκληρη η βάση Herbrand εκ των προτέρων. Επομένως, δεν είναι εφικτό, όταν ανιχνευθεί κενό για κάποιο κατηγόρημα, να παραχθεί και να εκτυπωθεί κάθε στοιχείο που θα άνηκε στη βάση και θα έπρεπε να αποτιμηθεί σε μηδέν. Επιπλέον, για μια βάση Herbrand με πολύ μεγάλο αριθμό ατόμων ενδέχεται το αποτέλεσμα της εκτυπώσης να είναι εξίσου μεγάλο, γεγονός που το καθιστά πιο δυσανάγνωστο από τη μορφή που προτείνεται παραπάνω.

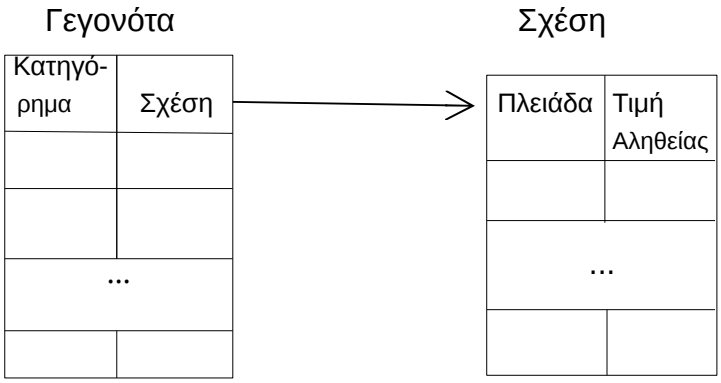
4. ΥΛΟΠΟΙΗΣΗ ΤΗΣ PrefLog ΣΤΟ ΣΥΣΤΗΜΑ IRIS

4.1 Αναπαράσταση τιμών αληθείας

Η βασικότερη διαφορά μεταξύ της γλωσσών IRIS και PrefLog είναι το μη πεπερασμένο πεδίο τιμών αληθείας που παρέχει η δεύτερη. Κατά τη διάρκεια υπολογισμού του ελάχιστου μοντέλου ενός προγράμματος, κάθε γεγονός που παραγεται χαρακτηρίζεται από μια τιμή αληθείας, η οποία ενδέχεται να μεταβληθεί μέχρι να τελειώσει ο υπολογισμός. Γι' αυτόν το λόγο, στην παρούσα υλοποίηση της γλώσσας PrefLog έχει τροποποιηθεί ο τρόπος που αναπαρίστανται οι σχέσεις (Relations).
Η μορφή με την οποία αποθηκεύονται οι τιμές αληθείας είναι ως SimpleEntry, δηλαδή ως την απλούστερη μορφή Map της Java, όπου αποτελείται μόνο από ένα ζεύγος κλειδιού - τιμής (key - value). Για κλειδί εισάγεται ένα γράμμα, που αναλόγως την περίπτωση, περιγράφει αν πρόκειται για αληθές(“T”) ή ψευδές(“F”) γεγονός ή μηδέν(“Z”). Η αντίστοιχη τιμή για κάθε κλειδί είναι η σειρά της τιμής αληθείας που έχει παραχθεί για το εκάστοτε γεγονός, δηλαδή για τιμή αληθείας ίση με T0, τότε το κλειδί είναι “T” και η τιμή του είναι 0, δηλαδή, ένα ζεύγος της μορφής <”T”, 0>. Για λόγους απόδοσης, εφόσον δεν παράγεται η βάση Herbrand για την αποτίμηση, τότε δεν αποθηκεύονται τα άτομα με τιμή αληθείας F_0.

4.2 Αναπαράσταση σχέσεων

Μια σχέση (Relation) αντιπροσωπεύει την αντιστοίχιση ενός πλήθους από πλειάδες (tuples) με τις τιμές αληθείας τους.
Αρχικά, στο σύστημα IRIS οι σχέσεις αναπαριστώνται ως μια λίστα από πλειάδες, που δεν επιτρέπει την ύπαρξη διπλότυπων.
Στην PrefLog όμως κάθε πλειάδα χαρακτηρίζεται από μια τιμή αληθείας, επομένως η αντίστοιχη αναπαράσταση που υλοποιήθηκε είναι ένα HashMap, με κλειδιά τις διάφορες πλειάδες και τιμές τις τιμές αληθείας που τους αντιστοιχούν. Τα γεγονότα εκφράζονται , τόσο στην IRIS όσο και στην PrefLog, ως ένα HashMap με κλειδιά τα ονόματα των διάφορων κατηγορημάτων και τιμές τις σχέσεις που έχουν παραχθεί γι' αυτά.

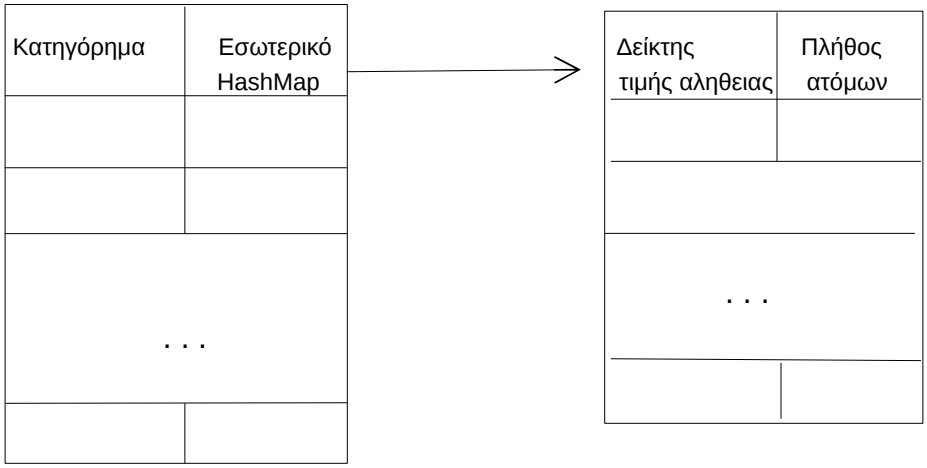


Σχήμα 1: Αναπαράσταση δομής για τις τιμές αληθείας

Ο λόγος που αποφασίστηκε η συγκεκριμένη προσέγγιση είναι ότι αυτές οι δομές εξασφαλίζουν καλύτερη απόδοση κατά την εκτέλεση του αλγορίθμου αποτίμησης. Σε πολλά σημεία απαιτείται η ανάγνωση όλων των καταχωρήσεων μιας σχέσης, επομένως η πολυπλκότητα είναι O(n+s), όπου n ο αριθμός των καταχωρήσεων και s το μέγεθος του HashMap. Το πιο κρίσιμο σημείο όμως είναι όταν, κατά την αποτίμηση, με οποιονδήποτε από τους δύο αλγορίθμους που περιγράφηκαν παραπάνω, παράγεται κάποια νέα σχέση. Τότε, θα πρέπει να συγκριθούν όλες τις πλειάδες που περιέχει η νέα σχέση με αυτές που έχουν ήδη παραχθεί, ώστε να διαπιστωθεί αν υπάρχουν ήδη με ίδια ή διαφορετική τιμή αληθείας ή αν παράγονται για πρώτη φορά. Στην περίπτωση που κάποια πλειάδα υπάρχει ήδη με μικρότερη τιμή αληθείας, τότε πρέπει να αντικατασταθεί από την καινούρια. Η βέλτιστη και η μέση πολυπλοκότητα για να εκτελεστεί μια εισαγωγή, αναζήτηση ή διαγραφή σε ένα HashMap είναι O(1), ενώ η χείριστη είναι O(n). Σκοπός είναι η συνολική πολυπλοκότητα για το συγκεκριμένο έλεγχο να εξαρτάται κυρίως από το διάβασμα ολόκληρης της σχέσης που παράχθηκε.

Μια επιπλέον δομή που χρησιμοποιήθηκε είναι για να κρατάμε την απαραίτητη πληροφορία που χρειαζόμαστε για τον έλεγχο των κενών κατά τη διάρκεια του αλγορίθμου. Όπως αναφέρθηκε στην ενότητα 3.3 σχετικά με την ανίχνευση κενών, πρέπει να γνωρίζουμε αν τα γεγονότα που παράγονται σταθεροποιούνται όσον αφορά την τιμή αληθείας τους. Στην περίπτωση που δε συμβαίνει αυτό, είναι απαραίτητο να γνωρίζουμε σε ποιό κατηγόρημα θα πρέπει να ανατεθεί η ενδιάμεση τιμή αληθείας μηδέν. Για αυτούς τους λόγους επιλέχθηκε να αποθηκεύονται σε ένα HashMap οι αναγακίες πληροφορίες. Το κλειδί της δομής είναι το όνομα του κατηγορήματος και ως τιμή έχει εσωτερικό HashMap όπου αναπαριστά το πλήθος των πλειάδων που έχουν τιμή αληθείας για κάθε σειρά τιμών αληθείας από 0 μέχρι το σημείο που έχει προχωρήσει η αποτίμηση. Ο ρόλος του εσωτερικού HashMap είναι για να μπορούμε να ελέγχουμε αν υπάρχει κάποιος κανόνας του οποίου τα γεγονότα να αυξάνουν όλα διαρκώς την τιμή αληθείας τους, χωρίς να σταθεροποιείται ποτέ κάποιο σε ένα συγκεκριμένο επίπεδο. Κρατώντας, λοιπόν, ως πληροφορία πόσα γεγονότα παράχθηκαν για κάθε σειρά (order) τιμών αληθείας, γνωρίζουμε ακριβώς πόσα είναι σταθερά και πόσα εξακολουθούν να μεταβάλλονται όσο προχωράει η αποτίμηση.

Κάθε φορά που προστίθεται μια πλειάδα ενός κατηγορήματος αυξάνεται κατά ένα το πλήθος των ατόμων με τιμή αληθείας σε αυτή τη σειρά. Αν σε κάποια πλειάδα αντικατασταθεί η τιμή αληθείας με κάποια μεγαλύτερη, τότε θα πρέπει να μειωθεί το πλήθος των ατόμων για τη σειρά της παλιάς τιμής και να αυξηθεί για τη νέα. Αν παρατηρηθεί ότι το πλήθος των ατόμων έχει γίνει μηδέν, τότε θεωρούμε ότι εχει δημιουργηθεί κενό, διότι τα άτομα που θα έπρεπε να είχαν σταθεροποιηθεί στο προηγούμενο επίπεδο, εξακολουθούν να αυξάνουν την τιμή αληθείας τους. Στην πλειάδα που παρουσιάστηκε το κενό ανατίθεται η τιμή αληθείας <”Z”, 0> και ολοκληρώνεται η αποτίμηση.



Σχήμα 2: Αναπαράσταση δομής για το κενό

4.3 Σημασιολογία τελεστών της PrefLog και λεπτομέρειες υλοποίησης

Σε αυτή την ενότητα παρουσιάζεται πιο αναλυτικά η σημασιολογία των νέων τελεστών, καθώς και τι αλλαγές ήταν απαραίτητο να γίνουν για να ορισθούν, εφόσον η υλοποίηση της IRIS δεν τους υποστήριζει.

4.3.1 Τελεστές σύζευξης και διάζευξης

Οι *and* και *or* χρησιμοποιούνται για να δηλώσουν σύζευξη ή διάζευξη μεταξύ των υποστόχων ενός κανόνα, αντίστοιχα. Έχουν διαφορετική σημασιολογία, εφόσον ο and επιλέγει τη μικρότερη τιμή αληθείας μεταξύ των υποστόχων, ενώ αντιθέτως ο or επιλέγει τη μεγαλύτερη. Επομένως, επειδή αυτή η διαφορά δεν μπορεί να αναπαρασταθεί με το απλό κόμμα (“,”) που χωρίζει τους υποστόχους ενός σύνθετου κανόνα στην IRIS, δε χρησιμοποιείται πια στην PrefLog και αντικαθίσταται από τους and και or.

Η επιλογή της μικρότερης τιμής αληθείας γίνεται στα επιμέρους στοιχεία που απαρτίζουν τον μεταγλωττισμένο κανόνα που παράγεται από τον αρχικό κανόνα εισόδου μετά τη φάση της μεταγλώττισης, όπως περιγράφηκε παραπάνω.

Για την περίπτωση του or δε χρειάζεται να ορίσουμε κάτι επιπλέον, αφού με την επεξεργασία και μετατροπή των κανόνων του προγράμματος, δεν υφίσταται πια ως αυτούσιος τελεστής μετά τη διαδικασία του parsing. Παρόλα αυτά, έχει τροποποιηθεί η

φάση της αποτίμησης. Κάθε φορά που παράγονται γεγονότα για τον ίδιο κανόνα, τότε επικρατεί η μεγαλύτερη τιμή αληθείας σε όσα έχουν ήδη υπολογιστεί ξανα, κι έτσι εκφράζεται στην πράξη η ιδιότητα του τελεστή διάζευξης.

```
mother(?x, ?y) :- parent(?x, ?y) and female(?x).
```

Εικόνα 11: Αναπαράσταση τελεστή and

4.3.2 Τελεστές opt και alt

Οι τελεστές opt^n και alt^n χρησιμοποιούνται για να εκφράσουν προτίμηση. Ο opt αναπαριστά την έννοια του “προαιρετικά” (optionally), ενώ ο alt εκφράζει το “εναλλακτικά” (alternatively). Η νιοστή δύναμη υπάρχει διότι κάθε τελεστής μπορεί να εφαρμοστεί πολλαπλές φορές σε έναν στόχο, μεταβάλλοντας αναλόγως και την τελική τιμή αλήθειας.

$$||\ opt^n\ ||(u) = \begin{cases} F_{k+n}, & \text{αν } u = F_k \\ 0, & \text{αν } u = 0 \\ T_k, & \text{αν } u = T_k \end{cases}$$

$$||\ alt^n\ ||(u) = \begin{cases} F_k, & \text{αν } u = F_k \\ 0, & \text{αν } u = 0 \\ T_{k+n}, & \text{αν } u = T_k \end{cases}$$

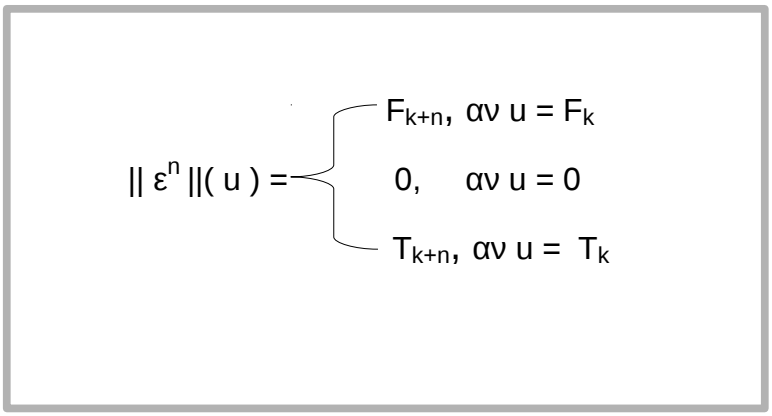
Εικόνα 12: Ορισμοί τελεστών opt και alt

Όπως φαίνεται κι από τους ορισμούς της Εικόνας 9, ο τελεστής opt επιστρέφει την ίδια τιμή αληθείας που επιστρέφει το u, αν αυτό είναι σωστό, αλλιώς επιστρέφει κάτι που είναι λιγότερο λάθος, αν το u είναι λάθος. Αυτό συμβαίνει γιατί η απουσία μιας προαιρετικής προϋπόθεσης δεν είναι τόσο σημαντική όσο η απουσία μιας υποχρεωτικής προϋπόθεσης. Κατά αντίστοιχο τρόπο, ερμηνεύεται και ο τελεστής alt. Η ύπαρξη μιας εναλλακτικής επιλογής δεν είναι τόσο ωφέλιμη όσο είναι η ύπαρξη μιας πρωταρχικής.

Με ανάλογο τρόπο, όπως και για τον or, αυτοί οι τελεστές δεν υφίστανται αυτούσιοι κατά τις φάσεις επεξεργασίας του προγράμματος, γιατί έχουν μετατραπεί οι κανόνες ώστε να αποτελούνται μόνο από ε και and τελεστές.

4.3.3 Τελεστής ε

Ο ε^n είναι ένας μοναδιαίος τελεστής που εκφράζει προτιμήσεις. Επίσης, είναι βασικό δομικό στοιχείο των υπόλοιπων τελεστών.



Εικόνα 13: Ορισμός τελεστή ε

Παρατηρούμε από τον ορισμό που δίνεται στην παραπάνω εικόνα ότι ο εν λόγω τελεστής αυξάνει την τιμή αληθείας του u κατά τον εκθέτη n, ανεξάρτητα από το αν το u είναι αληθές ή ψευδές. Για n = 0, έχουμε ότι $|| \varepsilon^0 ||(u) = u$.

Ο τελεστής ε αναπαρίσταται ως χαρακτηριστικό του εκάστοτε λεκτικού, μέσω ενός ακέραιου πεδίου που υποδηλώνει την ύπαρξη ή όχι του τελεστή, ανάλογα αν έχει τιμή μηδέν ή είναι θετικός ακέραιος, αντίστοιχα. Το πεδίο αυτό ουσιαστικά αποθηκεύει την τιμή του εκθέτη του τελεστή. Κατά τη φάση της μεταγλώττισης αυτή η πληροφορία μεταφέρεται με τον ίδιο τρόπο στο μεταγλωττισμένο κανόνα, γιατί παύουν να υπάρχουν τα λεκτικά στην αρχική τους μορφή, μετά το πέρας της. Στη συνέχεια, στην αποτίμηση, όποτε συναντάται πληροφορία που σχετίζεται με τον τελεστή ε, τότε αυξάνεται η τιμή αληθείας της αποτιμώμενης πλειάδας κατά τον εκθέτη του ε.

5. ΜΕΤΡΗΣΕΙΣ ΚΑΙ ΣΥΓΚΡΙΣΕΙΣ

Σε αυτή την ενότητα παρουσιάζονται συγκρίσεις μεταξύ των δύο αλγορίθμων αποτίμησης που αναλύθηκαν, για διάφορα προβλήματα εισόδου. Αρχικά, παρουσιάζονται τα προγράμματα εισόδου με βάση τα οποία γίνονται όλες οι μετρήσεις κι έπειτα οι διάφορες συγκρίσεις.

5.1 Προγράμματα εισόδου

Το ακόλουθο πρόγραμμα εισόδου επιλέγει την κατάλληλη πτήση σύμφωνα με τους περιορισμούς και τις προτιμήσεις που έχει θέσει ο χρήστης.

```
?- flight(?X).

flight(?f):- to("athens","boston",?f) and
has_stopover(?f) and
opt carrier(?f,"aegean").
has_stopover(?f) :- stopover(?f,"rome") or alt stopover(?f, "london").

to("athens","boston","fl1").
to("athens","boston","fl2").
to("athens","boston","fl3").
stopover("fl1","rome").
stopover("fl2","london").
stopover("fl3","rome").
carrier("fl1","british").
carrier("fl2","aegean").
carrier("fl3","aegean").
```

Εικόνα 14: Πρόγραμμα εισόδου flight

Το πρόγραμμα εισόδου που ακολουθεί επιλέγει τη βέλτιστη διαδρομή ανάμεσα σε δύο σημεία. Ανάλογα με τους διαθέσιμους δρόμους, πρώτη επιλογή είναι ένας δρόμος διπλής κυκλοφορίας, μετά ένας μονής και τέλος ο χωματόδρομος.

```
?- ppath(?X, ?Y).

ppath(?X,?Y) :- p(?X,?Y).
ppath(?X,?Y) :- p(?X,?Z) and ppath(?Z,?Y).
p(?X,?Y) :- e(?X,?Y, "two lane") or alt e(?X,?Y, "one lane") or alt alt e(?X,?Y,"dirt").

e('a','b',"two lane").
e('a','c',"one lane").
e('b','c',"two lane").
e('c','d',"one lane").
e('a','d',"dirt").
e('c','e',"dirt").
```

Εικόνα 15: Πρόγραμμα εισόδου ppath

Το παρακάτω πρόγραμμα παρουσιάζει μια ρεαλιστική περίπτωση όπου ένα αντικείμενο (ή οτιδήποτε άλλο) αρέσει σε κάποιον είτε λόγω προσωπικών προτιμήσεων είτε επειδή αρέσει σε κάποιον φίλο του, ακόμα κι αν δεν είναι η δική του πρώτη επιλογή.

```
?- likes("john",?X).

likes("john",?X) :- goodquality(?X) and opt likes("paul",?X).
likes("paul",?X) :- goodquality(?X) and opt likes("john",?X).

goodquality("object").
```

Εικόνα 16: Πρόγραμμα εισόδου likes

Τέλος, παρουσίαζεται ένα παράδειγμα προγράμματος που παρόλο που είναι πολύ απλό, χρειάζεται άπειρο πλήθος βημάτων κατά την αποτίμησή του. Ατυό συμβαίνει λόγω του τελεστή ε στον πρώτο κανόνα. Το πρόγραμμα εμφανίζει κενό και γι' αυτό το λόγο αποτιμάται στην ενδιάμεση τιμή αληθείας 0.

```
?- p(?X).

p(?x):- ε p(?x).
q(?x):- r(?x).
r(?x):- s(?x).

s('a').
```

Εικόνα 17: Πρόγραμμα εισόδου p

5.2 Συγκρίσεις χρόνων εκτέλεσης των προγραμμάτων εισόδου

Τα αποτελέσματα των παραπάνω προγραμμάτων για καθένα από τους δύο αλγορίθμους αποτίμησης φαίνονται στον ακόλουθο πίνακα.

Πρόγραμμα Εισόδου	**Αφελής Αποτίμηση**	**Ημι-αφελής αποτίμηση**
flight	35ms	28ms
ppath	39ms	34ms
likes	27ms	27ms
p	19ms	20ms

Πίνακας 1: Συγκρίσεις χρόνων εκτέλεσης

Εφόσον τα συγκεκριμένα προβλήματα δεν είναι ιδιαίτερα μεγάλα σε έκταση παρουσιάζουν παρόμοιους χρόνους εκτέλεσης. Παρόλα αυτά όμως, είναι εμφανές ότι η ημι-αφελής αποτίμηση παρουσιάζει καλύτερη απόδοση, αφού στα δύο πρώτα προγράμματα εισόδου έχει καλύτερους χρόνους εκτέλεσης. Σε εκείνα τα προγράμματα απαιτούνται περισσότεροι υπολογισμοί απ’ότι στα δύο τελευταία, που λόγω των κενών τερματίζει η αποτίμησή τους γρηγορότερα.

6. ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Η υλοποίηση που παρουσιάστηκε παρέχει τη δυνατότητα αναπαράστασης προτιμήσεων, καθώς επίσης και την αποτίμηση τόσο με τον αφελή αλγόριθμο, όσο και με τον ημι-αφελή, που βελτιώνει τους χρόνους εκτέλεσης και την απόδοση. Παρόλα αυτά, είναι επιθυμητό στο μέλλον να επεκταθεί η υλοποίηση ώστε να υποστηρίζει συναρτησιακά σύμβολα, καθώς λογικοί κανόνες με συναρτησιακά σύμβολα είναι εξίσου ισχυροί με μια μηχανή Turing.

Ένας ακόμα μελλοντικός στόχος είναι να υποστηρίζονται εκφράσεις μέσα σε παρενθέσεις στο σώμα των κανόνων. Αυτό θα ενισχύσει ακόμα περισσότερο την εκφραστικότητα της γλώσσας, εφόσον θα επιτρέπεται να γραφτούν πιο σύνθετα προγράματα και θα μπορούν να εκφραστούν πιο εξειδικευμένες προτιμήσεις.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός όρος**
First order logic	Λογική πρώτης τάξης
Herbrand Universe	Herbrand Σύμπαν
Herbrand Base	Herbrand Βάση
Herbrand Interpretation	Herbrand Ερμηνεία
Least Herbrand Model	Ελάχιστο μοντέλο Herbrand
Fixpoint Semantics	Σημασιολογία σταθερού σημείου
Top-down evaluation strategy	Από-πάνω-προς-τα-κάτω στρατηγική αποτίμησης
Bottom-up evaluation strategy	Από-κάτω-προς-τα-πάνω στρτηγική αποτίμησης
Naive evaluation	Αφελής αποτίμηση
Semi-naive evaluation	Ημι-αφελής αποτίμηση
Tuple	Πλειάδα
Relation	Σχέση
Preferential operators	Τελεστές προτίμησης
Stratification	Στρωματοποίηση
Program doubling	Διπλασιασμός προγράμματος
Program optimization	Μεταγλώττιση προγράμματος
Rule safety processing	Επεξεργασία ασφαλών κανόνων
Rule reordering	Επανατοποθέτηση κανόνων
Rule compilation	Εκτιμητής
Naive evaluator	Αφελής εκτιμητής
Semi-naive evaluator	Ημι-αφελής εκτιμητής
Order	Σειρά

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

IRIS Reasoner	Integrated Rule Inference System
EDB	Extensional Database
IDB	Intensional Database

ΑΝΑΦΟΡΕΣ

[1] P.Rondogiannis and A.Troumpoukis, “Expressing Preferences in Logic Programming using an Infinite-Valued Logic”.

[2] Π.Σταματόπουλος, Ι.Καράλη, “Σημειώσεις Λογικού Προγραμματισμού” Available: http://cgi.di.uoa.gr/~takis/Logic_Programming_Slides.pdf

[3] Barry Bishop and Florian Fischer, “IRIS - Integrated Rule Inference System ”

[4] “Computational Logic Logic Programming: Model and Fixpoint Semantics”, Available: http://cliplab.org/~logalg/slides/PS/6_semantics.pdf