ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ
ΣΠΟΥΔΩΝ ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Σύστημα Αναζήτησης Βιβλιογραφικών Μεταδεδομένων στο Σημασιολογικό Διαδίκτυο

Διπλωματική εργασία

της

Κομψαρά Σοφίας (Α.Μ. 05/30)

Επιβλέπων καθηγητής: Βασιλειάδης Νικόλαος
Εξεταστής: Βλαχάβας Ιωάννης

Θεσσαλονίκη 2007

Πρόλογος

Το αντικείμενο της παρούσας πτυχιακής εργασίας είναι η μελέτη των στοιχείων που συνθέτουν το Σημασιολογικό Διαδίκτυο και η ανάπτυξη μίας εφαρμογής ερωταπαντήσεων σε μεταδεδομένα του Σημασιολογικού Διαδικτύου στη γλώσσα Prolog που αφορούν δημοσιεύσεις επιστημονικών εργασιών της ομάδας Λογικού Προγραμματισμού και Ευφυών Συστημάτων LPIS του Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης.

Η ανάπτυξη του παρόντος συστήματος ανέδειξε τα ποικίλα πλεονεκτήματα και τις παρεχόμενες δυνατότητες της Prolog ως γλώσσας ανάπτυξης συστημάτων για το Σημασιολογικό Διαδίκτυο και παρουσίασε τόσο ερευνητικό – αναφορικά με τις δυνατές επεκτάσεις της - όσο και πρακτικό ενδιαφέρον.

Θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντα της πτυχιακής εργασίας, κ. Βασιλειάδη Νικόλαο, Επίκουρο Καθηγητή του τμήματος Πληροφορικής του Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης, για την άψογη συνεργασία μας, καθώς και για την παροχή υποστηρικτικού υλικού. Επίσης, θα ήθελα να ευχαριστήσω και τον κ. Jan Wielemaker, καθηγητή του τμήματος Human Computer Studies (HCS) του Πανεπιστημίου του Άμστερνταμ και βασικό συγγραφέα της SWI-Prolog, για την πολύτιμη βοήθειά του σε ένα κρίσιμο σημείο κατά την διάρκεια ανάπτυξης του συστήματος.

Κομψαρά Σοφία
Θεσσαλονίκη, Σεπτέμβριος 2007

Περιεχόμενα

1. Εισαγωγή

Στόχος της παρούσας διπλωματικής εργασίας είναι η μελέτη των στοιχείων που συνθέτουν το Σημασιολογικό Διαδίκτυο και στην συνέχεια η ανάπτυξη ενός συστήματος ερωταπαντήσεων σε μεταδεδομένα του Σημασιολογικού Διαδικτύου τα οποία βρίσκονται σε μορφή RDF. Πιο συγκεκριμένα, τα μεταδεδομένα αυτά αφορούν βιβλιογραφικά δεδομένα επιστημονικών εργασιών και δημοσιεύσεων της ερευνητικής ομάδας Λογικού Προγραμματισμού και Ευφυών Συστημάτων LPIS και εξάγονται δυναμικά στο διαδίκτυο από μία βάση δεδομένων.

Το Σημασιολογικό Διαδίκτυο αποτελεί μία επέκταση και βελτίωση του υπάρχοντος διαδικτύου στην οποία η δόμηση και οργάνωση των πληροφοριών να τις καθιστά ευκολότερα επεξεργάσιμες από τους υπολογιστές με αποτέλεσμα να αυξάνεται η απόδοση και να καθίσταται αποτελεσματικότερη η επικοινωνία ανθρώπων και υπολογιστών.

Η γλώσσα RDF είναι μία γλώσσα γενικού σκοπού η οποία αναπτύχθηκε για την αναπαράσταση πληροφοριών σχετικών με πόρους του Παγκόσμιου Ιστού.

Το σύστημα που αναπτύχθηκε δέχεται από τον χρήστη κάποια κριτήρια σχετικά με την αναζήτηση επιστημονικών εργασιών μέσω μίας διαδικτυακής φόρμας, στην συνέχεια επεξεργάζεται τα RDF δεδομένα βάσει αυτών των κριτηρίων και επιστρέφει τα αποτελέσματα της αναζήτησης στον χρήστη με την μορφή HTML σελίδας.

Η υλοποίηση του συστήματος πραγματοποιήθηκε στην SWI-Prolog με χρήση αφ' ενός των βιβλιοθηκών της για την ανάλυση και επεξεργασία RDF δεδομένων και αφ' ετέρου των βιβλιοθηκών της για την δημιουργία και επεξεργασία διαδικτυακών φορμών και ιστοσελίδων.
Η εργασία αποτελείται από 4 κεφάλαια και 1 παράρτημα τα οποία πραγματεύονται τα θέματα που αναφέρθηκαν παραπάνω.

Στο δεύτερο κεφάλαιο γίνεται παρουσίαση του Σημασιολογικού Διαδικτύου και των βασικών χαρακτηριστικών του. Επιπλέον, παρουσιάζονται οι γλώσσες RDF και RDF Schema οι οποίες έχουν αναπτυχθεί για την αναπαράσταση δεδομένων στο Σημασιολογικό Διαδίκτυο καθώς και η RDF/XML αναπαράσταση. Επίσης γίνεται μία εισαγωγή στις οντολογίες, παρουσιάζονται κάποιες περιπτώσεις χρήσης τους καθώς και η Γλώσσα Οντολογιών Διαδικτύου (OWL). Τέλος, παρουσιάζεται σύντομα η έννοια του λογικού προγραμματισμού και γίνεται μία εισαγωγή στον μηχανισμό εκτέλεσης της γλώσσας λογικού προγραμματισμού Prolog, η οποία χρησιμοποιείται και για την ανάπτυξη της εφαρμογής.

Στο τρίτο κεφάλαιο πραγματοποιείται η ανάλυση και ο σχεδιασμός του συστήματος. Αρχικά, παρουσιάζεται η δομή και το περιεχόμενο του RDF εγγράφου και στην συνέχεια περιγράφεται το RDF Schema το οποίο καθορίζει την οργάνωση του εγγράφου. Επιπλέον, πραγματοποιείται και μία ανάλυση των απαιτήσεων του συστήματος που αναπτύχθηκε και η οποία αποτέλεσε οδηγό κατά την διάρκεια της φάσης του σχεδιασμού και της υλοποίησης.

Επιπλέον, παρουσιάζονται οι βιβλιοθήκες Semantic Web και HTTP της SWI-Prolog που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος. Στην συνέχεια ακολουθεί μία αφαιρετική περιγραφή της αρχιτεκτονικής του συστήματος η οποία, τέλος, ακολουθείται από μία αναλυτική παρουσίαση και περιγραφή της υλοποίησης και λειτουργίας των βασικών τμημάτων που απαρτίζουν το σύστημα.

Το τέταρτο κεφάλαιο αποτελείται από μία σύνοψη της διαδικασίας ανάπτυξης του συστήματος, κάποια συμπεράσματα και παρατηρήσεις που έγιναν καθώς και προτάσεις για μελλοντική επέκταση της παρούσας εργασίας.

Το Παράρτημα Α περιλαμβάνει το RDF έγγραφο που περιέχει τα βιβλιογραφικά δεδομένα πάνω στα οποία εκτελούνται οι λειτουργίες αναζήτησης από το σύστημα καθώς επίσης και το RDF Schema, το οποίο χρησιμοποιήθηκε κατά την εξαγωγή των δεδομένων αυτών.

Το Παράρτημα Β περιλαμβάνει: (α) το αρχείο rules.pl με τους κανόνες που υλοποιούν τους διάφορους τύπους αναζήτησης, (β) το αρχείο http_user.pl το οποίο υλοποιεί την html φόρμα εισαγωγής των κριτηρίων αναζήτησης και (γ) το αρχείο http_data.pl το οποίο χειρίζεται τα δεδομένα εισόδου - κριτήρια αναζήτησης - πραγματοποιεί την αντίστοιχη αναζήτηση και επιστρέφει τα αποτελέσματα σε μορφή html σελίδας στον χρήστη.

2. Σημασιολογικό Διαδίκτυο

2.1 Εισαγωγή

Το Διαδίκτυο και ο παγκόσμιος ιστός είναι μια από τις σημαντικότερες εξελίξεις της εποχής μας. Άλλαξε και συνεχίζει να αλλάζει την ζωή μας με γρήγορους ρυθμούς και μας προσφέρει πρωτόγνωρες δυνατότητες χωρίς καν να χρησιμοποιούμε την τεράστια υπολογιστική δύναμη των υπολογιστών παρά μόνο για την αποθήκευση και την εμφάνιση των σελίδων. Το Σημασιολογικό Διαδίκτυο είναι το επόμενο βήμα.

Ο όρος Semantic Web καθώς και η αρχιτεκτονική για την υλοποίησή του προτάθηκαν από τον Tim Berners-Lee, τον εφευρέτη του σημερινού Παγκόσμιου Ιστού. Το Semantic Web υιοθετήθηκε από το World Wide Web Consortium (W3C) ο οποίος είναι ένας οργανισμός που στοχεύει στην ανάπτυξη και εξέλιξη του Διαδικτύου και των πρωτοκόλλων που το υποστηρίζουν.

Το Σημασιολογικό Διαδίκτυο – σημασιολογικός ή νοηματικός ιστός - είναι ένα όραμα για την εξέλιξη του Διαδικτύου έτσι ώστε οι πληροφορίες που υπάρχουν και διακινούνται σε αυτό να είναι επεξεργάσιμες από τους υπολογιστές. Στην σημερινή μορφή του Διαδικτύου οι πληροφορίες προορίζονται αποκλειστικά για ανθρώπινη ‘κατανάλωση’ ενώ οι υπολογιστές χρησιμοποιούνται απλά για την αποθήκευση, μετάδοση και εμφάνισή τους. Η μόνη ευρέως διαδεδομένη χρήση των υπολογιστών για επεξεργασία των πληροφοριών στο Διαδίκτυο είναι οι μηχανές αναζήτησης των οποίων η απόδοση είναι πολύ χαμηλή, όπως είναι πλέον κοινά αποδεκτό. Μοναδικός λοιπόν αποδέκτης των πληροφοριών αυτών στο Διαδίκτυο είναι ο τελικός χρήστης που τις βλέπει στην οθόνη του υπολογιστή του. Η προοπτική αυτή της επεξεργασίας του τεράστιου όγκου πληροφοριών του Διαδικτύου από τους υπολογιστές ανοίγει τον δρόμο για νέες επαναστατικές εφαρμογές.

Το Σημασιολογικό Διαδίκτυο είναι ουσιαστικά μία βελτίωση του παραδοσιακού Διαδικτύου κυρίως αναφορικά με την δόμηση και αναπαράσταση της πληροφορίας σε μορφή τέτοια ώστε να είναι κατανοητή και επεξεργάσιμη από τους υπολογιστές. Η σημερινή αναπαράσταση που προορίζεται για χρήση από τους ανθρώπους θα αντικατασταθεί από αναπαράσταση κατανοητή στους υπολογιστές. Από αυτή την άποψη λοιπόν, το Σημασιολογικό Διαδίκτυο είναι ένα πρόβλημα αναπαράστασης της γνώσης (knowledge representation) από και για τους υπολογιστές.

Πιο συγκεκριμένα, μέχρι σήμερα η πληροφορία στο Διαδίκτυο είναι μορφοποιημένη έτσι ώστε να είναι κατανοητή και επεξεργάσιμη από τον άνθρωπο και η κυρίαρχη γλώσσα που χρησιμοποιείται για την κατασκευή ιστοσελίδων είναι η HTML. Για παράδειγμα, μία ιστοσελίδα ενός φυσιοθεραπευτικού κέντρου θα έμοιαζε ως εξής:

```
<h1>Πρότυπο θεραπευτήριο Physiopraxis</h1>
Καλώς ήρθατε στην ιστοσελίδα του θεραπευτηρίου μας. Αισθάνεστε πόνο ή είχατε κάποιο τραυματισμό; Αφήστε το προσωπικό μας Παπαγεωργίου Κωνσταντίνο, Νικολάου Μαρία (η συμπαθέσταστη γραμματέας μας) και Ανδρεάδη Δημήτριο να φροντίσουν τόσο το σώμα όσο και την ψυχική σας διάθεση.
<h2>Ωρες λειτουργίας</h2>
Δευτέρα 11π.μ. – 7 μ.μ. <br>
Τρίτη 11 π.μ. – 7 μ.μ. <br>
Τετάρτη 3 π.μ. – 7 μ.μ.<br>
Πέμπτη  3 π.μ. – 7 μ.μ.<br>
Παρασκευή 11 π.μ. – 7 μ.μ.
Το κέντρο παραμένει κλειστό κατά την διάρκεια των
<\a href = "…."> επίσημων αργιών και γιορτών </a>
```

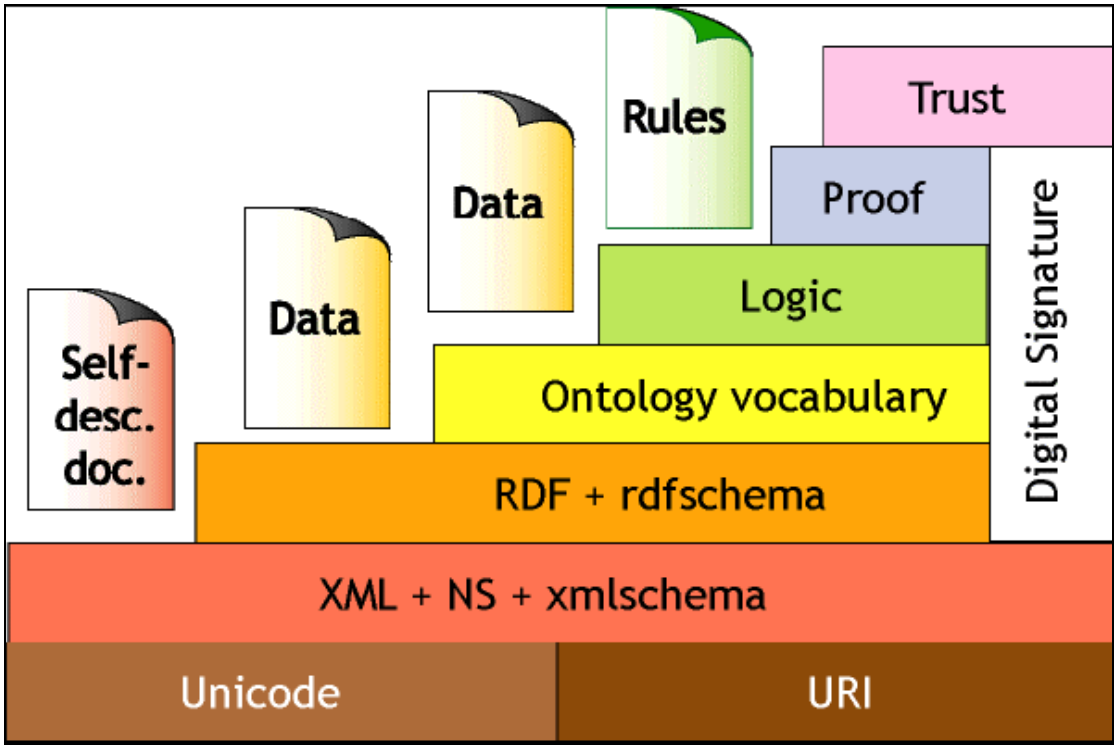
Η μορφοποίηση αυτή της πληροφορίας είναι ικανοποιητικά μορφοποιημένη για τον άνθρωπο αλλά όχι για τις μηχανές. Για παράδειγμα, η αναζήτηση με λέξεις κλειδιά θα μπορούσε να γίνει για τις λέξεις «θεραπευτήριο» ή «τραυματισμό» και ένας ευφυής πράκτορας στο Διαδίκτυο ίσως να μπορούσε να αναγνωρίσει το προσωπικό του θεραπευτηρίου. Όμως, θα ήταν αδύνατο να ξεχωρίσει την γραμματέα από τους θεραπευτές για παράδειγμα ή να εντοπίσει με ακρίβεια τις μέρες λειτουργίας αφού κάτι τέτοιο θα προϋπέθετε να ακολουθηθεί ο σύνδεσμος.

Η λύση σε τέτοιου είδους προβλήματα μπορεί να δοθεί εάν αντικατασταθεί η HTML από πιο κατάλληλες γλώσσες οι οποίες θα επιτρέπουν σε μία ιστοσελίδα όχι μόνο να διατηρεί πληροφορίες μορφοποίησης του περιεχομένου της αλλά και πληροφορίες για το ίδιο το περιεχόμενο. Ο όρος μεταδεδομένα (metadata) αναφέρεται ακριβώς σε αυτό, δεδομένα σχετικά με τα δεδομένα. Στο παραπάνω παράδειγμα, θα μπορούσαν να υπάρχουν πληροφορίες όπως:

```
<company>
      <treatmentOffered>Φυσιοθεραπεία</treatmentOffered>
      <companyName>Πρότυπο φυσιοθεραπευτήριο</companyName>
      <staff>
            <therapist>Παπαγεωργίου</therapist>
            <therapist>Ανδρεάδης</therapist>
            <secretary>Νικολάου</secretary>
      </staff>
</company>
```

Τέλος, βασικό συστατικό του Σημασιολογικού Διαδικτύου πέρα από μέθοδο αναπαράστασης της γνώσης είναι και ένας μηχανισμός που να μας επιτρέπει να επεξεργαστούμε την γνώση αυτή. Αυτός ο μηχανισμός θα πρέπει να υποστηρίζει την δυνατότητα λογικής επεξεργασίας των πληροφοριών με σκοπό την εξαγωγή συμπερασμάτων, την δημιουργία νέας γνώσης, την υποστήριξη στη λήψη αποφάσεων και την αυτόματη εκτέλεση ενεργειών.

Στο Σχήμα 1, παρατίθεται ένα σχηματικό διάγραμμα της όψης του Σημασιολογικού Διαδικτύου, έτσι όπως το οραματίστηκε ο δημιουργός του Tim Berners-Lee και γίνεται μια συνοπτική παρουσίαση των επιμέρους τμημάτων του.



Σχήμα 1: Αναπαράσταση της δομής του Σημασιολογικού Διαδικτύου από τον Tim Berners-Lee

2.1.1 Το βασικό μοντέλο των δηλώσεων

Κατά την προσπάθεια δημιουργίας ενός Διαδικτύου πληροφοριών σημασιολογικού περιεχομένου η αρχή της μινιμαλιστικής σχεδίασης απαιτεί την ύπαρξη ενός κοινού μοντέλου πάνω στο οποίο θα λειτουργεί κάθε μελλοντική εφαρμογή. Το μοντέλο αυτό θα πρέπει να διαθέτει δύο χαρακτηριστικά: *γενικότητα* και πολλά *διαδοχικά επίπεδα οργάνωσης*.

Το γενικό αυτό μοντέλο που χρησιμοποιείται είναι η γλώσσα RDF (Resource Description Framework). Το βασικό αυτό μοντέλο αποτελείται μόνο από ισχυρισμούς (assertions) και αναφορές σχετικές με αυτούς (quotations). Έχοντας ένα σύνολο από γεγονότα είναι εύκολο να πει κάποιος αν μία πρόταση είναι αληθής ή όχι, οπότε η γλώσσα σε αυτό το επίπεδο μπορεί να διαχειριστεί την πλειοψηφία των εφαρμογών που αφορούν απλή αναπαράσταση δεδομένων, όπως η αναγνώριση πνευματικών δικαιωμάτων ιδιοκτησίας, οι πληροφορίες για προσωπικά δεδομένα (P3P) κ.α.

Στο επίπεδο αυτό όμως, δεν δίνεται η δυνατότητα για έκφραση της άρνησης ή εξαγωγή επαγωγικών συμπερασμάτων, κάτι που δημιουργεί πολλούς περιορισμούς. Δεν είναι λοιπόν προφανές γιατί να αναπτύξει κάποιος μια εφαρμογή σε RDF. Η απάντηση είναι ότι, όσο απλή και περιορισμένη κι αν είναι η πληροφορία σε μία RDF εφαρμογή, έχει την δυνατότητα μετέπειτα να συνδυαστεί με πληροφορίες άλλων εφαρμογών στο Διαδίκτυο. Επομένως, τα

προγράμματα που χρησιμοποιούνται στο Διαδίκτυο πρέπει να έχουν ένα κοινό πλαίσιο ώστε να μπορούν να συνδυάζουν και να επεξεργάζονται πληροφορίες από διαφορετικές εφαρμογές.

2.1.2 Το επίπεδο δομής

Το βασικό μοντέλο της γλώσσας RDF παρέχει μεγάλη ελευθερία κινήσεων, χωρίς όμως να παρέχει ποικιλία εργαλείων. Σε επόμενο επίπεδο χρειάζεται η δυνατότητα να δηλώνονται νέες ιδιότητες και να τίθενται περιορισμοί στον τρόπο χρήσης τους. Συνήθως, επιθυμούμε να περιορίσουμε τους τύπους αντικειμένων πάνω στα οποία μπορούν να εφαρμοστούν. Αυτή η δυνατότητα, με την οποία μπορούμε να πραγματοποιήσουμε στοιχειώδεις ελέγχους πάνω σε ένα έγγραφο, προσφέρεται από το επίπεδο δομής.

Η γλώσσα του επιπέδου δομής υποστηρίζει δηλώσεις που καθορίζουν αν είναι επιτρεπτή κάποια πρόταση. Για παράδειγμα, μπορεί να γίνει έλεγχος ότι στο δίπλωμα οδήγησης κάποιου, το πεδίο «όνομα» αναφέρεται σε όνομα ανθρώπου και όχι σε μοντέλο αυτοκινήτου. Οι περιορισμοί που εκφράζονται με τη γλώσσα του επιπέδου δομής θα ήταν εύκολο να επεκταθούν ώστε να συμπεριλάβουν και λογικούς ισχυρισμούς, κάτι το οποίο όμως γίνεται σε ανώτερο επίπεδο, καθώς το επίπεδο δομής προορίζεται για εφαρμογή μόνο σε μηχανισμούς ελέγχου δομής και όχι σε μηχανισμούς εξαγωγής συμπερασμάτων. Το λογικό επίπεδο λοιπόν παρέχει επιπλέον τη δυνατότητα αναπαράστασης κατηγορηματικής λογικής (δηλαδή τη χρήση των λογικών τελεστών (not, and, or) και την δυνατότητα συσχέτισης των δεδομένων (π.χ. αν ισχύει το χ, τότε ισχύουν όλα τα y).

Ένα παράδειγμα εφαρμογής του λογικού επιπέδου φαίνεται όταν έχουμε δύο βάσεις δεδομένων που έχουν δημιουργηθεί ξεχωριστά, τοποθετήθηκαν στο Διαδίκτυο, συνδέθηκαν με σημασιολογικούς συνδέσμους, έτσι ώστε να υπάρχει δυνατότητα ερωτήματα που απευθύνονται στη μία να μετασχηματίζονται αυτόματα ώστε να τα επεξεργάζεται και η άλλη. Έτσι ενώ αρχικά φαινόταν ότι για την επίλυση διαφόρων θεμάτων θα χρειαζόταν μία εντελώς καινούρια γλώσσα, τώρα αυτό γίνεται απλά και μόνο με την συγγραφή του σωστού RDF κώδικα. Για να οριστεί λοιπόν μία νέα γλώσσα για μια συγκεκριμένη εφαρμογή χρειάζονται δύο πράγματα:

- Να οριστεί ο βαθμός επέμβασης του χρήστη στη λειτουργία της συλλογιστικής μηχανής και του υποσυνόλου της RDF που πρέπει αυτός να γνωρίζει.

- Να οριστούν κάποιες συναρτήσεις για τον τρόπο έκφρασης των ισχυρισμών μέσα στα όρια των περιορισμών που θα θέτει η γλώσσα.

2.1.3 Το λογικό επίπεδο

Το επόμενο επίπεδο είναι το επίπεδο λογικής. Αυτό προκύπτει επειδή χρειαζόμαστε τρόπους να εισάγουμε λογική στα προγράμματά μας ώστε να είναι κατανοητοί ισχυρισμοί όπως οι κανόνες που επιτρέπουν την μετατροπή ενός τύπου εγγράφου σε έναν άλλο τύπο, τον

έλεγχο ενός εγγράφου ως προς την συνέπειά του ή την λύση ενός ερωτήματος μέσω μετατροπής άγνωστων όρων σε γνωστούς.

2.1.4 Απόδειξη ελέγχου αξιοπιστίας

Το RDF μοντέλο δεν αναφέρει τίποτα σχετικά με την μορφή του μηχανισμού εξαγωγής συμπερασμάτων για τον έλεγχο αξιοπιστίας στο Διαδίκτυο. Συνήθως, στις περισσότερες εφαρμογές ο έλεγχος γίνεται με βάση κάποιους κοινούς κανόνες και η άλλη πλευρά πρέπει να δώσει μία γενική απόδειξη αξιοπιστίας.

Για παράδειγμα, για να επιτραπεί σε κάποιον η πρόσβαση σε ένα δικτυακό τόπο, μπορεί να του ζητηθεί να παραδώσει ένα έγγραφο το οποίο να αποδεικνύει στον εξυπηρετητή τον λόγο για τον οποίο του επιτράπηκε η πρόσβαση. Η απόδειξη αυτή θα αποτελείται από μια αλυσίδα δηλώσεων και κανόνων εξαγωγής συμπερασμάτων με αναφορές σε όλο το υλικό υποστηριζόμενο από το δικτυακό τόπο. Το ίδιο θα ισχύει και για τις ηλεκτρονικές συναλλαγές που απαιτούν εχεμύθεια και αφορούν κυρίως το ηλεκτρονικό εμπόριο. Τα έγγραφα που θα διακινούνται στο δίκτυο θα περιλαμβάνουν και συμπληρωματικές πληροφορίες που θα αφορούν την αξιοπιστία τους έτσι ώστε αν τεθεί ανάλογο θέμα, η απάντηση-απόδειξη να είναι εύκολο να κατασκευαστεί υπολογιστικά.

Στο πρωτόκολλο HTTP, η αναζήτηση μιας πληροφορίας θα πρέπει να περιλαμβάνει την απόδειξη ότι ο χρήστης έχει το δικαίωμα να λάβει απάντηση. Η προσκόμιση της πληροφορίας στον χρήστη θα είναι και η επιβεβαίωση του συστήματος ότι έλαβε τις επιθυμητές αποδείξεις αξιοπιστίας.

2.1.5 Ψηφιακές υπογραφές

Οι ψηφιακές υπογραφές είναι μια αξιόλογη τεχνολογία αποτελώντας απαραίτητο εργαλείο για τον έλεγχο της αξιοπιστίας των συλλογιστικών συστημάτων. Προκειμένου ο μηχανισμός εξαγωγής συμπερασμάτων να είναι σε θέση να συμπεριλάβει την απόδειξη αξιοπιστίας των εγγράφων πρέπει να γίνει επέκταση του λογικού μοντέλου ώστε να συμπεριλαμβάνει τα κλειδιά με τα οποία έχουν υπογραφεί τα έγγραφα. Αυτό σημαίνει ότι ο μηχανισμός εξαγωγής συμπερασμάτων θα είναι στενά συνδεδεμένος με τον μηχανισμό ελέγχου της αξιοπιστίας των εγγράφων. Τα έγγραφα δεν θα αναλύονται μόνο σε δένδρα συλλογισμών, αλλά και σε δένδρα με βάση το ποιος έχει «υπογράψει» αυτούς τους συλλογισμούς. Η απόδειξη αξιοπιστίας για τους συλλογισμούς ενός εγγράφου θα είναι ο έλεγχος της ψηφιακής τους υπογραφής. Το αποτέλεσμα θα είναι ένα σύστημα το οποίο θα μπορεί να εκφράσει και να αιτιολογήσει τις σχέσεις ανάμεσα σε όλο το εύρος των συστημάτων ελέγχου αξιοπιστίας.

2.2 Το μοντέλο αναπαράστασης δεδομένων RDF

Το RDF (Resource Description Framework) είναι το πρότυπο που υιοθετήθηκε από τον W3C για την περιγραφή πληροφοριακών πόρων και γενικότερα για την αναπαράσταση γνώσης στο περιβάλλον του Διαδικτύου. Συγκεκριμένα, προορίζεται για την αναπαράσταση μεταδεδομένων (metadata) σχετικών με πόρους του Διαδικτύου, δηλαδή συμπληρωματικών πληροφοριών όπως είναι ο τίτλος, ο συντάκτης και η ημερομηνία τελευταίας τροποποίησης μιας ιστοσελίδας, η άδεια χρήσης για ένα έγγραφο στο διαδίκτυο. Γενικά, με την έννοια πόρος εννοούμε οτιδήποτε είναι αναγνωρίσιμο στο Διαδίκτυο, οτιδήποτε για το οποίο θέλουμε να δηλώσουμε κάτι ή να το περιγράψουμε, ανεξάρτητα με το αν είναι δυνατή η άμεση ανάκτησή του από το Διαδίκτυο. Τέτοια παραδείγματα πληροφοριών που υφίστανται εικονικά στο Διαδίκτυο αλλά δεν είναι εφικτή η ανάκτησή τους είναι οι πληροφορίες για προϊόντα διαθέσιμα μέσω των ηλεκτρονικών καταστημάτων (όπως πληροφορίες για τις προδιαγραφές, τις τιμές και τη διαθεσιμότητα των προϊόντων), ή η περιγραφή για τις προτιμήσεις ενός χρήστη του διαδικτύου σχετικά με τον τρόπο παρουσίασης της πληροφορίας.

Γενικά, πόρος μπορεί να είναι μια σελίδα στο Διαδίκτυο ή ένα δικτυακός τόπος ή φυσικά αντικείμενα ή έννοιες ή και μία λέξη, οτιδήποτε. Οι πόροι δηλώνονται με τα Καθολικά Αναγνωριστικά Πόρων (Universal Resource Identifiers – URI), τα οποία είναι παγκόσμια, μοναδικά και μόνιμα αναγνωριστικά και αποτελούν ταυτότητα των πόρων που περιγράφουν. Συσχετίζοντας ένα πόρο με ένα URI δίνεται η δυνατότητα σε οποιονδήποτε να αναφερθεί σε αυτόν τον πόρο ή και να ανακτήσει μία αναπαράστασή του.

Το RDF προορίζεται για περιπτώσεις στις οποίες η πληροφορία είναι αναγκαίο να γίνεται αντικείμενο επεξεργασίας από εφαρμογές και όχι απλά να προορίζεται για απεικόνιση στον άνθρωπο. Το RDF ουσιαστικά προσφέρει ένα κοινό πλαίσιο έκφρασης της πληροφορίας έτσι ώστε αυτή να ανταλλάσσεται ανάμεσα στις εφαρμογές χωρίς καθόλου απώλεια νοήματος.

2.2.1 Σύνταξη

Το RDF βασίζεται στην ιδέα ότι οι πόροι μπορούν να περιγραφούν με βάση κάποιες ιδιότητές (properties) τους οι οποίες παίρνουν κάποιες τιμές (values). Όπως αναφέρεται και παραπάνω, στην RDF, τα στοιχεία αναπαριστώνται με την χρήση διευθύνσεων προσδιοριστών - URIs - και κάθε πόρος μπορεί να περιγραφεί με βάση τις ιδιότητες και τις αντίστοιχες τιμές τους.

Το RDF είναι ένα απλό μοντέλο δεδομένων: όλες οι προτάσεις – δεδομένα του RDF αποτελούνται από μια τριπλέτα της μορφής: [Υποκείμενο (Subject), Σχέση (Predicate), Αντικείμενο (Object)] όπου το Υποκείμενο και η Σχέση δηλώνονται με ένα URI ενώ το Αντικείμενο μπορεί να δηλώνεται επίσης με ένα URI ή μπορεί να είναι και ένα αλφαριθμητικό. Πιο συγκεκριμένα, το στοιχείο για το οποίο γίνεται λόγος καλείται subject (υποκείμενο), το

στοιχείο που προσδιορίζει μια ιδιότητα ή ένα χαρακτηριστικό του υποκειμένου ονομάζεται predicate (κατηγόρημα) και το στοιχείο που δίνει τιμή σε αυτή την ιδιότητα καλείται object (αντικείμενο). Η περιγραφή ενός διαδικτυακού πόρου επομένως μπορεί να γίνει με τριπλέτες αυτής της μορφής, που ονομάζονται προτάσεις (statements) και οι οποίες ορίζουν τις τιμές των ιδιοτήτων του.

Για παράδειγμα, έστω ότι κάποιος `David Billington` είναι ο ιδιοκτήτης (owner) μίας ιστοσελίδας με διεύθυνση `http://www.cit.gu.edu.au/~db`. Οι όροι αυτοί εκφρασμένοι σε rdf είναι: υποκείμενο είναι ηURL διεύθυνση `http://www.cit.gu.edu.au/~db` , κατηγόρημα είναι η λέξη `owner` και αντικείμενο η συμβολοσειρά `David Billington`.

Οι προτάσεις RDF παριστάνονται συνήθως ως κατευθυνόμενοι γράφοι με ετικέτες. Το Σχήμα 2 σχήμα αποτελεί τον γράφο της πρότασης του παραδείγματος:

Σχήμα 2: Αναπαράσταση μίας τυπικής πρότασης RDF με μορφή γράφου

2.2.2 Η γλώσσα περιγραφής λεξιλογίου RDF SCHEMA

Η γλώσσα RDF είναι ένα μοντέλο δεδομένων ανεξάρτητο από το εκάστοτε συγκεκριμένο πεδίο χρήσης καθώς δεν καθορίζει την σημασιολογία του πεδίου. Η RDF δεν παρέχει κανένα μηχανισμό για την περιγραφή των σχέσεων που μπορεί να συνδέουν τις ιδιότητες δύο διαφορετικών πόρων. Αυτόν τον ρόλο τον αναλαμβάνει η περιγραφική γλώσσα λεξιλογίου της RDF, η γλώσσα RDF Schema (RDFS), η οποία ορίζει ένα λεξικό - εκφρασμένο επίσης σε RDF - για να εκφράζονται οι κατηγορίες (κλάσεις) των πόρων, οι ιδιότητες των πόρων και οι μεταξύ τους σχέσεις. Η RDF Schema θεωρείται ότι είναι η σημασιολογική επέκταση της RDF, καθώς παρέχει μηχανισμούς για την περιγραφή συνόλων από σχετικούς μεταξύ τους πόρους και τις σχέσεις που τους συνδέουν. Στην RDFS ορίζονται τα περιγραφικά λεξικά της RDF τα οποία καθορίζουν διάφορα χαρακτηριστικά των πόρων όπως το πεδίο τιμών και το πεδίο ορισμού των ιδιοτήτων τους. Η γλώσσα RDFS είναι παρόμοια με τις γλώσσες αντικειμενοστραφούς προγραμματισμού με την διαφορά ότι η RDFS, αντί να ορίζει μία κλάση με βάση τις ιδιότητες που μπορεί να έχουν τα στιγμιότυπά της, περιγράφει τις ιδιότητες αυτές με βάση τις κλάσεις πόρων τις οποίες αφορούν. Το μεγάλο πλεονέκτημα που απορρέει από αυτή την προσανατολισμένη στις ιδιότητες προσέγγιση είναι το ότι επιτρέπει στον καθένα να

επεκτείνει την περιγραφή ενός πόρου χωρίς να χρειάζεται να ορίσει ξανά την περιγραφή της κλάσης του πόρου αυτού.

Στην RDF Schema οι δηλώσεις είναι πάντα περιγραφικές. Μπορεί να περιλαμβάνουν περιορισμούς, μόνο όμως αν μια εφαρμογή θέλει να τους συμπεριλάβει. Το μόνο που κάνει η RDF Schema είναι να παρέχει τον τρόπο για να δηλωθούν τέτοιου είδους πληροφορίες. Από εκεί και πέρα εξαρτάται από την εκάστοτε εφαρμογή ο τρόπος διαχείρισης των μεταδεδομένων.

- **Κλάσεις**

Μία κλάση είναι ένα σύνολο από διαδικτυακούς πόρους οι οποίοι καλούνται στιγμιότυπα της κλάσης. Η ιδιότητα `rdf:type` χρησιμοποιείται για να δηλώσει ότι ένας πόρος αποτελεί στιγμιότυπο μιας κλάσης.

Η RDFS επίσης δίνει την δυνατότητα ορισμού μίας ιεραρχίας των κλάσεων. Αν μια κλάση Α είναι υποκλάση μιας άλλης κλάσης Β, τότε όλα τα στιγμιότυπα της Α είναι και στιγμιότυπα της Β. Η ιδιότητα `rdf:subClassOf` συσχετίζει μία κλάση με τις υπερκλάσεις της.

Τέλος, υπάρχουν και κάποιες θεμελιώδεις κλάσεις στην RDFS όπως η `rdfs:Resource` που είναι η κλάση όλων των πόρων, η `rdfs:Class` που είναι η κλάση όλων των κλάσεων και η `rdfs:Literal` που είναι η κλάση όλων των συμβολοσειρών.

- **Ιδιότητες**

Εκτός από την περιγραφή συγκεκριμένων κλάσεων υπάρχει η δυνατότητα να περιγραφούν οι συγκεκριμένες ιδιότητες που χαρακτηρίζουν αυτές τις κλάσεις. Μια RDF ιδιότητα είναι η σχέση που υπάρχει μεταξύ ενός πόρου-υποκειμένου και ενός πόρου-αντικειμένου. Οι ιδιότητες μπορούν να περιγραφούν χρησιμοποιώντας τις εξής κλάσεις:

- ✓ η κλάση `rdfs:subPropertyOf` ορίζει την ιεραρχία των ιδιοτήτων, για παράδειγμα η κλάση father μπορεί να περιγραφεί με την κλάση `rdfs:subPropertyOf` ως υποκλάση της κλάσης parent αφού κάθε οντότητα που έχει την ιδιότητα πατέρας έχει επίσης την ιδιότητα γονιός
- ✓ η `rdfs:domain` ορίζει το πεδίο ορισμού μίας ιδιότητας και
- ✓ η `rdfs:range` το πεδίο τιμών της.

2.2.3 Αναπαράσταση RDF/XML

Εκτός από την αναπαράσταση της RDF σε γράφους, υπάρχει η δυνατότητα αναπαράστασης με το συντακτικό της XML (σύνταξη RDF/XML) ώστε να υπάρχει συμβατότητα με τα υπάρχοντα πρότυπα και εφαρμογές διαδικτύου.

Έστω για παράδειγμα, ως πεδίο μελέτης έχουμε τα μαθήματα και τους καθηγητές του πανεπιστημίου Griffith το έτος 2001. Η σύνταξη RDF/XML φαίνεται παρακάτω:

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:xsd ="http://www.w3.org/2001/XLMSchema#"
xmlns:uni="http://www.mydomain.org/uni-ns">
      <rdf:Description rdf:about="949352">
            <uni:name>Grigoris Antoniou</uni:name>
            <uni:title>Professor</uni:title>
      </rdf:Description>
      <rdf:Description rdf:about="949318">
            <uni:name>David Billington</uni:name>
            <uni:title>Associate Professor</uni:title>
            <uni:age rdf:datatype="&xsd:integer">27<uni:age>
      </rdf:Description>
      <rdf:Description rdf:about="949111">
            <uni:name>Michael Maher</uni:name>
            <uni:title>Professor</uni:title>
      </rdf:Description>
      <rdf:Description rdf:about="CIT1111">
            <uni:courseName>Discrete Mathematics</uni:courseName>
            <uni:isTaughtBy>David Billington</uni:isTaughtBy>
      </rdf:Description>
      <rdf:Description rdf:about="CIT1112">
            <uni:courseName>Concrete Mathematics</uni:courseName>
            <uni:isTaughtBy>Grigoris Antoniou</uni:isTaughtBy>
      </rdf:Description>
      <rdf:Description rdf:about="CIT2112">
            <uni:courseName>Programming III</uni:courseName>
            <uni:isTaughtBy>Michael Maher</uni:isTaughtBy>
      </rdf:Description>
      <rdf:Description rdf:about="CIT3112">
            <uni:courseName>Theory of Computation</uni:courseName>
            <uni:isTaughtBy>David Billington</uni:isTaughtBy>
      </rdf:Description>
      <rdf:Description rdf:about="CIT3116">
            <uni:courseName>Knowledge Representation</uni:courseName>
            <uni:isTaughtBy>Grigoris Antoniou</uni:isTaughtBy>
      </rdf:Description>
</rdf:RDF>
```

Αρχικά, πρέπει να γίνει μία παρατήρηση σχετικά με την χρήση των συντομεύσεων (namespaces) οι οποίες στην RDF/XML σύνταξη παραπέμπουν σε έγγραφα RDF που ορίζουν πόρους οι οποίοι χρησιμοποιούνται στην συνέχεια στο έγγραφο RDF που τις εισάγει. Τα περιεχόμενα του στοιχείου `rdf:Description` ονομάζονται στοιχεία ιδιοτήτων. Στο παρακάτω παράδειγμα,

```
<rdf:Description rdf:about="CIT3116">
      <uni:courseName> Knowledge Representation</uni:courseName>
      <uni:isTaughtBy>Grigoris Antoniou</uni:isTaughtBy>
</rdf:Description>
```

τα στοιχεία `uni:courseName` και `uni:isTaughtBy` ορίζουν ζεύγη ιδιοτήτων-τιμών για το στοιχείο `CIT3116` και ουσιαστικά αντιστοιχούν σε δύο δηλώσεις RDF. Τα δύο αυτά

στοιχεία ιδιοτήτων διαβάζονται συζευκτικά δηλαδή το μάθημα λέγεται 'Knowledge Representation' *και* διδάσκεται από τον Grigoris Antoniou.

Το παραπάνω παράδειγμα δεν είναι ικανοποιητικό γιατί η σχέση μεταξύ μαθημάτων και διδασκόντων ορίζεται μόνο έμμεσα μέσω της χρήσης του ίδιου ονοματεπώνυμου. Δηλαδή, ο David Billington που διδάσκει το CIT3112 μπορεί να μην είναι ο ίδιος με το άτομο με αριθμό ID949318 που τυχαίνει να λέγεται David Billington. Αυτό θα μπορούσε να αποφευχθεί με την χρήση του χαρακτηριστικού rdf:resource ως εξής.

```
<rdf:Description rdf:about="CIT1111">
      <uni:courseName>Discrete Mathematics</uni:courseName>
      <uni:isTaughtBy rdf:resource="949318"/>
</rdf:Description>
<rdf:Description rdf:about="949318">
      <uni:name>David Billington</uni:name>
      <uni:title>Associate Professor</uni:title>
</rdf:Description>
```

Στο παράδειγμά μας, όλες οι περιγραφές εμπίπτουν σε δύο κατηγορίες: μαθήματα και διδάσκοντες. Αυτό και πάλι δεν είναι προφανές στις μηχανές όσο στον άνθρωπο. Τέτοιου είδους δηλώσεις που αφορούν τον τύπο των στοιχείων μπορούμε να κάνουμε χρησιμοποιώντας το στοιχείο rdf:type, όπως παρακάτω. Με αυτόν τον τρόπο επίσης εισάγουμε και μία στοιχειώδη δομή στο έγγραφό μας.

```
<rdf:Description rdf:ID="CIT1111">
      <rdf:type rdf:resource="http://www.mydomain.org/unins#course"/>
      <uni:courseName>Discrete Mathematics</uni:courseName>
      <uni:isTaughtBy rdf:resource="#949318"/>
</rdf:Description>
<rdf:Description rdf:ID="949318">
      <rdf:type
      rdf:resource="http://www.mydomain.org/unins#lecturer"/>
      <uni:name>David Billington</uni:name>
      <uni:title>Associate Professor</uni:title>
</rdf:Description>
```

Επιπλέον, υπάρχουν οι συλλογές (container elements) τα οποία χρησιμεύουν για την πραγματοποίηση δηλώσεων για ομάδες πόρων ή ιδιοτήτων συνολικά. Υπάρχουν τρία είδη τέτοιων στοιχείων:

- ✓ rdf:Bag , το οποίο αντιπροσωπεύει μία μη ταξινομημένη συλλογή π.χ. μέλη ενός συμβουλίου
- ✓ rdf:Seq , το οποίο αντιπροσωπεύει μία ταξινομημένη συλλογή π.χ. στοιχεία μίας ατζέντας
- ✓ rdf:Alt , το οποίο αντιπροσωπεύει ένα σύνολο από εναλλακτικά στοιχεία π.χ. πολλές μεταφράσεις του ίδιου κειμένου.

Στα μέλη ενός container element αναφερόμαστε ως εξής: rdf:_1, rdf:_2 κ.ο.κ.

Για παράδειγμα, αν θέλαμε να δηλώσουμε τα μαθήματα που διδάσκονται από τον `Grigoris Antoniou`:

```
<uni:lecturer rdf:ID="949352"
      uni:name="Grigoris Antoniou"
      uni:title="Professor">
      <uni:coursesTaught>
            <rdf:Bag>
                  <rdf_1 rdf:resource="#CIT1112"/>
                  <rdf_2 rdf:resource="#CIT3116"/>
            </rdf:Bag>
      </uni:coursesTaught>
</uni:lecturer>
```

Όπως έχει προαναφερθεί, συχνά επιθυμούμε να κάνουμε δηλώσεις σχετικές με άλλες δηλώσεις. Για να γίνει αυτό πρέπει να μπορούμε να αναφερθούμε σε μία δήλωση με βάση ένα αναγνωριστικό της. Αυτό επιτυγχάνεται με ένα μηχανισμό που μετατρέπει μία δήλωση σε πόρο. Για παράδειγμα, η δήλωση:

```
<rdf:Description rdf:about="#949352">
      <uni:name>Grigoris Antoniou</uni:name>
</rdf:Description>
```

μετατρέπεται σε:

```
<rdf:Statement rdf:ID="StatementAbout949352">
      <rdf:subject rdf:resource="#949352"/>
      <rdf:predicate
      rdf:resource="http://www.mydomain.org/unins#name"/>
      <rdf:object>Grigoris Antoniou</rdf:object>
</rdf:Description>
```

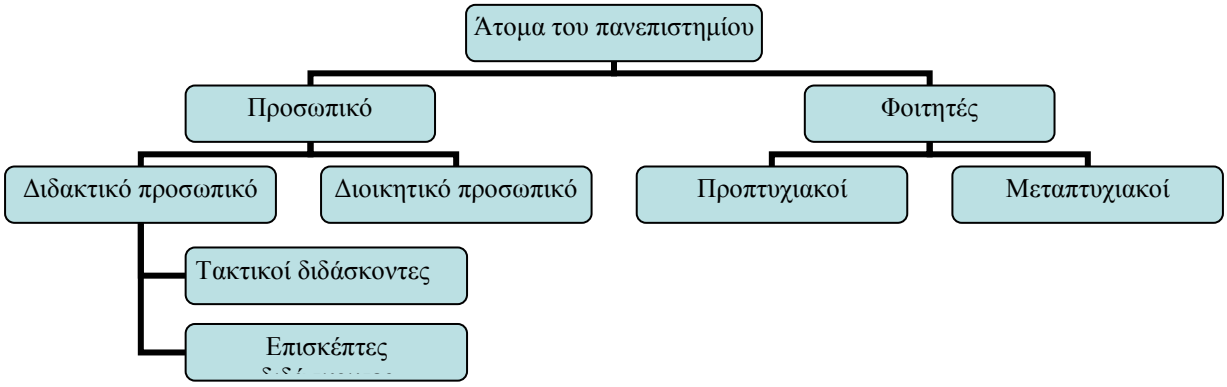
Τα `rdf:subject`, `rdf:predicate`, και `rdf:object` μας επιτρέπουν να αναφερόμαστε στα μέρη μίας δήλωσης RDF.

2.3 Οντολογίες

Στα πλαίσια του Σημασιολογικού Διαδικτύου *οντολογία* είναι μία αυστηρή περιγραφή των αντικειμένων-πόρων και των σχέσεων μεταξύ τους. Οι οντολογίες επιτρέπουν σε μία κοινότητα χρηστών να έχουν κοινή ονοματολογία και κοινή αντίληψη για τα αντικείμενα-πόρους τα οποία δηλώνουν ή χρησιμοποιούν. Επίσης, οι οντολογίες επιτρέπουν στους χρήστες να περιγράψουν τον ‘κόσμο’ που χρησιμοποιούν με ένα τέτοιο τρόπο ώστε οι υπολογιστές να μπορούν να επεξεργαστούν και συνδέσουν δεδομένα από διαφορετικούς κόσμους.

Τυπικά μία οντολογία αποτελείται από μία πεπερασμένη λίστα από όρους (terms) και τις μεταξύ τους σχέσεις relationships). Οι όροι αυτοί περιγράφουν σημαντικές έννοιες (κλάσεις αντικειμένων) του πεδίου. Οι σχέσεις τυπικά περιλαμβάνουν την ιεραρχία των κλάσεων (υποκλάσεις) καθώς και πληροφορίες όπως ιδιότητες κλάσεων, περιορισμούς τιμών και λογικούς περιορισμούς. Για παράδειγμα, αν το πεδίο μελέτης είναι ένα πανεπιστήμιο, το διδακτικό προσωπικό, οι φοιτητές, τα τμήματα και οι αίθουσες διαλέξεων είναι μερικές

σημαντικές έννοιες. Επιπλέον, μεταξύ τους υπάρχουν σχέσεις ιεραρχίας (όλοι οι βοηθοί εργαστηρίων είναι διδακτικό προσωπικό), ιδιότητες (ο Χ διδάσκει το Υ), περιορισμοί τιμών (μόνο το διδακτικό προσωπικό μίας σχολής μπορεί να διδάξει μαθήματα) και λογικοί περιορισμοί (κάθε σχολή πρέπει να διαθέτει τουλάχιστον 10 μέλη διδακτικού προσωπικού). Οι σχέσεις ιεραρχίας του παραδείγματος φαίνονται παρακάτω στο Σχήμα 3:



Σχήμα 3: Τυπικό παράδειγμα ιεραρχίας οντολογιών

Η χρήση οντολογιών είναι πολύ χρήσιμη στην οργάνωση και στην πλοήγηση διαδικτυακών τόπων καθώς και στην βελτίωση της ακρίβειας των αναζητήσεων στο Διαδίκτυο. Μέσω των οντολογιών, οι πληροφορίες θα ανταλλάσσονται μεταξύ των εφαρμογών, καθιστώντας τα προγράμματα ικανά να συλλέγουν και να επεξεργάζονται τα περιεχόμενα του Διαδικτύου.

Η παροχή υπηρεσιών μέσω του διαδικτύου είναι ένας από τους κλάδους όπου θα γίνει δυναμική χρήση των οντολογιών. Οι οντολογίες που θα δημιουργηθούν θα είναι κατανοητές από τις μηχανές και οι πράκτορες (agents) θα εντοπίζουν τις παρεχόμενες υπηρεσίες και θα αυτοματοποιούν την χρήση τους. Με το να εκφραστούν οι παρεχόμενες υπηρεσίες σε μια γλώσσα οντολογιών, θα δίνεται η δυνατότητα με βοήθεια της ιεραρχίας και της αρχής της κληρονομικότητας οι πράκτορες να βρίσκουν τα κοινά στοιχεία ανάμεσα στο αντικείμενο της αναζήτησης και τις ιδιότητες κλάσεων και υποκλάσεων ή και μέσω σημασιολογικών συνδέσμων. Για παράδειγμα, κάποιος που ενδιαφέρεται να αγοράσει τριαντάφυλλα κάνοντας μια αναζήτηση, μπορεί να λάβει ως απάντηση συνδέσμους σε ανθοπωλεία αν και κάτι τέτοιο δεν ταιριάζει ακριβώς με την αρχική επιλογή αναζήτησής του.

2.3.1 Περιπτώσεις χρήσης οντολογιών

2.3.1.1 Διαδικτυακές πύλες

Μία διαδικτυακή πύλη είναι μία τοποθεσία Διαδικτύου η οποία παρέχει πληροφορίες σχετικές με ένα συγκεκριμένο θέμα. Για να είναι επιτυχημένη μία πύλη πρέπει να αποτελεί αφετηρία για τον εντοπισμό ενδιαφέροντος περιεχομένου, το οποίο συνήθως υποβάλλεται από τους χρήστες και οργανώνεται σε ευρετήρια κάτω από κάποιο υπο-θέμα. Ωστόσο, αυτή η απλή αναπαράσταση συχνά δεν επαρκεί ώστε η κοινότητα χρηστών να αναζητά αποτελεσματικά πληροφορίες. Για πιο έξυπνη αναζήτηση οι πύλες μπορούν να ορίζουν για την κοινότητα χρηστών μία οντολογία, η οποία θα παρέχει την απαραίτητη ορολογία για την περιγραφή του περιεχομένου. Για παράδειγμα, μία οντολογία θα μπορούσε να περιλαμβάνει όρους όπως "journal paper", "publication", "person", "author" και δηλώσεις όπως "all journal papers are publications" και "the authors of all publications are people". Αυτές οι δηλώσεις συνδυαζόμενες με κάποια γεγονότα μπορούν να οδηγήσουν στην εξαγωγή συμπερασμάτων με άλλα γεγονότα τα οποία επαγωγικά ισχύουν. Με αυτό τον τρόπο οι χρήστες θα μπορούν να ανακτήσουν αποτελέσματα αναζητήσεων που δεν είναι δυνατό να ανακτηθούν με τα συμβατικά συστήματα αναζητήσεων στο Διαδίκτυο.

2.3.1.2 Συλλογές πολυμέσων

Οι οντολογίες μπορούν επίσης να χρησιμοποιηθούν και για να προσδώσουν σημασιολογικό περιεχόμενο σε συλλογές εικόνων, βίντεο, ήχου κ.α. Καθώς είναι πολύ δυσκολότερο για τις μηχανές να εξάγουν σημασιολογικές πληροφορίες από πολυμέσα παρά από κείμενο, αυτά τα αντικείμενα πολυμέσων συνήθως τοποθετούνται σε ευρετήρια με χρήση κάποιων ετικετών. Η αναζήτηση όμως αυτών των αντικειμένων με λέξεις-κλειδιά δεν είναι αποτελεσματική καθώς διαφορετικοί άνθρωποι μπορούν να τα περιγράψουν με τελείως διαφορετικούς τρόπους. Εδώ έρχονται οι οντολογίες οι οποίες μπορούν να διατηρούν πρόσθετες πληροφορίες σχετικές με το πεδίο αναζήτησης ώστε να βελτιώνεται η αναζήτηση εικόνας και ήχου και να υπερβαίνει την απλή αναζήτηση με ταίριασμα λέξεων-κλειδιών.

Οι οντολογίες πολυμέσων μπορούν να είναι δύο τύπων: βασισμένες στο μέσο και βασισμένες στο περιεχόμενο. Οι οντολογίες της πρώτης κατηγορίας θα μπορούσαν να έχουν ταξινομήσεις ανάλογα με τον τύπο πολυμέσου και τις ιδιότητές του. Για παράδειγμα, το βίντεο θα μπορούσε να έχει ως ιδιότητες την διάρκεια και τις αλλαγές σκηνής. Οι οντολογίες βασισμένες στο περιεχόμενο θα μπορούσαν να περιγράφουν το θέμα με βάση τους συμμετέχοντες ή ο σκηνικό. Το σημαντικό είναι ότι τέτοιου είδους οντολογίες είναι ανεξάρτητες του θέματος και επομένως μπορούν εύκολα να χρησιμοποιούνται σε πολλές περιπτώσεις.

Ένα παράδειγμα συλλογής πολυμέσων που θα αφορούσε έπιπλα αντίκες θα χρησιμοποιούσε μία οντολογία για τις αντίκες. Η οντολογία θα χρησιμοποιούσε αφ' ενός μία ταξινόμηση ανάλογα με τον τύπο επίπλου. Επιπλέον, η οντολογία θα έπρεπε να μπορεί να εκφράσει προσδιοριστική γνώση (definitional knowledge) δηλαδή αν, για παράδειγμα, κάποιος χρήστης αναζητούσε έπιπλα με στιλ περιόδου 'Late Georgian' θα έπρεπε να είναι δυνατό να βγει το συμπέρασμα ότι αναζητούνται έπιπλα με τιμή της ιδιότητας 'date.created' ανάμεσα από 1760 και 1811 και ότι η ιδιότητα 'culture' έχει την τιμή British. Αυτή η δυνατότητα της γνώσης υπόβαθρου (background knowledge) αυξάνει σημαντικά την αποτελεσματικότητα των αναζητήσεων. Ένα επιπλέον χαρακτηριστικό είναι η αναπαράσταση της εξ' ορισμού γνώσης (π.χ. ότι μία ντουλάπα περιόδου 'Late Georgian' εφόσον απουσιάζουν περαιτέρω πληροφορίες συμπεραίνεται ότι είναι από ξύλο 'mahogany'. Αυτό δίνει την δυνατότητα σε μία αναζήτηση χρήστη του τύπου 'antique mahogany storage furniture' να επιστρέψει αποτελέσματα όπως εικόνες από Late Georgian ντουλάπες ακόμα και αν η περιγραφή της εικόνας δεν αναφέρει πουθενά για υλικό το ξύλο.

2.3.1.3 Πράκτορες και υπηρεσίες

Το Σημασιολογικό Διαδίκτυο δίνει στους πράκτορες την δυνατότητα να κατανοούν και να ενσωματώνουν διαφορετικούς πόρους πληροφοριών. Ένα κλασσικό παράδειγμα είναι αυτό ενός οργανωτή κοινωνικών εκδηλώσεων ο οποίος λαμβάνοντας υπόψη του τις προτιμήσεις του χρήστη σχεδιάζει τις δραστηριότητες ενός απογεύματός του. Αυτό το είδος πράκτορα απαιτεί οντολογίες του πεδίου εφαρμογής που θα αντιπροσωπεύουν τα ξενοδοχεία, τα εστιατόρια κ.λ.π. καθώς επίσης και οντολογίες υπηρεσιών που θα αντιπροσωπεύουν τις πραγματικές υπηρεσίες. Αυτές οι οντολογίες θα επιτρέπουν την συλλογή των απαραίτητων πληροφοριών οι οποίες θα προέρχονται από πύλες, ιστοσελίδες παροχής υπηρεσιών και το ευρύτερο διαδίκτυο.

2.3.2 Γλώσσα Οντολογιών Διαδικτύου

Η RDF και η RDF Schema διαθέτουν πολύ περιορισμένη εκφραστικότητα η οποία δεν επαρκεί για κάποιες περιπτώσεις χρήσης του Σημασιολογικού Διαδικτύου. Για παράδειγμα:

- ✓ Δεν δίνεται δυνατότητα ορισμού τοπικής ισχύος για ιδιότητες. Στο RDF Schema δεν μπορούμε να δηλώσουμε ότι μία ιδιότητα ισχύει μόνο για μερικές κλάσεις (π.χ. για μία ιδιότητα eats δεν μπορούμε να δηλώσουμε ότι η αγελάδα τρώει μόνο φυτά ενώ άλλα ζώα μπορεί να τρώνε και σάρκα)
- ✓ Δεν είναι δυνατός ο ορισμός δύο αμοιβαίως αποκλειόμενων κλάσεων (π.χ. οι κλάσεις male και female) παρά μόνο ο ορισμός ιεραρχίας κλάσεων.
- ✓ Δεν παρέχεται η δυνατότητα να ορίσουμε λογικούς συνδυασμούς κλάσεων όπως ένωση ή τομή δύο κλάσεων

- ✓ Επιπλέον, στην RDF και RDF Schema δεν υπάρχει η δυνατότητα να οριστεί περιορισμός στο πλήθος των διακριτών τιμών που μπορεί να πάρει μία ιδιότητα
- ✓ Τέλος, δεν παρέχεται η δυνατότητα ορισμού κάποιων ιδιαίτερων χαρακτηριστικών για τις ιδιότητες όπως για παράδειγμα ο ορισμός ότι μία ιδιότητα είναι αντίστροφη μίας άλλης.

Η λεξικολογική επέκταση της RDF η οποία χρησιμοποιείται για τη δημιουργία στιγμιότυπων των οντολογιών του Διαδικτύου είναι η OWL (Web Ontology Language). Παρέχει επιπλέον λεξικό για την περιγραφή κλάσεων και ιδιοτήτων: συσχετίσεις κλάσεων (π.χ. αμοιβαίος αποκλεισμός), πλήθος (π.χ. 'ακριβώς ένα'), ισότητα, περισσότερους τύπους ιδιοτήτων, χαρακτηριστικά ιδιοτήτων (π.χ. συμμετρία) και ορισμό κλάσεων με απαρίθμηση των μελών τους.

Η OWL παρέχει τρεις πολύ εκφραστικές υπο-γλώσσες:

- Η OWL Lite, χρησιμοποιείται από αυτούς που χρειάζονται κλάσεις

δομημένες σε ιεραρχία και απλούς περιορισμούς για τα χαρακτηριστικά.

- Η OWL DL, υποστηρίζει χρήστες που επιθυμούν μεγάλη εκφραστικότητα,

χωρίς όμως να χάνουν σε πληρότητα (η OWL DL εγγυάται ότι όλα τα λογικά συμπεράσματα μπορούν να υπολογιστούν) και περατότητα (το σύστημα ολοκληρώνει όλους τους υπολογισμούς σε πεπερασμένο χρόνο). Παρέχει όλα τα μέρη της γλώσσας OWL, μόνο που αυτά μπορούν να χρησιμοποιηθούν κάτω από ορισμένες συνθήκες, για παράδειγμα, ενώ μία κλάση μπορεί να είναι υποκλάση μίας άλλης δεν μπορεί να αποτελεί στιγμιότυπο μία άλλης κλάσης.

- Η OWL Full, προορίζεται για χρήστες που επιθυμούν μεγάλη εκφραστικότητα σε συνδυασμό με την συντακτική ελευθερία που προσφέρει η RDF, χωρίς όμως υπολογιστικές εγγυήσεις. Η OWL Full μπορεί να θεωρηθεί επέκταση της OWL ενώ οι άλλες δύο ως επεκτάσεις μίας αυστηρής μορφής της RDF.

Η OWL παρέχει ικανοποιητική εκφραστικότητα για να χρησιμοποιηθεί στην πράξη. Ωστόσο νέες επεκτάσεις ήδη έχουν ξεκινήσει να εμφανίζονται οι οποίες θα παρέχουν περισσότερα λογικά χαρακτηριστικά, συμπεριλαμβανομένων και κανόνων.

Η λεπτομερής περιγραφή της OWL δεν περιέχεται στον αντικειμενικό σκοπό της παρούσας εργασίας επομένως θα παρουσιαστούν τα βασικά χαρακτηριστικά της μέσω ενός παραδείγματος.

<u>Παράδειγμα</u>:

Ορισμός προβλήματος: Δημιουργία μιας οντολογίας για τον τόπο προέλευσης/ παραγωγής των κρασιών. Υπάρχει μεγάλη ποικιλία στους τύπους των κρασιών, το είδος των οποίων εξαρτάται από την περιοχή παραγωγής. Εδώ, η περιοχή παραγωγής μπορεί να είναι από μια χώρα μέχρι

ένα συγκεκριμένο αμπέλι. Θα μπορούσε, λοιπόν, κανείς να διακρίνει 4 τύπους περιοχών παραγωγής κρασιού
ανάλογα με το μέγεθός τους:
1.χώρα
2.περιοχή
3.πόλη
4.αμπέλι
Επιπλέον, χρειάζεται να αναπαραστήσουμε και μερικές από τις σχέσεις που
συνδέουν αυτούς τους τύπους περιοχών:
-οι περιοχές είναι τμήματα των χωρών
-οι περιοχές μπορεί να διαιρούνται και σε μικρότερης έκτασης τμήματα (υπο-
περιοχές)
-οι πόλεις ανήκουν σε περιοχές
-τα αμπέλια είναι τμήματα των πόλεων.
Μέσω της οντολογίας, είναι επιθυμητό να μπορεί κανείς να συμπεράνει πως αν ένα κρασί προέρχεται από ένα συγκεκριμένο αμπέλι, τότε το κρασί αυτό έχει μια συγκεκριμένη χώρα προέλευσης (που προκύπτει μέσα από τις παραπάνω σχέσεις).
ΠΑΡΑΔΟΧΗ: Η υποκλάση πόλη δεν ορίζεται εδώ για λόγους απλότητας και
καθώς μπορεί να υποτεθεί ότι μια πόλη ως τόπος παραγωγής περιλαμβάνει και την
ευρύτερη περιοχή.

Ορισμός Κλάσεων

```
1. <owl:Class rdf:ID="&vin;ProductionArea"/>
2. <owl:Class rdf:ID="&vin;Country">
3.    <rdfs:subClassOf rdf:resource="&vin;ProductionArea"/>
4. </owl:Class>
5. <owl:Class rdf:ID="&vin;Region">
6.    <rdfs:subClassOf rdf:resource="&vin;ProductionArea"/>
7. </owl:Class>
8. <owl:Class rdf:ID="&vin;Vineyard">
9.    <rdfs:subClassOf rdf:resource="&vin;ProductionArea"/>
10. </owl:Class>
```

Αντιστοιχία σε Τριάδες

```
11. vin:ProductionArea rdf:type owl:Class.
12. vin:Country rdfs:subClassOf vin:ProductionArea.
13. vin:Region rdfs:subClassOf vin:ProductionArea.
14. vin:Vineyard rdfs:subClassOf vin:ProductionArea.
```

Ορισμός Ιδιοτήτων

```
15. <owl:ObjectProperty rdf:ID="&vin;hasSubArea">
16.   <rdf:type rdf:resource="&owl;TransitiveProperty"/>
17. </owl:ObjectProperty>
18. <owl:ObjectProperty rdf:ID="&vin;subAreaOf">
19.   <owl:inverseOf rdf:resource="&vin;hasSubArea"/>
```

```
20. </owl:ObjectProperty>
21. <owl:ObjectProperty rdf:ID=”&vin;hasRegion:>
22.   <rdfs:subPropertyOf rdf:resource=”&vin;hasSubArea”/>
23.   <rdfs:domain rdf:resource=”&vinRegion”/>
24. </owl:ObjectProperty>
25. <owl:ObjectProperty rdf:ID=”&vin;regionOf”>
26.   <rdf:type rdf:resource=”&owl;FunctionalProperty”/>
27.   <owl:inverseOf rdf:resource=”&vin;hasRegion”/>
28.   <rdfs:range rdf:resource=”&vin;Region”/>
29. </owl:ObjectProperty>
30. <owl:ObjectProperty rdf:ID=”&vin;hasSubRegion”>
31.   <rdfs:subPropertyOf rdf:resource=”&vin;hasSubArea”/>
32.   <rdfs:range rdf:resource=”&vin;Region”/>
33. </owl:ObjectProperty>
34. <owl:ObjectProperty rdf:ID=”&vin;subRegionOf”>
35.   <owl:inverseOf rdf:resource=”&vin;hasSubRegion”/>
36.   <rdfs:range rdf:resource=”&vin;Region”/>
37.   <rdf:type rdf:resource=”&owl;FunctionalProperty”/>
38. </owl:ObjectProperty>
39. <owl:ObjectProperty rdf:ID=”&vin;hasVineyard”>
40.   <rdfs:subPropertyOf rdf:resource=”&vin;hasSubArea”/>
41.   <rdfs:range rdf:resource=”&vin;Vineyard”/>
42. </owl:ObjectProperty>
43. <owl:ObjectProperty rdf:ID=”&vin;vineyardRegion”>
44.   <owl:inverseOf rdf:resource=”&vin;has Vineyard”/>
45.   <rdfs:range rdf:resource=”&vin;Region”/>
46.   <rdf:type rdf:resource=”&owl;FunctionalProperty”/>
47. </owl:ObjectProperty>
```

Αντιστοίχιση σε Τριάδες

```
48. vin:hasSubArea rdf:type rdf:Property.
49. vin:hasSubArea rdf:type owl:TransitiveProperty.
50. vin:hasSubArea owl:inverseOf vin:hasSubArea.
51. vin:hasRegion rdfs:subPropertyOf vin:hasSubArea.
52. vin:hasRegion owl:range vin:Region.
53. vin:regionOf owl:inverseOf vin:hasRegion.
54. vin:regionOf owl:range vin:Country.
55. vin:regionOf rdf:type owl:FunctionalProperty
56. vin:hasSubRegion rdfs:subPropertyOf vin:hasSubArea.
57. vin:hasSubRegion owl:range vin:Region.
58. vin:subRegionOf owl:inverseOf vin:hasSubRegion.
59. vin:subRegionOf owl:range vin:Region.
60. vin:subRegionOf rdf:type owl:FunctionalProperty
61. vin:hasVineyard rdfs:subPropertyOf vin:hasSubArea.
62. vin:hasVineyard owl:range vin:Vineyard.
63. vin:vineyardRegion owl:inverseOf vin:hasVineyard.
64. vin:vineyardRegion owl:range vin:Region.
65. vin:vineyardRegion rdf:type owl:FunctionalProperty
```

Εξήγηση Κλάσεων

Οι πιο βασικές έννοιες μιας οντολογίας πρέπει να ανήκουν σε κλάσεις, που ανήκουν σε ιεραρχικά δένδρα. Κάθε μονάδα (individual) στην OWL, είναι αντικείμενο της κλάσης owl:Thing**,** που είναι υπερκλάση όλων**.** Ο ορισμός κάθε κλάσης έχει δύο τμήματα: τη δήλωση του ονόματος της κλάσης και την λίστα των περιορισμών της.

Η OWL αναγνωρίζει μόνο τις κλάσεις που ορίζονται από δηλώσεις παρόμοιες με εκείνη της γραμμής 2, με το όνομα της κλάσης να δίνεται μετά τη δήλωση rdf:ID. Τυπικά, μετά από

μια τέτοια δήλωση, δεν γνωρίζουμε τίποτα άλλο για την κλάση, εκτός από την ύπαρξή της. Το όνομα μιας κλάσης μπορεί να είναι προσδιοριστικό του περιεχομένου της για έναν άνθρωπο, για την OWL όμως, δεν προσφέρει καμιά επιπλέον πληροφορία. Παρά το γεγονός ότι μια κλάση δημιουργείται, δεν σημαίνει απαραίτητα πως έχει μέλη (αντικείμενα). Σημειώνεται εδώ, ότι σε ένα OWL έγγραφο, οι δηλώσεις των κλάσεων δεν είναι απαραίτητο να είναι συγκεντρωμένες σε ένα σημείο. Η δήλωση του ονόματος μιας κλάσης (μετά την rdf:ID) μπορεί να γίνει με χρήση της πλήρους διεύθυνσης ή και με χρήση μιας τμηματικής, αν η βασική διεύθυνση έχει οριστεί στην αρχή του προγράμματος. Στην συγκεκριμένη περίπτωση, αυτό γίνεται για τη δήλωση της κλάσης vin**,** με τις δύο εντολές**:**

```
<!ENTITY vin "http://www.w3.org/TR/2003/CR-owl-guide-
20030818/wine#">
<rdf:RDF xmls:vin="http://www.w3.org/TR/2003/CR-owl-guide-
20030818/">
```

Έτσι, η δήλωση &vin;ProductionArea αντιστοιχεί στην πλήρη διεύθυνση http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine#ProductionArea. Στη γραμμή 3, έχουμε τον περιορισμό ότι η κλάση Country είναι υποκλάση της κλάσης ProductionArea**.** Αυτή η ιδιότητα συσχετίζει μια πιο ειδική οντότητα με μια πιο γενική. Σημειώνεται τέλος ότι, όπως είναι γνωστό, η ιδιότητα της υποκλάσης είναι μεταβατική. Έτσι, αν η X είναι υποκλάση της Y και η Y υποκλάση της Z, τότε και η X είναι υποκλάση της Z.

Εξήγηση Τριάδων

Οι τριάδες των γραμμών 11-14 και 48-65 προσδιορίζουν με σαφήνεια τις σχέσεις που συνδέουν τις ιδιότητες και τις κλάσεις της συγκεκριμένης οντολογίας. Ιδιαίτερη αναφορά γίνεται στη δήλωση της γραμμής 11, όπου δηλώνεται ότι το υποκείμενο (vin:ProductionArea) είναι ένας πόρος που αποτελεί στιγμιότυπο της γενικής κλάσης owl:Class (αντικείμενο).

Εξήγηση Ιδιοτήτων

Μια ιδιότητα είναι μια δυαδική σχέση. Στην OWL υπάρχουν δύο τύποι ιδιοτήτων:

- οι ιδιότητες τύπου δεδομένων (datatype properties), δηλαδή σχέσεις μεταξύ των στιγμιότυπων των κλάσεων και των τύπων δεδομένων της RDF ή της XML.
- οι ιδιότητες αντικειμένων, δηλαδή σχέσεις μεταξύ των στιγμιότυπων δύο κλάσεων.

Η δήλωση μιας ιδιότητας και το όνομά της γίνονται με μια δήλωση παρόμοια με εκείνη της γραμμής 15, όπου έχουμε τον ορισμό της ιδιότητας hasSubArea. Κατά των ορισμό μιας ιδιότητας, υπάρχουν πολλοί τρόποι για να τεθούν οι απαραίτητοι περιορισμοί. Η OWL παρέχει τη δυνατότητα να οριστούν πεδία ορισμού και τιμών. Ακόμη, μια ιδιότητα μπορεί να οριστεί ως υπο-ιδιότητα μιας άλλης. Έτσι, στη γραμμή 23, η δήλωση rdfs:domain καθορίζει ότι το πεδίο ορισμού της ιδιότητας hasRegion είναι το σύνολο των μελών της κλάσης Region**.** Η δήλωση στη γραμμή 28, ορίζει ως πεδίο τιμών της regionOf το σύνολο τιμών της Region**.** Και η δήλωση της γραμμής 22, ορίζει ότι η ιδιότητα hasRegion είναι υπο-ιδιότητα της hasSubArea**.**

Γίνεται λοιπόν φανερό, πως οι ιδιότητες, όπως και οι κλάσεις, μπορούν κι εκείνες να οργανωθούν σε ιεραρχία, με τη χρήση των παραπάνω περιορισμών. Τέλος, η ιδιότητα inverseOf της γραμμής 27 δηλώνει ότι οι ιδιότητες regionOf και hasRegion είναι αντίστροφες.

Cardinality

Η ιδιότητα owl:cardinality επιτρέπει τον ακριβή ορισμό του αριθμού των στοιχείων σε μια ιδιότητα. Όταν χρησιμοποιείται μια δήλωση της μορφής:

<owl:cardinality>1</owl:cardinality>

σημαίνει ότι η συγκεκριμένη ιδιότητα αυτής της κλάσης παίρνει ακριβώς *μία* τιμή.

Επίσης, οι δηλώσεις owl:maxCardinality και owl:minCardinality χρησιμοποιούνται για τον ορισμό ανώτατου και κατώτατου ορίου αντίστοιχα.

2.4 Λογική

Βασικό συστατικό του Σημασιολογικού Διαδικτύου πέρα από την μέθοδο αναπαράστασης της γνώσης είναι και ένας μηχανισμός που να επιτρέπει την επεξεργασία της γνώσης αυτής. Αυτός ο μηχανισμός θα πρέπει να υποστηρίζει την δυνατότητα λογικής επεξεργασίας των πληροφοριών με σκοπό την εξαγωγή συμπερασμάτων, την δημιουργίας νέας γνώσης, την υποστήριξη στην λήψη αποφάσεων και την αυτόματη εκτέλεση ενεργειών. Ο μηχανισμός αυτός κάνει χρήση κυρίως της λογικής (logic).

Η λογική είναι το πεδίο που μελετά τις αρχές της συλλογιστικής και έχει τις ρίζες της στον Αριστοτέλη. Σε γενικές γραμμές η λογική προσφέρει μία τυπική γλώσσα για την αναπαράσταση της γνώσης. Επιπλέον, παρέχει μία τυπική σημασιολογία (semantics) καθώς και την δυνατότητα συναγωγής συμπερασμάτων από την υπάρχουσα γνώση.

Έστω για παράδειγμα ότι γνωρίζουμε τα εξής:

«όλοι οι καθηγητές είναι μέλη του διδακτικού προσωπικού μίας σχολής»,

«όλα τα μέλη διδακτικού προσωπικού είναι μέλη του γενικού προσωπικού μίας σχολής» και

«ο Γιώργος είναι καθηγητής».

Στην κατηγορηματική λογική αυτή η γνώση θα είχε την εξής αναπαράσταση:

$prof(X) \rightarrow faculty(X)$

$faculty(X) \rightarrow staff(X)$

$prof(george)$

Από τα παραπάνω μπορούν να συναχθούν τα εξής συμπεράσματα:

$faculty(george)$

$staff(george)$

$prof(X) \rightarrow staff(X)$

Ένα από τα πιο σημαντικά πλεονεκτήματα της λογικής είναι ότι παρέχει την δυνατότητα αιτιολόγησης των συμπερασμάτων δηλαδή είναι δυνατόν να εντοπιστούν τα επιμέρους βήματα της συναγωγής ενός συμπεράσματος και να αιτιολογηθούν με βάση την υπάρχουσα γνώση.

2.4.1 Προτασιακή λογική

Η προτασιακή λογική (propositional logic) είναι το πιο απλό σύστημα συμβολικής λογικής. Η προτασιακή λογική ασχολείται με προτάσεις που αποφαίνονται γιά κάτι, δηλαδή κάνουν κάποιες διαπιστώσεις. Οι πιο απλές προτάσεις ονομάζονται ατομικές προτάσεις ή άτομα (atomic propositions, atoms). Οι προτάσεις αυτές συνδεόμενες με τους λογικούς συνδέσμους (logical connectives) συνθέτουν τις σύνθετες προτάσεις (compound propositions). Οι λογικοί σύνδεσμοι είναι : ¬ (not, άρνηση), ∧ (and, και, σύζευξη), ∨ (or, ή, διάζευξη), → (implies, συνεπαγωγή), ↔ (equivalent, ισοδυναμία).

Η αναπαράσταση της γνώσης στην προτασιακή λογική είναι πολύ στοιχειώδης : Βασίζεται στην ανάθεση κάποιων προτάσεων της φυσικής (ή άλλης) γλώσσας σε ατομικές προτάσεις, και η χρησιμοποίηση συνδέσμων για την σύνθεση προτάσεων που αφορούν πιο σύνθετα φαινόμενα. Η αναπαράσταση της γνώσης βέβαια στην προτασιακή λογική συναρτάται άμεσα με την σημασιολογία (semantics), η οποία με την σειρά της συναρτάται με την ερμηνεία (interpretation) των προτάσεων της προτασιακή λογική. Η μόνη δυνατή ερμηνεία μιας πρότασης της προτασιακή λογική είναι η εκτίμησή της (valuation), η απονομή δηλαδή μιας\ τιμής αλήθειας (truth value) στην πρόταση. Η εκτίμηση μιας πρότασης στη προτασιακή λογική ορίζεται με βάση τις εκτιμήσεις των ατομικών προτάσεων λαμβάνοντας υπόψη τους πίνακες αλήθειας (truth tables) των λογικών συνδέσμων. Οι πίνακες αλήθειας των λογικών συνδέσμων ορίζονται παρακάτω. Υπάρχουν δύο μόνο τιμές αλήθειας : αληθές (true) και ψευδές (false). Οι τιμές αυτές αναπαρίστανται και σαν 1 ή t, και 0 ή f αντίστοιχα.

I(p)	I(q)	I(¬p)	I(p∧q)	I(p∨q)	I(p→q)	I(p↔q)
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Πίνακας 1: Πίνακες αλήθειας των λογικών συνδέσμων

Η ερμηνεία μιας πρότασης στην προτασιακή λογική

Δεδομένης μιας πρότασης φ, έστω p1,p2,..,pn, τα άτομα της πρότασης. Μια ερμηνεία I της φ είναι μια ανάθεση τιμών αλήθειας σε κάθε άτομο pi, i=1,...,n. Η ανάθεση μιας τιμής αλήθειας ενός ατόμου p στην I συμβολίζεται με I(p)=1 (ανάθεση αληθούς τιμής), I(p)=0 (ανάθεση ψευδούς τιμής).

Η αποτίμηση της τιμής αλήθειας μιας πρότασης στην προτασιακή λογική

Η πρόταση φ είναι αληθής κάτω από μια ερμηνεία I εάν η φ αποτιμάται ως αληθής στην I. Αλλιώς η φ είναι ψευδής στην ερμηνεία I.

Παράδειγμα

Ας θεωρήσουμε την πρόταση (p∧q)→(r↔¬s). Η πρόταση διαθέτει 4 άτομα, οπότε υπάρχουν 16 ερμηνείες. Για κάθε ερμηνεία η πρόταση έχει τιμή αλήθεια ή ψέμα. Με βάση τον πίνακα αλήθειας μπορούμε να βρούμε ποιες ερμηνείες ικανοποιούν, δίνουν εκτίμηση 1, στην πρόταση.

I(p)	I(q)	I(r)	I(s)	I(¬s)	I(p∧q)	I(r↔¬s)	I((p∧q)→(r↔¬s))
0	0	0	0	1	0	0	1
0	0	0	1	0	0	1	1
0	0	1	0	1	0	1	1
0	0	1	1	0	0	0	1
0	1	0	0	1	0	0	1
0	1	0	1	0	0	1	1
0	1	1	0	1	0	1	1
0	1	1	1	0	0	0	1
1	0	0	0	1	0	0	1
1	0	0	1	0	0	1	1
1	0	1	0	1	0	1	1
1	0	1	1	0	0	0	1
1	1	0	0	1	1	0	0
1	1	0	1	0	1	1	1
1	1	1	0	1	1	1	1
1	1	1	1	0	1	0	0

Πίνακας 2: Πίνακας αληθείας της πρότασης (p∧q)→(r↔¬s)

2.4.2 Κατηγορηματική λογική

Η προτασιακή λογική είναι πολύ στοιχειώδες μέσον για την αναπαράσταση και επεξεργασία της γνώσης, γιατί έχει πολύ περιορισμένες δυνατότητες. Για παράδειγμα, από τις προτάσεις "σε κάθε νησί μπορείς να πας με πλοίο", "η Κέρκυρα είναι νησί", εύκολα συμπεραίνουμε ότι "στην Κέρκυρα μπορείς να πας με πλοίο". Η προτασιακή λογική όμως δεν έχει τρόπο να αναπαραστήσει την γνώση των απλών αυτών προτάσεων και να εφαρμόσει συμπερασματολογία ακόμα και σε τόσο απλές προτάσεις. Ο λόγος είναι (η εννοούμενη από την φυσική γλώσσα) ύπαρξη καθολικού ποσοδείκτη (κάθε νησί...) που ταυτόχρονα προϋποθέτει την ύπαρξη μεταβλητών. Αντίθετα, στη κατηγορηματική λογική α' τάξης (First Order Predicate Logic, FOPL) μπορούμε να αναπαραστήσουμε γνώση που αναφέρεται σε ένα σύνολο

αντικειμένων και στις σχέσεις τους, και να εξετάσουμε ως προς την αλήθεια τους προτάσεις για αυτά τα αντικείμενα.

Για να γράψουμε προτάσεις στην προτασιακή λογική πρέπει να εισάγουμε με αυστηρό τρόπο τις εξής τρεις έννοιες, τους όρους (terms), τα κατηγορηματικά σύμβολα (predicate symbols, predicates), και τους ποσοδείκτες (quantifiers).

- ***Όρος (term)***

Ένας όρος ορίζεται ως εξής :

- Μια σταθερά είναι ένας όρος.

- Μια μεταβλητή είναι ένας όρος.

- Εάν f είναι ένα συναρτησιακό σύμβολο n-ορισμάτων, και τ1,τ2,...,τn είναι όροι, τότε το f(τ1,τ2,...,τn) είναι επίσης όρος.

- ***Άτομο - Κατηγόρημα***

Εάν p είναι ένα κατηγορηματικό σύμβολο n-θέσεων και t1,t2,...,tn είναι όροι τότε το p(t1,t2,...,tn) είναι άτομο ή κατηγόρημα.

Για παράδειγμα, τα παρακάτω είναι κατηγορήματα:

άνθρωπος(πέτρος), project(leader(X), programmer(peter),secretary(helen))

- ***Ποσοδείκτες***

Οι ποσοδείκτες είναι δύο σύμβολα, το ∃ (Exists, Υπαρχει) και το ∀ (All, για κάθε) που ονομάζονται αντίστοιχα υπαρξιακός και καθολικός ποσοδείκτης.

Έτσι αν γράψουμε (∃X) άνθρωπος(X)... εννοούμε ότι '*υπάρχει τουλάχιστον ένας άνθρωπος που* ...' , ενώ αν γράψουμε (∀X) άνθρωπος(X)... εννοούμε *ότι 'για κάθε άνθρωπο που* ...' . Όλα τα παραπάνω τα συνδυάζουμε κατάλληλα ώστε να σχηματίσουμε προτάσεις της κατηγορηματικής λογικής, εκ των οποίων μερικά χαρακτηριστικά παραδείγματα είναι και τα παρακάτω:

(∀X) (άνθρωπος(X) → θνητός(X))

(∃X) (άνθρωπος(X) ∧ έλληνας(X))

(∀X) (φοιτητής(X) → (∃Z) (πανεπιστήμιο(Z) ∧ πηγαίνει(X,Z)))

Μία μορφή προτάσεων της κατηγορηματικής λογικής, που χρησιμοποιείται από την Prolog όπως θα δούμε παρακάτω, είναι και οι προτάσεις clause ή φράσεις. Μια φράση είναι μια πεπερασμένη ακολουθία από μηδέν ή περισσότερα λεκτικά σε διάζευξη, όπου ένα λεκτικό είναι ένα θετικό ή αρνητικό άτομο.

Μια φράση τύπου Horn είναι μια φράση που έχει ένα το πολύ θετικό λεκτικό, όπως τα παραδείγματα που παρουσιάζονται παρακάτω:

¬a(X,S) ∨ ¬a(X,T) ∨ a(X, τ(S,T))

¬a(f(S,T),T) ∨ u(S,T)

2.4.3 Λογικός προγραμματισμός

Κατά μια πρώτη προσέγγιση, ο λογικός προγραμματισμός είναι μια υπολογιστική κωδικοποίηση που συνδυάζει δύο κύριες αρχές:

- χρησιμοποιεί λογική για να αναπαραστήσει τη γνώση,
- χρησιμοποιεί το μηχανισμό των αποδείξεων και των συμπερασμάτων για να επεξεργαστεί γνώση.

Στο πλαίσιο επίλυσης ενός προβλήματος, η πρώτη αρχή αφορά την αναπαράσταση υποθέσεων και συμπερασμάτων, ενώ η δεύτερη, αφορά τον ορισμό και την παραγωγή των λογικών σχέσεων μεταξύ υποθέσεων και συμπερασμάτων. Ο γενικός σκοπός σε κάθε τέτοιο πλαίσιο είναι να συνάγεται το επιθυμητό αποτέλεσμα από τις δεδομένες υποθέσεις, με έναν τρόπο που είναι υπολογιστικά βιώσιμος.

Ο λογικός προγραμματισμός μπορεί να θεωρηθεί ως ο ακρογωνιαίος λίθος του προγραμματισμού με βάση τη γνώση (knowledge-based programming). Η εκφραστικότητα της λογικής δίνει την δυνατότητα κωδικοποίησης υποθέσεων στο πλαίσιο του προβλήματος με έναν άμεσο τρόπο ανεξάρτητο της υλοποίησης. Αντίστροφα, μπορεί κανείς να αποκωδικοποιήσει μια τέτοια κωδικοποίηση με σκοπό να επανακτήσει τις υπονοούμενες υποθέσεις. Αυτό είναι και το βασικό πλεονέκτημά του, το ότι δηλαδή δίνει την δυνατότητα σε κάποιον να εκφράσει τη γνώση αποκλειστικά με έναν τρόπο ανεξάρτητο της μηχανής, φτιάχνοντας προγράμματα πιο συμπαγή, ευέλικτα και κατανοητά.

Ένα πρόγραμμα λογικής είναι ένα σύνολο από αξιώματα, ή κανόνες, τα οποία ορίζουν σχέσεις μεταξύ αντικειμένων. Οι βασικές δομές του λογικού προγραμματισμού είναι δύο: οι *όροι (terms)* και οι *δηλώσεις (statements)*. Υπάρχουν τρία διαφορετικά είδη δηλώσεων: τα γεγονότα (facts), τα ερωτήματα (queries) και οι κανόνες (rules), ενώ υπάρχει μία μοναδική δομή, ο λογικός όρος (logical term).

- **Γεγονότα**

Το πιο απλό είδος δήλωσης είναι το *γεγονός (fact)*. Ένα γεγονός είναι ένας τρόπος με τον οποίο δηλώνεται η ύπαρξη μίας σχέσης (relation) μεταξύ αντικειμένων. Για παράδειγμα: `father(abraham, isaac)`. Το γεγονός αυτό δηλώνει ότι ο abraham είναι πατέρας του isaac ή αλλιώς ότι μεταξύ των abraham και isaac υπάρχει η σχέση 'πατέρας'. Μία εναλλακτική ονομασίας της σχέσης είναι ο όρος *κατηγόρημα (predicate)*. Τα ονόματα των μεμονωμένων στοιχείων λέγονται άτομα (atoms). Ένα πεπερασμένο σύνολο από γεγονότα αποτελεί ένα πρόγραμμα (program), την απλούστερη μορφή ενός προγράμματος λογικής.

- **Ερωτήματα**

Μία δεύτερη μορφή δήλωσης είναι ένα *ερώτημα (query)*. Ένα ερώτημα ελέγχει αν υπάρχει μία συγκεκριμένη σχέση μεταξύ δύο αντικειμένων και αποτελεί τρόπο ανάκτησης πληροφοριών από ένα πρόγραμμα. Για παράδειγμα, το ερώτημα `father(abraham,`

`isaac)?` ελέγχει αν υπάρχει η σχέση 'πατέρας' μεταξύ των abraham και isaac. Ένα γεγονός τελειώνει με μία τελεία (.) και ένα ερώτημα με ένα ερωτηματικό (?). Η οντότητα χωρίς την τελεία ή το ερωτηματικό καλείται στόχος (goal). Δηλαδή, ένα γεγονός P. δηλώνει ότι ο στόχος P είναι αληθής ενώ ένα ερώτημα P? ελέγχει αν ο στόχος P αληθεύει ή όχι. Η απάντηση ερωτημάτων χρησιμοποιώντας ένα πρόγραμμα που περιέχει γεγονότα προκαλεί αναζήτηση του προγράμματος για ένα γεγονός που υπονοεί ή είναι όμοιο με το ζητούμενο οπότε επιστρέφει *yes* ή διαφορετικά επιστρέφει *no*.

- **Κανόνες**

Η τρίτη μορφή δήλωσης είναι ο *κανόνας (rule)*. Ένας κανόνας αποτελεί μία γενική δήλωση σχετικά με τα αντικείμενα και τις μεταξύ τους σχέσεις και χρησιμοποιείται όταν θέλουμε να δηλώσουμε ότι ένα γεγονός εξαρτάται από ένα σύνολο γεγονότων. Ένας κανόνας στην Prolog αποτελείται από την κεφαλή (head) και το σώμα (body). Η κεφαλή περιλαμβάνει το γεγονός το οποίο ορίζει ο κανόνας και το σώμα περιέχει την σύζευξη των γεγονότων που πρέπει να ισχύουν για να είναι αληθής η κεφαλή και να ικανοποιείται ο κανόνας. Για παράδειγμα, ο παρακάτω κανόνας ο οποίος ορίζει ότι η X είναι αδελφή του Y αν η X είναι γυναίκα και έχουν τους ίδιους γονείς:

```
sister_of(X,Y):-
      female(X),
      parents(X,M,F),
      parents(Y,M,F).
```

Η ερμηνεία (meaning) *M(P)*, ενός λογικού προγράμματος *P*, είναι το σύνολο των στοιχειωδών μοναδιαίων φράσεων, δηλαδή στοιχειωδών κατηγορημάτων, που συμπεραίνονται από το *P*. Ο βασικός τρόπος χρήσης ενός λογικού προγράμματος είναι με τον ορισμό μίας βάσης γνώσης η οποία δεν είναι τίποτα παραπάνω από ένα σύνολο γεγονότων και κανόνων.

2.4.4 Το μοντέλο εκτέλεσης της Prolog

Η Prolog βασίζεται στην κατηγορηματική λογική α΄ επιπέδου (FOPL), η οποία και περιγράφτηκε στην προηγούμενη ενότητα, χρησιμοποιεί όμως προτάσεις σε μορφή clause ή φράσεις (clausal form) και πιο συγκεκριμένα φράσεις σε μορφή Horn (Horn clause form).

Η Prolog απαντά στα ερωτήματα που υποβάλλει ο χρήστης προσπαθώντας να ικανοποιήσει μία σύζευξη στόχων (goals) που το εκάστοτε ερώτημα περιλαμβάνει. Ένας στόχος μπορεί να ικανοποιηθεί είτε άμεσα με ενοποίηση με ένα γεγονός (fact) είτε να αναχθεί στην ικανοποίηση υπο-στόχων μέσω ενοποίησης με ένα κανόνα (rule). Κάθε φορά που ένας στόχος δεν μπορεί να ικανοποιηθεί η Prolog εκτελεί οπισθοδρόμηση (backtracking), δηλαδή επιστροφή σε κάποιους στόχους που έχουν ήδη ικανοποιηθεί προηγουμένως με σκοπό να τους ικανοποιήσει εκ νέου.

Παρακάτω παρατίθεται ένα παράδειγμα για την πληρέστερη κατανόηση του μηχανισμού εκτέλεσης της Prolog. Θεωρώντας τον κανόνα του παραπάνω παραδείγματος `sister_of` ζητάμε από την Prolog να μάθουμε αν η mary είναι αδερφή του john. Το ερώτημα που πρέπει να απαντηθεί είναι: `sister_of(mary,john).`

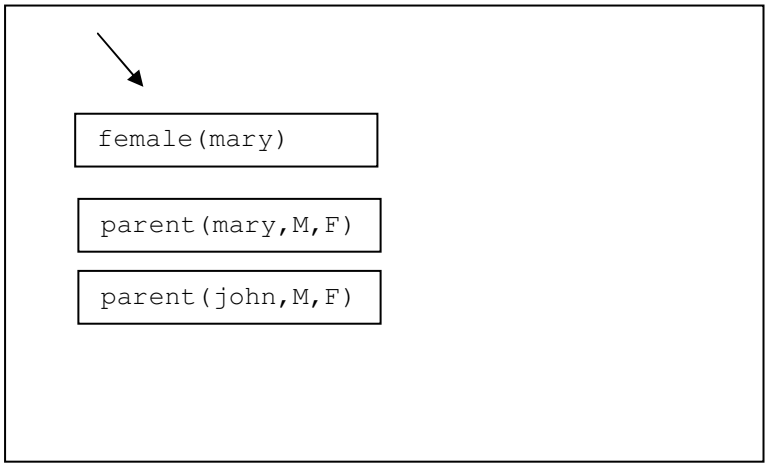
Έστω ότι στην βάση γνώσης υπάρχουν οι εξής κανόνες και γεγονότα:

```
parent(C,M,F) :-
      mother(C,M), father(C,F).
mother(john,ann).
mother(mary,ann).
father(mary,fred).
father(john,fred).
```

Με την ενοποίηση των μεταβλητών X, Y με τα mary, john η απάντηση του ερωτήματος ανάγεται στην ικανοποίηση των εξής στόχων:

```
female(mary),
parents(mary,M,F),
parents(john,M,F).
```

Η ροή ικανοποίησης των στόχων παρουσιάζεται βήμα προς βήμα στα Σχήματα 4.1- 4.3:

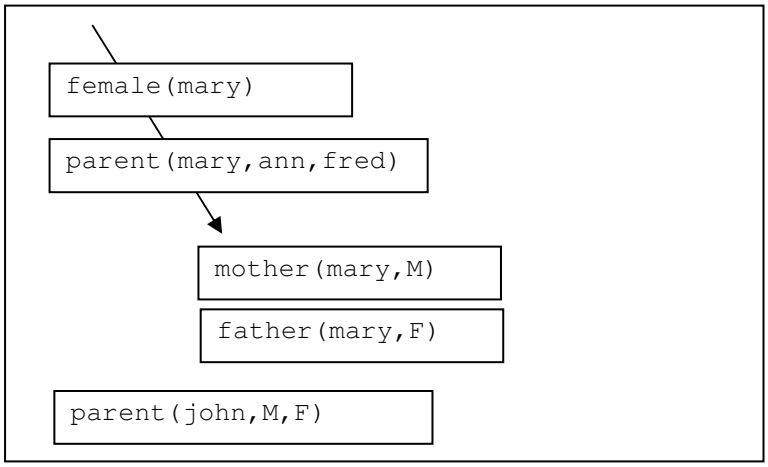


Σχήμα 4: Τυπική ροή ικανοποίησης στόχων από τον μηχανισμό εκτέλεσης της Prolog

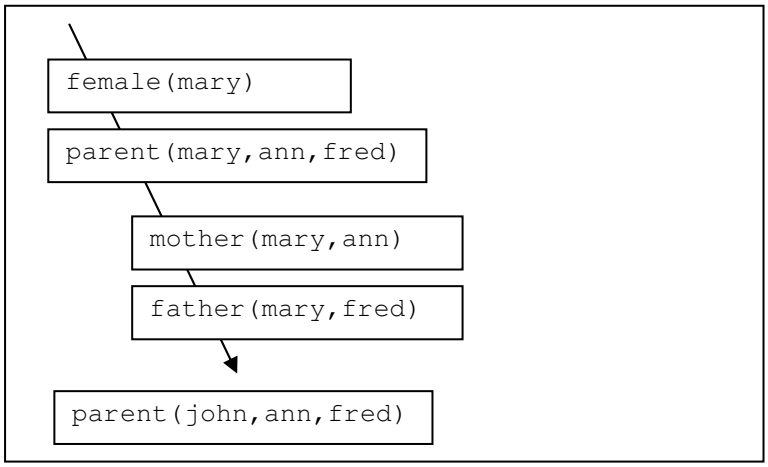
Η ικανοποίηση ενός στόχου περιλαμβάνει αναζήτηση στην βάση γνώσης για γεγονότα ή κανόνες που ταιριάζουν, στην συνέχεια μαρκάρισμα του σημείου στην βάση γνώσης και συνέχεια με την ικανοποίηση υπο-στόχων που τυχόν υπάρχουν.

Μετά την ικανοποίηση του στόχου `female(mary)` ο στόχος `parent(mary,M,F)` ικανοποιείται με ενοποίηση των `M` με `ann` και `F` με `fred` οπότε ανάγεται στην ικανοποίηση των υπο-στόχων `mother(mary,M)` και `father(mary,F)` όπως φαίνεται στο Σχήμα 4.1.

Οι υποστόχοι αυτοί ικανοποιούνται από τα γεγονότα της βάσης γνώσης `mother(mary,ann)` και `father(mary,fred).`, οπότε και η ροή εκτέλεσης οδηγείται στον στόχο `parent(john,M,F)` όπως παρουσιάζεται και στο Σχήμα 4.2.



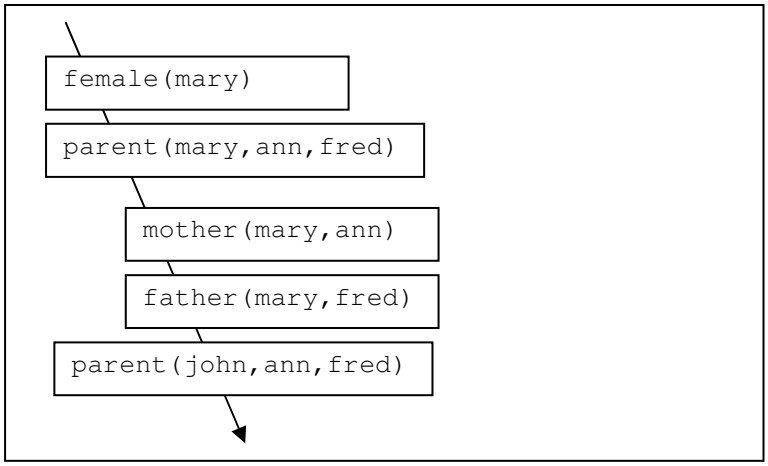
Σχήμα 4.1: Πρώτο βήμα εκτέλεσης



Σχήμα 4.2: Δεύτερο βήμα εκτέλεσης

Η ικανοποίηση αυτού του στόχου ανάγεται στην ικανοποίηση των υπο-στόχων `mother(john,ann)`, `father(john,fred)`, οι οποίοι και ικανοποιούνται από τα αντίστοιχα γεγονότα της βάσης γνώσης, οπότε το ερώτημα επιτυγχάνει όπως φαίνεται και στο Σχήμα 4.3.

Όπως αναφέρθηκε παραπάνω, όταν ένας στόχος αποτύχει να ικανοποιηθεί, τότε ο μηχανισμός εκτέλεσης οπισθοδρομεί στο αμέσως προηγούμενο σημείο όπου ικανοποιήθηκε ένας κανόνας με σκοπό να το ικανοποιήσει εναλλακτικά. Όλες οι μεταβλητές του γίνονται πάλι ανεξάρτητες και η Prolog αρχίζει να ψάχνει την βάση γνώσης από το σημείο εκείνο και μετά για γεγονότα ή κανόνες που ενοποιούνται με τον ζητούμενο στόχο. Αν βρεθεί κάποιος η εκτέλεση συνεχίζεται κανονικά. Αν υπάρξει πάλι αποτυχία τότε γίνεται άλλη μία οπισθοδρόμηση ένα βήμα πιο πίσω. Στο παράδειγμά μας, αν ο στόχος `parent(john,ann,fred)` αποτύγχανε τότε η Prolog θα οπισθοδρομούσε στον `parent(mary,ann,fred)` επιχειρώντας να τον ικανοποιήσει εναλλακτικά.



Σχήμα 4.3: Τελικό βήμα εκτέλεσης

3. Σχεδιασμός και Υλοποίηση Συστήματος

3.1 Δομή και περιεχόμενα του RDF εγγράφου

3.1.1 Το RDF έγγραφο

Όπως αναφέρθηκε και στην περιγραφή της παρούσας εργασίας, το σύστημα που κατασκευάστηκε πραγματοποιεί αναζητήσεις πάνω σε rdf δεδομένα τα οποία εξάγονται δυναμικά από μία βάση δεδομένων Access. Το RDF έγγραφο που παράγεται περιέχει όλες τις δημοσιεύσεις και τους συγγραφείς που είναι καταχωρημένοι στη βάση δεδομένων με τη μορφή πόρων των κλάσεων που έχουν οριστεί στο RDF Schema. Το RDF έγγραφο βρίσκεται στο Διαδίκτυο στην διεύθυνση http://lpis.csd.auth.gr/rdf-publications.asp .

Για παράδειγμα, η καταχώρηση της δημοσίευσης *Tsoumakas G., Vrakas D., Bassiliades N., Vlahavas I., "Using the k-nearest problems for adaptive multicriteria planning", 3rd Hellenic Conference on Artificial Intelligence (SETN '04), G. Vouros and T. Panayiotopoulos, Springer-Verlag, LNAI 3025, 132-141, Samos, Greece, 2004* έχει ως εξής:

```
<LPISPubs:ConferenceProceedings  rdf:ID="entry146">
      <LPISPubs:publicationTitle>Using the k-nearest problems for
      adaptive multicriteria planning</LPISPubs:publicationTitle>
      <LPISPubs:authors rdf:parseType="Collection">
            <rdf:Description rdf:about="http://lpis.csd.auth.gr/rdf-
                  publications.asp#author6"/>
            <rdf:Description rdf:about="http://lpis.csd.auth.gr/rdf-
                  publications.asp#author8"/>
            <rdf:Description rdf:about="http://lpis.csd.auth.gr/rdf-
                  publications.asp#author9"/>
            <rdf:Description rdf:about="http://lpis.csd.auth.gr/rdf-
                  publications.asp#author2"/>
      </LPISPubs:authors>
      <LPISPubs:mediaTitle>Proc. 3rd Hellenic Conference on Artificial
      Intellligence (SETN '04)</LPISPubs:mediaTitle>
      <LPISPubs:mediaEditors>G. Vouros and T.Panayiotopoulos
      </LPISPubs:mediaEditors>
      <LPISPubs:mediaPublisher>SpringerVerlag
      </LPISPubs:mediaPublisher>
      <LPISPubs:mediaVolInfo>LNAI 3025</LPISPubs:mediaVolInfo>
      <LPISPubs:publicationPagesInMedium>132-141
      </LPISPubs:publicationPagesInMedium>
      <LPISPubs:publicationLocation>Samos,Greece
      </LPISPubs:publicationLocation>
      <LPISPubs:publicationYear
      rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2004
      </LPISPubs:publicationYear>
      <LPISPubs:publicationNoOfPages
            rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">10
      </LPISPubs:publicationNoOfPages>
      <LPISPubs:publicationFileName>tsoumakas setn04.pdf
      </LPISPubs:publicationFileName>
      <LPISPubs:publicationAbstract></LPISPubs:publicationAbstract>
```

```
      <LPISPubs:publicationPubURL></LPISPubs:publicationPubURL>
      <LPISPubs:keywords>
            <rdf:Seq>
                  <rdf:_1>Machine Learning</rdf:_1>
                  <rdf:_2>Planning</rdf:_2>
                  <rdf:_3>Prediction</rdf:_3>
            </rdf:Seq>
      </LPISPubs:keywords>
</LPISPubs:ConferenceProceedings>
```

Μία δημοσίευση, όπως φαίνεται και παραπάνω, περιλαμβάνει στοιχεία όπως ο τίτλος της (`publicationTitle`), μία λίστα με τους συγγραφείς της (`authors`), τον τίτλο του μέσου δημοσίευσης (`mediaTitle`), τα ονόματα των συγγραφέων του μέσου (`mediaEditors`), των εκδοτών του μέσου (`mediaPublisher`), τον μέσο όρο των σελίδων (`publicationPagesInMedium`), τον τόπο της έκδοσης της δημοσίευσης (`publicationLocation`) και το έτος δημοσίευσης (`publicationYear`).

Επίσης, η καταχώρηση ενός συγγραφέα, για παράδειγμα του *Βασιλειάδη Ν.* με διεύθυνση *http://lpis.csd.auth.gr/people/nbassili/*, είναι η παρακάτω:

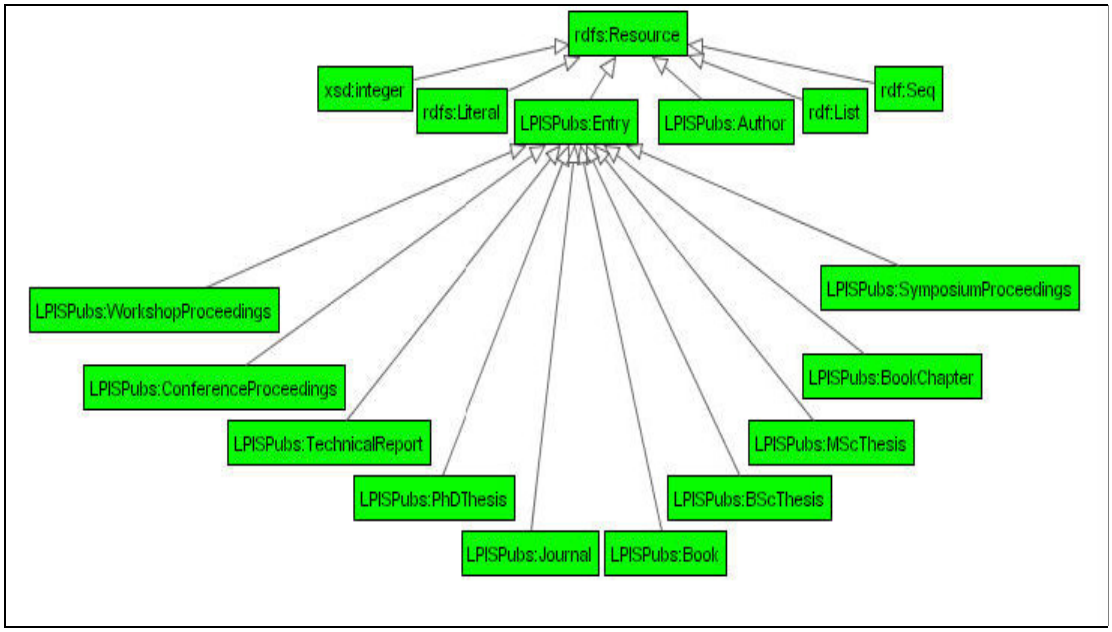
```
<LPISPubs:Author rdf:ID="author9">
      <LPISPubs:authorName>N</LPISPubs:authorName>
      <LPISPubs:authorSurName>Bassiliades</LPISPubs:authorSurName>
      <LPISPubs:authorURL>http://lpis.csd.auth.gr/people/nbassili/
      </LPISPubs:authorURL>
</LPISPubs:Author>
```

3.1.2 Το RDF Schema

Η δομή του RDF εγγράφου βασίζεται σε ένα RDF Schema το οποίο βρίσκεται στην διεύθυνση http://lpis.csd.auth.gr/publications.rdfs#. Στο RDF Schema υπάρχουν δύο βασικές κλάσεις: η κλάση Entry, που αφορά τις δημοσιεύσεις και η κλάση Author, η οποία περιλαμβάνει τους συγγραφείς και τα στοιχεία τους. Η κλάση Entry έχει δέκα υποκλάσεις, κάθε μία από τις οποίες αντιστοιχεί σε διαφορετικό είδος δημοσίευσης. Ο ορισμός αυτών των υποκλάσεων έγινε με βάση τις τιμές του πεδίου Media Type, που περιγράφει τα διαφορετικά είδη δημοσιεύσεων στη βάση δεδομένων. Στο Σχήμα 5 φαίνεται η ιεραρχία των κλάσεων και των υποκλάσεων του RDF Schema.

Τα πεδία που περιγράφουν μια δημοσίευση στη βάση δεδομένων από όπου έγινε η εξαγωγή των rdf δεδομένων έγιναν ιδιότητες με πεδίο ορισμού την κλάση Entry. Οι ιδιότητες που υπάρχουν στο RDF Schema παρέχουν τη δυνατότητα για την επαρκή περιγραφή μιας δημοσίευσης αλλά και των στοιχείων ενός συγγραφέα. Ο ορισμός των ιδιοτήτων είναι απλός

και οι περισσότερες από αυτές αφορούν τις δημοσιεύσεις. Αυτό δηλώνεται από το πεδίο



Σχήμα 5:Η οργάνωση των κλάσεων στο RDF Schema

ορισμού τους που στην προηγούμενη περίπτωση ορίζεται να είναι η κλάση Entry, ενώ για τις ιδιότητες που αφορούν τους συγγραφείς το πεδίο ορισμού είναι αντίστοιχα η κλάση Author. Όσον αφορά τα πεδία τιμών των ιδιοτήτων (range), στις περισσότερες περιπτώσεις αυτά είναι αλφαριθμητικά (rdfs:Literal). Εξαίρεση αποτελούν κάποιες ιδιότητες, όπως ο χρόνος έκδοσης (PublicationYear), που ορίζονται ως ακέραιοι αριθμοί (xsd:integer). Επίσης, υπάρχει και η ιδιότητα authors, που είναι ιδιότητα της κλάσης των δημοσιεύσεων και έχει ως πεδίο τιμών μια λίστα (rdf:List), για να εμφανίζει τους συγγραφείς μιας δημοσίευσης διατεταγμένους σε σειρά. Παρόμοια είναι και η περίπτωση των λέξεων-κλειδιών (keywords) που έχει ως πεδίο τιμών μια «ακολουθία» (rdf:Seq) και πάλι για τον ίδιο λόγο. Ο λόγος που χρησιμοποιήθηκαν δύο διαφορετικοί τρόποι για την κατάταξη είναι απλός: και στις δύο περιπτώσεις επιτυγχάνεται η κατά προτεραιότητα κατάταξη, όμως στην πρώτη περίπτωση το σύνολο που παράγεται είναι κλειστό (δεν υπάρχουν άλλοι συγγραφείς εκτός από τους συμπεριληφθέντες), ενώ στη δεύτερη το σύνολο είναι θεωρητικά ανοιχτό, δηλαδή μπορεί κάποιος να προσθέσει και άλλες λέξεις κλειδιά αν το επιθυμεί.

3.2 Απαιτήσεις συστήματος

Στην παρούσα ενότητα προσδιορίζονται οι απαιτήσεις από το προς ανάπτυξη σύστημα οι οποίες κατά κύριο λόγο αφορούν τον προσδιορισμό των κριτηρίων βάσει των οποίων θα πραγματοποιούνται οι αναζητήσεις των επιστημονικών εργασιών στο σύστημα.

Μελετώντας τα δεδομένα προκύπτουν συμπεράσματα σχετικά με το ποια από τα χαρακτηριστικά μίας δημοσίευσης μπορούν να χρησιμοποιηθούν ως κριτήρια των

αναζητήσεων. Οι πιο χαρακτηριστικές πληροφορίες που διατηρούνται για μια επιστημονική εργασία και μπορούν να αποτελέσουν κριτήρια αναζήτησής της από τους χρήστες είναι τα εξής: το/α επίθετο/α του/των συγγραφέων, οι λέξεις κλειδιά, ο τίτλος της, η χρονολογία δημοσίευσης, το όνομα του εκδότη και ο τύπος της δημοσίευσης δηλαδή αν πρόκειται για βιβλίο, δημοσίευση σε επιστημονικό περιοδικό, ανακοίνωση σε συνέδριο, εκπόνηση διδακτορικού κ.λ.π. . Επομένως, μία απλή αναζήτηση επιστημονικής εργασίας θα μπορούσε να γίνει βάσει ενός εκ των παρακάτω κριτηρίων:

- το επίθετο ενός ή περισσοτέρων συγγραφέων που συμμετείχαν στην συγγραφή της
- μία ή περισσότερες λέξεις κλειδιά
- τον πλήρη τίτλο της δημοσίευσης
- την χρονολογία δημοσίευσης
- το όνομα του εκδότη
- τον τύπο της δημοσίευσης

Επιπλέον, πέραν των απλών αναζητήσεων, το σύστημα θα πρέπει να δίνει την δυνατότητα σύνθετων αναζητήσεων βάσει πολλαπλών κριτηρίων. Μία κατηγορία σύνθετης αναζήτησης αφορά τον συνδυασμό των παραπάνω κριτηρίων σε λογική σύζευξη μεταξύ τους, το κάθε ένα με όλα τα υπόλοιπα. Για παράδειγμα, θα μπορούσε να γίνει αναζήτηση μίας εργασίας με βάση το όνομα ενός εκ των συγγραφέων της *και* της χρονολογίας δημοσίευσής της. Επίσης, μία ακόμα κατηγορία σύνθετης αναζήτησης αφορά την χρήση των παραπάνω κριτηρίων σε λογική διάζευξη μεταξύ τους. Για παράδειγμα, αναζήτηση μίας επιστημονικής εργασίας με βάση τον πλήρη τίτλο της *ή* μία λέξη κλειδί που περιλαμβάνεται στον τίτλο της. Σε αυτό το σημείο κρίνονται αναγκαίες δύο διευκρινίσεις: α) στις αναζητήσεις με βάση το επίθετο του συγγραφέα ή συγγραφέων παίζει ρόλο και η σειρά τους στη λίστα συγγραφέων, για παράδειγμα αναζήτηση των εργασιών στις οποίες πρώτο όνομα είναι ο Χ και δεύτερο όνομα ο Υ, και β) στην αναζήτηση με λογική σύζευξη κριτηρίων προβλέπονται όλοι οι συνδυασμοί κριτηρίων εκτός από τις αναζητήσεις με σύζευξη του πλήρους τίτλου μίας εργασίας και λέξεων κλειδιών του τίτλου. Η τελευταία παρατήρηση προκύπτει λογικά διότι δεν έχει νόημα να αναζητηθεί μία εργασία με βάση μία λέξη κλειδί του τίτλου της και ταυτόχρονα με βάση τον πλήρη τίτλο της αφού ο τελευταίος αρκεί για τον προσδιορισμό της.

3. 3 Τεχνικές προδιαγραφές

Η υλοποίηση του παρόντος συστήματος πραγματοποιήθηκε με χρήση της SWI-Prolog 5.6 και των βιβλιοθηκών της που αφορούν αφ' ενός την ανάλυση και επεξεργασία RDF μεταδεδομένων και αφ' ετέρου την δημιουργία και επεξεργασία ιστοσελίδων. Η SWI-Prolog αποτελεί μία υλοποίηση ανοιχτού κώδικα της γλώσσας προγραμματισμού Prolog που

χρησιμοποιείται κυρίως για διδακτικούς σκοπούς. Από το 1987 βρίσκεται σε συνεχή διαδικασία ανάπτυξης και ο κύριος συγγραφέας της είναι ο Jan Wielemaker.

3.3.1 Η βιβλιοθήκη SWI-Prolog Semantic Web (SemWeb library)

Η βιβλιοθήκη SWI-Prolog Semantic Web (SemWeb library) είναι μία βιβλιοθήκη που επιτρέπει στην SWI-Prolog να ενσωματώνει δεδομένα που συμφωνούν με τα πρότυπα του W3C για το Σημασιολογικό Διαδίκτυο. Περιλαμβάνει πακέτα που επιτρέπουν την πραγματοποίηση ανάγνωσης, αναζήτησης και αποθήκευσης εγγράφων του Σημασιολογικού Διαδικτύου.

Κεντρικό δόμημα (module) της βιβλιοθήκης είναι το `rdf_db` το οποίο παρέχει δυνατότητες αποθήκευσης και πραγματοποίησης βασικών ερωτημάτων στις rdf τριπλέτες. Στην παρούσα εφαρμογή χρησιμοποιήθηκαν μόνο οι δυνατότητες πραγματοποίησης ερωτημάτων κυρίως μέσω του κατηγορήματος `rdf(?Subject, ?Predicate, ?Object)` το οποίο πραγματοποιεί στοιχειώδη αναζήτηση τριπλέτων. Τα `Subject` και `Predicate` είναι άτομα που αντιπροσωπεύουν το πλήρες URL του περιγραφόμενου πόρου, ενώ το `Object` είναι είτε ένα άτομο που αναπαριστά ένα πόρο είτε του τύπου `literal(Value)` αν το αντικείμενο είναι τύπου κειμένου, ακεραίου κ.λ.π. Ένα ακόμα χρήσιμο κατηγόρημα στην αναζήτηση τριπλέτων είναι και το `rdf_has((?Subject, ?Predicate, ?Object)` το οποίο εκμεταλλεύεται την rdfs σχέση `subPropertyOf`. Το ερώτημα αυτό επιστρέφει κάθε τριπλέτα της οποίας το κατηγόρημα (predicate) είναι ίδιο με το `Predicate` ή μπορεί να προσεγγιστεί αναδρομικά από αυτό μέσω της ιδιότητας `subPropertyOf`.

Επίσης, το δόμημα `rdf_db` παρέχει δυνατότητα φόρτωσης δεδομένων από ένα rdf αρχείο καθώς και αποθήκευσης σε αρχείο. Στην παρούσα εργασία χρησιμοποιείται το κατηγόρημα `rdf_load(+InOrList)` το οποίο φορτώνει rdf τριπλέτες από την ροή εισόδου `In` η οποία μπορεί να είναι όνομα ενός αρχείου ή οποιαδήποτε άλλη έγκυρη είσοδος.

Ένα άλλο δόμημα της βιβλιοθήκης είναι και το `rdf_edit` το οποίο δίνει δυνατότητες τροποποίησης της βάσης δεδομένων. Μερικά από τα κατηγορήματα που παρέχονται είναι και τα εξής:

- `rdf_assert(+Subject, +Predicate, +Object)`
  Εισάγει μία νέα τριπλέτα στην βάση δεδομένων
- `rdf_retractall(?Subject, ?Predicate, ?Object)`
  Διαγράφει από την βάση δεδομένων όλες τις τριπλέτες που ταιριάζουν με την ζητούμενη
- `rdf_update(+Subject, +Predicate, +Object, +Action)`

Αντικαθιστά ένα από τα τρία πεδία της τριπλέτας ανάλογα με την τιμή της `Action`, η οποία μπορεί να είναι μία από τις εξής:

`subject(Resource)` οπότε αντικαθίσταται το πρώτο πεδίο της τριπλέτας

`predicate(Resource)` οπότε αντικαθίσταται το δεύτερο πεδίο της τριπλέτας

`object(Resource)` οπότε αντικαθίσταται το τρίτο πεδίο της τριπλέτας

Το δόμημα αυτό δεν χρησιμοποιήθηκε στην παρούσα εργασία καθώς το μόνο ζητούμενο ήταν η πραγματοποίηση αναζητήσεων από το σύστημα, αλλά θα ήταν πολύ χρήσιμο σε μία ενδεχόμενη μελλοντικά επέκταση του συστήματος ώστε εκτός από αναζητήσεις να εκτελεί και εισαγωγή, διαγραφή ή τροποποίηση τριπλετών στην βάση των rdf δεδομένων.

Επιπλέον, πολύ χρήσιμη είναι η δυνατότητα να πραγματοποιούμε αναζητήσεις στις rdf τριπλέτες σε όρους της γλώσσας RDFS. Αυτό επιτυγχάνεται με την χρήση του module rdfs της βιβλιοθήκης, το οποίο εκμεταλλεύεται τις σχέσεις `rdfs:subPropertyOf`, `rdfs:subClassOf` και `rdf:type`. Τα κατηγορήματα με τα οποία επιτυγχάνεται αυτό είναι τα παρακάτω:

- `rdfs_subproperty_of(?SubProperty, ?Property)`

  Το κατηγόρημα αυτό ικανοποιείται αν οι ιδιότητες `SubProperty` και `Property` συνδέονται με την σχέση `rdfs:subPropertyOf`
- `rdfs_subclass_of(?SubClass, ?Class)`

  Το κατηγόρημα αυτό ικανοποιείται αν οι κλάσεις `SubClass` και `Class` συνδέονται με την σχέση `rdfs:subClassOf`
- `rdfs_class_property(+Class, ?Property)`

  Το κατηγόρημα αυτό ικανοποιείται αν το πεδίο τιμών της ιδιότητας `Property` περιλαμβάνει την κλάση `Class`
- `rdfs_individual_of(?Resource, ?Class)`

  Το κατηγόρημα αυτό ικανοποιείται αν ο πόρος `Resource` είναι στιγμιότυπο της ζητούμενης κλάσης

Το δόμημα rdfs επιπλέον παρέχει μερικά χρήσιμα κατηγορήματα για τον χειρισμό συλλογών αντικειμένων όπως οι λίστες που παράγονται από την rdf δομή `rdf:parseType=Collection`. Δύο από αυτά που χρησιμοποιούνται και στην εφαρμογή μας είναι τα εξής:

- `rdfs_member(?Resource, +Set)`

Το κατηγόρημα αυτό χρησιμοποιείται για να ελέγξει αν ένας πόρος `Resource` είναι μέλος του `Set` ή χρησιμοποιείται για να παράγει όλα τα μέλη του `Set`, το οποίο μπορεί να είναι τύπου `rdf:List` ή `rdf:Container`.

- `rdfs_list_to_prolog_list(+Set, –List)`

Το κατηγόρημα αυτό μετατρέπει το `Set` που είναι τύπου `rdf:List` σε μία λίστα αντικειμένων της Prolog ώστε να είναι μετά δυνατή η επεξεργασία της με την γλώσσα Prolog.

3.3.2 Υποστήριξη του πρωτοκόλλου HTTP από την SWI-Prolog

Το πακέτο HTTP (SWI-Prolog HTTP package) είναι ένα σύνολο βιβλιοθηκών οι οποίες παρέχουν υποστήριξη του HTTP πρωτοκόλλου. Το πακέτο αυτό περιλαμβάνει μία σειρά από βιβλιοθήκες που προσφέρουν δυνατότητα πρόσβασης δεδομένων που βρίσκονται σε HTTP διακομιστές καθώς επίσης δίνουν στην ίδια την Prolog δυνατότητες HTTP διακομιστή (server – client libraries). Στην παρούσα εργασία χρησιμοποιήθηκε μόνο η http server βιβλιοθήκη που αφορά την μετατροπή της ίδιας της Prolog σε HTTP διακομιστή καθώς το ζητούμενο είναι η είσοδος παραμέτρων, η επεξεργασία των παρεχόμενων RDF μεταδεδομένων με βάση αυτές τις παραμέτρους και η έξοδος των αποτελεσμάτων προς τον χρήστη σε html μορφή. Η βιβλιοθήκη αυτή αποτελείται από δύο τμήματα: το πρώτο αφορά την διαχείριση της σύνδεσης με τον διακομιστή και έχει τρεις διαφορετικές υλοποιήσεις ανάλογα με τον επιθυμητό τύπο διακομιστή, ενώ το δεύτερο τμήμα αφορά την αποκωδικοποίηση της http αίτησης που παράγεται, την κλήση του κώδικα του χρήστη που χειρίζεται την αίτηση αυτή και τέλος την κωδικοποίηση της απάντησης σε κατάλληλη μορφή. Το πλεονέκτημα αυτού του διαχωρισμού είναι προφανές διότι κατά αυτόν τον τρόπο ο κώδικας του χρήστη που υλοποιεί την λειτουργικότητα του συστήματος είναι ανεξάρτητος από τον τύπο του διακομιστή οπότε και καθίσταται εύκολη η αλλαγή τύπου διακομιστή οποιαδήποτε στιγμή.

Η SWI-Prolog τρεις διαφορετικούς τύπους διακομιστή:

- Διακομιστής οδηγούμενος από συμβάντα (event-driven server) με χρήση της βιβλιοθήκης `library(xpce_httpd)`. Αυτός ο τύπος διακομιστή μπορεί να χειρίζεται αιτήσεις από πολλούς πελάτες (clients) ταυτόχρονα αλλά κάθε φορά εξυπηρετεί μόνο μία αίτηση μπλοκάροντας όλες τις υπόλοιπες διαδικασίες μέχρι αυτή να ολοκληρωθεί, οπότε ουσιαστικά οι αιτήσεις πολλαπλών διακομιστών σειριοποιούνται σε μία διαδικασία Prolog. Για τον λόγο αυτό αυτός ο τύπος διακομιστή είναι κατάλληλος μόνο για απλές εφαρμογές σε αυστηρά ελεγχόμενο περιβάλλον ή για πραγματοποίηση αποσφαλμάτωσης (debugging).
- Πολύ-νηματικός διακομιστής (multi threaded server) με χρήση της βιβλιοθήκης `library(thread_httpd)`. Αυτός ο τύπος διακομιστή είναι πιο γρήγορος και επιτρέπει τον χειρισμό πολλαπλών αιτήσεων ταυτόχρονα καθώς εκτελεί τον κώδικα του χρήστη παράλληλα μέσα από ένα σύνολο νημάτων (worker threads). Ως συνέπεια παρουσιάζει μεγαλύτερη δυσκολία στην αποσφαλμάτωση αλλά ωστόσο προσφέρει γρήγορη επικοινωνία με πολλαπλούς πελάτες

- Ένας διακομιστής για κάθε πελάτη με χρήση της βιβλιοθήκης `library(inetd_httpd)`. Σε αυτή την υλοποίηση δημιουργείται ένας διακομιστής για κάθε πελάτη και προφανώς είναι εφικτή λύση μόνο στην περίπτωση διακομιστών που απαιτούν μικρό χρόνο εκκίνησης και στις περιπτώσεις που είναι κρίσιμο η κατάρρευση ενός διακομιστή να μην επηρεάζει τους υπόλοιπους της εφαρμογής.

Η εφαρμογή μας, όπως άλλωστε και η πλειοψηφία των εφαρμογών, χρησιμοποιεί έναν πολύ- νηματικό διακομιστή λόγω των προφανών πλεονεκτημάτων του που αναφέρθηκαν και παραπάνω.

Εκτός από το τμήμα που υλοποιεί τον διακομιστή, η εφαρμογή αποτελείται από ένα ακόμα τμήμα - το 'σώμα' του διακομιστή (server-body) - που περιέχει τον κώδικα που είναι υπεύθυνος για τον χειρισμό της αίτησης και την παραγωγή μίας απάντησης (reply). Ο κώδικας αυτός καλείται με όρισμα την εξυπηρετούμενη αίτηση και ο ρόλος του είναι παρόμοιος με ένα cgi αρχείο, δηλαδή πρέπει να παράγει μία επικεφαλίδα (header) που θα δηλώνει τουλάχιστον το πεδίο `Content-type` της ιστοσελίδας και ένα σώμα (body) με το περιεχόμενό της. Αυτό το τμήμα της εφαρμογής χρησιμοποιεί διάφορες βιβλιοθήκες του πακέτου όπως για παράδειγμα την βιβλιοθήκη `library(http/http_parameters)` η οποία παρέχει δύο κατηγορήματα που επιτρέπουν την ανάκτηση παραμέτρων από μία html φόρμα όπως η φόρμα κριτηρίων αναζήτησης που χρησιμοποιεί η εφαρμογή μας. Τα κατηγορήματα αυτά είναι τα

- `http_parameters(+Request,?Parameters)` και
- `http_parameters(+Request, ?Parameters, +Options).`

Και τα δύο δέχονται ως όρισμα την αίτηση `Request` και μία μερικώς ορισμένη λίστα `Parameters` των παραμέτρων. Κάθε στοιχείο της λίστας αυτής είναι της μορφής `Name(-Value, +Options)` όπου `Name` είναι το όνομα της παραμέτρου, `Value` η τιμή που αυτή λαμβάνει μετά τον χειρισμό της αίτησης και `Options` μία λίστα με επιλογές που καθορίζουν τον τύπο δεδομένων της παραμέτρου, την εξ' ορισμού τιμή της κ.α. Αν μία παράμετρος λείπει τότε παράγεται η εξαίρεση `error(existence_error (form_data, Name),_ ).`
Οι επιλογές που διατίθενται είναι οι παρακάτω:

✓ `default (Default)` → αν η παράμετρος λείπει τότε η τιμή της ταυτοποιείται με την τιμή `Default`.

✓ `optional (true)` → αν η παράμετρος λείπει τότε η μεταβλητή Value, που αντιπροσωπεύει την τιμή της παραμέτρου, παραμένει αδέσμευτη χωρίς να παράγεται λάθος.

✓ `zero_or_more` → επιτρέπει καμία ή περισσότερες από μία εμφανίσεις της ίδιας μεταβλητής

✓ `one of(List)` → επιτυγχάνει όταν η τιμή της παραμέτρου είναι μία εκ των στοιχείων της `List`

✓ `length>N, length>=N, length<N, length=<N` → επιτυγχάνει όταν η παράμετρος είναι άτομο μήκους ανάλογου με τον περιορισμό σε σχέση με το N.

✓ `number` → μετατρέπει την τιμή της μεταβλητής σε αριθμό

✓ `integer` → μετατρέπει την τιμή της μεταβλητής σε ακέραιο

✓ `float` → μετατρέπει την τιμή της μεταβλητής σε αριθμό κινητής υποδιαστολής

Το δεύτερο κατηγόρημα παρέχει δύο επιπλέον επιλογές:

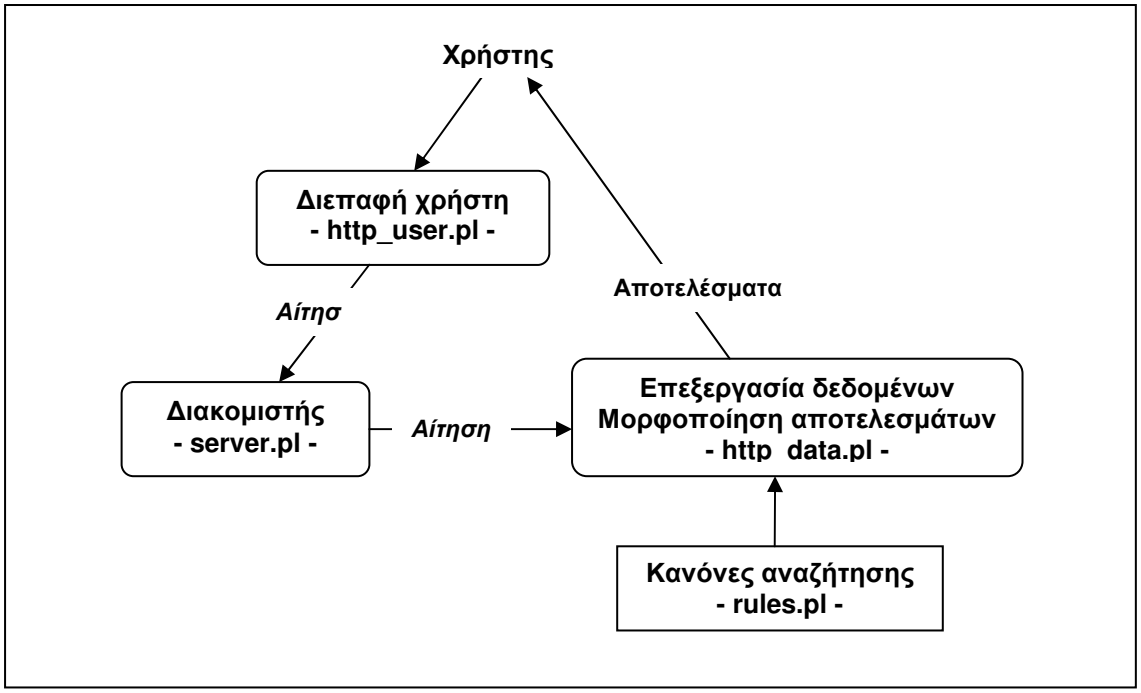
✓ `form_data(-Data)` → επιστρέφει το σύνολο των παραμέτρων σε μία λίστα από ζεύγη `Name=Value`.

✓ `attribute_declarations(:Goal)` → αν μία παράμετρος δεν έχει ορισμένες επιλογές τότε καλείται ο κανόνας `Goal` που καθορίζει αυτές τις επιλογές. Είναι πολύ χρήσιμη επιλογή όταν οι ίδιες παράμετροι χρησιμοποιούνται από διαφορετικές κλήσεις αιτήσεων.

Μία ακόμα βιβλιοθήκη που χρησιμοποιείται από το 'σώμα' του διακομιστή είναι η `library(http/html_write)` η οποία αφορά την παραγωγή δομημένων html εγγράφων από δομές δεδομένων της Prolog. Το κύριο κατηγόρημα αυτής της βιβλιοθήκης είναι το `html(:Spec)`, το οποίο μεταφράζει τον ορισμό μίας html σελίδας σε μία λίστα από άτομα τα οποία μπορούν στην συνέχεια να γραφτούν στην έξοδο με την βοήθεια του κατηγορήματος `print_html/[1,2]`. Ο ορισμός `Spec` μπορεί να είναι στοιχείο - ή και λίστα στοιχείων – της μορφής `Tag(Attributes)` ή `Tag(Attributes, Content)`, όπου Tag είναι το όνομα του html στοιχείου, `Content` το περιεχόμενό του και `Attributes` μία λίστα από ιδιότητες του στοιχείου της μορφής `Name(Value)` ή `Name=Value`.

3.4 Αρχιτεκτονική και λειτουργία

Το σύστημα αφαιρετικά αποτελείται από δύο κυρίως τμήματα: το τμήμα που υλοποιεί τον διακομιστή και το τμήμα που περιλαμβάνει τον κώδικα που χειρίζεται την είσοδο από τον χρήστη, την επεξεργασία της εισόδου και την έξοδο των αποτελεσμάτων πίσω στον χρήστη.
Το τμήμα που αφορά την δημιουργία του διακομιστή υλοποιείται από το αρχείο server.pl ενώ το δεύτερο από το αρχείο http_data.pl καθώς και το αρχείο rules.pl το οποίο περιέχει το σώμα των κανόνων που υλοποιούν την λειτουργία της αναζήτησης. Επιπλέον, η επικοινωνία μεταξύ χρήστη και συστήματος επιτυγχάνεται μέσω της διεπαφής χρήστη η οποία υλοποιείται από το αρχείο http_user.pl. Η δομή αυτή απεικονίζεται διαγραμματικά στο Σχήμα 6.



Σχήμα 6: Δομή του συστήματος

Κατά την εκκίνηση του συστήματος δημιουργείται ο διακομιστής και στέλνεται μία αίτηση (request) η οποία έχει ως αποτέλεσμα την δημιουργία της html φόρμας εισαγωγής των κριτηρίων της αναζήτησης. Αυτή η φόρμα αποτελεί την διεπαφή του χρήστη μέσω της οποίας ο τελευταίος επικοινωνεί με το σύστημα. Όταν συμπληρωθεί και υποβληθεί αυτή η φόρμα στέλνεται μία άλλη αίτηση (request), μαζί με τις παραμέτρους της αναζήτησης, την οποία χειρίζεται το τμήμα του συστήματος που υλοποιεί την διαδικασία της επεξεργασίας (http_data.pl). Η επεξεργασία πραγματοποιείται με την βοήθεια του σώματος των κανόνων αναζήτησης οι οποίοι περιέχονται στο αρχείο rules.pl όπως αναφέρθηκε και παραπάνω. Τέλος, τα αποτελέσματα που παράγονται επιστρέφονται στον χρήστη υπό μορφή html φόρμας.

Στις ακόλουθες ενότητες, περιγράφεται αναλυτικά η τυπική λειτουργία του συστήματος καθώς και το σώμα κώδικα το οποίο υλοποιεί τα διάφορα τμήματα του συστήματος όπως αυτά περιγράφηκαν παραπάνω.

3.4.1 Δημιουργία του διακομιστή

Για την δημιουργία του διακομιστή χρησιμοποιείται η βιβλιοθήκη http/thread_httpd.pl που δίνει την δυνατότητα δημιουργίας ενός πολύ-νηματικού διακομιστή (multi-threaded server) ο οποίος μπορεί να εξυπηρετεί ταυτόχρονα και γρηγορότερα ένα πλήθος πελατών (clients) με χρήση αντίστοιχου αριθμού νημάτων (worker-threads). Στο αρχείο parms.pl καθορίζονται παράμετροι σχετικές με τον διακομιστή, όπως η θύρα (port), ο μέγιστος χρόνος αδράνειας για μία σύνδεση

(max_idle_time) και ο αριθμός των νημάτων (workers) όπως φαίνεται στο παρακάτω τμήμα κώδικα:

```
setting(port(3020)).
setting(max_idle_time(3600)).
setting(http_options([ local(infinite),
                       global(infinite),
                       trail(infinite),
                       workers(2)
                     ])
        ).
```

3.4.2 Εκκίνηση και λειτουργία του συστήματος

Το σύστημα ενεργοποιείται με την εκτέλεση του αρχείου run.pl το οποίο περιέχει εντολές για την δημιουργία και εκκίνηση του διακομιστή. Στην συνέχεια κατευθύνουμε τον φυλλομετρητή μας προς την διεύθυνση http://localhost:3020.

Ουσιαστικά, η λειτουργία του συστήματος βασίζεται σε δύο αρχεία: το `http_user.pl` και το `http_data.pl`. Το πρώτο περιέχει τον κώδικα που χρειάζεται για την δημιουργία της διεπαφής του χρήστη, δηλαδή παράγει όλα τα στοιχεία HTML τα οποία συνθέτουν την φόρμα αναζητήσεων. Το δεύτερο περιέχει τον κώδικα που χειρίζεται την πραγματοποίηση των αναζητήσεων, δηλαδή δέχεται τα κριτήρια και τις παραμέτρους της αναζήτησης που ορίστηκαν από τον χρήστη και πυροδοτεί ανάλογα τον αντίστοιχο κανόνα. Επιπλέον, αναλαμβάνει την μορφοποίηση των αποτελεσμάτων που επιστρέφονται από μία αναζήτηση σε μία φόρμα HTML φιλική προς τον χρήστη.

3.4.3 Κανόνες αναζήτησης

Η λειτουργία του παρόντος συστήματος βασίζεται σε ένα σώμα κανόνων, οι οποίοι βρίσκονται στο αρχείο rules.pl, και καθορίζουν τους τρόπους με τους οποίους μπορούν να πραγματοποιούνται οι αναζητήσεις πάνω στα rdf δεδομένα. Τα δεδομένα αυτά φορτώνονται κατά την εκκίνηση του συστήματος από το αρχείο publications.rdf το οποίο βρίσκεται τοπικά αποθηκευμένο κι είναι προσβάσιμο με την εντολή `:-rdf_load('c:/publications.rdf').` Ωστόσο, υπάρχει δυνατότητα φόρτωσης του αρχείου από το διαδίκτυο με την εντολή

`:-load_files('lpis.csd.auth.gr/publications.rdf',[stream(Input)])`

αν το αρχείο βρίσκεται στη διεύθυνση `lpis.csd.auth.gr/publications.rdf`.

Όλοι οι κανόνες που έχουν οριστεί δέχονται ως είσοδο τα δεδομένα της αναζήτησης (π.χ. τίτλο δημοσίευσης) και επιστρέφουν τους κωδικούς των δημοσιεύσεων που εντοπίστηκαν σε μία λίστα List. Ο πρώτος κανόνας υλοποιεί την αναζήτηση με βάση τον τίτλο δηλαδή εντοπίζει μία δημοσίευση που έχει ως τίτλο της τον ζητούμενο.

```
publicationtitle(Title,List):-
      findall(Entry,rdf(Entry,pubrdf:'publicationTitle',literal(Title)),List).
```

Ο δεύτερος κανόνας πραγματοποιεί αναζήτηση με βάση το όνομα του εκδότη, δηλαδή προσδιορίζει όλες τις δημοσιεύσεις οι οποίες εκδόθηκαν από ένα συγκεκριμένο εκδότη.

```
publicationpublisher(Publisher,List):-
       findall(Entry,rdf(Entry,pubrdf:'mediaPublisher',literal(Publisher)),
       List).
```

Ο τρίτος κανόνας υλοποιεί την αναζήτηση με βάση το όνομα ενός εκ των συγγραφέων. Εδώ πολύ σημαντικό ρόλο παίζει η θέση που έχει ο ζητούμενος συγγραφέας στην λίστα συγγραφέων της εκάστοτε δημοσίευσης. Για τον λόγο αυτό ο κανόνας αυτός δέχεται ως κριτήρια της αναζήτησης που υλοποιεί τόσο το όνομα του συγγραφέα όσο και την θέση του. Επιπλέον, υπάρχει η περίπτωση αναζήτησης ενός συγγραφέα ανεξαρτήτως θέσης στην λίστα συγγραφέων, οπότε υπάρχουν δύο διαφορετικοί ορισμοί του κανόνα ανάλογα με την περίπτωση. Η θέση του συγγραφέα καθορίζεται από την μεταβλητή Position η οποία παίρνει τιμές 1-3 αν αναζητείται συγγραφέας στην πρώτη, δεύτερη ή τρίτη θέση της λίστας ή την τιμή 0 αν αναζητείται ο συγγραφέας σε οποιαδήποτε θέση.

```
author(AuthorSurname,0,List):-
       isAuthor(AuthorID,AuthorSurname,_AuthorName),
       findall(Entry,(rdf(Entry,pubrdf:'authors',AuthorsList),rdf(AuthorsList,
       rdf:'type',rdf:'List'),rdfs_list_to_prolog_list(AuthorsList,PrList),
       member(AuthorID,PrList)),List).
author(AuthorSurname,Position,List):-
       Position\=0,
       isAuthor(AuthorID,AuthorSurname,_AuthorName),
       findall(Entry,(rdf(Entry,pubrdf:'authors',AuthorsList),rdf(AuthorsList,
       rdf:'type',rdf:'List'),rdfs_list_to_prolog_list(AuthorsList,PrList),
       nth1(Position,PrList,AuthorID)),List).
```

Ο τέταρτος κανόνας υλοποιεί την αναζήτηση με βάση μία λέξη κλειδί την οποία εισάγει ο χρήστης.

```
publicationkeyword(Keyword,List):-
      findall(Entry,(rdf(Entry,pubrdf:'keywords',Description),rdf(Description,
      rdf:'type',rdf:'Seq'),rdfs_member(literal(Keyword),Description)),List).
```

Ο πέμπτος κανόνας υλοποιεί την αναζήτηση με βάση το έτος δημοσίευσης. Ο κανόνας αυτός δίνει την δυνατότητα να οριστεί και χρονικό διάστημα εκτός από μεμονωμένη χρονολογία της δημοσίευσης, για παράδειγμα «να βρεθούν όλες οι δημοσιεύσεις που έγιναν *πριν* το 2002». Η δυνατότητα αυτή παρέχεται μέσω μίας παραμέτρου `Operator` η οποία ορίζει το χρονικό διάστημα αναζήτησης και μπορεί να πάρει μία από τις τιμές *κατά την διάρκεια* (=), *πριν* (<), *μετά* (>), *πριν ή και κατά την διάρκεια* (<=), *κατά την διάρκεια ή και μετά* (>=).

```
publicationyear(PubYear,Operator,List):-
       atom_number(PubYear,PublicationYear),
       Comparison=..[Operator,PYear,PublicationYear],
       findall(Entry,(rdf(Entry,pubrdf:'publicationYear',literal(type(xsd:
       'integer',Year))),atom_number(Year,PYear),call(Comparison)),List).
```

Ο έκτος κανόνας υλοποιεί την αναζήτηση με βάση το είδος της δημοσίευσης το οποίο μπορεί να είναι για παράδειγμα πρακτικά συνεδρίου, διδακτορική διατριβή, βιβλίο, περιοδικό κ.α.

```
publicationtype(Type,List):-
       rdf_global_term(pubrdf:Type, XXX),
       findall(Entry,rdf(Entry,rdf:type,XXX),List).
```

Τέλος, υπάρχει και ένας βοηθητικός κανόνας που αντιστοιχίζει ονοματεπώνυμα συγγραφέων με τους κωδικούς τους και αντίστροφα. Αυτός είναι απαραίτητος κατά την αναζήτηση με βάση το ονοματεπώνυμο συγγραφέα γιατί στα rdf δεδομένα η κάθε δημοσίευση αναφέρεται στους συγγραφείς της μέσω μίας λίστας η οποία περιέχει τους κωδικούς των συγγραφέων και όχι τα ονοματεπώνυμά τους.

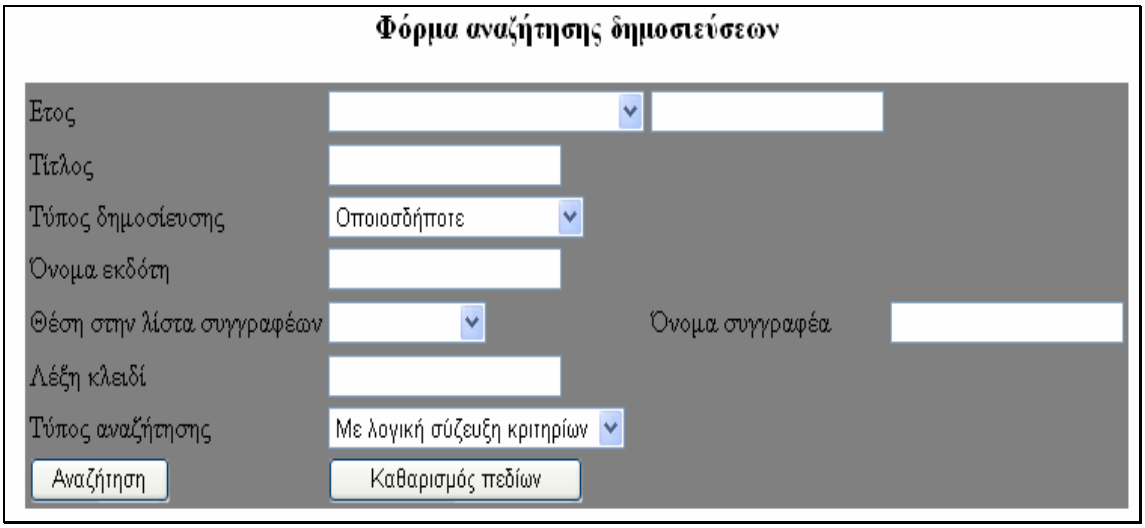
```
isAuthor(AuthorID,Surname,Name):-
       rdf(AuthorID,pubrdf:'authorSurName',literal(Surname)),
       (rdf(AuthorID,pubrdf:'authorName',literal(Name)) -> true; Name='').
```

3.4.4 Διεπαφή χρήστη

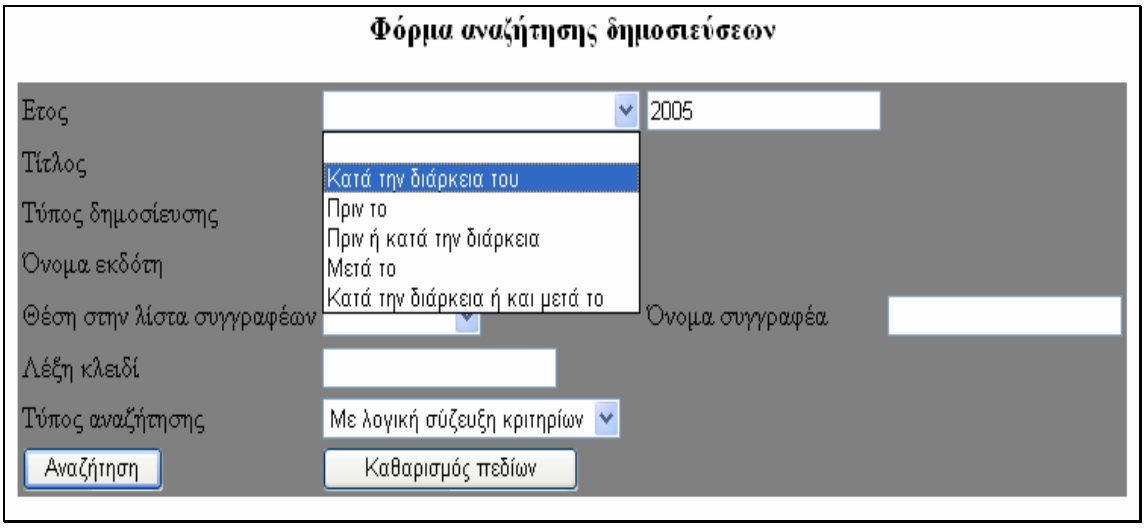
Μετά την εκκίνηση του διακομιστή στέλνεται μία αίτηση και παράγεται μία διαδικτυακή φόρμα (html form) στην οποία ο χρήστης καλείται να εισάγει τα κριτήρια της αναζήτησης που θέλει να πραγματοποιήσει. Αυτή η φόρμα εισαγωγής κριτηρίων φαίνεται στο Σχήμα 7.

Σε αυτή την φόρμα ο χρήστης έχει την δυνατότητα να εισάγει πρώτον ως κριτήριο αναζήτησης το χρονικό διάστημα μέσα στο οποίο έγινε/αν η/οι ζητούμενη/ες δημοσίευση/σεις εισάγοντας την χρονολογία και το διάστημα συσχετισμού (πριν, μετά, κατά την διάρκεια κ.λ.π.) όπως φαίνεται και στο Σχήμα 7.1.

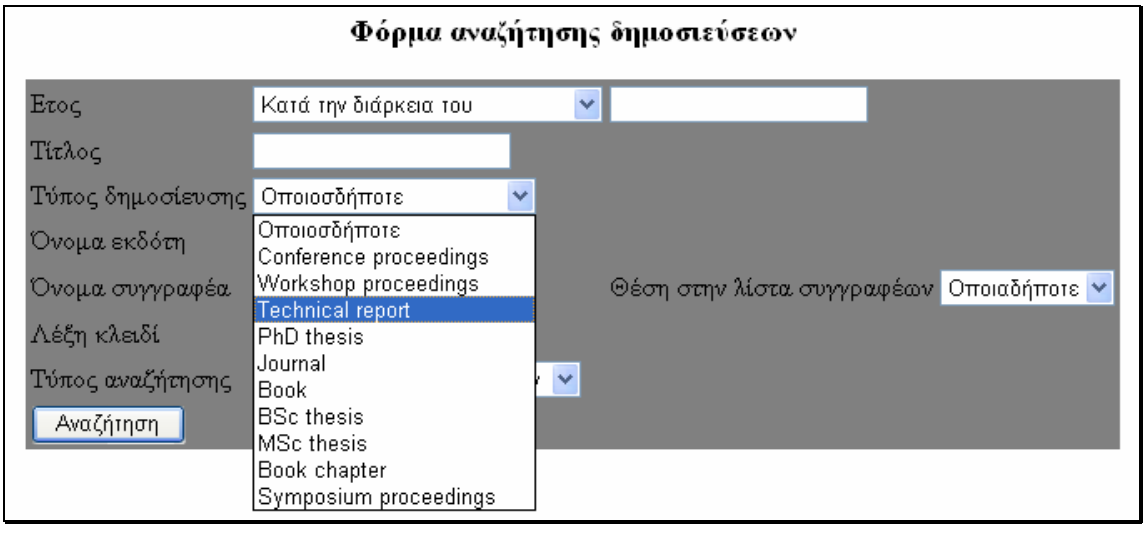
Επίσης, μπορεί να εισάγει ως κριτήρια αναζήτησης το είδος της δημοσίευσης, δηλαδή αν πρόκειται για βιβλίο, πρακτικά συνεδρίου, διδακτορική διατριβή κ.α., κάνοντας την αντίστοιχη επιλογή από ένα μενού, όπως φαίνεται στο Σχήμα 7.2.



Σχήμα 7: Φόρμα αναζήτησης του συστήματος



Σχήμα 7.1 : Αναζήτηση με βάση το έτος δημοσίευσης

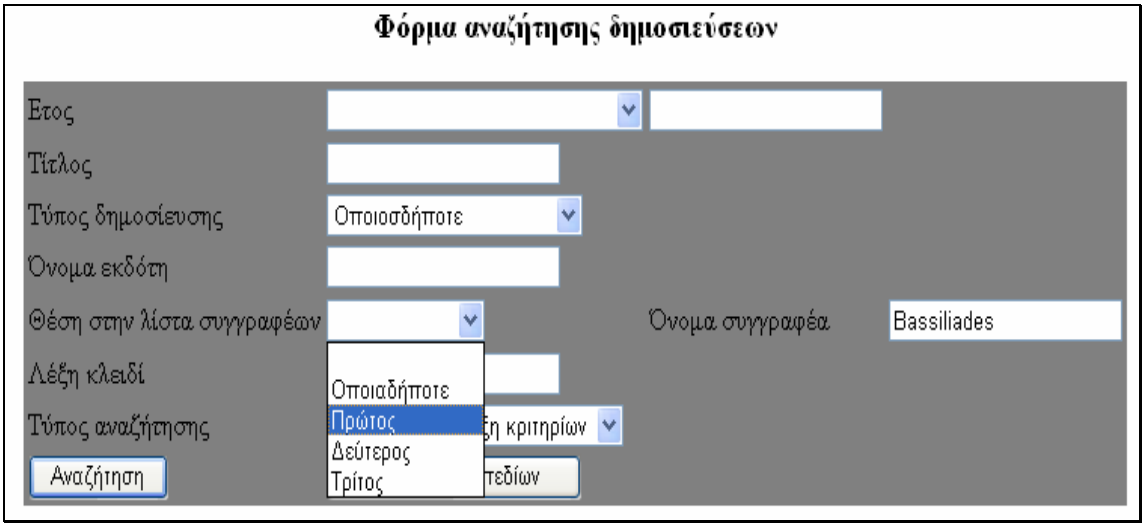


Σχήμα 7.2: Αναζήτηση με βάση τον τύπο της δημοσίευσης

Επιπλέον, ο χρήστης μπορεί να ορίσει ως κριτήρια αναζήτησης τον τίτλο της δημοσίευσης, το όνομα του εκδότη ή και μία λέξη κλειδί εισάγοντας τους ζητούμενους όρους στα αντίστοιχα πεδία κειμένου της φόρμας, όπως για παράδειγμα στο Σχήμα 7.3.

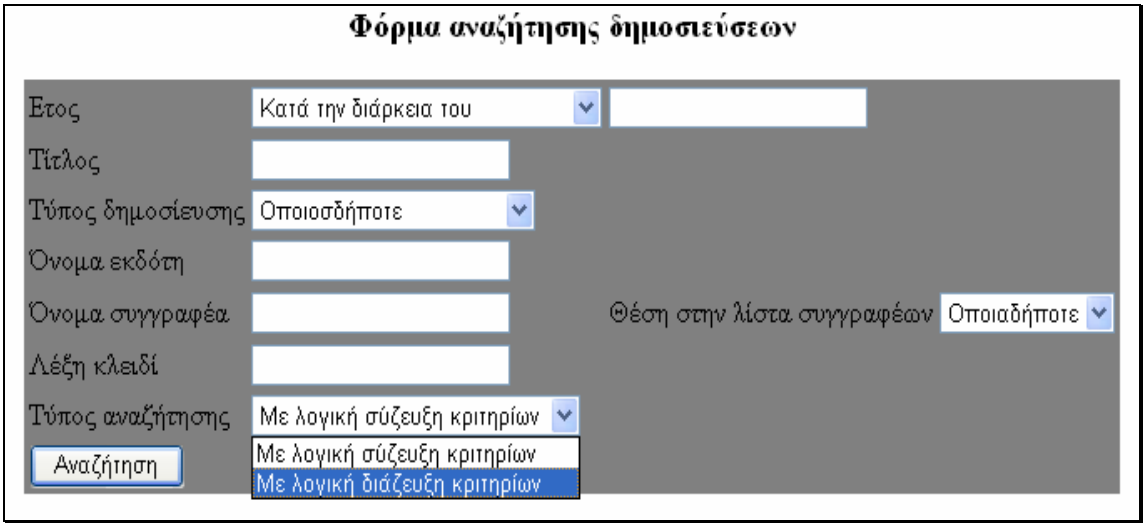
Σχήμα 7.3: Αναζήτηση με βάση μία λέξη κλειδί

Μία πολύ σημαντική δυνατότητα είναι αυτή της αναζήτησης με βάση το όνομα ενός εκ των συγγραφέων και μάλιστα με βάση και την θέση στην λίστα συγγραφέων, δηλαδή αν ο ζητούμενος συγγραφέας είναι πρώτος, δεύτερος ή τρίτος στην συγγραφή της ζητούμενης εργασίας. Αυτό επιτυγχάνεται, όπως φαίνεται και παρακάτω, αν ο χρήστης εισάγει το ονοματεπώνυμο του ζητούμενου συγγραφέα και επιλέξει στη συνέχεια από το μενού την θέση του συγγραφέα στην λίστα συγγραφέων της δημοσίευσης (οι επιλογές είναι πρώτος, δεύτερος, τρίτος ή σε οποιαδήποτε θέση), όπως φαίνεται και στο Σχήμα 7.4.



Σχήμα 7.4: Αναζήτηση με βάση το όνομα συγγραφέα

Τέλος, το σύστημα δίνει την δυνατότητα πραγματοποίησης σύνθετων αναζητήσεων με λογική σύζευξη (and) ή λογική διάζευξη (or) των κριτηρίων. Αυτό επιτυγχάνεται αν επιλεγεί ο αντίστοιχος τύπος αναζήτησης από το ανάλογο μενού στην φόρμα, όπως φαίνεται παρακάτω στο Σχήμα 7.5.



Σχήμα 7.5: Ορισμός τύπου σύνθετης αναζήτησης

Με την υποβολή της φόρμας στέλνεται μία αίτηση (Request) προς τον διακομιστή και ανακτώνται όλες οι παράμετροι της φόρμας. Στην συνέχεια ενεργοποιείται ο κανόνας `define_criteria` ο οποίος δημιουργεί την λίστα `Criteria` με τα ονόματα των παραμέτρων αναζήτησης που έχουν τιμή, την λίστα `Values` με τις τιμές των αντίστοιχων παραμέτρων και την λίστα `Ops` με τους τελεστές – αν είναι απαραίτητοι – για τα κριτήρια που έχουν οριστεί, όπως φαίνεται παρακάτω:

```
define_criteria([],C,V,O,Criteria,Values,Ops):-
       Ops=O,
       Criteria=C,
       Values=V.

define_criteria(L,C,V,O,Criteria,Values,Ops):-
       L=[Name=Value|T],
       ((Name=='operator',Value\=='99'|Name=='position',Value\=='99')->
       (atom_number(Value,Val),append(O,[Val],O1));O1=O ),
              ((Name\=='operator',Name\=='position',Name\=='searchtype',
              Name\=='publicationtype',Value\=='',Value\=='any')
              ->
              (append(C,[Name],C1),append(V,[Value],V1));
                     ((Name=='publicationtype',Value\=='any')
                     ->
                     (convert_case(Value,Val),append(C,[Name],C1),
                     append(V,[Val],V1) );C1=C,V1=V)),
       define_criteria(T,C1,V1,O1,Criteria,Values,Ops).
```

Οι τρεις αυτές λίστες αποτελούν ορίσματα για τον κανόνα `make_query` ο οποίος κατασκευάζει το ερώτημα, ή τα ερωτήματα στην περίπτωση σύνθετης αναζήτησης, που υλοποιούν την συγκεκριμένη αναζήτηση. Ο κανόνας αυτός φαίνεται παρακάτω και λειτουργεί ως εξής: αναδρομικά ανακτώνται ένα ένα τα κριτήρια της λίστας `Criteria` και κατασκευάζεται ο κανόνας (`Term`) που υλοποιεί την αντίστοιχη αναζήτηση. Για παράδειγμα, αν στην λίστα `Criteria` περιλαμβάνονται τα κριτήρια publicationtype και publicationtitle τότε οι κανόνες που θα κατασκευαστούν είναι αντίστοιχα οι `publicationtype(Type,List)` και `publicationtitle(Title,List)`.

```
make_query(Criteria,Values,Ops,[],Terms):-
       Criteria=[Criterion | T],
       Values=[Value | R],
       ((Criterion=='publicationyear'|Criterion=='author')
       ->
       (Ops=[Op|Q],Term=..[Criterion,Value,Op,Matches]);
       (Term=..[Criterion,Value,Matches],Q=Ops)),
       F=[Term],
make_query(T,R,Q,F,Terms).

make_query(Criteria,Values,Ops,F,Terms):-
       Criteria=[Criterion | T],
       Values=[Value | R],
       ((Criterion=='publicationyear'|Criterion=='author')
       ->
       (Ops=[Op|Q],Term=..[Criterion,Value,Op,Matches]);
       (Term=..[Criterion,Value,Matches],Q=Ops)),
       append(F,[Term],Ft),
make_query(T,R,Q,Ft,Terms).

make_query([],Values,Ops,F,Terms):-
       Terms=F.
```

Οι κανόνες που κατασκευάζονται με αυτόν τον τρόπο επιστρέφονται σε μία λίστα Terms η οποία στην συνέχεια αποτελεί όρισμα του αναδρομικού κανόνα `run_query` ο οποίος ουσιαστικά είναι αυτός που εκτελεί τα ερωτήματα (queries) που κατασκευάστηκαν και επιστρέφει τα σύνολα των εγγραφών που ικανοποιούν τα κριτήρια. Ο κανόνας `run_query` είναι ο παρακάτω:

```
run_query([],[]):- !.
run_query([H|T],[M1|M2]):-
       call(H),
       H =.. L,
       last(L,M1),
       run_query(T,M2).
```

Όπως αναφέρθηκε και παραπάνω, το σύστημα παρέχει την δυνατότητα σύνθετων αναζητήσεων με σύζευξη ή διάζευξη κριτηρίων. Επομένως λοιπόν, μετά την εκτέλεση των ερωτημάτων για κάθε κριτήριο αναζήτησης χωριστά απαιτείται ο συνδυασμός των αποτελεσμάτων τους στην περίπτωση που έχουμε σύνθετη αναζήτηση ανάλογα με τον τύπο αναζήτησης που έχει ορίσει ο χρήστης. Στην περίπτωση που ο χρήστης δεν έχει ορίσει τύπο αναζήτησης εξ΄ ορισμού θεωρείται επιλεγμένη η σύζευξη κριτηρίων, γεγονός που επιτρέπει

στην εφαρμογή να λειτουργεί με συνέπεια και ομοιομορφία ως προς τον κώδικα αφού ακόμα και αν ο χρήστης έχει επιλέξει μόνο ένα κριτήριο (απλή αναζήτησης) πραγματοποιείται σύζευξη του μοναδικού συνόλου των αποτελεσμάτων με το ίδιο το σύνολο, γεγονός που επιτυγχάνει πάντα επιστρέφοντας το ίδιο σύνολο αποτελεσμάτων. Ο κανόνας που επιτελεί αυτόν τον συνδυασμό των επιμέρους αποτελεσμάτων είναι ο `combine_results` που δέχεται ως παράμετρο τις επιμέρους λίστες αποτελεσμάτων και την μεταβλητή `Searchtype` η οποία καθορίζει τον τύπο αναζήτησης που έχει επιλέξει ο χρήστης. Αυτό το τμήμα κώδικα φαίνεται παρακάτω:

```
combine_results(and,MatchesIn,MatchesOut) :-
       multi_intersection(MatchesIn,MatchesOut).

combine_results(or,MatchesIn,MatchesOut) :-
       multi_union(MatchesIn,MatchesOut).

multi_intersection([H],H).

multi_intersection([H|T],R) :-
       multi_intersection(T,T1),
       intersection(H,T1,R).

multi_union([],[]).

multi_union([H|T],R) :-
       multi_union(T,T1),
       union(H,T1,R).
```

Τέλος, τα αποτελέσματα που προκύπτουν επιστρέφονται σε μία λίστα `Matches` η οποία όμως περιέχει τους κωδικούς αριθμούς εγγραφής των δημοσιεύσεων που ικανοποιούν τα κριτήρια της αναζήτησης. Απαιτείται λοιπόν πρώτα για κάθε εγγραφή η ανάκτηση όλων των στοιχείων της δημοσίευσης την οποία αυτή αντιπροσωπεύει προτού την εμφάνιση των αποτελεσμάτων. Αυτό πραγματοποιείται από τον κανόνα `publicationdata` ο οποίος ανακτά όλα τα στοιχεία μίας δημοσίευσης και τα επιστρέφει σε ένα αλφαριθμητικό. Ο κανόνας αυτός φαίνεται παρακάτω:

```
publicationdata(Entry,ResultStr):-
       listauthors(Entry,AList),
       authornames(AList,[],Authors),
       (rdf(Entry,pubrdf:'publicationTitle',literal(ATitle))
       ->
       true ; ATitle=''),
       string_to_atom(Title,ATitle),
       append(Authors,[Title],List1),
       (rdf(Entry,pubrdf:'mediaTitle',literal(MediaTitle))
       ->
       true ; MediaTitle=''),
       string_to_atom(MediaTitle1,MediaTitle),
       append(List1,[MediaTitle1],List2),
       (rdf(Entry,pubrdf:'mediaEditors',literal(MediaEditors))
       ->
       true ; MediaEditors=''),
       string_to_atom(MediaEditors1,MediaEditors),
       append(List2,[MediaEditors1],List3),
       (rdf(Entry,pubrdf:'mediaPublisher',literal(MediaPublisher))
       ->
```

```
       true ; MediaPublisher=''),
       string_to_atom(MediaPublisher1,MediaPublisher),
       append(List3,[MediaPublisher1],List4),
       (rdf(Entry,pubrdf:'mediaVolInfo',literal(MediaVolInfo))
       ->
       true ; MediaVolInfo=''),
       string_to_atom(MediaVolInfo1,MediaVolInfo),
       append(List4,[MediaVolInfo1],List5),
       (rdf(Entry,pubrdf:'publicationPagesInMedium',literal(PublicationPagesIn
       Medium))
       ->
       true ; PublicationPagesInMedium=''),
       string_to_atom(PublicationPagesInMedium1,PublicationPagesInMedium),
       append(List5,[PublicationPagesInMedium1],List6),
       (rdf(Entry,pubrdf:'publicationLocation',literal(PublicationLocation))
       ->
       true ; PublicationLocation=''),
       string_to_atom(PublicationLocation1,PublicationLocation),
       append(List6,[PublicationLocation1],List7),
       (rdf(Entry,pubrdf:'publicationYear',literal(type(_,APublicationYear)))
       ->
       true ; APublicationYear=''),
       string_to_atom(PublicationYear,APublicationYear),
       append(List7,[PublicationYear],List),
       list_to_string(List,'',ResultStr).
```

Για κάθε εγγραφή δημοσίευσης λοιπόν απαιτείται η κλήση του κανόνα αυτού και η επιστροφή των στοιχείων της σε ένα αλφαριθμητικό έτοιμο για εκτύπωση. Αυτή η λειτουργία επιτελείται από τον κανόνα `build_html` ο οποίος κατασκευάζει την δομή μίας html σελίδας με περιεχόμενό της τα αλφαριθμητικά με τα στοιχεία κάθε δημοσίευσης που επιστράφηκε ως αποτέλεσμα της αναζήτησης και είναι ο παρακάτω:

```
build_html(Matches,Results) :-
       build_html_aux(Matches,R),
       Results =
       page([ title(['Αποτελέσματα αναζήτησης'])
            ],
            [ h2(align(center),['Αποτελέσματα αναζήτησης']) | R ]
           ).

build_html_aux([],[]) :- !.

build_html_aux([H|T],[p(S)|Rest]) :-
       publicationdata(H,S),
       build_html_aux(T,Rest).
```

Τέλος, χρειάζεται η εμφάνιση της σελίδας αυτής στο διαδίκτυο γεγονός το οποίο επιτελείται από τον κανόνα :

```
reply_page(Results) :-
       phrase(Results, HTML),
       format('Content-type: text/html~n~n'),
       print_html(HTML).
```

Ένα παράδειγμα σελίδας αποτελεσμάτων μίας αναζήτησης είναι και η παρακάτω:

Αποτελέσματα αναζήτησης

Δημοσιεύσεις που βρέθηκαν

- Bassiliades N, Vlahavas I, Elmagarmid A, E-DEVICE: An Extensible Knowledge Base System with Multiple Rule Support, IEEE Transactions on Knowledge and Data Engineering, Vol. 12, No. 5, 82 844, 2000,
- Bassiliades N, Vlahavas I, DEVICE: Compiling Production Rules into Event-Driven Rules Using Complex Events, Information and Software Technology, Elsevier, Vol. 39, No. 5, 331-342, 1997,
- Bassiliades N, Vlahavas I, Processing Production Rules in DEVICE, an Active Knowledge Base System, Data & Knowledge Engineering, Elsevier, Vol. 24, No. 2, 117-155, 1997,
- Bassiliades N, Vlahavas I, Hierarchical Query Execution in a Parallel Object-Oriented Database System, Parallel Computing, Elsevier, Vol. 22, No. 7, 1017-1048, 1996,
- Bassiliades N, Gray P, CoLan: A Functional Constraint Language and Its Implementation, Data & Knowledge Engineering, Vol. 14, No. 3, 203-249, 1995,
- Bassiliades N, Vlahavas I, Constraint Checking in a Parallel Object-Oriented Database System, Journal of Parallel Algorithms and Applications, Gordon and Breach, Vol. 5, No. 3-4, 129-147, 1995,
- Bassiliades N, Vlahavas I, PRACTIC: A Concurrent Object Data Model for a Parallel Object-Oriented Database System, Information Sciences, Elsevier, Vol. 86 (1-3), 149-178, 1995,
- Bassiliades N, Vlahavas I, Elmagarmid A, Houstis E, Interbase-KB: A Knowledge-based Multidatabase Sustem for Data Warehousing, IEEE Transactions on Knowledge and Data Engineering, Vol. 1 No. 5, 1188-1205, 2003,
- Bassiliades N, Vlahavas I, R-DEVICE: An Object-Oriented Knowledge Base System for RDF Metadata, International Journal on Semantic Web and Information Systems, Amit Sheth , Miltiadis D. Lytras, Idea Group, Vol. 2, No. 2 (to appear), 2006,

4. Συμπεράσματα – Μελλοντική εργασία

Το Σημασιολογικό Διαδίκτυο είναι ένας τομέας ακόμα υπό ανάπτυξη και συνεπώς προσφέρει μεγάλο εύρος εφαρμογών. Η χρήση της SWI-Prolog για την ανάπτυξη του συστήματος ανέδειξε τα ποικίλα πλεονεκτήματά της ως γλώσσας ανάπτυξης εφαρμογών του Σημασιολογικού Διαδικτύου.

Αρχικά, ο σκοπός ήταν να χρησιμοποιηθεί η βιβλιοθήκη PiLLoW για την δημιουργία της διεπαφής του συστήματος με το Διαδίκτυο αλλά ωστόσο δεν υπήρχε η απαραίτητη τεκμηρίωση και υποστήριξη της βιβλιοθήκης που να επιτρέπει την αποτελεσματική χρήση της. Συνεπώς, χρησιμοποιήθηκαν τα πακέτα της SWI-Prolog που αφορούν την υποστήριξη του HTTP πρωτοκόλλου και τα οποία αποδείχθηκαν πολύ αποτελεσματικά καθώς επέτρεψαν την δημιουργία της html σελίδων εύκολα και πολύ αποτελεσματικά.

Το παρόν σύστημα αποτελεί συνέχεια μίας προγενέστερης εργασίας η οποία αφορούσε την εξαγωγή rdf δεδομένων δυναμικά από μία σχεσιακή βάση δεδομένων. Μία μελλοντική επέκταση θα μπορούσε να περιλαμβάνει περισσότερους τύπους ερωτημάτων καθώς και την δυνατότητα εισαγωγής νέων, διαγραφής και τροποποίησης των υπαρχόντων βιβλιογραφικών δεδομένων έτσι ώστε από ένα απλό σύστημα αναζητήσεων να μεταβούμε σε ένα πλήρες σύστημα διαχείρισης των βιβλιογραφικών μεταδεδομένων. Επιπλέον, κάποιες επεκτάσεις θα μπορούσαν να αφορούν τον τρόπο παρουσίασης των αποτελεσμάτων των αναζητήσεων όπως για παράδειγμα ταξινόμηση των αποτελεσμάτων κατά ένα πεδίο, μέγιστο αριθμό αποτελεσμάτων ανά σελίδα κ.λ.π. Ακόμα παραπέρα, θα ήταν δυνατό να πραγματοποιηθεί παραμετροποίηση της εφαρμογής, δηλαδή προσαρμογή της σε οποιαδήποτε RDF έγγραφο και όχι απαραίτητα σε βιβλιογραφικά δεδομένα.

5. Βιβλιογραφία

[1] Βλαχάβας Ι., Κεφαλάς Π., Βασιλειάδης Ν., Ρεφανίδης Ι., Κόκκορας Φ., Σακελλαρίου Η., «Τεχνητή Νοημοσύνη», Γ΄ έκδοση, Β. Γκιούρδας, 2006

[2] Grigoris Antoniou and Frank van Harmelen, 'A Semantic Web Primer', 2004

[3] Berners-Lee T., "Semantic Web Road Map", September, 1998

[4] Manola F., Miller E., "RDF Primer" World Wide Web Consortium Working

[5] W.F. Clocksin, C.S. Mellish, 'Programming in Prolog', 4th edition

[6] Leon Sterling, Ehud Shapiro, 'The Art of Prolog – Advanced Programming Techniques', 2nd edition, The MIT Press

[7] Jan Wielemaker, SWI-Prolog 5.6 Reference Manual, University of Amsterdam Human-Computer Studies

[8] Jan Wielemaker, 'SWI-Prolog/XPCE Semantic Web Library', University of Amsterdam

[9] Jan Wielemaker, SWI-Prolog HTTP support, University of Amsterdam

Παράρτημα Α

Στο παράρτημα αυτό περιλαμβάνεται το RDF έγγραφο που περιέχει τα βιβλιογραφικά δεδομένα πάνω στα οποία εκτελούνται οι αναζητήσεις από το σύστημα καθώς και το RDF Schema με βάση το οποίο είναι οργανωμένα τα δεδομένα αυτά.

1. Το RDF έγγραφο

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:LPISPubs="http://lpis.csd.auth.gr/publications.rdfs#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<LPISPubs:WorkshopProceedings rdf:ID="entry97">
<LPISPubs:publicationID>97</LPISPubs:publicationID>
<LPISPubs:publicationTitle>Constraint Maintenance Using Generated
Methods in the P/FDM Object-Oriented
Database</LPISPubs:publicationTitle>
<LPISPubs:authors rdf:parseType="Collection">
<rdf:Description rdf:about="http://lpis.csd.auth.gr/rdfpublications.
asp#author66"/>
<rdf:Description rdf:about="http://lpis.csd.auth.gr/rdfpublications.
asp#author62"/>
<rdf:Description rdf:about="http://lpis.csd.auth.gr/rdfpublications.
asp#author9"/>
</LPISPubs:authors>
<LPISPubs:mediaTitle>Proc. 1st International Workshop on Rules in
Database Systems</LPISPubs:mediaTitle>
<LPISPubs:mediaEditors>N.W. Paton and M.H.
Williams</LPISPubs:mediaEditors>
<LPISPubs:mediaPublisher>Springer-Verlag</LPISPubs:mediaPublisher>
<LPISPubs:publicationPagesInMedium>364-
381</LPISPubs:publicationPagesInMedium>
<LPISPubs:publicationLocation>Edinburgh, Scotland, 30 August- 1
September 1993</LPISPubs:publicationLocation>
<LPISPubs:publicationYear>1994</LPISPubs:publicationYear>
<LPISPubs:publicationNoOfPages>17</LPISPubs:publicationNoOfPages>
<LPISPubs:publicationFileName>RIDS93.ps.gz</LPISPubs:publicationFileN
ame>
<LPISPubs:publicationRelatedURL>http://lpis.csd.auth.gr/people/nbassi
li/systems/colan.html</LPISPubs:publicationRelatedURL>
<LPISPubs:publicationRelatedURLText>CoLan</LPISPubs:publicationRelate
dURLText>
<LPISPubs:publicationAbstract></LPISPubs:publicationAbstract>
<LPISPubs:keywords>
<rdf:Seq>
<rdf:_1>Semantic Integrity Constraints</rdf:_1>
<rdf:_2>Object-Oriented Databases</rdf:_2>
<rdf:_3>Functional Data Model</rdf:_3>
<rdf:_4>Constraint Compilation</rdf:_4>
<rdf:_5>Incremental Constraint Checking</rdf:_5>
<rdf:_6>Meta-Data</rdf:_6>
</rdf:Seq>
</LPISPubs:keywords>
</LPISPubs:WorkshopProceedings>
<LPISPubs:Journal rdf:ID="entry98">
<LPISPubs:publicationID>98</LPISPubs:publicationID>
<LPISPubs:publicationTitle>The AND/OR Parallel Prolog Machine APIM:
Execution Model and Abstract Design</LPISPubs:publicationTitle>
<LPISPubs:authors rdf:parseType="Collection">
<rdf:Description rdf:about="http://lpis.csd.auth.gr/rdfpublications.
asp#author2"/>
<rdf:Description rdf:about="http://lpis.csd.auth.gr/rdfpublications.
```

```
asp#author50"/>
</LPISPubs:authors>
<LPISPubs:mediaTitle>Journal of Programming
Languages</LPISPubs:mediaTitle>
<LPISPubs:mediaPublisher>Chapman &amp;amp; Hall</LPISPubs:mediaPublisher>
<LPISPubs:mediaVolInfo>Vol. 1</LPISPubs:mediaVolInfo>
<LPISPubs:publicationPagesInMedium>245-
261</LPISPubs:publicationPagesInMedium>
<LPISPubs:publicationYear>1993</LPISPubs:publicationYear>
<LPISPubs:publicationNoOfPages>16</LPISPubs:publicationNoOfPages>
<LPISPubs:keywords>
<rdf:Seq>
</rdf:Seq>
</LPISPubs:keywords>
</LPISPubs:Journal>
<LPISPubs:Journal rdf:ID="entry99">
<LPISPubs:publicationID>99</LPISPubs:publicationID>
<LPISPubs:publicationTitle>A Novel Flow Control and Switching
Strategy for Preventing Hot Spot Congestion in Multistage
Networks</LPISPubs:publicationTitle>
<LPISPubs:authors rdf:parseType="Collection">
<rdf:Description rdf:about="http://lpis.csd.auth.gr/rdfpublications.
asp#author45"/>
<rdf:Description rdf:about="http://lpis.csd.auth.gr/rdfpublications.
asp#author2"/>
</LPISPubs:authors>
<LPISPubs:mediaTitle>Microprocessors and
Microsystems</LPISPubs:mediaTitle>
<LPISPubs:mediaPublisher>Butterworth</LPISPubs:mediaPublisher>
<LPISPubs:mediaVolInfo>Vol. 17 (7)</LPISPubs:mediaVolInfo>
<LPISPubs:publicationYear>1993</LPISPubs:publicationYear>
<LPISPubs:publicationNoOfPages>0</LPISPubs:publicationNoOfPages>
<LPISPubs:keywords>
<rdf:Seq>
</rdf:Seq>
</LPISPubs:keywords>
</LPISPubs:Journal>
…
<LPISPubs:Author rdf:ID="author1">
<LPISPubs:authorName>F</LPISPubs:authorName>
<LPISPubs:authorSurName>Kokkoras</LPISPubs:authorSurName>
<LPISPubs:authorURL>http://lpis.csd.auth.gr/people/fotis.html</LPISPu
bs:authorURL>
</LPISPubs:Author>
<LPISPubs:Author rdf:ID="author2">
<LPISPubs:authorName>I</LPISPubs:authorName>
<LPISPubs:authorSurName>Vlahavas</LPISPubs:authorSurName>
<LPISPubs:authorURL>http://www.csd.auth.gr/~vlahavas/</LPISPubs:autho
rURL>
</LPISPubs:Author>
<LPISPubs:Author rdf:ID="author6">
<LPISPubs:authorName>G</LPISPubs:authorName>
<LPISPubs:authorSurName>Tsoumakas</LPISPubs:authorSurName>
<LPISPubs:authorURL>http://users.auth.gr/~greg/</LPISPubs:authorURL>
</LPISPubs:Author> .....
```

2. To RDF Schema

```
<?xml version='1.0'?>
<!DOCTYPE rdf:RDF [
<!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
<!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
<!ENTITYLPISPubs 'http://lpis.csd.auth.gr/publications.rdfs#'>
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
```

```
<rdf:RDF xmlns:rdf="&rdf;"
xmlns:rdfs="&rdfs;"
xmlns:LPISPubs="&LPISPubs;"
xmlns:xsd="&xsd;">
<rdfs:Class rdf:about="&LPISPubs;Entry">
<rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="&LPISPubs;Book">
<rdfs:subClassOf rdf:resource="&LPISPubs;Entry"/>
</rdfs:Class>
<rdfs:Class rdf:about="&LPISPubs;BookChapter">
<rdfs:subClassOf rdf:resource="&LPISPubs;Entry"/>
</rdfs:Class>
<rdfs:Class rdf:about="&LPISPubs;TechnicalReport">
<rdfs:subClassOf rdf:resource="&LPISPubs;Entry"/>
</rdfs:Class>
<rdfs:Class rdf:about="&LPISPubs;Journal">
<rdfs:subClassOf rdf:resource="&LPISPubs;Entry"/>
</rdfs:Class>
<rdfs:Class rdf:about="&LPISPubs;WorkshopProceedings">
<rdfs:subClassOf rdf:resource="&LPISPubs;Entry"/>
</rdfs:Class>
<rdfs:Class rdf:about="&LPISPubs;SymposiumProceedings">
<rdfs:subClassOf rdf:resource="&LPISPubs;Entry"/>
</rdfs:Class>
<rdfs:Class rdf:about="&LPISPubs;BScThesis">
<rdfs:subClassOf rdf:resource="&LPISPubs;Entry"/>
</rdfs:Class>
<rdfs:Class rdf:about="&LPISPubs;MScThesis">
<rdfs:subClassOf rdf:resource="&LPISPubs;Entry"/>
</rdfs:Class>
<rdfs:Class rdf:about="&LPISPubs;PhDThesis">
<rdfs:subClassOf rdf:resource="&LPISPubs;Entry"/>
</rdfs:Class>
<rdfs:Class rdf:about="&LPISPubs;ConferenceProceedings">
<rdfs:subClassOf rdf:resource="&LPISPubs;Entry"/>
</rdfs:Class>
<rdfs:Class rdf:about="&LPISPubs;Author">
<rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdf:Property rdf:about="&LPISPubs;authors">
<rdfs:domain rdf:resource="&LPISPubs;Entry"/>
<rdfs:range rdf:resource="&rdf;List"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;authorName">
<rdfs:domain rdf:resource="&LPISPubs;Author"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;authorMiddleName">
<rdfs:domain rdf:resource="&LPISPubs;Author"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;authorSurName">
<rdfs:domain rdf:resource="&LPISPubs;Author"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;authorURL">
<rdfs:domain rdf:resource="&LPISPubs;Author"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;authorEmail">
<rdfs:domain rdf:resource="&LPISPubs;Author"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;keywords">
<rdfs:domain rdf:resource="&LPISPubs;Entry"/>
<rdfs:range rdf:resource="&rdf;Seq"/>
</rdf:Property>
```

```
<rdf:Property rdf:about="&LPISPubs;publicationTitle">
<rdfs:domain rdf:resource="&LPISPubs;Entry"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;mediaTitle">
<rdfs:domain rdf:resource="&LPISPubs;Entry"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;mediaPublisher">
<rdfs:domain rdf:resource="&LPISPubs;Entry"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;mediaEditors">
<rdfs:domain rdf:resource="&LPISPubs;Entry"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;mediaVolInfo">
<rdfs:domain rdf:resource="&LPISPubs;Entry"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;publicationYear">
<rdfs:domain rdf:resource="&LPISPubs;Entry"/>
<rdfs:range rdf:resource="&xsd;integer"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;publicationPagesInMedium">
<rdfs:domain rdf:resource="&LPISPubs;Entry"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;publicationAbstract">
<rdfs:domain rdf:resource="&LPISPubs;Entry"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&LPISPubs;publicationFileName">
<rdfs:domain rdf:resource="&LPISPubs;Entry"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
```

Παράρτημα Β

Το δεύτερο παράρτημα περιλαμβάνει α)το αρχείο που περιέχει τους κανόνες με βάση τους οποίους πραγματοποιείται η αναζήτηση στα βιβλιογραφικά δεδομένα β) το αρχείο που χειρίζεται την επικοινωνία του συστήματος με τον χρήστη (διεπαφή χρήστη) και γ) το αρχείο που χειρίζεται την εκτέλεση της αναζήτησης και την επιστροφή των αποτελεσμάτων.

1. Το αρχείο κανόνων rules.pl

```
:- module(rules,
       [ publicationtitle/2,
         publicationpublisher/2,
         author/3,
         publicationkeyword/2,
         publicationyear/3,
         publicationtype/2,
         publicationdata/2
       ]).
:rdf_register_ns(pubrdf,'http://lpis.csd.auth.gr/publications.rdfs#',
[force(true)]).
:-rdf_register_ns(pub,'__file://c:/publications.rdf#',[force(true)]).
:-rdf_register_ns(xsd,'http://www.w3.org/2001/XMLSchema#',[force(true)]).
:-rdf_assert(rdf:'Container',rdfs:'subClassOf',rdf:'Resource').
:-rdf_assert(rdf:'Seq',rdfs:'subClassOf',rdf:'Container').

:-rdf_load('c:/publications.rdf').

publicationtitle(Title,List):-
      findall(Entry,rdf(Entry,pubrdf:'publicationTitle',literal(Title)),List).

publicationpublisher(Publisher,List):-
       findall(Entry,rdf(Entry,pubrdf:'mediaPublisher',
       literal(Publisher)),List).

isAuthor(AuthorID,Surname,Name):-
       rdf(AuthorID,pubrdf:'authorSurName',literal(Surname)),
       (rdf(AuthorID,pubrdf:'authorName',literal(Name)) -> true; Name='').

author(AuthorSurname,0,List):-
       isAuthor(AuthorID,AuthorSurname,_AuthorName),
       findall(Entry,(rdf(Entry,pubrdf:'authors',AuthorsList),
       rdf(AuthorsList,rdf:'type',rdf:'List'),
       rdfs_list_to_prolog_list(AuthorsList,PrList),
       member(AuthorID,PrList)),List).
author(AuthorSurname,Position,List):-
       Position\=0,
       isAuthor(AuthorID,AuthorSurname,_AuthorName),
       findall(Entry,(rdf(Entry,pubrdf:'authors',AuthorsList),
       rdf(AuthorsList,rdf:'type',rdf:'List'),
       rdfs_list_to_prolog_list(AuthorsList,PrList),
       nth1(Position,PrList,AuthorID)),List).

publicationkeyword(Keyword,List):-
       findall(Entry,(rdf(Entry,pubrdf:'keywords',Description),
       rdf(Description,rdf:'type',rdf:'Seq'),
       rdfs_member(literal(Keyword),Description)),List).

publicationyear(PubYear,Operator,List):-
       atom_number(PubYear,PublicationYear),
```

```
       Comparison=..[Operator,PYear,PublicationYear],
       findall(Entry,(rdf(Entry,pubrdf:'publicationYear',
       literal(type(xsd:'integer',Year))),atom_number(Year,PYear),
       call(Comparison)),List).

publicationtype(Type,List):-
       rdf_global_term(pubrdf:Type, XXX),
       findall(Entry,rdf(Entry,rdf:type,XXX),List).

listauthors(Entry,AList):-
       rdf(Entry,pubrdf:'authors',List),
       rdf(List,rdf:'type',rdf:'List'),
       rdfs_list_to_prolog_list(List,AList).

authornames([],AuthorNames,Authors):-
       Authors=AuthorNames.

authornames([Head|Tail],HList,Authors):-
       isAuthor(Head,Surname,Name),
       string_concat(Surname,' ',SurnameF),
       string_concat(SurnameF,Name,FullName),
       append(HList,[FullName],AuthorNames),
       authornames(Tail,AuthorNames,Authors).

list_to_string([],MStr,Str):-
       Str=MStr.
list_to_string([H|T],MS,FStr):-
       string_to_atom(H,''),
       list_to_string(T,MS,FStr).
list_to_string([H|T],MS,FStr):-
       string_concat(MS,H,Str1),
       string_concat(Str1,', ',Str2),
       list_to_string(T,Str2,FStr).


publicationdata(Entry,ResultStr):-
       listauthors(Entry,AList),
       authornames(AList,[],Authors),
       (rdf(Entry,pubrdf:'publicationTitle',literal(ATitle))
       ->
       true ; ATitle=''),
       string_to_atom(Title,ATitle),
       append(Authors,[Title],List1),
       (rdf(Entry,pubrdf:'mediaTitle',literal(MediaTitle))
       ->
       true ; MediaTitle=''),
       string_to_atom(MediaTitle1,MediaTitle),
       append(List1,[MediaTitle1],List2),
       (rdf(Entry,pubrdf:'mediaEditors',literal(MediaEditors))
       ->
       true ; MediaEditors=''),
       string_to_atom(MediaEditors1,MediaEditors),
       append(List2,[MediaEditors1],List3),
       (rdf(Entry,pubrdf:'mediaPublisher',literal(MediaPublisher))
       ->
       true ; MediaPublisher=''),
       string_to_atom(MediaPublisher1,MediaPublisher),
       append(List3,[MediaPublisher1],List4),
       (rdf(Entry,pubrdf:'mediaVolInfo',literal(MediaVolInfo))
       ->
       true ; MediaVolInfo=''),
       string_to_atom(MediaVolInfo1,MediaVolInfo),
       append(List4,[MediaVolInfo1],List5),
       (rdf(Entry,pubrdf:'publicationPagesInMedium',literal(PublicationPagesIn
       Medium))
       ->
       true ; PublicationPagesInMedium=''),
       string_to_atom(PublicationPagesInMedium1,PublicationPagesInMedium),
       append(List5,[PublicationPagesInMedium1],List6),
```

```
       (rdf(Entry,pubrdf:'publicationLocation',literal(PublicationLocation))
       ->
       true ; PublicationLocation=''),
       string_to_atom(PublicationLocation1,PublicationLocation),
       append(List6,[PublicationLocation1],List7),
       (rdf(Entry,pubrdf:'publicationYear',literal(type(_,APublicationYear)))
       ->
       true ; APublicationYear=''),
       string_to_atom(PublicationYear,APublicationYear),
       append(List7,[PublicationYear],List),
       list_to_string(List,'',ResultStr).
```

2. Το αρχείο http_user.pl

```
:- module(http_user,
         [ reply_user/1
         ]).
:- use_module(server).
:- use_module(library('http/http_open')).
:- use_module(library('http/thread_httpd')).
:- use_module(library('http/html_write')).
:- use_module(library('http/mimetype')).
:- use_module(http_data).
:- use_module(parms).
:- use_module(semweb(rdf_db)).

reply_user(Request) :-
       memberchk(path(Path), Request),
       reply(Path, Request).

:- discontiguous
       reply/2.

reply('/', _Request) :- !,
       (   setting(title(Title))
       ->
       true ; Title = 'Σύστημα αναζήτησης επιστημονικών δημοσιεύσεων'
       ),
       phrase(html([ head(title(Title)),
              h3(align(center),'Φόρμα αναζήτησης δημοσιεύσεων'),
              form([ name(query),
                     action('../evaluateQuery'),
                     method('GET')
                   ],
                   [ table([align(center),bgcolor(gray)],
                           [tr([td('Ετος'),
                                td(\operator),
                                td(input([ type(text),
                                           name(publicationyear)
                                         ])
                                   )
                               ]),
                            tr([td('Τίτλος'),
                                td(input([ type(text),
                                           name(publicationtitle)
                                         ])
                                   )
                               ]),
                            tr([td('Τύπος δημοσίευσης'),
                                td(\type)
                              ]),
                            tr([td('Όνομα εκδότη'),
                                td(input([ type(text),
                                           name(publicationpublisher)
                                         ])
```

```
                                    )
                                ]),
                               tr([td('Θέση στην λίστα συγγραφέων'),
                                   td(\position),
                                   td('Όνομα συγγραφέα'),
                                   td(input([ type(text),
                                              name(author)
                                            ])
                                     )
                                ]),
                               tr([td('Λέξη κλειδί'),
                                   td(input([ type(text),
                                              name(publicationkeyword)
                                            ])
                                     )
                                ]),
                               tr([td('Τύπος αναζήτησης'),
                                   td(\searchtype)
                                ]),
                               tr([td(input([ type(submit),
                                              value('Αναζήτηση')
                                            ])
                                     ),
                                   td(input([type(reset),
                                             value('Καθαρισμός πεδίων')
                                            ])
                                     )
                                ])
                              ])
                         ])
                  ]),HTML),
      format('Content-type: text/html~n~n'),
      print_html(HTML).
```

```
operator -->
      html(select(name(operator),
                  [option([value(99)],''),
                   option([value(=)],'Κατά την διάρκεια του'),
                   option([value(<)],'Πριν το'),
                   option([value(=<)],'Πριν ή κατά την διάρκεια'),
                   option([value(>)],'Μετά το'),
                   option([value(>=)],'Κατά την διάρκεια ή και μετά το')
                  ]
                )
           ).

type -->
      html(select(name(publicationtype),
                  [option([value(any)],'Οποιοσδήποτε'),
                   option([value(conferenceproceedings)],
                          'Conference proceedings'),
                   option([value(workshopproceedings)],
                          'Workshop proceedings'),
                   option([value(symposiumproceedings)],
                          'Symposium proceedings'),
                   option([value(technicalreport)],
                          'Technical report'),
                   option([value(book)],'Book'),
                   option([value(bookchapter)],'Book chapter'),
                   option([value(journal)],'Journal'),
                   option([value(phd)],'PhD thesis'),
                   option([value(bsc)],'BSc thesis'),
                   option([value(msc)],'MSc thesis')
                  ]
                  )
          ).
```

```
position -->
       html(select(name(position),
                   [option([value(99)],''),
                    option([value(0)],'Οποιαδήποτε'),
                    option([value(1)],'Πρώτος'),
                    option([value(2)],'Δεύτερος'),
                    option([value(3)],'Τρίτος')
                   ]
                   )
            ).

searchtype -->
       html(select(name(searchtype),
                   [option([value(and)],'Με λογική σύζευξη κριτηρίων'),
                     option([value(or)],'Με λογική διάζευξη κριτηρίων')
                   ]
                    )
           ).
```

3. Το αρχείο http_data.pl

```
:- module(sesame_http_body,
         [ sesame_reply_body/1
         ]).

:- use_module(parms).
:- use_module(library('http/http_parameters')).
:- use_module(rdf_html).
%:- use_module(rdf_io).
%:- use_module(semweb(rdf_edit)).
:- use_module(semweb(rdfs)).
:- use_module(semweb(rdf_db)).
:- use_module(library('http/html_write')).
:- use_module(library('http/http_open')).
:- use_module(library(memfile)).
%:- use_module(library(rdf_ntriples)).
%:- use_module(library(debug)).
:-use_module(rules).


sesame_reply_body(Request) :-
       memberchk(path(Path), Request),
       reply(Path, Request).

:- discontiguous
       reply/2.

reply('/evaluateQuery', Request) :- !,
        http_parameters(Request,
                        [
                          operator(Operator),
                          publicationyear(Publicationyear),
                          publicationtitle(Publicationtitle),
                          publicationtype(Publicationtype),
                          publicationpublisher(Publicationpublisher),
                          position(Position),
                          author(Author),
                          publicationkeyword(Publicationkeyword),
                          searchtype(Searchtype)
                        ],
                        [ attribute_declarations(attribute_decl), form_data(Data)
                        ]),

       define_criteria(Data,[],[],[],Criteria,Values,Ops),
       make_query(Criteria,Values,Ops,[],Terms),
```

```
       run_query(Terms,Matches1),
       combine_results(Searchtype,Matches1,Matches),
       build_html(Matches,Results),
       reply_page(Results).

       define_criteria([],C,V,O,Criteria,Values,Ops):-
       Ops=O,
       Criteria=C,
       Values=V.

       define_criteria(L,C,V,O,Criteria,Values,Ops):-
              L=[Name=Value|T],
              ((Name=='operator',Value\=='99'|Name=='position',Value\=='99')
              ->
              (atom_number(Value,Val),append(O,[Val],O1));O1=O ),
              ((Name\=='operator',Name\=='position',Name\=='searchtype',
              Name\=='publicationtype',Value\=='',Value\=='any')
              ->
                     (append(C,[Name],C1),append(V,[Value],V1));
                     ((Name=='publicationtype',Value\=='any')
                     ->
                            (convert_case(Value,Val),append(C,[Name],C1),
                            append(V,[Val],V1) );C1=C,V1=V)),
              define_criteria(T,C1,V1,O1,Criteria,Values,Ops).


       make_query(Criteria,Values,Ops,[],Terms):-
              Criteria=[Criterion | T],
              Values=[Value | R],
              ((Criterion=='publicationyear'|Criterion=='author')
              ->
              (Ops=[Op|Q],Term=..[Criterion,Value,Op,Matches]);
              (Term=..[Criterion,Value,Matches],Q=Ops)),
              F=[Term],
              make_query(T,R,Q,F,Terms).

make_query(Criteria,Values,Ops,F,Terms):-
       Criteria=[Criterion | T],
       Values=[Value | R],
       ((Criterion=='publicationyear'|Criterion=='author')
       ->
       (Ops=[Op|Q],Term=..[Criterion,Value,Op,Matches]);
       (Term=..[Criterion,Value,Matches],Q=Ops)),
       append(F,[Term],Ft),
       make_query(T,R,Q,Ft,Terms).

make_query([],Values,Ops,F,Terms):-
       Terms=F.

run_query([],[]):- !.

run_query([H|T],[M1|M2]):-
       call(H),
       H =.. L,
       last(L,M1),
       run_query(T,M2).


combine_results(and,MatchesIn,MatchesOut) :-
       multi_intersection(MatchesIn,MatchesOut).

combine_results(or,MatchesIn,MatchesOut) :-
       multi_union(MatchesIn,MatchesOut).

multi_intersection([H],H).

multi_intersection([H|T],R) :-
       multi_intersection(T,T1),
```

```
       intersection(H,T1,R).

multi_union([],[]).

multi_union([H|T],R) :-
       multi_union(T,T1),
       union(H,T1,R).

upcase_first(A,Au):-
       sub_atom(A,0,1,After,H1),
       sub_atom(A,1,Len,0,H2),
       upcase_atom(H1,H3),
       atom_concat(H3,H2,Au).

convert_case(S,Su):-
       ((atom_concat(A,'proceedings',S)|atom_concat(A,'report',S)|
       atom_concat(A,'chapter',S) )
       ->
       atom_concat(A,B,S),upcase_first(A,A1),upcase_first(B,B1),
       atom_concat(A1,B1,Su) ;
       (S=='phd'
              ->
              Su='PhD';( S=='msc'
                     ->
                     Su='MSc';( S=='bsc'
                            ->
                            Su='BSc' ;
                            upcase_first(S,Su)
                               )
                        )
       )).

build_html(Matches,Results) :-
       build_html_aux(Matches,R),
       Results =
       page([ title(['Αποτελέσματα αναζήτησης'])
                           ],
                    [ h2(align(center),['Αποτελέσματα αναζήτησης']),
                     ul(b('Δημοσιεύσεις που βρέθηκαν')) | R ]).

build_html_aux([],[]) :- !.

build_html_aux([H|T],[p(S)|Rest]) :-
       publicationdata(H,S),
       build_html_aux(T,Rest).


attribute_decl(operator,
              [ default('='),
               type(oneof(['=','<','<=','>','>=']))
              ]).
attribute_decl(publicationyear,
              [ optional(true)
              ]).
attribute_decl(publicationtitle,
              [ optional(true)
              ]).
attribute_decl(publicationtype,
              [ type(oneof(['any',
                            'conferenceproceedings',
                            'workshopproceedings',
                            'symposiumproceedings',
                            'technicalreport','journal','book',
                            'bookchapter',
                            'bsc',
                            'msc',
                            'phd'])),
               optional(true)
```

```
              ]).
attribute_decl(publicationpublisher,
              [
               optional(true)
              ]).
attribute_decl(author,
              [
               optional(true)
              ]).
attribute_decl(position,
              [
               default('0'),
               type(oneof(['0', '1','2','3'])),
               number
              ]).
attribute_decl(publicationkeyword,
              [
               optional(true)
              ]).
attribute_decl(searchtype,
              [
               optional(true),
               type(oneof(['and', 'or']))
              ]).


reply_page(Results) :-
       phrase(Results, HTML),
       format('Content-type: text/html~n~n'),
       print_html(HTML).
```