ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Μετατροπή Datalog Βάσης Γνώσης σε OWL Οντολογία και αντίστροφα

Δημήτριος Χ. Σιδέρης

Επιβλέπουσα **Ιζαμπώ Καράλη,** Επίκουρη Καθηγήτρια

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2019

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Μετατροπή Datalog Βάσης Γνώσης σε OWL Οντολογία και αντίστροφα

Δημήτριος Χ. Σιδέρης
Α.Μ.: 1115201400182

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Ιζαμπώ Καράλη,** Επίκουρη Καθηγήτρια

ΠΕΡΙΛΗΨΗ

Η Datalog είναι μια γλώσσα που η λογική της βασίζεται σε ένα υποσύνολο της λογικής πρώτης τάξης, στις προτάσεις Horn, και την σχεσιακή άλγεβρα. Η OWL είναι μια οικογένεια γλωσσών που η λογική της βασίζεται στις λογικές περιγραφών και απότελεί ένα σημαντικό χαρακτηριστικό του Σημασιολογικού Ιστού. Και οι δύο γλώσσες χρησιμοποιούνται για την απόθήκευση και την αναζήτηση γνώσης. Ωστόσο οι μεθοδολογίες αυτές έχουν τόσο κοινά σημεία όσο και διαφορές.

Στα πλαίσια αυτής της πτυχιακής, θα θεωρήσουμε ένα υποσύνολο και των δύο γλωσσών που μπορεί να μετατραπεί από την μία γλώσσα στην άλλη. Στην συνέχεια θα αναπτύξουμε τις μεθοδολογίες που απαιτούνται για τις μετατροπές αυτές. Κατά την πτυχιακή αυτή, αναπτύχθηκαν και δύο λογισμικά που λαμβάνουν ως είσοδο δεδομένα της μίας μεθοδολογίας και τα μετατρέπουν στην άλλη.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Σημασιολογικός Ιστός

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: OWL, Datalog, Λογικές Περιγραφών, Προτάσεις Horn, Μεταγλώττιση

ABSTRACT

Datalog is a language whose logic is based on a subset of first-order logic, Horn Clauses, and relational algebra. OWL is a family of languages whose logic is based on Description Logics and is an important feature of the Semantic Web. Both languages are used for storing and searching for knowledge. However, these methodologies have both similarities and differences.

Within this thesis, we will consider a subset of both languages that can be converted from one language to another. We will then develop the methodologies required for these conversions. During this thesis, two applications were developed that take data from one methodology as input and convert it to the other.

SUBJECT AREA: Knowledge Bases

KEYWORDS: OWL, Datalog, Description Logics, Horn Clauses, Knowledge Base Transformation

Η πτυχιακή αυτή εργασία αφιερώνεται στην οικογένειά μου.

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας Πτυχιακής Εργασίας θα ήθελα να ευχαριστήσω την επιβλέπουσα, επίκουρη καθηγήτρια Ιζαμπώ Καράλη, για τη συνεργασία, την εμπιστοσύνη, την επιμονή και την πολύτιμη συμβολή της στην ολοκλήρωση της.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εργασία με τίτλο “Μετατροπή δεδομένων Datalog σε OWL και το αντίστροφο” εκπονήθηκε για το τμήμα Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών - ΕΚΠΑ.

1. ΕΙΣΑΓΩΓΗ

Η Datalog είναι μία γλώσσα αναπαράστασης και επερωτήσεων σε βάσεις δεδομένων. Βασίζεται και αποτελεί συντακτικό υποσύνολο της Prolog, αλλά διαφέρει σημαντικά από αυτήν καθώς η λογική της είναι πιο κοντά στην σχεσιακή άλγεβρα από τον λογικό προγραμματισμό. Η OWL είναι μία οικογένεια γλωσσών που χρησιμοποιείται για την αναπαράσταση γνώσης στο περιβάλλον του Παγκόσμιου Ιστού. Η λογική της βασίζεται στην έννοια της οντολογίας, μίας δομής αναπαράστασης εννοιών. αποτελεί σημαντικό στοιχείο του Σημασιολογικού Ιστού. Και οι δύο γλώσσες χρησιμοποιούνται για την αποθήκευση και την αναζήτηση γνώσης, απλά χρησιμοποιούν διαφορετικές μεθόδους αναπαράστασης. Υπάρχουν όμως πολλοί λόγοι για να μετατρέψει κανείς δεδομένα από την μία γλώσσα στην άλλη. Αρχικά σε Datalog, ως παλαιότερη γλώσσα, έχουν αναπτυχθεί πολλές βάσεις γνώσης, οπότε και είναι σίγουρα χρήσιμη η γρήγορη και αποτελεσματική μετάφραση των ήδη υπαρχουσών βάσεων. Επίσης είναι πιθανό να επιθυμήσει κάποιος να κάνει ενοποίηση των OWL συστημάτων σε Datalog ή να εκμεταλλευτεί μεθοδολογίες της Datalog για οντολογίες της OWL. Όλοι οι παραπάνω λόγοι κάνουν χρήσιμο τον ορισμό κανόνων και την κατασκευή λογισμικού που μετατρέπει δεδομένα από τον ένα φορμαλισμό στον άλλο.

Η πτυχιακή αυτή βασίστηκε κυρίως στο paper των Grosof, Horrocks, Volz, Decker [13]. Το συγκεκριμένο paper είχε ως στόχο το DLP-Fusion, δηλαδή την ορισμό του υποσυνόλου των λογικών περιγραφών (DL) που να μπορεί να αντιστοιχηθεί σε ένα υποσύνολο του λογικού προγραμματισμού (LP). Αρχικά στο paper ορίζονται τα βασικά σημεία των δύο λογικών. Στην συνέχεια επιβάλλονται εκφραστικοί περιορισμοί στην συνένωση τους. Τέλος ορίζονται οι κανόνες και τα αξιώματα των δύο λογικών που είναι εκφραστικά ισοδύναμοι και ορίζονται μερικοί βασικοί κανόνες για την έκφραση της ιδιότητας της αναδρομής.

Στα πλαίσια της πτυχιακής χρησιμοποιήθηκε αυτό το υποσύνολο, καθώς η Datalog βασίζεται στις προτάσεις Horn και, κατ' επέκταση στον λογικό προγραμματισμό και η OWL στις λογικές περιγραφών. Ωστόσο αυτό το υποσύνολο επαυξήθηκε με ένα υποσύνολο κανόνων της Datalog με n-ary κατηγορήματα, το οποίο υπακούει όμως σε πολλούς περιορισμούς. Το υποσύνολο αυτό μεταφράστηκε σε εκφράσεις της Datalog και της OWL. Επίσης φτιάχτηκε ένα σύστημα μετατροπής δεδομένων Datalog σε OWL, καθώς και ένα σύστημα που μετατρέπει δεδομένα από OWL σε Datalog. Τα δύο συστήματα αποτελούν παραδοσιακοί μεταγλωττιστές που στην έξοδο τους παράγουν OWL ή Datalog πρόγραμμα, αντίστοιχα.

Η πτυχιακή εργασία χωρίζεται σε 8 κεφάλαια:

- Το πρώτο κεφάλαιο περιγράφει την λογική πρώτης τάξης (FOL)
- Το δεύτερο κεφάλαιο περιγράφει ένα υποσύνολο της FOL, τις προτάσεις Horn (Horn Clauses), στην λογική των οποίων βασίζεται ο λογικός προγραμματισμός και η Datalog
- Το τρίτο κεφάλαιο περιγράφει την Datalog σαν γλώσσα. Αναφέρει το συντακτικό της και τις βασικές διαφορές της από την Prolog, οι οποίες και την καθιστούν την γλώσσα που χρησιμοποιείται στα πλαίσια της πτυχιακής αυτής
- Το τέταρτο κεφάλαιο περιγράφει την έννοια του Σημασιολογικού Ιστού
- Το πέμπτο κεφάλαιο περιγράφει την λογική των Λογικών Περιγραφών

- Το έκτο κεφάλαιο περιγράφει την OWL σαν οικογένεια γλωσσών. Αναλύει την ιστορία της, αναφέρει τις γλώσσες που ανήκουν σε αυτή και δείχνει τις πολλές συντάξεις της.
- Το έβδομο κεφάλαιο περιγράφει με μεγάλη λεπτομέρεια την λογικής της μετατροπής δεδομένων των δύο γλωσσών από την μία στην άλλη. Αρχικά ορίζει τους εκφραστικούς περιορισμούς αυτής της λογικής και ορίζει την μετατροπή n-ary σε binary κατηγορημάτων που χρησιμοποιείται στα πλαίσια αυτής της πτυχιακής (Σημ. η μετατροπή αυτή δεν είναι μοναδική). Στην συνέχεια ορίζεται λεπτομερώς το υποσύνολο μετατροπής και οι κανόνες για την αναδρομή. Μετά περιγράφονται πολύ αναλυτικά τα δύο λογισμικά που κατασκευάστηκαν και το περιβάλλον τους. Στο πλαίσιο αυτό δίνονται οι γραμματικές σε BNF μορφή των δυο υποσυνόλων των γλωσσών που μετατρέπονται από την μία στην άλλη και δίνονται τεχνικές λεπτομέρειες.
- Το όγδοο και τελευταίο κεφάλαιο περιγράφει τα προβλήματα που προέκυψαν κατά την διάρκεια της πτυχιακής και δεν έχουν ακόμα επιλυθεί.

2. ΛΟΓΙΚΗ ΠΡΩΤΗΣ ΤΑΞΗΣ

2.1 Εισαγωγή

Η λογική πρώτης τάξης (First Order Logic – FOL) είναι μια μορφή λογικής με εφαρμογές στα μαθηματικά, τη φιλοσοφία, τη γλωσσολογία και την επιστήμη των υπολογιστών. Τα θεμέλιά της αναπτύχθηκαν ανεξάρτητα από τους Gottlob Frege και Charles Sanders Peirce τον 19ο αιώνα. Αποτελεί μια εκφραστική λογική που καλύπτει πολλές από τις απαιτήσεις που έχουμε από μια γλώσσα αναπαράστασης γνώσης.

Το επίθεμα “πρώτης τάξης” διακρίνει τη λογική πρώτης τάξης από λογικές υψηλότερης τάξης, στις οποίες υπάρχουν κατηγορήματα που έχουν κατηγορήματα ή συναρτησιακά σύμβολα, σαν ορίσματα συναρτησιακών συμβόλων. Στην λογική πρώτης τάξης, τα κατηγορήματα συσχετίζονται συχνά με σύνολα. Σε λογικές υψηλότερης τάξης, τα κατηγορήματα μπορούν να ερμηνευτούν ως σύνολα συνόλων.

Υπάρχουν πολλά συμπερασματικά συστήματα για τη λογική πρώτης τάξης που είναι τόσο συνεπή (όλες οι αποδεδειγμένες δηλώσεις είναι αληθείς σε όλα τα μοντέλα) όσο και πλήρη (όλες οι δηλώσεις που είναι αληθινές σε όλα τα μοντέλα είναι αποδείξιμες). Αν και η λογική πρώτης τάξης είναι ημιαποφασίσιμη, έχει σημειωθεί μεγάλη πρόοδος στο θεωρία αποδείξεών της. Η λογική πρώτης τάξης ικανοποιεί επίσης αρκετά μεταλογικά θεωρήματα, όπως το Θεώρημα Löwenheim-Skolem [24] και το θεώρημα πληρότητας του Gödel [25]. [1]

2.2 Βασικά

Ενώ η προτασιακή λογική (propositional logic) ασχολείται με απλές δηλωτικές προτάσεις, η λογική πρώτης τάξης καλύπτει επιπλέον τα κατηγορήματα και τους ποσοδείκτες.

Η λογική πρώτης τάξης χρησιμοποιεί μεταβλητές πάνω σε αντικείμενα και επιτρέπει τη χρήση προτάσεων που περιέχουν μεταβλητές, έτσι ώστε αντί για προτάσεις όπως *“ο Σωκράτης ένας άνθρωπος”* να μπορεί να έχει εκφράσεις με τη μορφή *“υπάρχει x τέτοιο ώστε το x είναι ο Σωκράτης και x είναι ένας άνθρωπος”* με το *“υπάρχει”* να είναι ένας ποσοδείκτης ενώ το *x* είναι μια μεταβλητή. Γενικώς η λογική πρώτης τάξης βασίζει την οντολογική λογική της στην υπόθεση ότι ο κόσμος αποτελείται από αντικείμενα (objects) πχ ο Γιώργος, η Μαίρη, που συμμετέχουν σε σχέσεις (relations) και συναρτησιακά σύμβολα (function symbols) πχ ο Γιώργος είναι δυνατός, ο Γιώργος αγαπάει την Μαίρη. Οι σχέσεις μεταξύ αντικειμένων μπορεί να αληθεύουν η όχι για τον κόσμο.

Οι σχέσεις μεταξύ των κατηγορημάτων μπορούν να δηλωθούν χρησιμοποιώντας λογικους συνδέσμους. Σκεφτείτε, για παράδειγμα, τον τύπο της πρώτης τάξης *"εάν ο α είναι φιλόσοφος, τότε είναι και διανοούμενος"*. Αυτός ο τύπος είναι μια υπό όρους δήλωση με τον όρο “*ο α είναι φιλόσοφος*” ως υπόθεση και τον όρο “*ο α είναι διανοούμενος*” ως συμπέρασμα του. Η αλήθεια αυτού του τύπου εξαρτάται από το αντικείμενο που υποδηλώνεται από το *α*, και από τις ερμηνείες των όρων “*είναι φιλόσοφος*” και “*είναι διανοούμενος*”.

Οι ποσοδείκτες μπορούν να εφαρμοστούν σε μεταβλητές σε έναν τύπο. Η μεταβλητή *α* στον προηγούμενο τύπο μπορεί να ποσοτικοποιηθεί καθολικά, για παράδειγμα, με τη φράση πρώτης τάξης *“Για κάθε α, εάν α είναι ένας φιλόσοφος, τότε είναι και διανοούμενος”*. Ο καθολικός ποσοδείκτης *“για κάθε”* στη φράση αυτή εκφράζει την ιδέα ότι ο ισχυρισμός *“αν κάποιος είναι φιλόσοφος, τότε είναι και διανοούμενος”* ισχύει για *όλες τις* επιλογές *κάποιου* α.

Η *άρνηση* της πρότασης *“Για κάθε α, αν α είναι ένας φιλόσοφος, τότε α είναι ένας διανοούμενος”* είναι λογικά ισοδύναμη με την φράση *“Υπάρχει α τέτοιο ώστε α είναι ένας φιλόσοφος και α δεν είναι ένας διανοούμενος”*. Ο υπαρξιακός ποσοδείκτης *"υπάρχει"* εκφράζει την ιδέα ότι ο ισχυρισμός *"ο α είναι φιλόσοφος και α δεν είναι διανοούμενος"* ισχύει για *οποιονδήποτε έναν α*.

Οι όροι *"είναι φιλόσοφος"* και *"είναι διανοούμενος"* λαμβάνουν μία μόνο μεταβλητή. Σε γενικές γραμμές, τα κατηγορήματα μπορούν να πάρουν πολλές μεταβλητές. Στην φράση *"Ο Σωκράτης είναι ο δάσκαλος του Πλάτωνα"*, το κατηγόρημα *"είναι ο δάσκαλος του"* παίρνει δύο μεταβλητές. [1]

2.3 Σύνταξη

Υπάρχουν δύο βασικά μέρη της λογικής πρώτης τάξης. Η σύνταξη καθορίζει ποιες συλλογές συμβόλων ορίζουν εκφράσεις στη λογική πρώτης τάξης, ενώ η σημασιολογία καθορίζει το νόημα πίσω από αυτές τις εκφράσεις.

2.3.1 Αλφάβητο

Σε αντίθεση με τις φυσικές γλώσσες, όπως η ελληνική γλώσσα, η λογική πρώτης τάξης είναι εντελώς τυπική, ώστε να μπορεί να προσδιοριστεί μηχανικά αν ορίζεται μια δεδομένη έκφραση. Υπάρχουν δύο βασικοί τύποι εκφράσεων: *οι όροι*, οι οποίοι διαισθητικά αντιπροσωπεύουν αντικείμενα και *οι τύποι*, οι οποίοι εκφράζουν διαισθητικά σχέσεις που μπορεί να είναι αληθείς ή ψευδείς. Οι όροι και οι τύποι είναι σειρές συμβόλων, όπου όλα τα σύμβολα σχηματίζουν το αλφάβητο της γλώσσας. Όπως συμβαίνει με όλες τις τυπικές γλώσσες, η φύση των ίδιων των συμβόλων είναι εκτός του πλαισίου της τυπικής λογικής. Συχνά θεωρούνται ως γράμματα και σύμβολα στίξης.

Το αλφάβητο αποτελείται απο λογικούς συνδέσμους, που έχουν πάντα το ίδιο νόημα και αλλα σύμβολα, των οποίων η έννοια ποικίλλει από την ερμηνεία. Για παράδειγμα, το σύμβολο ∧ είναι ενας λογικός σύνδεσμος και αντιπροσωπεύει πάντα το "και" δεν ερμηνεύεται ποτέ ως "ή". Από την άλλη πλευρά, ένα σύμβολο κατηγορηματικού τύπου όπως ο Phil(*x*) θα μπορούσε να ερμηνευθεί ως " *x* είναι φιλόσοφος", " *x* είναι ένας άνθρωπος που ονομάζεται Philip" ή οποιοδήποτε άλλο κατηγόρημα.

2.3.1.1 Λογικοί σύνδεσμοι

Ττο αλφάβητο περιλαμβάνει τα:

- Τα σύμβολα ποσοδεικτών: ∀ (καθολικός ποσοδείκτης) και ∃ (υπαρξιακός ποσοδείκτης)
- Οι λογικοί σύνδεσμοι: ∧ για σύζευξη, ∨ για διαζευξη, → για συνεπαγωγή, ↔ για διπλή συνεπαγωγή, ¬ για άρνηση.
- Παρενθέσεις, παρενθέσεις και άλλα σύμβολα στίξης. Η επιλογή τέτοιων συμβόλων ποικίλει ανάλογα με το πλαίσιο.
- Ένα άπειρο σύνολο μεταβλητών, που συχνά υποδηλώνεται με πεζά γράμματα στο τέλος του αλφαβήτου *x*, *y*, *z*,.... Συχνά χρησιμοποιούνται δείκτες για τη διάκριση μεταβλητών: *x*0, *x*1, *x*2,....
- Ένα σύμβολο ισότητας (μερικές φορές, σύμβολο ταυτότητας) =. [1] [2]

2.3.1.2 Άλλα σύμβολα

Άλλα σύμβολα που χρησιμοποιούνται είναι οι σταθερές, συναρτήσεις και τα κατηγορήματα. Δεν έχουν πάντα σταθερή σημασία, όπως προαναφέραμε. Συνήθως ορίζονται στην αγγλική γλώσσα. Τα συναρτησιακά σύμβολα, οι σταθερές και τα

κατηγορήματα ορίζονται με πρώτο γράμμα μικρό και οι μεταβλητές με κεφαλαίο. Ορίζονται τέσσερα σύνολα συμβόλων:

- Τα σύμβολα σταθερών, που αποτελεί ένα αριθμήσιμο απειροσύνολο. Παραδείγματα: bill,mary
- Τα σύμβολα μεταβλητών, που αποτελεί ένα αριθμήσιμο απειροσύνολο. Παραδείγματα X,Y
- Τα συναρτησιακά σύμβολα, που για κάθε θετικό αριθμό n έχουμε ένα σύνολο από n-ary συναρτήσεις. Παράδειγμα sine(X)
- Τα σύμβολα κατηγορημάτων όπου για κάθε θετικό αριθμό n έχουμε ένα σύνολο από n-ary κατηγορήματα. Παράδειγμα father(X,Y).

Τα τέσσερα σύνολα συνθέτουν ένα λεξιλόγιο (vocabulary). Χάρη σε αυτό μπορούμε να φτιάξουμε εκφράσεις της λογικής πρώτης τάξης. Οι εκφράσεις διακρίνονται σε όρους και τύπους.

2.3.1.2.1 Όροι

Οι όροι εκφράζουν αντικείμενα στον κόσμο. Όροι είναι όλες οι μεταβλητές, οι σταθερές και οι σύνθετοι όροι.

Παραδείγματα: mary, X, sine(X).

2.3.1.2.2 Τύποι

Οι τύποι εκφράζουν γεγονότα και καταστάσεις στον κόσμο που μοντελοποιούμε. Διακρίνονται ως εξής.

- Οι ατομικοί τύποι (atomic formulas) είναι τα απλά γεγονότα.

  Παράδειγμα: father(bob,john), john = elderSonOf(mary)

- Οι καλά διαμορφωμένοι τύποι (ή WFFs) είναι η γενική κατηγορία τύπων. Αυτοί αποτελούνται από ατομικούς τύπους συνδεδεμένους με σύζευξη ή διάζευξη καθώς και άρνηση και ποσοδείκτες.

  Παράδειγμα: (∃X)(ball(X) ∧ playsWith(paul, X))

2.3.2 Σημασιολογία

Μια ερμηνεία μιας γλώσσας πρώτης τάξης αποδίδει ένα νόημα σε κάθε σταθερά, συναρτησιακό σύμβολο και κατηγόρημα στη γλώσσα αυτή. Το αποτέλεσμα είναι ότι κάθε όρος έχει αντιστοιχηθεί σε ένα αντικείμενο που αντιπροσωπεύει, κάθε κατηγόρημα έχει αντιστοιχηθεί σε μια σχέση μεταξύ αντικειμένων και κάθε πρόταση έχει αντιστοιχηθεί με μια τιμή αλήθειας. Με τον τρόπο αυτό, μια ερμηνεία παρέχει σημασία στους όρους, τα κατηγορήματα και τους τύπους της γλώσσας.

Αρκετή πληροφορία από την φυσική γλώσσα μπορεί να εκφραστεί με την βοήθεια της λογικής πρώτης τάξης. Αυτό γίνεται κατανοητό αν αναλογιστούμε ότι:

- Ο κόσμος αποτελείται από αντικείμενα (σταθερές)
- Τα αντικείμενα έχουν σχέσεις με άλλα αντικείμενα (κατηγορήματα και συναρτησιακά σύμβολα)
- Οι προτάσεις που χρησιμοποιούμε αναφέρονται σε συγκεκριμένα αντικείμενα (όροι)

- Αυτά τα αντικείμενα έχουν απλές η πολύπλοκες σχέσεις μεταξύ τους που εκφράζονται μέσα από προτάσεις (τύποι)
- Οι προτάσεις μπορεί να είναι αληθείς η ψευδείς
- Συνήθως (όχι πάντα) τα ουσιαστικά αποτελούν σταθερές, τα επίθετα και τα ρήματα κατηγορήματα με τα επίθετα να είναι μοναδιαία κατηγορήματα και τα ρήματα σύνθετα.

Υπάρχουν και εκφράσεις όπως “κάθε”, “κάποιος” “αν...τότε” που επίσης μεταφράζονται σε λογική πρώτης τάξης. Για παράδειγμα το “κάθε” μεταφράζεται με καθολικό ποσοδείκτη, οι αόριστες αντωνυμίες με υπαρξιακό ποσοδείκτη και το “αν...τοτε” εκφράζει την συνεπαγωγή. Επίσης οι σύνδεσμοι “και”, “ή” παραπέμπουν σε σύζευξη και διάζευξη αντίστοιχα.

Παραδείγματα:

Ο bill είναι ο πατέρας του John = father(bill,john).

Ο bill είναι όμορφος = handsome(bill).

Κάποιος διέρρηξε το σπίτι μας = (∃X)(broke_in(X,our_place)).

Αν κάποιος είναι άνθρωπος τότε είναι θνητός (∀X) (human(X) → mortal(X))

Σημείωση: δεν είναι πάντα προφανής η μετάφραση από φυσική γλώσσα σε λογική πρώτης τάξης.

Παράδειγμα

Κάθε άνθρωπος κάνει όνειρα = (∀X)(human(X) → dreams(X)) [2]

2.4 Περιορισμοί

Η λογική πρώτης τάξης έχει πολλούς εκφραστικούς περιορισμούς. Αυτοί οφείλονται κυρίως στο ότι τα σύμβολά της εκφράζουν περιορισμένο μέρος των προτάσεων της φυσικής γλώσσας. Για παράδειγμα:

- Δεν μπορούμε να ορίσουμε ποσότητα (cardinality) στον υπαρξιακό ποσοδείκτη πχ υπάρχουν 3 άτομα που εργάζονται ως προγραμματιστές
- Δεν μπορούμε να ορίσουμε αν ένα μέγεθος είναι σχετικά μεγάλο η μικρό πχ ο Γιάννης είναι είναι πολύ δυνατός
- Δεν μπορούμε να εκφράσουμε χρονικές ακολουθίες πχ ο Δημήτρης πρώτα θα κοιμηθεί και μετά θα εργαστεί. [1]

2.5 Επέκταση ισότητας

Η πιο συνηθισμένη επέκταση της λογικής πρώτης τάξης, γνωστή ως λογική πρώτης τάξης με ισότητα, περιλαμβάνει το σύμβολο ισότητας ως ένα λογικό σύμβολο. Αυτή η προσέγγιση προσθέτει επίσης ορισμένα αξιώματα σχετικά με την ισότητα στο χρησιμοποιούμενο σύστημα έκφρασης. Αυτά τα αξίωμα ισότητας είναι:

- Ανακλαστικότητα. Για κάθε μεταβλητή x, $x = x$.
- Υποκατάσταση λειτουργιών. Για όλες τις μεταβλητές x και y και οποιοδήποτε σύμβολο λειτουργίας f,

  $x = y \rightarrow f(\ldots, x, \ldots) = f(\ldots, y, \ldots)$.
- Υποκατάσταση των τύπων. Για κάθε μεταβλητή x και y και κάθε τύπο φ (x), εάν φ' προκύπτει από την αντικατάσταση οποιουδήποτε αριθμού ελεύθερων

εμφανίσεων του *x* στο φ με *y*, έτσι ώστε αυτά να παραμένουν ελεύθερα περιστατικά *y*, τότε

$x = y \rightarrow (\varphi \rightarrow \varphi')$.

Αυτά είναι σχήματα (*schema*) αξιωμάτων, καθένα από τα οποία καθορίζει ένα άπειρο σύνολο αξιωμάτων. Το τρίτο σχήμα είναι γνωστό ως νόμος του Leibniz ή ως “η ταυτότητα των μη διακρίσημων”. Το δεύτερο σχήμα, που περιλαμβάνει το σύμβολο λειτουργίας *f*, είναι (ισοδύναμο) με μια ειδική περίπτωση του τρίτου σχήματος, χρησιμοποιώντας τον τύπο

$x = y \rightarrow (f (..., x,...) = z \rightarrow f (..., y,...) = z)$.

Πολλές άλλες ιδιότητες της ισότητας είναι απόροιες των παραπάνω αξιωμάτων, για παράδειγμα:

- Συμμετρία. Αν $x = y$ τότε $y = x$.
- Μεταβατικότητα. Αν $x = y$ και $y = z$ τότε $x = z$. [1]

3. HORN CLAUSES

3.1 Ορισμός

Οι προτάσεις Horn (Horn Clauses) είναι ένα υποσύνολο προτάσεων της λογικής πρώτης τάξης. Μια πρόταση Horn είναι μια σύζευξη πολλών λεκτικών όπου το πολύ ένα είναι θετικό.

Μια πρόταση Horn με ακριβώς ένα θετικό λεκτικό ορίζεται ως οριστική πρόταση Horn. Μια οριστική πρόταση χωρίς λεκτικά με άρνηση ονομάζεται γεγονός. Και μια πρόταση Horn χωρίς ένα θετικό λεκτικό ονομάζεται πρόταση στόχου (σημειώστε ότι η κενή πρόταση που αποτελείται από κανένα κατηγόρημα δεν είναι μια πρόταση στόχου). Αυτά τα τρία είδη προτάσεων Horn απεικονίζονται στο ακόλουθο προτεινόμενο παράδειγμα:

	Ένωση	**Συνεπαγωγή**	**Διαβάζεται διαισθητικά ως**
Οριστική πρόταση	$\neg p \vee \neg q \vee ... \vee \neg t \vee u$	$u \leftarrow p \wedge q \wedge ... \wedge t$	Αν *p* και *q* και... και *t* ισχύουν, τότε και το *u* ισχύει
Γεγονός	u	u	Το *u* ισχύει
πρόταση στόχου	$\neg p \vee q \vee ... \vee \neg t$	$false \leftarrow p \wedge q \wedge ... \wedge t$	Να δειχθεί ότι τα *p* και *q* και... και *t* ισχύουν

Όλες οι μεταβλητές σε μια πρόταση είναι καθολικά ποσοτικά προσδιορισμένες με το πεδίο εφαρμογής να είναι ολόκληρη η πρόταση. Έτσι, για παράδειγμα:

$\neg$ *άνθρωπος*(*X*) $\vee$ *θνητός* (*X*)

σημαίνει:

$\forall X$ ($\neg$*άνθρωπος*(*X*) $\vee$ *θνητός* (*X*))

που λογικά ισοδυναμεί με:

$\forall X$ (*άνθρωπος*(*X*) $\rightarrow$ *θνητός* (*X*)) [3]

3.2 Σημασία των προτάσεων Horn

Οι προτάσεις Horn είναι σημαντικές στην αυτόματη απόδειξη που επιτυγχάνεται από την ανάλυση της πρώτης τάξης, επειδή το αποτελεσμα της εφαρμογής του κανόνα της ανάλυσης (resolution) δύο προτάσεων Horn είναι επίσης μια πρόταση Horn, και το αποτελεσμα της εφαρμογής του κανόνα της ανάλυσης (resolution) μιας πρότασης στόχου και μιας άλλης πρότασης είναι μια πρόταση στόχου. Αυτές οι ιδιότητες των προτάσεων Horn μπορούν να οδηγήσουν σε μεγαλύτερη αποτελεσματικότητα στην απόδειξη ενός θεωρήματος (που αντιπροσωπεύεται ως άρνηση μιας πρότασης στόχου).

Οι προτάσεις Horn είναι επίσης η βάση του λογικού προγραμματισμού, όπου γράφουμε προτάσεις με τη μορφή μιας συνεπαγωγής:

($p \wedge q \wedge ... \wedge t$) $\rightarrow u$

Στην πραγματικότητα, η επίλυση μιας πρότασης στόχου με μια συγκεκριμένη πρόταση για την παραγωγή μιας νέας πρότασης στόχου είναι η βάση του κανόνα συμπερασμού για την ανάλυση SLD, που χρησιμοποιείται για την υλοποίηση του λογικού προγραμματισμού στη γλώσσα προγραμματισμού Prolog.

Στο λογικό προγραμματισμό μια οριστική πρόταση συμπεριφέρεται ως διαδικασία απόδειξης στόχου. Για παράδειγμα, η πρόταση Horn που γράφτηκε παραπάνω συμπεριφέρεται ως η διαδικασία:

να αποδειχθεί u αρκεί να αποδειχθεί οτι ισχύει το p και q και... και t.

Για να τονιστεί αυτή η αντίστροφη χρήση της πρότασης, συχνά γράφεται με την αντίστροφη μορφή:

$u \leftarrow (p \land q \land ... \land t)$

Στην Prolog αυτό γράφεται ως εξής:

u:- p, q,..., t.

Στον λογικό προγραμματισμό και στην Datalog, ο υπολογισμός ερωτημάτων γίνονται θεωρόντας την άρνηση ενός προβλήματος που πρέπει να επιλυθεί ως πρόταση στόχου. Για παράδειγμα, το πρόβλημα της επίλυσης του παρακάτω:

$\exists X (p \land q \land ... \land t)$

ανάγεται στην εκτέλεση του προγράμματος με την λογικά ισοδύναμη πρόταση στόχο:

$\forall X (false \leftarrow p \land q \land ... \land t)$

Στην Prolog αυτό γράφεται ως εξής:

:- *p*, q,..., t.

Στην συνέχεια, με την χρήση ανάλυσης, η επίλυση του προβλήματος οδηγεί σε αντίφαση, η οποία αντιπροσωπεύεται από την κενή πρόταση (ή "ψευδές"). Η λύση του προβλήματος είναι μια αντικατάσταση των όρων για τις μεταβλητές στην πρόταση του στόχου, που μπορεί να εξαχθεί από την απόδειξη της αντίφασης. Χρησιμοποιούμενες με αυτόν τον τρόπο, οι προτάσεις Horn είναι παρόμοιες με τις συνηθισμένες επερωτήσεις (*query*) σε σχεσιακές βάσεις δεδομένων. Η λογική των προτάσεων Horn ισοδυναμεί με την υπολογιστική ισχύ μιας μηχανής Turing. [3]

4. DATALOG

Η Datalog είναι μια γλώσσα λογικών επερωτήσεων (query) και κανόνων και αποτελεί μοντέλο δεδομένων για βάσεις δεδομένων. Είναι μία μίξη της σχεσιακής άλγεβρας, την θεωρία πίσω από την SQL, και του λογικού προγραμματισμού και πιο συγκεκριμένα της Prolog, της οποίας αποτελεί συντακτικό υποσύνολο. Παρόλα αυτά, η λογική της είναι πιο κοντά στην σχεσιακή άλγεβρα, παρά στον λογικό προγραμματισμό. Δημιουργήθηκε για να επαυξήσει την θεωρία των βάσεων (database theory) με κάποιες αρχές του λογικού προγραμματισμού, κυρίως την δυνατότητα αναδρομής. [4]

Η εκτέλεση των ερωτημάτων σε Datalog βασίζεται στην λογική πρώτης τάξης και τις προτάσεις Horn (Horn clauses). Εκτελεί bottom-up αναζητήσεις (από κάτω προς τα πάνω) σε αντίθεση με την Prolog που εκτελεί top-down αναζητήσεις (από πάνω προς τα κάτω). [16]

4.1 Συντακτικο

Το συντακτικό της Datalog είναι παρόμοιο με της Prolog, με την έννοια ότι βασίζεται σε λογικά κατηγορήματα, που γράφονται ως εξής:

awesome(bill)

(Το παραπάνω σημαίνει ότι ο bill έχει την ιδιότητα awesome Στο παραπάνω παράδειγμα ο bill είναι ένα άτομο και το Awesome εκφράζει μια ιδιότητα του bill.)

Ή και ως εξής:

parent(bill,mary)

(Το παραπάνω σημαίνει ότι ο bill και η mary έχουν μία σχέση μεταξύ τους, την parent)

Είναι δυνατό να συνδέονται περισσότερα από δύο άτομα μεταξύ τους με μία σχέση.

Στο παράδειγμα παρακάτω ορίζονται δύο γεγονότα (facts) :

parent(bill,mary).

parent(mary,john).

(Το παραπάνω σημαίνει ότι ο γονιός της mary είναι ο bill και ο γονιός του john είναι η mary.)

Σε αυτό το παράδειγμα ορίζονται δύο κανόνες:

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

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

Ο κανόνας ορίζεται από δύο μέρη:

- Την κεφαλή (αριστερά του :-) : Αποτελείται από ένα μοναδικό ατομικό τύπο.
- Το σώμα (δεξιά του :-) : Αποτελείται από έναν ή περισσότερους ατομικούς τύπους.

Τα κεφαλαία γράμματα δεν εκφράζουν άτομα. Εκφράζουν μεταβλητές και αντικαθιστώνται κατά την διάρκεια εκτέλεσης του ερωτήματος από υπάρχοντα άτομα. Μεταβλητές είναι όλες οι λέξεις που αρχίζουν με κεφαλαίο γράμμα ενώ άτομα, όλες όσες αρχίζουν με μικρό. Στο παράδειγμα ο πρώτος κανόνας μπορεί να διαβαστεί σαν *ο X είναι πρόγονος του Y αν ο X είναι γονιός του Y* και ο δεύτερος κανόνας σαν *ο X είναι πρόγονος του Y αν ο X είναι γονιός κάποιου Z και ο Z είναι πρόγονος του Y*. Η σειρά των κανόνων δεν έχει σημασία στην Datalog, αντίθετα με την Prolog που χρησιμοποιεί τους κανόνες με την σειρά που είναι γραμμένοι για να αποτιμήσει το αποτέλεσμα.

?- ancestor(X,john).

Αυτό είναι μία επερώτηση. Ρωτάει ποιοι είναι οι πρόγονοι του john. [4] [16]

4.2 Διαφορες με την Prolog

Το συντακτικό της Datalog, σε αντίθεση με την Prolog:

- δεν επιτρέπει σύνθετους όρους σαν ορίσματα, πχ. το p(1, 2) επιτρέπεται αλλά όχι το p(f1(1), 2),
- δεν δέχεται άρνηση και διάζευξη στο σώμα του κανόνα
- επιβάλλει περιορισμούς στη χρήση της αναδρομής, ώστε να αποτιμώνται όλα τα αναδρομικά ερωτήματα
- επιβάλλει κάθε μεταβλητή που εμφανίζεται στην κεφαλή του κανόνα, να εμφανίζεται και στο σώμα του κανόνα

Λόγω των παραπάνω περιορισμών όλα τα προγράμματα σε Datalog αποτιμώνται και τερματίζουν (σε αντίθεση με τα προγράμματα σε Prolog). Παράδειγμα αποτελεί το εξής πρόγραμμα:

parent(bob,john).

parent(john,pam).

parent(pam,cat).

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

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

Οποιοδήποτε ερώτημα με το κατηγόρημα ancestor αποτιμάται σε Datalog. Σε Prolog, όμως, το ερώτημα:

?-ancestor(X,Y).

όπως και το ερώτημα:

?-ancestor(bob,X).

δεν αποτιμάται λόγω μη τερματισμού των αναδρομικών κλήσεων.

Αυτό έχει ως αποτέλεσμα, οι εντολές και τα κατηγορήματα ενός προγράμματος να μπορούν να δοθούν με οποιαδήποτε σειρά (πάλι σε αντίθεση με την Prolog). Υπάρχουν διάφοροι αλγόριθμοι για την αποδοτική εκτέλεση ερωτήσεων, όπως ο αλγόριθμος Magic Sets, ή η Counting Method.

4.3 Επεκτάσεις

Έχουν προταθεί επεκτάσεις της Datalog που επιτρέπουν τη χρήση αντικειμενοστραφούς προγραμματισμού, την σύζευξη, την άρνηση και την χρήση πράξεων. Αυτές οι επεκτάσεις, μολονότι χρήσιμες, έχουν αντίκτυπο στην λογική της Datalog. Στα πλαίσια της πτυχιακής αυτής θα χρησιμοποιήσουμε την καθαρή Datalog, χωρίς τις παραπάνω επεκτάσεις. [4] [16]

5. SEMANTIC WEB

5.1 Εισαγωγή

Ο Σημασιολογικός Ιστός αποτελεί επέκταση του Διαδικτύου μέσω προτύπων από την World Wide Web Consortium (W3C). Τα πρότυπα προωθούν κοινές μορφές δεδομένων και πρωτόκολλα ανταλλαγής στον Ιστό, το πιο ουσιαστικό είναι το Πλαίσιο Περιγραφής Πόρων (RDF). Σύμφωνα με το W3C, “ο Σημασιολογικός Ιστός παρέχει ένα κοινό πλαίσιο που επιτρέπει την κοινή χρήση και επαναχρησιμοποίηση δεδομένων ανάμεσα σε εφαρμογές, επιχειρήσεις και διαφορετικές κοινότητες”. Ως εκ τούτου, ο Σημασιολογικός Ιστός θεωρείται ο συνδετικός κρίκος μεταξύ εφαρμογών πληροφοριών και συστημάτων.

Ο όρος προτάθηκε απο τον Tim Berners-Lee και αναφερόταν σε έναν ιστό δεδομένων (ή διαδικτυακού τόπου δεδομένων) που να μπορεί να επικοινωνήσει με μηχανές. Ενώ οι επικριτές του αμφισβήτησαν τη σκοπιμότητά του, οι υποστηρικτές του υποστηρίζουν ότι οι εφαρμογές και την επιστήμη των υπολογιστών, της βιομηχανίας, της βιολογίας και των ανθρωπιστικών επιστημών έχουν ήδη αποδείξει την εγκυρότητα της αρχικής ιδέας.

Ο Berners-Lee εξέφρασε αρχικά το όραμά του για τον Σημασιολογικό Ιστό ως εξής:

Έχω ένα όνειρο για τον Ιστό [στον οποίο οι υπολογιστές] είναι σε θέση να αναλύσουν όλα τα δεδομένα στον Ιστό - το περιεχόμενο, τους συνδέσμους και τις συναλλαγές μεταξύ ανθρώπων και υπολογιστών. Ένας "Σημασιολογικός Ιστός", ο οποίος καθιστά αυτό δυνατό, δεν έχει ακόμη προκύψει, αλλά όταν συμβαίνει αυτό, οι καθημερινοί μηχανισμοί του εμπορίου, της γραφειοκρατίας και της καθημερινής μας ζωής θα αντιμετωπιστούν από μηχανήματα που μιλάνε με μηχανές. Οι " ευφυείς πράκτορες " που οι άνθρωποι έχουν ασχοληθεί για χρόνια θα υλοποιηθούν τελικά. [26]

Η ιδέα του μοντέλου σημασιολογικού δικτύου δημιουργήθηκε στις αρχές της δεκαετίας του 1960 από ερευνητές όπως ο γνωστικός επιστήμονας Allan M. Collins, ο γλωσσολόγος M. Ross Quillian και η ψυχολόγος Elizabeth F. Loftus ως μορφή που αντιπροσωπεύει τη σημασιολογικά δομημένη γνώση. Όταν εφαρμόζεται στο πλαίσιο του σύγχρονου διαδικτύου, επεκτείνει το δίκτυο των υπερσυνδεδεμένων σελίδων που διαβάζονται από τον άνθρωπο, εισάγοντας μεταδεδομένα σχετικά με τις σελίδες και με το πώς σχετίζονται μεταξύ τους. Αυτό επιτρέπει στους αυτοματοποιημένους πράκτορες να έχουν πιο έξυπνη πρόσβαση στον Ιστό και να εκτελούν περισσότερες εργασίες εξ ονόματος των χρηστών.

Ο όρος “Σημασιολογικός Ιστός” αναφέρεται κυρίως στις τεχνολογίες που θα του επιτρέψουν να γίνει πραγματικότητα. Αυτός οι τεχνολογίες είναι γνωστές ως “W3C standards” και περιλαμβάνουν:

- Resource Description Framework (RDF), για την περιγραφή των πληροφοριών
- RDF Schema (RDFS)
- Simple Knowledge Organization System (SKOS)
- SPARQL, μια RDF γλώσσα ερωτημάτων
- Notation3 (N3)
- N-Triples
- Turtle

- Web Ontology Language (OWL), μία οικογένεια γλωσσών που περιγράφει πληροφορία
- Rule Interchange Format (RIF)

Πρόκειται για τεχνολογίες που έχουν χρήση σε πλαίσια επεξεργασίας πληροφοριών και ανταλλαγής δεδομένων. Επιπλέον, έχουν προκύψει άλλες τεχνολογίες με παρόμοιους στόχους, όπως τα microformats.

Φυσικά για την υλοποίηση του Σημασιολογικού Ιστού είναι απαραίτητη η χρήση και πολλών φυσικών πόρων. Οπότε είναι απαραίτητη η χρήση:

- Σέρβερς που επεξεργάζονται υπάρχοντα συστήματα δεδομένων χρησιμοποιώντας τα πρότυπα RDF και SPARQL. Πολλές εφαρμογές μετατρέπουν δεδομένα σε RDF μορφή.
- Σελίδων που είναι επαυξημένες με σημασιολογικές πληροφορίες. Θα πρέπει να γίνονται κατανοητές από μηχανές επεξεργασίας, πληροφορίες του εγγράφου που μπορούν να γίνουν κατανοητές από άνθρωπο.
- Κοινών λεξιλογίων μεταδεδομένων (οντολογίες) και χάρτες μεταξύ λεξιλογίων που να επιτρέπουν στους δημιουργούς εγγράφων να γνωρίζουν πώς να επισημάνουν τα έγγραφά τους έτσι ώστε οι πράκτορες να μπορούν να χρησιμοποιήσουν τις πληροφορίες στα παρεχόμενα μεταδεδομένα.
- Αυτοματοποιημένων πράκτορων για την εκτέλεση εργασιών για χρήστες του σημασιολογικού ιστού χρησιμοποιώντας αυτά τα δεδομένα.
- Υπηρεσιών του διαδικτύου που να παρέχουν πληροφορίες ειδικά σε πράκτορες.

Τέτοιες υπηρεσίες θα μπορούσαν να είναι χρήσιμες στις δικτυακές μηχανές αναζήτησης ή θα μπορούσαν να χρησιμοποιηθούν για τη διαχείριση της γνώσης μέσα σε έναν οργανισμό. Στις ενδοεπιχειρησιακές εφαρμογές, οι υπηρεσίες αυτές παρέχουν:

- Διευκόλυνση της ενσωμάτωσης πληροφοριών από μικτές πηγές
- Διευκρίνηση ασαφειών στην εταιρική ορολογία
- Βελτίωση της ανάκτησης πληροφοριών
- Προσδιορισμός σχετικών πληροφοριών σε σχέση με συγκεκριμένο τομέα
- Παροχή υποστήριξης λήψης αποφάσεων [6]

5.2 Προκλήσεις

Ορισμένες από τις προκλήσεις για τον Σημασιολογικό ιστό είναι η απεραντοσύνη (Vastness), η ασάφεια, η αβεβαιότητα, η ασυνέπεια και η εξαπάτηση. Τα αυτοματοποιημένα συστήματα συλλογιστικής θα πρέπει να αντιμετωπίσουν όλα αυτά τα ζητήματα, προκειμένου να τηρήσουν τον στόχο του Σημασιολογικού Ιστού.

- Απεραντοσύνη: Ο Παγκόσμιος Ιστός περιέχει πολλά δισεκατομμύρια σελίδες που περιέχουν πολλές φορές παρόμοια πληροφορία και παρόμοιους όρους.
- Ασάφεια: Αυτό περιλαμβάνει έννοιες χωρίς ξεκάθαρο νόημα όπως "ψηλός" ή “δυνατός”. Το πρόβλημα προκύπτει από την ασάφεια των ερωτημάτων των

χρηστών, των εννοιών που χρησιμοποιούνται από τους παρόχους περιεχομένου, της αντιστοίχισης των όρων ερωτήματος με τους όρους των παρόχων και της προσπάθειας συνδυασμού διαφορετικών βάσεων γνώσης με αλληλεπικαλυπτόμενες αλλά με ελάχιστα διαφορετικές έννοιες. Ο χειρισμός ασαφών εννοιών γίνεται συνήθως με την χρήση της ασαφούς λογικής (fuzzy logic)

- Αβεβαιότητα: Αυτό περιλαμβάνει ακριβείς έννοιες με αβέβαιες ισχύ αλήθειας. Για παράδειγμα, η έκφραση “σε λίγο θα βρέξει”. Ο χειρισμός αβέβαιης πληροφορίας γίνεται με μεθοδολογίες χειρισμού αβεβαιότητας πχ θεωρία πιθανοτήτων.

- Ανακολουθία: Κατά την δημιουργία μεγάλων οντολογιών είναι αναπόφευκτο να συνδυαστούν οντολογίες από διαφορετικές πηγές οπότε και θα υπάρξουν αντιφάσεις και ανακολουθίες.

- Εξαπάτηση : Πολλές φορές ο παραγωγός των πληροφοριών σκοπίμως επιθυμεί να παραπλανήσει τον καταναλωτή των πληροφοριών

Αυτός ο κατάλογος των προκλήσεων είναι ενδεικτικός και όχι εξαντλητικός. Για την αντιμετώπιση αυτών των προβλημάτων θα πρέπει να γίνει έρευνα και να δημιουργηθούν επεκτάσεις στις ήδη υπάρχουσες τεχνολογίες. [6]

6. ΛΟΓΙΚΕΣ ΠΕΡΙΓΡΑΦΩΝ

6.1 Εισαγωγή

Οι λογικές περιγραφών (Description Logics DL) είναι μια οικογένεια τυπικών γλωσσών αναπαράστασης γνώσης. Πολλές DLs είναι πιο εκφραστικές από την προτασιακή λογική, αλλά λιγότερο εκφραστικές από τη λογική πρώτης τάξης. Σε αντίθεση με την τελευταία, τα βασικά προβλήματα συλλογιστικής για τις DL είναι αποφασιστικότητας και έχουν σχεδιαστεί και εφαρμοστεί αποτελεσματικές διαδικασίες υπολογισμού για αυτά τα προβλήματα. Υπάρχουν διάφορες επεκτάσεις στις λογικές περιγραφών πχ χωρικές, χρονικές, υποστήριξη ασάφειας κτλ. Γενικά με χρήση διαφορετικών συνόλων μαθηματικών συνδέσμων επιτυγχάνουμε διαφορετική εκφραστικότητα με αντιστάθμισμα στην πολυπλοκότητα της συλλογιστικής που απαιτείται.

Οι DL χρησιμοποιούνται στην τεχνητή νοημοσύνη για να περιγράψουν και να αιτιολογήσουν τις σχετικές έννοιες ενός πεδίου εφαρμογής (γνωστή ως *ορολογική γνώση*). Μία εφαρμογή των DL και της OWL είναι στη βιοϊατρική πληροφορική, όπου οι DL βοηθούν στην κωδικοποίηση της βιοϊατρικής γνώσης.

Μια λογική περιγραφής μοντελοποιεί τις *έννοιες (Concept)*, τους *ρόλους (Role)* και τα *άτομα (Individual)* και τις σχέσεις μεταξύ τους. Η θεμελιώδης έννοια της μοντελοποίησης μιάς DL είναι το *αξίωμα (axiom)*- μια λογική δήλωση σχετικά με τους ρόλους και / ή τις έννοιες. [7]

6.2 Ορολογία σε σύγκριση με την λογική πρώτης τάξης και την OWL

Η επιστημονική κοινότητα χρησιμοποιεί διαφορετική ορολογία από τη λογική πρώτης τάξης για λειτουργικά ισοδύναμες έννοιες. Η οικογένεια γλωσσών OWL χρησιμοποιεί και πάλι διαφορετική ορολογία. Οι αντιστοιχίες δίνονται στον παρακάτω πίνακα. [7]

Συνώνυμα

Λογική πρώτης τάξης	OWL	DL
σταθερά	Άτομο	Άτομο
μοναδιαίο κατηγόρημα	Κλάση	Έννοια
δυαδικό κατηγόρημα	Ιδιότητα	Ρόλος

6.3 Μοντελοποίηση

Στις DL, γίνεται διάκριση μεταξύ του αποκαλούμενου TBox (ορολογικό πλαίσιο) μιας βάσης γνώσης και του ABox (πλαίσιο επιβεβαίωσης). Το TBox περιέχει προτάσεις πού περιγράφουν τις σχέσεις μεταξύ των εννοιών και των ρόλων ενώ το ABox περιέχει προτάσεις που δηλώνουν ποια άτομα ανήκουν σε κάθε έννοια και σε κάθε ρόλο. Για παράδειγμα, η δήλωση:

Κάθε γυναίκα είναι άνθρωπος

ανήκει στο TBox, ενώ η δήλωση:

Η Mary είναι γυναίκα

ανήκει στο ABox.

Η διάκριση TBox / ABox δεν φαίνεται τόσο σημαντική με την πρώτη ματιά και, μάλιστα, τα δύο "είδη" των προτάσεων δεν αντιμετωπίζονται διαφορετικά στην λογική πρώτης

τάξης (της οποίας αποτελούν υποσύνολο οι περισσότερες DL). Όταν μεταφράζεται σε λογική πρώτης τάξης, κάθε αξίωμα του TBox απλά μετατρέπεται σε έναν κανόνα και κάθε αξίωμα του ABox μετατρέπεται σε γεγονός.

Πρωταρχικός λόγος για την εισαγωγή της διάκρισης είναι ότι ο διαχωρισμός μπορεί να είναι χρήσιμος όταν περιγράφονται και διατυπώνονται διαδικασίες υπολογισμού για διάφορες DL. Για παράδειγμα, ένας λογικός επεξεργαστής (reasoner) μπορεί να επεξεργαστεί χωριστά το TBox και το ABox, εν μέρει επειδή ορισμένα προβλήματα βασικών συμπερασμάτων συνδέονται με το ένα, αλλά όχι με το άλλο. Ένα άλλο παράδειγμα είναι ότι η πολυπλοκότητα του TBox μπορεί να επηρεάσει σε μεγάλο βαθμό την απόδοση μιας δεδομένης διαδικασίας υπολογισμού για μια συγκεκριμένη DL, ανεξάρτητα από το ABox.

Ο δευτερεύων λόγος είναι ότι η διάκριση μπορεί να έχει νόημα από πλευράς του μοντέλου της γνώσης. Γιά παράδειγμα: όταν η ιεραρχία μιας επιχείρησης είναι η ίδια σε κάθε υποκατάστημα, αλλά η ανάθεση στους υπαλλήλους διαφέρει σε κάθε τμήμα (επειδή υπάρχουν και άλλοι που εργάζονται εκεί), είναι λογικό να επαναχρησιμοποιηθεί το TBox για διαφορετικούς κλάδους που δεν χρησιμοποιείται το ίδιο ABox.

Υπάρχουν δύο χαρακτηριστικά γνωρίσματα λογικής περιγραφών που διαφέρουν απο άλλα μοτίβα περιγραφής δεδομένων.

- Οι DL δεν κάνουν την παραδοχή μοναδικού ονόματος (UNA)
- Δεν ισχύει η υπόθεση κλειστού κόσμου (CWA).

Χωρίς την UNA σημαίνει ότι δύο έννοιες με διαφορετικά ονόματα μπορούν μετα απο κάποια βήματα συμπερασμού να επιτραπούν να αποδειχθούν ισοδύναμες. Η υπόθεση ανοιχτού κόσμου (OWA), σε αντιδιαστολή με την CWA σημαίνει ότι η έλλειψη γνώσης ενός γεγονότος δεν συνεπάγεται αμέσως την αποδοχή της άρνησης ενός γεγονότος. [7]

6.4 Επίσημη περιγραφή

Όπως η λογική πρώτης τάξης, μια σύνταξη ορίζει ποιες συλλογές συμβόλων είναι αποδεκτές εκφράσεις σε μια λογική περιγραφής, και η σημασιολογία καθορίζει το νόημά τους. Σε αντίθεση με την λογική πρώτης τάξης, οι DL έχουν αρκετές γνωστές συντακτικές παραλλαγές.

6.4.1 Σύνταξη

Η σύνταξη ενός μέλους των λογικών περιγραφής χαρακτηρίζεται από τα συνθετικά του που μπορούν να χρησιμοποιηθούν για τον σχηματισμό εννοιών. Ορισμένα συνθετικά των γλωσσών σχετίζονται με τους λογικούς συνδέσμους της λογικής πρώτης τάξης (FOL), όπως η *τομή* ή η διάζευξη των εννοιών, η *ένωση* των εννοιών, η *άρνηση* ή το *συμπλήρωμα* των εννοιών, ο *καθολικός περιορισμός* και ο *υπαρξιακός περιορισμός*.

Επίσης όσον αφορά την ονοματολογία ισχύουν οι εξής κανόνες:

- τα ονόματα σταθερών είναι γραμμένα συνήθως με κεφαλαία
- τα ονόματα εννοιών έχουν μόνο το πρώτο γράμμα κεφαλαίο
- τα ονόματα ρόλων είναι με μικρά

Σημείωση: Αυτοί οι περιορισμοί δεν είναι απαραίτητοι στην οικογένεια γλωσσών OWL.

Η σύνταξη των εκφράσεων DL χρησιμοποιεί τα παρακάτω σύμβολα. [7]

Έστω C και D είναι έννοιες και R είναι ένας ρόλος.

Συμβολική σημείωση

Σύμβολο	Περιγραφή	Παράδειγμα	Ανάγνωση
⊤	Το ⊤ εκφράζει κάθε άτομο που έχει οριστεί στην βάση	⊤	Όλα τα στοιχεία
⊥	Το ⊥ εκφράζει κανένα άτομο	⊥	Κανένα στοιχείο
⊓	Σύζευξη εννοιών	C ⊓ D	C και D
⊔	Διάζευξη εννοιών	C ⊔ D	C ή D
¬	Άρνηση ή συμπλήρωμα έννοιας	¬ C	Όχι C
∀	Καθολικός περιορισμός	∀R.C	Όλοι οι R-διάδοχοι βρίσκονται στο C
∃	Υπαρξιακός περιορισμός	∃R.C	Τουλάχιστον ένας διάδοχος R υπάρχει στο C
⊑	Συμπερίληψη έννοιας	C ⊑ D	Όλα τα C είναι D
≡	Ισοδυναμία εννοιών	C ≡ D	Το C είναι ισοδύναμο με το D

6.4.2 Σημασιολογία

Η σημασιολογία των λογικών περιγραφής ορίζεται από την ερμηνεία των εννοιών ως συνόλων ατόμων και ρόλων ως συνόλων διατεταγμένων ζευγών ατόμων. Αυτά τα άτομα συνήθως λαμβάνονται από συγκεκριμένο πεδίο. Τα ονόματα των εννοιών και τα ονόματα των ρόλων αποκαλούνται ατομικές έννοιες και ατομικοί ρόλοι αντίστοιχα. Η σημασιολογία των μη-ατομικών εννοιών και ρόλων ορίζεται στη συνέχεια με όρους ατομικών εννοιών και ρόλων. Αυτό γίνεται χρησιμοποιώντας έναν αναδρομικό ορισμό παρόμοιο με τη σύνταξη.

Οι βασικές έννοιες της σημασιολογίας των DL ορίζονται ως εξής:

- κάθε ερμηνεία έχει ένα πεδίο ορισμού (domain)
- οι σταθερές είναι στοιχεία του πεδίου ορισμού (elements of domain)
- οι έννοιες αποτελούν υποσύνολα του πεδίου ορισμού (subsets of domain)
- οι ρόλοι αποτελούν δυαδικές σχέσεις του πεδίου ορισμού (binary relation of domain)

Η σημασιολογία περίπλοκων εκφράσεων των DL βασίζεται σε εκφράσεις μεταξύ συνόλων. Όποτε και η λογική των Abox και Tbox καθορίζεται από την θεωρία συνόλων.[8]

6.4.3 Ονοματολογία

Οι λογικές περιγραφής έχουν διαφορετικά ονόματα ανάλογα με ποια μέρη της λογικής υλοποιούν. Κατά κανόνα βασίζονται στις εξής γλώσσες:

- την $\mathcal{AL}$ (Attribute Language). Αυτή επιτρέπει άρνηση,τομή εννοιών, καθολικούς και στοιχειώδεις υπαρξιακούς περιορισμούς
- την $\mathcal{FL}$ (Frame-based description Language) που επιτρέπει καθολικούς και κάποιους υπαρξιακούς περιορισμούς, τομή εννοιών και περιορισμούς στους ρόλους
- την $\mathcal{EL}$ (Existential language) που επιτρέπει υπαρξιακό περιορισμό και τομή εννοιών

Παράλληλα υποστηρίζονται και οι παρακάτω επεκτάσεις:

- $\mathcal{F}$ που υποστηρίζει περιορισμούς συναρτησιακής φύσεως (functional number restrictions)
- $\mathcal{E}$ που έχει υπαρξιακό περιορισμό
- $\mathcal{U}$ που υποστηρίζει ένωση εννοιών
- $\mathcal{C}$ που υποστηρίζει άρνηση εννοιών
- $\mathcal{H}$ που υποστηρίζει ιεραρχίες ρόλων
- $\mathcal{R}$ που υποστηρίζει ασύνδετες έννοιες (disjoint) και την αυτοπαθή ιδιότητα των ρόλων
- $\mathcal{O}$ που υποστηρίζει αριθμητικούς περιορισμούς (nominals)
- $\mathcal{I}$ που υποστηρίζει αντίστροφους ρόλους (inverse)
- $\mathcal{N}$ που υποστηρίζει απλούς περιορισμούς με cardinality (number restrictions)
- $\mathcal{Q}$ που υποστηρίζει περίπλοκους περιορισμούς με cardinality (quantified number restrictions)
- $(\mathcal{D})$ που υποστηρίζει την χρήση datatypes δηλαδή τύπων δεδομένων

Κάθε γλώσσα παίρνει τα γράμματα από την γλώσσα και τις επεκτάσεις που χρησιμοποιεί. Παράδειγμα: η $\mathcal{ALC}$ (που αποτελεί την πιο βασική DL) υποστηρίζει άρνηση, τομή εννοιών, καθολικούς και κάποιους υπαρξιακούς περιορισμούς (γιατί έχει το $\mathcal{AL}$ στο όνομα) και περίπλοκη άρνηση εννοιών (γιατί έχει το $\mathcal{C}$).

Ιδιαίτερη αναφορά αξίζει η γλώσσα $\mathcal{S}$ που είναι η $\mathcal{ALC}$ επαυξημένη με την μεταβατική ιδιότητα των ρόλων. Το S χρησιμοποιείται κανονικά σαν γράμμα γλωσσών των περιγραφικών λογικών.

Η OWL 2 χρησιμοποιεί την $\mathcal{SROIQ}(\mathcal{D})$ που περιλαμβάνει την $\mathcal{S}$ γλώσσα με τις επεκτάσεις $\mathcal{R}$, $\mathcal{O}$, $\mathcal{I}$, $\mathcal{Q}$, $(\mathcal{D})$ ενώ παλαιότερες εκδόσεις χρησιμοποιούν υποσύνολα αυτής. Συγκεκριμένα η OWL-DL χρησιμοποιεί την $\mathcal{SHOIN}(\mathcal{D})$ και η OWL-Lite την $\mathcal{SHIF}$. [7]

6.5 Σχέση με Λογική πρώτης τάξης

Πολλές DLs είναι υποσύνολα της λογικής πρώτης τάξης (FOL). Ωστόσο, ορισμένες DL έχουν χαρακτηριστικά που δεν καλύπτονται από τη FOL. Αυτό περιλαμβάνει *συγκεκριμένους τύπους* (όπως ακέραιοι ή συμβολοσειρές, που μπορούν να χρησιμοποιηθούν ως εύρος για ρόλους όπως hasAge ή hasName). [7]

7. WEB ONTOLOGY LANGUAGE (OWL)

7.1 Εισαγωγή

Η Web Ontology Language (OWL) είναι μία οικογένεια γλωσσών που χρησιμοποιούνται για την αναπαράσταση οντολογιών στον σημασιολογικό ιστό. Οι οντολογίες είναι ένας τρόπος για να περιγράψουμε την δομή της γνώσης σε διάφορα περιβάλλοντα και επίπεδα. Σε μία οντολογία ορίζονται ομάδες με αντικείμενα και οι σχέσεις μεταξύ αυτών. Οι οντολογίες θυμίζουν την έννοια των κλάσεων στα πλαίσια του αντικειμενοστραφούς προγραμματισμού αλλά υπάρχουν κάποιες σημαντικές διαφορές. Οι κλάσεις υποτίθεται ότι αναπαριστούν πληροφορία μέσα σε κώδικα και έχουν σταθερή δομή και μεθόδους, αλλά οι οντολογίες του σημασιολογικού ιστού υποτίθεται οτι αναπαριστούν πληροφορία από το internet οπότε και η δομή τους θα μεταβάλλεται.

Η τυπική θεμελίωση των οντολογιών και της OWL βασίζεται συνήθως στις λογικές περιγραφών. Η OWL είναι ένας τρόπος αναπαράστασης γνώσης για αντικείμενα, ομάδες και σχέσεις μεταξύ αντικειμένων.

Οι γλώσσες OWL χαρακτηρίζονται από επίσημη σημασιολογία. Μια απο τις μεθοδολογίες απόδοσης σημασιολογίας βασίζεται στο Resource Description Framework (RDF).

Η OWL έχει τραβήξει αρκετό ακαδημαϊκό ενδιαφέρον ανά τα έτη. Τον Οκτώβριο του 2007, έγινε μία προσπάθεια επέκτασης της OWL με αρκετές προσθήκες που κατέληξε στην ανακοίνωση της δημιουργίας μίας νέας έκδοσης της OWL στην 27η Οκτωβρίου 2009. Αυτή η νέα έκδοση είναι η OWL2 που θα χρησιμοποιηθεί στα πλαίσια της παρούσας πτυχιακής, αν και θα αναφερθεί ως OWL, επειδή θα αποτελέσει τον εκπρόσωπο της οικογένειας γλωσσών OWL. [5] [18] [19]

7.2 Συντακτικό

Τα αντικείμενα με τα οποία αναπαρίσταται η γνώση ονομάζονται άτομα (individuals). Ομάδες απο αντικείμενα ονομάζονται κλάσεις (class). Σχέσεις μεταξύ αντικειμένων ονομάζονται ιδιότητες (property). Αντικείμενα, κλασεις και ιδιότητες αποτελούν τα οντότητες (entity). Αυτές μπορούν να συνδυαστούν με συνθετικά (constructors) για να δημιουργησουν εκφράσεις (expression). Για να αναπαραστήσουμε γνώση στην OWL ορίζουμε δηλώσεις. Αυτές οι δηλώσεις ονομάζονται αξιώματα (axioms).

Η OWL συντάσσεται με πολλούς τρόπους. Αυτοι είναι:

- Η Functional-Style syntax. Αυτή η σύνταξη χρησιμοποιήθηκε για να αποτελέσει βάση για την υλοποίηση πολλών εργαλείων της OWL2 όπως APIs και reasoners. Αυτή θα χρησιμοποιηθεί στα πλαίσια αυτής της πτυχιακής.

- Η RDF/XML syntax: Επρόκειτο για σύνταξη RDF/XML, εκφράσεων OWL. Με αυτήν είναι δυνατή η έκφραση άλλων συντάξεων όπως η Turtle syntax.

- Η Manchester syntax: Μία απλή σύνταξη που γίνεται εύκολα κατανοητή απο τους χρήστες.

- Η OWL XML syntax: Είναι σύνταξη σε XML εκφράσεων OWL ορισμένη μέσω XML schema

Παράδειγμα χρήσης των διαφορετικών συντάξεων:

Jack is a person but not a parent. (ο jack είναι άνθρωπος αλλά όχι γονέας)

Functional-Style Syntax

```
ClassAssertion(
   ObjectIntersectionOf(:Person  ObjectComplementOf(:Parent)) :Jack )
```

RDF/XML Syntax

```
<rdf:Description rdf:about="Jack">
  <rdf:type>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="Person"/>
        <owl:Class>
          <owl:complementOf rdf:resource="Parent"/>
        </owl:Class>
     </owl:intersectionOf>
    </owl:Class>
  </rdf:type>
</rdf:Description>
```

Turtle Syntax

```
:Jack rdf:type [
  rdf:type owl:Class;
  owl:intersectionOf ( :Person
                       [ rdf:type owl:Class;
                         owl:complementOf  :Parent ]
                     )
].
```

Manchester Syntax

```
Individual: Jack
Types: Person and not Parent
```

OWL/XML Syntax

```
<ClassAssertion>
  <ObjectIntersectionOf>
    <Class IRI="Person"/>
    <ObjectComplementOf>
      <Class IRI="Parent"/>
    </ObjectComplementOf>
```

```
    </ObjectIntersectionOf>
    <NamedIndividual IRI="Jack"/>
</ClassAssertion> [17]
```

7.3 Περιορισμοί OWL

Το συντακτικό της OWL έχει όμως τον περιορισμό, ότι δεν υποστηρίζει n-ary σχέσεις. Για παράδειγμα, ο ορισμός μίας τριμερούς σχέσης του στυλ:

“Οι γονείς του bob είναι ο john και η Pam”

δεν ορίζεται σε OWL. Είναι απαραίτητη η παράκαμψη αυτού με την χρήση άλλων σχέσεων.

8. ΛΟΓΙΚΗ ΜΕΤΑΤΡΟΠΗΣ DATALOG ΣΕ OWL ΚΑΙ ΑΝΤΙΣΤΡΟΦΑ

Η λογική με την οποία γίνεται η μετατροπή Datalog σε OWL και το αντίστροφο και ακολουθήθηκε σε αυτή την πτυχιακή, ορίζεται στο paper των Grosof, Horrocs, Volz και Decker [13]. Εκεί αποτιμάται η τομή μεταξύ της λογικής πρώτης τάξης που βασίζονται οι προτάσεις Horn, στις οποίες βασίζεται η Datalog, και της λογικής DAML-OIL των Description Logic Programs (DLP), η οποία είναι ισοδύναμη με την $\mathcal{SHOIQ}(\mathcal{D})$ (στην οποία βασίζεται η OWL). Στο paper αναφέρεται ότι η εύρεση κοινού τόπου μεταξύ της DAML-OIL και της λογικής πρώτης τάξης δεν έχει οριστεί ακόμα, καθώς η λογική πρώτης τάξης, έχει τ μειονέκτημα, ότι δεν είναι αποφασίσιμη και οι αλγόριθμοι υπολογισμού που έχουν υιοθετηθεί έχουν υψηλή πολυπλοκότητα. Κατάλληλος περιορισμός των προτάσεων Horn δεν έχει τα παραπάνω μειονεκτήματα ενώ κρατάει τα πλεονεκτήματα όποτε η εύρεση τομής της με την DAML-OIL είναι εφικτή. Ο περιορισμός αυτός είναι οι def-Horn προτάσεις, οι οποίες, επιπροσθέτως παρέχουν την δυνατότητα έκφρασης κανόνων.

Πρώτα όμως θα ορίσουμε μία έννοια που χρησιμοποιείται πολύ στα πλαίσια της πτυχιακής εργασίας, την έννοια του *λογικού μονοπατιού*. Η έννοια αυτή διαφέρει ανάμεσα στις δύο γλώσσες.

Λογικό μονοπάτι σε Datalog αποτελεί οποιοδήποτε σύνολο κατηγορημάτων που έχουν τις εξής ιδιότητες:

- Αν ένα κατηγόρημα είναι δυαδικό (binary) πρέπει η πρώτη μεταβλητή να είναι δεύτερη σε ακριβώς ένα ευθύ κατηγόρημα ή πρώτη σε ένα αντεστραμμένο και η δεύτερη μεταβλητή να είναι πρώτη σε ακριβώς ένα ευθύ κατηγόρημα ή δεύτερη σε ένα αντεστραμμένο, με εξαίρεση ακριβώς μία πρώτη και μια δεύτερη μεταβλητή. Επίσης δεν μπορεί να έχει ίδιες μεταβλητές με κανένα άλλο δυαδικό κατηγόρημα στο σύνολο.
- Αν ένα κατηγόρημα είναι μοναδιαίο (unary) πρέπει η μεταβλητή του να είναι πρώτη ή δεύτερη σε κάποιο δυαδικό κατηγόρημα, με εξαίρεση την πρώτη μεταβλητή που δεν είναι δεύτερη σε κανένα δυαδικό κατηγόρημα.

Οπότε p(X,Y),q(Y),r(Y,Z),t(Z) είναι λογικό μονοπάτι βάσει της περιγραφής αλλά p(X,Y),r(X,Z),q(Y,Z) δεν είναι γιατί δεν τηρείται ο περιορισμός και p(X,Y),q(X),r(Y,Z),t(Z) δεν λαμβάνεται ως λογικό μονοπάτι.

Λογικό μονοπάτι σε OWL είναι οποιαδήποτε απλή κλάση, ιδιότητα, τομή κλάσεων, κλάσεις που προκύπτουν από υπαρξιακούς και καθολικούς περιορισμούς και ιδιότητες που προκύπτουν από αλυσίδες ιδιοτήτων (*property chain*).

Στη συνέχεια, αναφέρουμε λεπτομέρειες του μετατρέψιμου υποσυνόλου των δύο φορμαλισμών καθώς και λεπτομέρειες της μετατροπής που ακολουθήθηκε στην πτυχιακή.

Σημείωση: Στην συνέχεια, για λόγους ευκολίας στην έκφραση, αναφέροντας τον όρο “κατηγόρημα” εννοούμε “ατομικό τύπο”.

8.1 Περιορισμοί

Δεν είναι δυνατό να μετατραπούν όλα τα δεδομένα Datalog σε OWL και ομοίως ούτε και το αντίστροφο. Υπάρχουν πολλοί περιορισμοί οι οποίοι οφείλονται στην φύση της DAML-OIL λογικής στην οποία βασίζεται η OWL και στους πολλούς περιορισμούς που έχουν οι φράσεις Horn στην λογική των οποίων βασίζεται η Datalog.

1)Καταρχήν, δεν αποδεικνύεται ισότητα μεταξύ δυο διαφορετικών ατόμων σε Datalog και λόγο αυτού, στην Datalog δεν είναι εφικτό να δειχτεί ότι:

Αν q(X, Y) και q(X, Z) τότε Y=Z

το οποίο ορίζεται σε OWL και είναι γνωστό ως συναρτησιακή ιδιότητα (Functional Property)

2)Ένας σημαντικός περιορισμός στην μετατροπή Datalog σε OWL θεωρείται το γεγονός, πως είναι αδύνατη η περιγραφή κλάσεων στην OWL, οι οποίες συνδυάζουν μεταβλητές που να ανήκουν σε δυο ή περισσότερα διαφορετικά λογικά μονοπάτια. Πχ. η σχέση:

a(X) :- b(X,Y),c(X,Z),d(Y,Z).

δεν υλοποιείται σε OWL, διότι οι μεταβλητές Y και Z ανήκουν σε διαφορετικά μονοπάτια. Ωστόσο η σχέση:

a(X) :- b(X,Y),c(Y,Z),d(Z).

υλοποιείται σε OWL, καθώς οι μεταβλητές X, Y και Z ανήκουν στο ίδιο λογικό μονοπάτι.

3)Κανόνες με ιδιότητες ορίζονται μόνο αν κάθε κατηγόρημα στο σώμα του κανόνα έχει διαφορετικό ζεύγος μεταβλητών οι οποίες να σχηματίζουν ένα λογικό μονοπάτι. Αλλιώς η μεταγλώττιση είναι αδύνατη.

4)Στην OWL, δεν δημιουργούνται εμφανίσεις ιδιοτήτων, οι οποίες να παράγονται από απλές κλάσεις. Οπότε η σχέση:

a(X,Y):-b(X),c(Y).

δεν μπορεί να υλοποιηθεί σε OWL. Αντιθέτως, μπορούμε να παράγουμε εμφανίσεις κλάσεων χρησιμοποιώντας ιδιότητες, αρκεί να σχηματίζονται διακριτά λογικά μονοπάτια.

5)Στην Datalog δεν ορίζεται το ότι όλα τα instances μιας βάσης έχουν ένα χαρακτηριστικό, εάν δεν έχει οριστεί ρητά με έναν κανόνα. Πχ. Κάθε άνθρωπος έχει ένα πατέρα. Αυτό ορίζεται εύκολα σε OWL, εν αντιθέσει με την Datalog, στην οποία αυτό είναι αδύνατο να εκφραστεί, σε περίπτωση που δεν έχει οριστεί ένας πατέρας για κάθε instance της κλάσης άνθρωπος.

6)Όπως έχει προαναφερθεί, καμία μορφή άρνησης και μετρήματος (cardinality) δεν υλοποιείται σε Datalog. Επιπροσθέτως, είναι αδύνατο να γίνουν πράξεις σε Datalog. Εξαιτίας αυτού, είναι μάταιη οποιαδήποτε κατασκευή συμπληρώματος κλάσης ή ιδιότητας (Disjoint Classes) σε Datalog, παρόλο που υπάρχει η δυνατότητα διάζευξης. Πχ. Η συμπληρωματική σχέση ένας άνθρωπος είναι άντρας ή γυναίκα. Ενώ είναι δυνατόν να εκφραστεί σε Datalog το ότι ένας άνθρωπος είναι άντρας ή γυναίκα, δεν μπορεί να οριστεί το ότι ένας άνθρωπος είναι μόνο άντρας ή γυναίκα, αποκλείοντας την πιθανότητα να είναι ένα τρίτο φύλο Man :- not Woman όπου κάθε Person ή θα ανήκει στην κλάση Man ή στην Woman, δεν υλοποιείται σε Datalog, αλλά υλοποιείται σε OWL.

8.2 n-ary to binary

Ένα ακόμα πρόβλημα της μετατροπής Datalog σε OWL είναι ότι η Datalog βασίζεται στην n-ary λογική πρώτης τάξης (με πολλές μεταβλητές), ενώ η OWL στην binary (έχει μέχρι δύο μεταβλητές).

Όπως γίνεται αντιληπτό, δεν μετατρέπονται απευθείας n-ary κατηγορήματα σε OWL

Ωστόσο, μπορούν να διασπαστούν τα κατηγορήματα με n- μεταβλητές ως εξής:

p(X1, X2,….., Xn) = p1 (X1, i), p2 (X2, i), …., pn (Xn, i).

Όπου P1, P2,…., Pn κατηγορήματα, που προκύπτουν από το κατηγόρημα P, X1,….,Xn οι μεταβλητές του πραγματικού κατηγορήματος και i ένας δείκτης γεγονότος, μοναδικός για κάθε γεγονός που ορίζεται στην Datalog.

Πχ p(john, doe, 35) = p1(john, 1), p2(doe, 1), p3(35, 1).

p(jane, dae, 34) = p1(jane, 2), p2(dae, 2), p3 (34, 2).

Όπου john, doe, 35, jane, dae, 34 γνωστές σταθερές

Όταν θέλουμε να ελέγξουμε αν ένα σπασμένο κατηγόρημα p(α, β, γ) ισχύει, τότε χρησιμοποιούμε το εξής ερώτημα:

?-p1 (john, i), p2 (doe, i),p3 (35, i).

Με john, doe, 35 σταθερές και i λογική μεταβλητή, διότι δεν είναι δυνατό να γνωρίζουμε τον δείκτη γεγονότος

Ομοίως, ψάχνουμε και όλα τα instances του κατηγορήματος

?-p1 (X, i), p2 (Y, i), p3 (Z, i).

Με X, Y, Z μεταβλητές και i λογική μεταβλητή, ίδια και στα τρία κατηγορήματα

Με αυτό τον τρόπο, μπορούν να μετατραπούν και οι κανόνες από n-ary σε binary λογική.

Πχ έστω ότι θέλουμε να δείξουμε ότι p(A, B, Γ) :- n(A, B, Γ).

Χρησιμοποιώντας ότι έχουμε μέχρι τώρα, ο κανόνας σχηματίζεται έτσι:

p1(A,I), p2(B,I), p3(Γ,I) :- n1(A,I), n2(B,I), n3(Γ,I).

Όπου I λογική μεταβλητή και A,B,Γ απλές μεταβλητές

Η Datalog δεν επιτρέπει το παραπάνω, λόγο σύζευξης στη κεφαλή του κανόνα. Αλλά αυτό σπάει σε:

p1 (A,I) :- n1 (A,I), n2 (B,I), n3 (Γ,I).

p2 (B,I):- n1 (A,I), n2 (B,I), n3 (Γ,I).

p3 (Γ,I) :- n1 (A,I), n2 (B,I), n3 (Γ,I).

Στον κανόνα 1, δεν υπάρχουν Β και Γ στη κεφαλή του κανόνα. Επομένως, αυτές οι μεταβλητές στο σώμα του κώδικα μπορούν να έχουν οποιαδήποτε τιμή και δεν επηρεάζουν το αποτέλεσμα. Ομοίως και στους κανόνες 2,3. Συμπεραίνουμε πως οι κανόνες μετατρέπονται σε

p1 (A,I) :- n1 (A,I).

p2 (B,I) :- n2 (B,I).

p3 (Γ,I) :- n3 (Γ,I).

Που μετατρέπεται σε OWL.

Οι κανόνες με n-ary κατηγορήματα δεν μπορούν να έχουν παραπάνω από ένα κατηγόρημα στο σώμα του κανόνα. Αυτό θα δημιουργούσε κανόνες που τα

κατηγορήματα στο σώμα του κανόνα έχουν τις ίδιες μεταβλητές, και αυτό, όπως θα δούμε παρακάτω δεν μεταφράζεται σε OWL.

Αλλά υπό αυτό το πρίσμα, κανόνες της μορφής

p(X,Y,Z,X) :- a(X,Y,Z).

Μετατρέπονται σε OWL, εφόσον μπορούν να γραφτούν με την παραπάνω μορφή. Όμως, κανόνες που περιέχουν απλές κλάσεις ή ιδιότητες δεν μπορούν, γιατί αντιτίθενται στην λογική αυτή (δεν αποτελούν ιδιότητες με μια μεταβλητή και ένα δείκτη γεγονότος).

Παράδειγμα

p(X,X,Z,Z) :- d(X,Z).

Το κατηγόρημα d(X,Z) δεν σπάει σε d1(X,i) και d2(Z,i) διότι είναι ήδη δυαδικό κατηγόρημα. Επίσης δεν είναι υποκλάση ούτε του p1(X,i) ούτε του p3(Z,i) επειδή δεν υπάρχει ταύτιση μεταξύ των μεταβλητών. Άρα ο κανόνας δεν μεταφράζεται. [20] [21] [22] [23]

8.3 Μετατρέψιμο υποσύνολο

Οι παρακάτω κανόνες μπορούν να μετατραπούν από Datalog σε OWL και το αντίστροφο:

- Απλό γεγονός
- Υποκλάση και Υποιδιότητα
- Συμμετρία
- Αυτοπαθής ιδιότητα
- Τομή κλάσεων
- Ένωση Κλάσεων (Όχι πάντα)
- Ίσα κατηγορήματα
- Αντίστροφες ιδιότητες
- Η αντίστροφη μιας κλάσης.
- Καθολικός περιορισμός
- Υπαρξιακός περιορισμός
- Αλυσίδα ιδιοτήτων
- Μεταβατική ιδιότητα

8.3.1 Απλό γεγονός

Το απλό γεγονός μιας κλάσης αναπαρίσταται στην Datalog ως:

person(andrew).

Και μιας ιδιότητας ως:

likes(andrew, maria).

Με andrew, maria γνωστές μεταβλητές

Στην OWL υπάρχουν οι εντολές ClassAssertion και ObjectPropertyAssertion που συνδέουν άτομα με κλάσεις ή ιδιότητες.

ClassAssertion(a:person a:andrew)

ObjectPropertyAssertion(a:likes a:andrew a:maria)

Σημείωση: Υπάρχει και η εντολή DataPropertyAssertion που συνδυάζει άτομο με κάποια σταθερά, όπως έναν αριθμό.

DataPropertyAssertion (a:hasAge a:andrew "19"^^xsd:integer)

Στα πλαίσια της παρούσας πτυχιακής δεν θα ασχοληθούμε με αυτό.

8.3.2 Υποκλάση και Υποιδιότητα

Η έννοια της υποκλάσης σε Datalog ορίζεται ως:

awesome (X) :- has_cool_clothes (X).

Η έννοια της υποιδιότητας σε Datalog ορίζεται ως:

has_pet (X,Y) :- has_dog(X,Y).

Στην OWL η έννοια της υποκλάσης εκφράζεται μέσω της εντολής SubClassOf, ενώ της υποιδιότητας, SubObjectPropertyOf.

SubClassOf(a:has_cool_clothes a:awesome)

SubObjectPropertyOf(a:has_dog a:has_pet)

Σημείωση: Υπάρχει και η εντολή SubDataPropertyOf που κάνει το ίδιο για dataproperties, ιδιότητες μεταξύ ατόμων και σταθερών. Στα πλαίσια της παρούσας πτυχιακής δεν θα ασχοληθούμε με αυτό.

8.3.3 Συμμετρία

Η ιδιότητα της συμμετρίας μεταξύ ατόμων σε ιδιότητες σε Datalog εκφράζεται ως:

friend (X, Y) :- friend (Y, X).

Σε OWL εκφράζεται ως:

SymmetricObjectProperty(a:friend)

8.3.4 Αυτοπαθής ιδιότητα

Αυτή η ιδιότητα εκφράζεται στην Datalog ως εξής:

likes (X, X).

Σε OWL εκφράζεται ως εξής:

ReflexiveObjectProperty(a:likes)

8.3.5 Τομή κλάσεων

Αν η τομή δυο κλάσεων είναι υποκλάση μιας άλλης, τότε, στην Datalog, ισχύει ότι:

americanHot(X):- americanPizza (X), hotPizza (X).

Σε OWL χρησιμοποιούμε την ObjectIntersectionOf ως εξής:

SubClassOf (ObjectIntersectionOf(pizza:americanPizza, pizza:hotPizza) pizza:americanHot)

Αν όμως η τομή των δύο κλάσεων είναι υπερκλάση μιας άλλης, τότε, στην Datalog, ισχύει ότι:

spicyPizza(X):- hotPizza (X). hotSaucePizza (X):- hotPizza (X). (δύο κανόνες)

Ενώ σε OWL την χρησιμοποιούμε έτσι:

SubClassOf(pizza:hotPizza ObjectIntersectionOf(pizza:spicyPizza, pizza:hotSaucePizza))

Σημείωση: δεν ορίζεται τομή ιδιοτήτων σε OWL.

8.3.6 Ένωση Κλάσεων

Αν η ένωση δύο κλάσεων είναι υποκλάση μιας άλλης, τότε, στην Datalog, ισχύει ότι:

person (X) :- man (X). person (X) :- woman (X). (δύο κανόνες)

Ενώ στην OWL χρησιμοποιούμε την ObjectUnionOf ως εξής:

SubClassOf(ObjectUnionOf(a:man a:woman) a:person)

Η εντολή, ωστόσο, SubClassOf(a:person ObjectUnionOf(a:man a:woman)) δεν μετατρέπεται σε Datalog, γιατί δημιουργεί ένωση κλάσεων στην κεφαλή του κανόνα.

Σημείωση: δεν ορίζεται ένωση ιδιοτήτων σε OWL.

8.3.7 Ίσα κατηγορήματα.

Η ισότητα κλάσεων σε Datalog εκφράζεται ως:

kid(X) :- young(X). young(X) :- kid(X). (δύο κανόνες)

Ενώ, η ισότητα ιδιοτήτων σε Datalog εκφράζεται ως:

wouldgooutwith (X, Y) :- likes (X, Y). likes (X, Y) :- wouldgooutwith (X, Y).

Σε OWL εκφράζεται ως:

EquivalentClasses(a:kid a:young) ή

equivalentProperty (a:likes a: wouldgooutwith)

Μπορεί να οριστεί ισότητα και μεταξύ τομών κλάσεων, αλλά όχι ιδιοτήτων, λόγω του ότι δεν ορίζεται τομή ιδιοτήτων. Η λογική με την οποία επιτυγχάνεται αυτό είναι παρόμοια με την λογική του ορισμού υποιδιοτήτων.

Πχ

EquivalentClasses(a:Kid ObjectIntersectionOf(a:young a:oblivious))

μετατρέπεται σε:

kid(X) :- young(X),oblivious(X). young(X) :- kid(X), oblivious(X) :- kid(X). (3 κανόνες)

Σημείωση: Στην περίπτωση ισότητας κλάσεων πρέπει και τα δύο ορίσματα του αξιώματος να είναι κλάσεις ή τομή κλάσεων. Στην περίπτωση ισότητας ιδιοτήτων, τα δύο ορίσματα πρέπει να είναι απλές κλάσεις, καθώς δεν ορίζεται τομή ιδιοτήτων. Αλλιώς δεν είναι δυνατή η μετάφραση σε Datalog.

8.3.8 Αντίστροφες ιδιότητες

Αυτό σε Datalog εκφράζεται ως:

hasBase (X,Y) :- isBaseOf (Y,X). isBaseOf(X,Y) :- hasBase (Y,X). (δύο κανόνες)

Ενώ σε OWL ως:

InverseObjectProperties(pizza:hasBase pizza:isBaseOf)

Σημείωση: Πρέπει και τα δύο ορίσματα του αξιώματος να είναι απλές ιδιότητες αλλιώς δεν είναι δυνατή η μετάφραση σε Datalog.

8.3.9 Το αντίστροφο μιας κλάσης

Σε Datalog αλλάζουμε την φορά των μεταβλητών ως εξής:

P(X,Y) :- fatherOf(Y,X).

Σε OWL χρησιμοποιούμε το αξίωμα:

ObjectInverseOf(a:fatherOf)

8.3.10 Καθολικός Περιορισμός (universal restriction)

Αυτός σε Datalog εκφράζεται ως:

D(X) :- cheesypizza (Y), hasTopping (Y,X).

Ενώ, σε OWL με την χρήση του:

ObjectAllValuesFrom (pizza:hasTopping pizza:cheesypizza)

8.3.11 Υπαρξιακός Περιορισμός (Existential Restriction)

Αυτός σε Datalog εκφράζεται ως:

D(X) :- hasTopping (X,Y), fourcheesestopping (Y).

Ενώ, σε OWL με την χρήση του:

ObjectSomeValuesFrom (pizza:hasTopping pizza:fourcheesestopping)

8.3.12 Αλυσίδα ιδιοτήτων

Αυτή σε Datalog εκφράζεται ως:

P(X,Y),Q(Y,Z)

σε OWL εκφράζεται ως

ObjectPropertyChain(P Q).

8.3.13 Μεταβατική ιδιότητα

Σε Datalog εκφράζεται ως:

hasIngredient (X,Z) :- hasIngredient (X,Y), hasIngredient (Y,Z).

Σε OWL χρησιμοποιείται η εντολή:

TransitiveObjectProperty(pizza:hasIngredient)

8.4 Αναδρομή

Μια σημαντική ιδιότητα και των δυο γλωσσών είναι η αναδρομική αναζήτηση δεδομένων. Σε Datalog αυτό υλοποιείται με χρήση αναδρομικών κατηγορημάτων. Αυτά αποτελούνται από την βάση της αναδρομής, που είναι ένας κανόνας με απλά κατηγορήματα και το βήμα της αναδρομής, που είναι κανόνες που περιέχουν το κατηγόρημα που είναι στην κεφαλή του κανόνα. Πχ.

parent (bob,john).

parent (john,pam).

parent (pam,cat).

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

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

Εδώ, η βάση της αναδρομής για το αναδρομικό κατηγόρημα ancestor είναι ο πρώτος κανόνας που έχει το κατηγόρημα parent. Και το βήμα της αναδρομής είναι ο δεύτερος κανόνας όπου αποτελείται από κατηγορήματα ancestor. Εναλλακτικά, το βήμα της αναδρομής θα μπορούσε να περιέχει τη βάση της αναδρομής με ένα κατηγόρημα ancestor ως εξής:

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

Η αναδρομική ιδιότητα σε OWL υλοποιείται με της χρήση της μεταβατικής ιδιότητας των ιδιοτήτων, για να περιγράψουμε το αναδρομικό βήμα και τους ορισμούς άλλων υποιδιοτήτων, για να εκφράσουμε την αναδρομική βάση ως εξής:

TransitiveObjectProperty(a:Ancestor)

SubObjectPropertyOf(a:parent a:Ancestor)

Εδώ, το ancestor έχει την αναδρομική ιδιότητα (αναδρομικό βήμα) και είναι υπερκλάση της κλάσης parent (αναδρομική βάση).

8.5 Η λογική της μετατροπής δεδομένων

Η λογική με την οποία μετατρέπουμε δεδομένα Datalog σε OWL και το αντίστροφο είναι παρόμοια με την λογική του front end των μεταγλωττιστών, με την έννοια ότι διαβάζουμε δεδομένα τα οποία υπακούν σε μια συγκεκριμένη γραμματική και έπειτα από επεξεργασία, καταλήγουμε σε ένα μοναδικό αποτέλεσμα Όποτε πριν αναφερθούμε σε λεπτομέρειες της λογική των μετατροπών, επιβάλλεται η περιγραφή της λογικής του front end των μεταγλωττιστών.

Μια απλή περιγραφή των μεταγλωττιστών όπως αναφέρεται στην Wikipedia είναι: “Μεταγλωττιστής ή μεταφραστής (compiler) ονομάζεται ένα πρόγραμμα υπολογιστή που διαβάζει κώδικα γραμμένο σε μια γλώσσα προγραμματισμού (την πηγαία γλώσσα) και τον μεταφράζει σε ισοδύναμο κώδικα σε μια άλλη γλώσσα προγραμματισμού (τη γλώσσα στόχο). Το κείμενο της εισόδου ονομάζεται *πηγαίος κώδικας* (source code), ενώ η έξοδος του προγράμματος, η οποία συχνά έχει δυαδική μορφή, *αντικειμενικός κώδικας* (object code).” [10] Χρησιμοποιούνται στην μετάφραση κώδικα γραμμένου σε γλώσσες προγραμματισμού υψηλού επιπέδου σε γλώσσες χαμηλότερου επιπέδου, συνήθως σε Assembly. αποτελείται από τρία μέρη, το front end, το middle end και το back end. Το front end ελέγχει αν η είσοδος είναι σωστά γραμμένη και παράγει μία ενδιάμεση αναπαράσταση (*intermediate representation* ή *IR*) του πηγαίου κώδικα. Το middle end επεξεργάζεται την ενδιάμεση αναπαράσταση και κάνει βελτιστοποιήσεις. Το back end μεταφράζει την βελτιστοποιημένη πλέον ενδιάμεση αναπαράσταση σε γλώσσα μηχανής.

Η λογική του front end των μεταγλωττιστών βασίζεται χοντρικά στην ανάγνωση προτάσεων που αποτελούνται από λέξεις (tokens), των οποίων η αλληλουχία ορίζεται από κανόνες. Αυτοί οι κανόνες αποτελούν μια γραμματική (grammar). Όταν η σύνταξη είναι λάθος, τότε πρέπει η μεταγλώττιση να σταματάει καθώς αρκετή πληροφορία δεν μπορεί να μεταγλωττιστεί, με αποτέλεσμα να χάνεται και τα αποτελέσματα μεταξύ του πηγαίου και του αντικείμενου κώδικα να μην ταυτίζονται.

Στην πράξη, η είσοδος αναλύεται μέσα από τρία στάδια:

- Την λεκτική ανάλυση (lexical analysis). Εδώ αναγνωρίζονται οι χαρακτήρες και μετατρέπονται σε μια λίστα από λέξεις (tokens).

- Την συντακτική ανάλυση (syntactic analysis, parsing). Ελέγχει εάν η παραγόμενη λίστα με λέξεις υπακούει στους κανόνες της γραμματικής. Εάν κάποιος κανόνας δεν ικανοποιείται, τότε η διαδικασία της μεταγλώττισης σταματάει εδώ.
- Την σημασιολογική ανάλυση (semantic analysis). Ανάλογα με τον τύπο του κανόνα που ισχύει, γίνονται συγκεκριμένες ενέργειες και παράγονται συγκεκριμένα αποτελέσματα Αυτές οι ενέργειες είναι καταγεγραμμένες με την μορφή κώδικα και ονομάζονται επισκέπτες (visitors). [10] [11]

Η γραμματική των υποσυνόλων της Datalog και της OWL που μπορούν να μεταφραστούν από τον ένα φορμαλισμό στον άλλο αναφέρεται παρακάτω.

8.6 Σχετικά με το περιβάλλον υλοποίησης

8.6.1 Java

Η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού σχεδιασμένη ώστε να έχει όσο το δυνατόν λιγότερες εξαρτήσεις εφαρμογής. Σκοπός της είναι να επιτρέπει στους προγραμματιστές να γράφουν μία φορά το πρόγραμμά τους και να εκτελείται οπουδήποτε (WORA). Αυτό σημαίνει ότι ο μεταγλωττισμένος κώδικας Java μπορεί να εκτελεστεί σε όλες τις πλατφόρμες ανεξαρτήτως λειτουργικού συστήματος. Οι εφαρμογές Java μεταγλωττίζονται σε bytecode που μπορεί να εκτελεστεί σε οποιαδήποτε εικονική μηχανή Java (JVM) ανεξάρτητα από την αρχιτεκτονική του υπολογιστή. Αν είναι επιθυμητό μπορεί να παραχθεί εκτελέσιμο πρόγραμμα. Το συντακτικό της Java είναι παρόμοιο με της C και C ++, αλλά είναι γλώσσα υψηλότερου επιπέδου. Το 2018, η Java ήταν μια από τις πιο δημοφιλείς γλώσσες και χρησιμοποιήθηκε περισσότερο για την παραγωγή web-εφαρμογών. [9]

Η Java αναπτύχθηκε αρχικά από τον James Gosling στην Sun Microsystems με το όνομα Oak και κυκλοφόρησε το 1995 ως βασική συνιστώσα της πλατφόρμας Java της Sun Microsystems. Οι αρχικοί μεταγλωττιστές Java, οι εικονικές μηχανές και οι βιβλιοθήκες υλοποιήθηκαν αρχικά από την Sun και η χρήση του απαιτούσε άδειες ιδιοκτησίας. Αργότερα η Sun Microsystems εξαγοράστηκε από την Oracle που είναι ιδιοκτήτρια της Java ως και σήμερα. Από τότε έχουν βγει πολλές εκδόσεις της Java με τελευταία την Java 12 αλλά δωρεάν για εμπορική χρήση διατίθεται η Java 8. Επίσης έχουν εξελιχθεί οι τεχνολογίες της Sun με την δημιουργία του GNU Compiler για Java και του GNU Classpath και έχει εκδοθεί πάρα πολύ λογισμικό που να έχει σχέση με την Java. Στα πλαίσια της πτυχιακής θα χρησιμοποιήσουμε τον JavaCC [14] (Java Compiler Compiler) που με την βοήθεια μίας δοσμένης γραμματικής και κώδικα γραμμένο σε Java δημιουργεί έναν μεταγλωττιστή της δοσμένης γραμματικής.

8.6.2 JavaCC

Ο JavaCC, γνωστός και ως Java Compiler Compiler, είναι μία γεννήτρια parser και lexical analyzer, δηλαδή γεννήτρια προγραμμάτων που κάνουν λεξιλογική και συντακτική ανάλυση σε μία δεδομένη είσοδο, γραμμένη σε Java. Συγκεκριμένα ο JavaCC χρησιμοποιεί μία EBNF γραμματική και την μετατρέπει σε έναν parser, γραμμένο σε Java που θα βρίσκει αν μια είσοδος υπακούει στην γραμματική η όχι. Οι parsers που δημιουργεί είναι top-down LL(k) parsers (Left-to-right, Leftmost derivation με k σύμβολα lookahead). Αυτό σημαίνει ότι ο parser διαβάζει την είσοδο από τα αριστερά στα δεξιά διαβάζοντας κάθε φορά k σύμβολα. Το lookahead βοηθά στην ανάγνωση παρόμοιων κανόνων, όπου ο JavaCC δεν μπορεί να διακρίνει με 1 σύμβολο lookahead. [12]

Το lookahead του JavaCC δεν είναι σταθερό σε όλη την γραμματική. Ο JavaCC επιτρέπει την χρήση πολλών διαφορετικών lookahead σε διαφορετικούς κανόνες. Τα

lookahead δεν χρειάζεται καν να είναι αριθμοί. Μπορούν να είναι και υποκανόνες, οπότε διαβάζονται τόσα σύμβολα όσα χρειάζεται για να ικανοποιηθεί ο κανόνας.

Ο JavaCC λειτουργεί σε συνεργασία με τον JTB (Java Tree Builder) [15]. Ο δεύτερος λαμβάνει σαν είσοδο μια γραμματική μέσα σε ένα jj αρχείο και δημιουργεί τα παρακάτω:

- Τον φάκελο syntaxtree που περιέχει κλάσεις για όλες τις μεταβλητές της γραμματικής που δώσαμε σαν είσοδο.
- Τον φάκελο visitor που έχει 8 κλάσεις με τους βασικούς visitors. Η λειτουργία των visitors θα αναλυθεί παρακάτω.
- Μία JavaCC γραμματική, σε άλλο .jj αρχείο, με τις μετατροπές που έπρεπε να γίνουν για να δημιουργηθεί το syntax tree.

Οι δυνατότητες του JavaCC δεν σταματούν εκεί. Μπορεί επίσης, όσο διαβάζει και ελέγχει την είσοδο, να κάνει και άλλες ενέργειες, και πιο συγκεκριμένα, να τρέξει παράπλευρο κώδικα γραμμένο σε Java. Αυτό γίνεται με την βοήθεια των visitors. Οι visitors είναι κλάσεις, γραμμένες σε Java, που κάθε φορά που διαβάζεται είσοδος που ταιριάζει σε μία μεταβλητή, εκτελεί δοσμένο, σε μορφή συνάρτησης, κώδικα. Η δημιουργία αυτών των κλάσεων γίνεται αυτόματα με την μεταγλώττιση και, μαζί με αυτές δημιουργούνται και πολλές συναρτήσεις visit, μία για κάθε λογική μεταβλητή μέσα στην γραμματική. Αυτές οι συναρτήσεις έχουν δύο ορίσματα, ένα αντικείμενο μιας κλάσης που συμβολίζει την λογική μεταβλητή και ένα όρισμα οποιουδήποτε τύπου, και τύπο επιστροφής οποιοδήποτε τύπο. Αυτές οι κλάσεις όμως δεν κάνουν κάτι ιδιαίτερο και η σωστή δημιουργία επισκεπτών προϋποθέτει την δημιουργία άλλων κλάσεων που κληρονομούν από αυτές. Σε αυτή την περίπτωση, είναι απαραίτητος ο ορισμός των τύπων του ορίσματος και του τύπου επιστροφής.

Στα πλαίσια της πτυχιακής γίνεται χρήση ενός depth first visitor για κάθε πρόγραμμα που διαβάζει τα tokens κάνοντας αναζήτηση κατά βάθος. Η λειτουργία του καθενός, βασίζεται σε ένα αρχείο .jj που έχει την γραμματική μαζί με τα σωστά lookahead, ώστε να είναι διακριτές οι περιπτώσεις. Έχουν οριστεί και για τους δύο μεταγλωττιστές, ο τύπος επιστροφής των visit συναρτήσεων να είναι String, ώστε να μεταφέρονται κομμάτια κανόνων από τα κάτω προς τα πάνω επίπεδα και ο τύπος του ορίσματος να είναι αριθμός. Το όρισμα είναι χρήσιμο όμως μόνο στον μεταγλωττιστή OWL σε Datalog.

8.7 Μετατροπή δεδομένων OWL σε Datalog

Η λογική με την οποία μετατρέπουμε δεδομένα OWL σε Datalog περιλαμβάνει την ανάγνωση των αξιωμάτων ξεχωριστά και την απλή μετατροπή τους σε κανόνες Datalog. Εδώ, δεν υφίσταται το πρόβλημα της μετατροπής n-ary κατηγορημάτων σε binary, το οποίο θα μας απασχολήσει ιδιαίτερα στην αντίστροφη διαδικασία. Ωστόσο, κάθε ένα αξίωμα μπορεί να φτιάξει από ένα μέχρι πολλούς κανόνες σε Datalog, το οποίο δεν αποτελεί ιδιαίτερο πρόβλημα.

Βεβαία, η OWL περιλαμβάνει τους ορισμούς των αξιωμάτων και των ατόμων όπως και ένα ιδιαίτερο συντακτικό, τα οποία δεν ορίζονται στην Datalog, οπότε πρέπει να αγνοηθούν κατά τη διάρκεια της μεταγλώττισης.

Η διαδικασία περιλαμβάνει την ανάγνωση ενός αξιώματος και την μετατροπή του σε Datalog, χρησιμοποιώντας τους παραπάνω κανόνες. Ωστόσο, πρέπει να δοθεί πολύ μεγάλη βάση στην τήρηση των περιορισμών, καθώς δεν είναι δυνατή η μετατροπή όλων των συνδέσμων των παραπάνω κανόνων από Datalog σε OWL. Συγκεκριμένα πρέπει να προσέξουμε:

1. Όταν ορίζονται υποκλάσεις και υποιδιότητες, το δεύτερο μέρος (το οποίο θα αποτελέσει την κεφαλή των κανόνων της Datalog) να αποτελείται μόνο από μια κλάση ή μια τομή κλάσεων. Συνδυασμοί που περιλαμβάνουν υπαρξιακό ή καθολικό περιορισμό δεν μπορούν να μεταφραστούν, καθώς αυτό θα δημιουργούσε σχέση της μορφής

P(X,Y),Q(Y) :-....

που δεν ορίζεται σε Datalog με κανένα τρόπο. Ο ορισμός τομής κλάσεων μέσα στην τομή κλάσεων είναι περιττός, οπότε ούτε και αυτός δεν γίνεται δεκτός.

2. Η ισότητα κατηγορημάτων και ο ορισμός των αντίστροφων κατηγορημάτων ορίζεται μόνο για κλάσεις ή ιδιότητες και τομές αυτών και όχι ενώσεις ή περιορισμούς.

Εδώ, σε αντίθεση με την Datalog, δεν υπάρχει πρόβλημα με την ονομασία των μεταβλητών, διότι δεν ορίζονται μεταβλητές στο σώμα των κανόνων και τα λογικά μονοπάτια που ορίζουν αυτοί είναι διακριτά.

8.8 Γραμματική μετατροπής OWL σε Datalog

Όπως όλοι οι μεταγλωττιστές, έτσι και τώρα ο μεταγλωττιστής OWL σε Datalog βασίζεται σε μια γραμματική και ανάλογα με τα μοτίβα που παρατηρούνται, εκτελούνται και οι ανάλογες ενέργειες.

Οι χαρακτήρες που χρησιμοποιεί αυτή η γραμματική είναι:

- Prefix (διαβάζεται αναγκαστικά, καθώς όλα τα προγράμματα owl την περιέχουν, και αγνοείται κατά την μεταγλώττιση)
- (
-)
- :
- Ontology (διαβάζεται αναγκαστικά, καθώς όλα τα προγράμματα owl την περιέχουν, και αγνοείται κατά την μεταγλώττιση)
- Declaration (διαβάζεται αναγκαστικά, καθώς όλα τα προγράμματα owl την περιέχουν, και αγνοείται κατά την μεταγλώττιση)
- AnnotationAssetion (διαβάζεται αναγκαστικά, καθώς όλα τα προγράμματα owl την περιέχουν, και αγνοείται κατά την μεταγλώττιση)
- ClassAssertion
- ObjectPropertyAssertion
- EquivalentClasses
- EquivalentProperty
- TransitiveObjectProperty
- ReflexiveObjectProperty
- SymmetricObjectProperty
- SubClassOf

- SubObjectPropertyOf
- ObjectUnionOf
- ObjectSomeValuesFrom
- ObjectAllValuesFrom
- ObjectIntersectionOf
- Identifier() ::- <IDENTIFIER> (μια οποιαδήποτε συμβολοσειρά. Έτσι αναπαριστώνται οι μεταβλητές, σταθερές, σχέσεις)

Η γραμματική μετατροπής Owl σε Datalog ξεκινάει με μερικά εισαγωγικά τα οποία δεν επεξεργάζονται και μετά μπαίνουμε στην ουσία του κώδικα, το κομμάτι που μπορεί να μεταφραστεί

```
void Goal() ::-
 Preprefix() ( PrefixStatement() )* "Ontology" "(" Classes() ( Statement() )* ")" <EOF>
```

Τα παρακάτω δεν έχουν καμιά αξία και δεν μεταφράζονται. Ούτε και το Classes μεταφράζεται αυτή την στιγμή. Χρησιμοποιήθηκε επειδή η δομή του μοιάζει γραμματικά με την λίστα από κλάσεις.

```
void Preprefix() ::-"Prefix" "(" ":" Identifier() ":" Identifier() ")"
```

```
void PrefixStatement() ::- "Prefix" "(" Identifier() ":" Identifier() ":" Identifier() ")"
```

Κάθε πρόγραμμα Datalog αποτελείται γραμματικά από πολλά είδη αξιωμάτων:

```
void Statement() ::-   Declaration()
                     |    AnnotationAssertion()
                     |    ClassAssertion()
                     |    PropertyAssertion()
                     |    Equivalent()
                     |    EquivalentP()
                     |    InverseObjectProperties()
                     |    TransitiveObjectProperty()
                     |    ReflexiveObjectProperty()
                     |    SymmetricObjectProperty()
                     |    SubClass()
                     |    SubProperty()
                     |    SubUnionClass()
```

Οι δηλώσεις και τα σχόλια είναι υποχρεωτικά στην OWL άλλα δεν έχουν καμιά πληροφορία που μετατρέπεται σε Datalog. Οπότε ανήκουν στην γραμματική, άλλα δεν μεταφράζονται

void Declaration() ::- "Declaration" "(" Identifier() "(" Identifier() ":" Identifier() ")" ")"

void AnnotationAssertion() ::- "AnnotationAssertion" "(" Classes() ")"

Εδώ γίνεται η μετάφραση του απλού γεγονότος μιας κλάσης:

void ClassAssertion() ::-

"ClassAssertion" "(" Identifier() ":" Identifier() Identifier() ":" Identifier() ")"

Αυτό μεταφράζεται σε Identifier2(Identifier4).

Η μετάφραση ενός απλού γεγονότος μιας ιδιότητας μεταφράζεται παρόμοια:

void PropertyAssertion() ::-

"ObjectPropertyAssertion" "(" Identifier() ":" Identifier() Identifier() ":" Identifier() Identifier() ":" Identifier() ")"

Η ισότητα κλάσεων είναι εδώ. Όπως έχουμε προαναφέρει ισότητα ορίζεται μόνο μεταξύ απλών κλάσεων η τομών μεταξύ αυτών:

void Equivalent() ::-

"EquivalentClasses" "(" Return() Return() ")"

Ωστόσο η ισότητα ιδιοτήτων ισχύει μόνο μεταξύ μεμονωμένων ιδιοτήτων, γιατί δεν ορίζονται τομές αυτών.

void EquivalentP() ::-

"equivalentProperty" "("SingleClass() SingleClass()")"

Με παρόμοια λογική μεταφράζεται η εντολή περί αντίστροφων κλάσεων.

void InverseObjectProperties() ::-

"InverseObjectProperties" "(" PropertyIntersection() PropertyIntersection()")"

Οι τρεις ιδιότητες που μεταφράζονται από owl σε Datalog ορίζονται και μεταφράζονται εύκολα.

void TransitiveObjectProperty() ::- "TransitiveObjectProperty" "(" Identifier() ":" Identifier() ")"

void ReflexiveObjectProperty() ::- "ReflexiveObjectProperty" "(" Identifier() ":" Identifier() ")"

void SymmetricObjectProperty() ::- "SymmetricObjectProperty" "(" Identifier() ":" Identifier() ")"

Εδώ ορίζεται η υποκλάση

void SubClass() ::- "SubClassOf" "(" Value() Return()")"

Το δεύτερο μέρος της εντολής μπορεί μόνο να είναι μία κλάση ή μία τομή κλάσεων.

void Return() ::- SingleClass()
| IntersectionOfClasses()

void IntersectionOfClasses() ::- "ObjectIntersectionOf" "(" Classes() ")"

void Classes() ::- (SingleClass())*

void SingleClass() ::- Identifier() ":" Identifier()

Το πρώτο μέρος μπορεί να είναι όμως μία τομή κλάσεων, ένας υπαρξιακός η καθολικός περιορισμός:

void Value() ::- SingleClass()
| ClassIntersection()
| SomeValues()
| AllValues()

Η τομή κλάσεων μπορεί και αυτή με την σειρά της να απότελείται από αλλές τομες κλάσεων, υπαρξιακούς ή καθολικούς περιορισμούς

void ClassIntersection() ::- "ObjectIntersectionOf" "(" ManyValues() ")"

void ManyValues() ::- (Value())*

Ο καθολικός και ο υπαρξιακός περιορισμός αποτελούνται από μία ένωση ιδιοτήτων και από το δεύτερο μέρος που είναι είτε μία μοναδική κλάση, είτε μία τομή από άλλες τομές κλάσεων καθώς και υπαρξιακούς ή καθολικούς περιορισμούς.

void SomeValues() ::- "ObjectSomeValuesFrom" "(" PropertyIntersection() From() ")"

void AllValues() ::- "ObjectAllValuesFrom" "(" PropertyIntersection() From() ")"

void From() ::- SingleClass()
| ClassIntersection()
| SomeValues()
| AllValues()

Ιδιαίτερη αναφορά αξίζει ο ορισμός της αλυσίδας ιδιοτήτων. Μπορεί κάποιες από αυτές να είναι αντεστραμμένες. Επίσης το κατηγόρημα "ObjectInverseOf" μπορεί να έχει μόνο μία ιδιότητα. Επίσης ενώ ορίζονται δυο inverseclasses το ένα μέσα στο άλλο, η γραμματική απαγορεύει αυτή την δυνατότητα ως περιττή.

void PropertyIntersection() ::- SingleClass()
| IntersectionOfClasses2()
| InverseClasses()

void IntersectionOfClasses2() ::- "ObjectPropertyChain" "(" ClassesWithInverse() ")"

void InverseClasses() ::- "ObjectInverseOf" "(" SingleClass()")"

void ClassesWithInverse() ::- (Classes2())*

void Classes2() ::- InverseClasses()
| SingleClass()

Δεδομένων των παραπάνω, ο ορισμός της υποιδιότητας δίδεται παρακάτω και είναι προφανής

void SubProperty() ::- "SubObjectPropertyOf" "(" PropertyIntersection() SingleClass() ")"

Επειδή δεν ορίζεται τομή ιδιοτήτων, το δεύτερο μέρος μπορεί αποκλειστικά και μόνο να είναι κλάση.

Η λογική περί ένωσης κλάσεων πρέπει να είναι ξεχωριστή από την υπόλοιπη γραμματική, λόγω της διαφορετικής λογικής.

void SubUnionClass() ::- "SubClassOf" "(" "ObjectUnionOf" "(" UnionClasses() ")" Return() ")"

void UnionClasses() ::- (UnionClass())*

void UnionClass() ::- Value()

Λαμβάνουμε μια λίστα με σώματα κανόνων από το πρώτο μέρος και απλές κλάσεις που θα αποτελέσουν τις κεφαλές των κανόνων από το δεύτερο. Στην συνέχεια για κάθε κεφαλή ορίζουμε έναν κανόνα για κάθε σώμα. Η λογική με την οποία παράγονται τα σώματα των κανόνων είναι όμοια με πριν, με την διαφορά ότι, αντί για ένα σώμα, έχουμε πολλά.

Προφανώς, επειδή δεν ορίζεται ένωση ιδιοτήτων, η παραπάνω λογική δεν εφαρμόζεται στις ιδιότητες.

8.9 Τεχνικες λεπτομέρειες μεταγλωττιστή OWL σε Datalog

Σε OWL δεν ορίζονται μεταβλητές στο σώμα του κανόνα. Οπότε, κατά την διάρκεια της μεταγλώττισης, πρέπει να φτιάξουμε τις δικές μας μεταβλητές. Χρησιμοποιήθηκαν οι μεταβλητές X1,X2,...,Xn, επειδή η κατασκευή τους θα προϋπόθετε μόνο την γνώση ενός index το οποίο μπορεί να περαστεί ως η παράμετρος στις συναρτήσεις visitors. Οπότε, όταν θέλουμε να μεταφράσουμε μια απλή κλάση, χρησιμοποιούμε την παράμετρο. Στις ιδιότητες, συνήθως, ως δεύτερη παράμετρο-μεταβλητή χρησιμοποιούμε την μεταβλητή αυξημένη κατά 1, ιδίως όταν μιλάμε για κανόνες με ιδιότητες. Σε κανόνες με κλάσεις, οι οποίοι μπορούν να περιέχουν ιδιότητες στο σώμα του κανόνα, δεν γίνεται συνήθως αυτό. Πρέπει να χρησιμοποιήσουμε άλλες παραμέτρους, ιδίως όταν έχουμε εμφωλευμένους ή τομή πολλών περιορισμών. Σε αυτή την περίπτωση, χρησιμοποιείται η μεταβλητή sec_item_val που αυξάνεται σε κάθε τέτοια περίπτωση και μπαίνει ως παράμετρος στο δεύτερο μέρος του περιορισμού. Αυτή γίνεται 1 κάθε φορά που αρχίζει νέος κάνονας, ώστε να είναι πάντα χρήσιμη και αυξάνεται κάθε φορά που διαβάζεται μία ιδιότητα στο σώμα του κανόνα, ώστε να αποτυπώνει την λογική των αλυσίδων ιδιοτήτων. Για την ορθή αποτύπωση των αλυσίδων ιδιοτήτων ορίζεται και η μεταβλητή argu_var. Όταν διαβάζεται μία αλυσίδα ιδιοτήτων, τότε αυτή η μεταβλητή χρησιμεύει ως παράμετρος για την πρώτη μεταβλητή του κατηγορήματος. Στο πρώτο κατηγόρημα της αλυσίδας ορίζεται συγκεκριμένα η τιμή του, ανάλογα με το επίπεδο της, ενώ στην συνέχεια ορίζεται ως sec_item_val-1, ώστε να αποτυπώσει την εναλλαγή των μεταβλητών.

Σε κάθε περίπτωση, όμως, η δημιουργία των σωμάτων των κανόνων δημιουργείται στην συνάρτηση argu_set, η οποία ελέγχει αν έχουμε αντίστροφο κατηγόρημα ή όχι και ορίζει την σειρά των μεταβλητών μέσα στο κατηγόρημα. Η συνάρτηση εκτελεί διαφορετικές ενέργειες αν είμαστε σε κανόνα με κλάση ή ιδιότητα, χάρη στην μεταβλητή single που είναι αληθής αν είμαστε σε κανόνα με κλάση. Στην συνέχεια αποθηκεύει την παραγόμενη συμβολοσειρά, στην συμβολοσειρά elems και την επιστρέφει, γιατί πολλές συναρτήσεις συνήθως χρησιμοποιούν μόνο ένα κατηγόρημα που θέλουν να αποτιμηθεί.

Η συμβολοσειρά elems χρησιμεύει ως λίστα για τα κατηγορήματα, ή τους κανόνες στην περίπτωση της ένωσης κλάσεων. Όλα τα δεδομένα αποθηκεύονται εκεί και διαχωρίζονται ή με κενό, στις πιο πολλές περιπτώσεις, ή με | στην περίπτωση της ένωσης κλάσεων, γιατί έχουμε πολλούς κανόνες. Σε κάθε περίπτωση, παίρνουμε τα κομμάτια της λίστας με την χρήση του Stringtokenizer και τα χρησιμοποιούμε στην εκτύπωση. Ο λόγος που δεν χρησιμοποιούμε μια λίστα από Strings είναι, ότι αυτό είναι πιο ακριβό στην μνήμη, απαιτεί μια δομή και δεν είναι απαραίτητη στην πράξη, καθώς η μορφή που επιστρέφει η argu_set είναι σχεδόν έτοιμη για εκτύπωση και χρειάζεται μόνο

η προσθήκη κομμάτων. Μόνο στην ισότητα κατηγορημάτων και στις αντίστροφες ιδιότητες, είναι απαραίτητη η αλλαγή του τρόπου εκτύπωσης.

Σε αντίθεση με την Datalog σε OWL, οι περισσότεροι περιορισμοί ικανοποιούνται σε επίπεδο γραμματικής, με τον κώδικα να πρέπει απλώς να εκτυπώνει σωστά τα στοιχεία χρησιμοποιώντας την παράμετρο και την sec_item_val οπού χρειάζεται. Ωστόσο η μεταχείριση των ιδιοτήτων των κλάσεων και η παραγωγή των γεγονότων δεν απότελούν περίπλοκες περιπτώσει καθώς είναι συγκεκριμένη η επιστροφή, όπως ορίζεται παραπάνω. Στον ορισμό ιδιοτήτων μάλιστα η έξοδος είναι συγκεκριμένη και μάλιστα χρησιμοποιεί τις μεταβλητές X,Y.

8.10 Μετατροπή δεδομένων Datalog σε OWL

Σε αντίθεση με την προηγούμενη διαδικασία, εδώ δεν δηλώνουμε τις μεταβλητές και τους κανόνες από πριν. Ωστόσο, είναι δυνατός ο συνδυασμός πολλών μεμονωμένων κανόνων της Datalog σε ένα αξίωμα της OWL, πχ. πολλοί κανόνες ορισμού υποκλάσης και υποιδιότητας συνενώνονται σε έναν κανόνα ορισμού υποκλάσης που χρησιμοποιεί ένωση κλάσεων. Φυσικά αυτό μπορεί να παρακαμφθεί, με συνέπεια να δημιουργηθούν παρά πολύ απλά αξιώματα, κάτι που δεν αποτελεί ιδιαίτερο πρόβλημα καθώς η βάση ακόμα θα περιέχει όλη την πληροφορία.

Συνοπτικά, το μεταφρασμένο πρόγραμμα σε OWL δεν θα περιλαμβάνει:

1. Ένωση κλάσεων, ως υποκλάση μιας άλλης. Θα ορίζουμε πολλά αξιώματα τα οποία θα μπορούν να ενωθούν σε μια ένωση κλάσεων.

2. Τομή κλάσεων, ως υπερκλάση μιας άλλης, με την ίδια λογική

3. Ισότητα κατηγορημάτων. Θα ορίζουμε ότι το ένα κατηγόρημα είναι υποκλάση ή υποιδιότητα του άλλου και το αντίστροφο.

4. Αντίστροφα κατηγορήματα. Θα ορίζουμε ότι το ένα κατηγόρημα είναι υποκλάση ή υποιδιότητα του αντιστρόφου κατηγορήματος του άλλου και το αντίστροφο.

Αξιοσημείωτο είναι ότι υπάρχει δυνατότητα να εκφραστεί ο καθολικός περιορισμός ως υπαρξιακός στο σώμα του κανόνα, αν σκεφτούμε ότι η έκφραση του υπαρξιακού περιορισμού σε Datalog είναι:

p(X) :- q(X,Y),t(Y)

ενώ του καθολικού είναι:

p(Y) :- q(X,Y),t(X).

Αν μεταφραστεί το κατηγόρημα Q με την αντίστροφη ιδιότητα του Q', τότε θα έχουμε:

p(Y) :- q'(Y,X),t(X).

το οποίο απότελεί υπαρξιακός περιορισμός.

Έτσι ο παρακάτω καθολικός περιορισμός:

p(X) :- z(Y,X),t(Y).

μεταφράζεται σε OWL ως:

SubClassOf(ObjectSomeValuesFrom(ObjectInverseOf(:z) :t)

Προσοχή πρέπει να δοθεί στο γεγονός ότι τα κατηγορήματα στο σώμα του κανόνα της Datalog μπορεί να είναι με οποιαδήποτε σειρά και να βγάζουν ισοδύναμους κανόνες σε OWL.

Στην Datalog τα λογικά μονοπάτια ορίζονται με την βοήθεια μεταβλητών. Οπότε είναι απαραίτητη η τήρηση των περιορισμών που αναφέραμε παραπάνω όσων αφορά την διαχείριση μεταβλητών στο σώμα του κανόνα.

8.11 Γραμματική μετατροπής Datalog σε OWL

Όπως όλοι οι μεταγλωττιστές έτσι και τώρα ο μεταγλωττιστής Datalog σε OWL βασίζεται σε μια γραμματική και ανάλογα με τα μοτίβα που παρατηρούνται, εκτελούνται και οι ανάλογες ενέργειες.

Οι χαρακτήρες που χρησιμοποιεί αυτή η γραμματική είναι:

- :-
- (
-)
- .
- ,
- Identifier() ::- <IDENTIFIER> (μια οποιαδήποτε συμβολοσειρά. Έτσι αναπαριστώνται οι μεταβλητές, σταθερές, σχέσεις)

Η γραμματική του υποσυνόλου της Datalog που μετατρέπεται σε OWL παρατίθεται παρακάτω:

Goal() ::- (Statement())* <EOF>

Προφανώς κάθε πρόγραμμα Datalog αποτελείται από πολλούς κανόνες, των οποίων η μορφή είναι ως εξής:

```
Statement() ::-      SingleAssertion()
              |      DoubleAssertion()
             |       MultiAssertion()
             |       SingleRule()
              |      DoubleRule()
              |      MultiRule()
```

Οι περιπτώσεις είναι οι εξής: το απλό γεγονός μιας κλάσης, μιας ιδιότητας ή και μιας n-ary ιδιότητας (με παραπάνω από δυο μεταβλητές) και ένας κανόνας μιας κλάσης, μιας ιδιότητας και μιας n-ary ιδιότητας.

Για το απλό γεγονός μιας κλάσης ισχύει ότι:

SingleAssertion() ::- Identifier() "(" Identifier() ")" "."

Το οποίο, όπως προαναφέραμε, μεταφράζεται σε OWL ως:

ClassAssertion(a:Identifier1 a:Identifier2)

Ομοίως, για το απλό γεγονός μιας ιδιότητας ισχύει ότι:

DoubleAssertion() ::- Identifier() "(" Identifier() "," Identifier() ")" "."

Το οποίο, όπως προαναφέραμε, μεταφράζεται σε OWL ως:

ObjectPropertyAssertion(a:Identifier1 a:Identifier2 a:Identifier3)

Αν ο κάνονας όπως ορίζεται σε Datalog έχει δυο ισοδύναμα identifiers μέσα στην παρένθεση που ξεκινάνε με κεφαλαίο, τότε έχουμε ορισμό της ιδιότητας ως αυτοπαθούς (reflexive object property)

Το απλό γεγονός μιας n-ary ιδιότητας πρέπει πάντα να έχει παραπάνω από δυο μεταβλητές. Οπότε ο κανόνας της γραμματικής πρέπει να περιέχει τουλάχιστον δυο μεταβλητές και μια λίστα από τις επιπλέον. Οπότε ισχύει ότι:

MultiAssertion() ::- MultiElem() "."

MultiElem() ::- Identifier() "(" Identifier() "," Identifier() "," Identifier() More() ")"

More() ::- (Temp())*

Temp() ::- "," Identifier()

Οι επιπλέον μεταβλητές του κατηγορήματος διαχωρίζονται με κόμμα που μπαίνει πριν από την συμβολοσειρά.

Η μετατροπή αυτού του n-ary κατηγορήματος σε OWL προϋποθέτει την δημιουργία πολλών σχέσεων της μορφής <identifier>1, <identifier>2 … <identifier>n και χρειάζεται ακόμα και ένα μοναδικό index. Οπότε, τελικά το γεγονός μεταφράζεται σε OWL ως

ObjectPropertyAssertion(a:<Identifier1>0 a:Identifier1 a:index)

ObjectPropertyAssertion(a:<Identifier1>1 a:Identifier2 a:index)

…..

ObjectPropertyAssertion(a:<Identifier1>n-1 a:Identifiern a:index)

Ο ορισμός των κανόνων για κλάσεις και ιδιότητες σε Datalog αποτελείται από literals απλών κλάσεων και ιδιοτήτων

Literal() ::- SingleClass()

| Property()

Το παρακάτω είναι το literal μιας απλής κλάσης:

SingleClass() ::- Identifier() "(" Identifier() ")"

Και αυτό μιας ιδιότητας:

Property() ::- Identifier() "(" Identifier() "," Identifier() ")"

Τα identifiers μέσα στην παρένθεση δεν θα μας χρειαστούν.

Ο ορισμός του απλού κανόνα είναι περίπλοκος με την έννοια, ότι μπορεί να περιέχει literals, τόσο κλάσεων, όσο και ιδιοτήτων. Οι ιδιότητες μπορεί να είναι ευθείες ή αντεστραμμένες. Η εναλλαγή μεταξύ των λογικών μονοπατιών μπορεί να γίνει σε οποιαδήποτε στιγμή. Οπότε πρέπει να ορίσουμε όλες τις περιπτώσεις.

SingleRule() ::- Identifier() "(" Identifier() ")" ":-" SingleRuleTail() "."

Η κεφαλή του κανόνα ορίζεται ξεκάθαρα και το σώμα έχει πολύ περίπλοκη ανάλυση.

Το πρώτο κατηγόρημα του σώματος μπορεί να είναι, είτε κλάση, είτε ιδιότητα, όπως και το υπόλοιπο σώμα. Η γραμματική δεν μπορεί να ανιχνεύσει τα λογικά μονοπάτια, κάτι που μπορεί να γίνει μόνο με τη βοήθεια κώδικα.

SingleRuleTail() ::- Literal() ClassTail()

ClassTail() ::- (ClassTailElem())*

ClassTailElem() ::- "," Literal()

Η λογική για τους κανόνες με ιδιότητες είναι αρκετά πιο απλή με την έννοια ότι εδώ δεν μπορούμε να έχουμε γεγονότα κλάσεων. Αντιθέτως, έχουμε μια λίστα από κατηγορήματα που ορίζεται όπως παραπάνω.

DoubleRule() ::- Identifier() "(" Identifier() "," Identifier() ")" ":-" DoubleRuleTail() "."

DoubleRuleTail() ::- Property() PropertyTail()

PropertyTail() ::- (PropertyTailElem())*

PropertyTailElem() ::- "," Property()

Οπότε διακρίνουμε τις εξής περιπτώσεις:

Αν η λίστα έχει μόνο ένα στοιχείο, τότε μεταφράζεται σε OWL ως:

SubObjectPropertyOf(a:Property1 a:Identifier1)

Ενώ αν έχει πολλά στοιχεία, τότε μεταφράζεται ως:

SubObjectPropertyOf(ObjectIntersectionOf(a:Property1… a:Propertyn) a:Identifier1)

Οφείλουμε, κατά την διάρκεια ελέγχου της λίστας, να ελέγξουμε, αν ένα από τα κατηγορήματα ταυτίζεται με αυτό της κεφαλής του κανόνα. Σε αυτή την περίπτωση, ορίζουμε το κατηγόρημα ως αναδρομικό (transitive object property)

Επίσης, οφείλουμε να ελέγξουμε, εάν το σώμα του κανόνα αποτελείται από μόνο ένα κατηγόρημα, ίδιο με το αρχικό αλλά αντεστραμμένο. Σε αυτή την περίπτωση, ορίζουμε το κατηγόρημα ως συμμετρικό.

Η επεξεργασία των n-ary κατηγορημάτων, σε επίπεδο γραμματικής, γίνεται όπως και πριν. Υπάρχουν όμως δύο σημαντικές διαφορές. Αρχικά δεν μπορούμε να χρησιμοποιήσουμε απλές κλάσεις ή ιδιότητες, καθώς αυτό θα παραβίαζε την λογική της μετάφρασης που έχουμε αναφέρει πιο πριν. Επίσης το σώμα του κανόνα μπορεί να έχει μόνο ένα κατηγόρημα, καθώς δεν ορίζεται τομή μεταξύ δύο ιδιοτήτων. Το καλό είναι ότι μπορούμε να χρησιμοποιήσουμε ένα literal με όσες μεταβλητές (πάνω από 2 προφανώς) θέλουμε, αρκεί αυτές να έχουν οριστεί στην κεφαλή του κανόνα, σύμφωνα με τους περιορισμούς της Datalog.

MultiRule() ::- MultiElem() ":-" MultiElem()"."

Το πρόβλημα είναι στην μετάφραση σε OWL. Πρέπει για κάθε παραγόμενη σχέση που σχηματίζεται μετά τον μετασχηματισμό, να γίνεται αντιστοιχία με όσες παραγόμενες σχέσεις από την κεφαλή του κανόνα έχουν την ίδια μεταβλητή (μπορεί να είναι παραπάνω από μια, όπως έχουμε προαναφέρει). Μετά από αυτή την αντιστοιχία, δημιουργούνται πολλές σχέσεις υποιδιότητας. Αυτή η διαδικασία γίνεται σε επίπεδο κώδικα.

8.12 Τεχνικές λεπτομέρειες μεταγλωττιστή Datalog σε OWL

Σε αντίθεση με τον μεταγλωττιστή OWL σε Datalog, οι περισσότεροι διαχωρισμοί μεταξύ των περιπτώσεων γίνονται σε επίπεδο κώδικα. Οπότε και έχουν οριστεί αρκετές υποστηρικτικές δομές. Όπως και πριν, πολλοί κανόνες απότιμώνται από πολλούς επισκέπτες. Οπότε τα δεδομένα τοποθετούνται σε λίστες και maps που είναι ορίσματα της κλάσης. Ο λόγος που δεν χρησιμοποιούμε String για την απόθήκευση των δεδομένων αυτή την φορά είναι ότι χρειαζόμαστε και πολλά indexes, όπως θα δούμε στην συνέχεια.

Σε αντίθεση με πριν, εδώ υπάρχουν πολλές μεταβλητές οι οποίες ορίζουν τα λογικά μονοπάτια. Αυτές διαφέρουν από κανόνα σε κανόνα. Οπότε, στα πλαίσια του κώδικα, οι μεταβλητές συνήθως αντικαθιστώνται με αριθμούς, που συμβολίζουν την σειρά τους μέσα στον κανόνα.

Η λογική με την οποία φτιάχνουμε τους κανόνες διαφέρει δραστικά ανάλογα με το αν έχουμε κανόνες με ιδιότητες και κλάσεις ή n-ary κατηγορήματα. Αυτό έχει σημασία γιατί η λογική δημιουργίας κανόνων και στις δυο περιπτώσεις διαφέρει. Όταν έχουμε κανόνες με ιδιότητες και κλάσεις, χρησιμοποιούμε την βοηθητική κλάση Simple_Literal που έχει σαν ορίσματα δυο αριθμούς, που στην πράξη είναι μεταβλητές, το όνομα της μεταβλητής και έναν δείκτη που δείχνει αν είναι αντεστραμμένο το κατηγόρημα στο σώμα του κανόνα η όχι. Οι δυο αριθμοί είναι οι x,y και πρέπει x < y αλλιώς το κατηγόρημα ορίζεται σαν αντίστροφο. Αν x = y, τότε μιλάμε για απλή κλάση αλλιώς μιλάμε για ιδιότητα.

Για τον σχηματισμό των κανόνων χρησιμοποιούμε το απλό Simple_Literal head, που συμβολίζει την κεφαλή του κανόνα, και την λίστα από Simple_Literal body, που αντιστοιχεί στο σώμα του κανόνα. Η κεφαλή σχηματίζεται αυτόματα, χάρη στην γραμματική, ενώ το σώμα γεμίζει, χάρη στους άλλους επισκέπτες (visitors).

Όταν θέλουμε να φτιάξουμε ένα καινούργιο Simple_Literal, πρώτα τοποθετούμε τις μεταβλητές που διαβάζουμε σε έναν map String σε Int, όπου ο αριθμός είναι η σειρά με την οποία διαβάστηκε η μεταβλητή. Ο map του αρχείου είναι ο variables_read και η αποθήκευση των μεταβλητών σε αυτόν γίνεται μέσω της μεθόδου variable_insert. Η εισαγωγή στο σώμα του κανόνα και η ρύθμιση, αν επρόκειτο για αντίστροφη ιδιότητα ή όχι, γίνεται μέσω της body_insert.

Αν μιλάμε για δημιουργία γεγονότος κλάσης η ιδιότητας, τότε όλα τα δεδομένα αποθηκεύονται στην κεφαλή του κανόνα, η οποία, φυσικά, μετά εκτυπώνεται. Αν μιλάμε

για κανόνα με ιδιότητες τότε ορίζουμε όλα τα Simple_Literal μέσα στο σώμα του κανόνα ως υποιδιότητες του Simple_Literal της κεφαλής. Οι κανόνες με κλάσεις προκύπτουν πιο πολύπλοκα όμως, καθώς μπορεί να περιέχουν ιδιότητες. Αρχικά, τα λογικά μονοπάτια που ορίζονται στο σώμα του κανόνα, μπορεί να μην καταλήγουν σε κλάση, αλλά σε ιδιότητα. Οπότε, σε κάθε τέτοια περίπτωση, πρέπει να ορίσουμε ένα ακόμα κατηγόρημα, για κάθε μεταβλητή που λείπει, το οποίο θα συμβολίζει το Owl:Thing. Στην συνέχεια ταξινομούμε τα Simple_Literal σε σχέση με τα x,y. Το μέτρο σύγκρισης είναι y*(αριθμός μεταβλητών) + x. Έτσι επιλύουμε την δυσκολία ότι μπορεί τα κατηγορήματα να είναι ανακατεμένα. Μετά χρησιμοποιούμε την μέθοδο body_ret για να φτιάξουμε την τελική μορφή του σώματος του κανόνα.

Πρέπει να προσέξουμε όμως:

- Αν, σε κανόνα με ιδιότητες, βρούμε κατηγόρημα με το ίδιο όνομα με της κεφαλής, τότε πρέπει να ορίσουμε την ιδιότητα ως μεταβατική
- Αν, σε γεγονός με ιδιότητα, έχουμε δυο ίδια ορίσματα με πρώτο γραμμα κεφαλαίο, ορίζουμε την ιδιότητα ως συμμετρική

Η λογική για τα n-ary κατηγορήματα διαφέρει σε τέτοιο βαθμό, που χρησιμοποιούμε διαφορετικές δομές. Συγκεκριμένα χρησιμοποιούμε την κλάση Multi_Literal που αποτελείται από το όνομα του κατηγορήματος και μια λίστα από μεταβλητές συμβολισμένες ως αριθμοί. Τώρα το σύνολο του κανόνα, μαζί με την κεφαλή, αποθηκεύεται σε μια μεγάλη λίστα την multi_rule επειδή η κεφαλή δεν είναι τόσο ευδιάκριτη στην δομή της γραμματικής. Η κεφαλή δεν παύει όμως να ξεχωρίζει, γιατί αποτελεί πάντα το πρώτο στοιχείο της λίστας αυτής. Τα ονόματα των μεταβλητών της κεφαλής αποθηκεύονται σε μια άλλη λίστα, την multi_variables. Τα ονόματα μας χρειάζονται, τόσο στην εκτύπωση του γεγονότος με multi_literal, όσο και στους κανόνες με literals. Η λίστα με τα κατηγορήματα δημιουργείται χωρίς την βοήθεια κάποιας μεθόδου, αφού όλο το βάρος πέφτει σε μια συνάρτηση visitor. Στην ανάθεση γεγονότων χρειάζεται να αποτυπώσουμε και ένα ξεχωριστό index γεγονότος, το οποίο αποτελεί όρισμα του νέου κατηγορήματος και αυξάνεται κάθε φορά που χρησιμοποιείται. Στους κανόνες γίνεται αντιστοίχιση των μεταβλητών της κεφαλής και των κατηγορημάτων. Η διαδικασία απαιτεί την χρήση των ονομάτων των μεταβλητών της κεφαλής.

Σε κάθε μια, όμως, από τις παραπάνω περιπτώσεις δημιουργείται ένα String με τον κανόνα μεταφρασμένο σε OWL. Όλοι οι κανόνες αποθηκεύονται σε ένα String, το rules, που εκτυπώνεται στο τέλος.

Κατά την διάρκεια ανάγνωσης των κανόνων, δημιουργούμε και ένα Set από Strings, το Declaration_Set, όπου αποθηκεύουμε όλες τις δηλώσεις. Στο τέλος αυτής, εκτυπώνουμε το σύνολο αυτό πριν τους κανόνες.

9. ΑΝΟΙΧΤΑ ΠΡΟΒΛΗΜΑΤΑ

Στο κεφάλαιο αυτό, κάνουμε μία σύνοψη των προβλημάτων που εμφανίζονται κατά την μετατροπή Datalog προγραμμάτων σε OWL και αντίστροφα, τα οποία δεν αντιμετωπίστηκαν στην πτυχιακή.

- Κάτι που δεν αναφέρονταν στο paper των Grosof, Horrocs, Volz και Decker[13] και μπορεί να μεταφραστεί από Datalog σε OWL είναι σχέσεις της μορφής:

  dogOwner(X):- has_dog(X,_).

  Σε OWL αυτό υλοποιείται ως:

  SubClassOf (ObjectSomeValuesFrom (:has_dog owl:thing) dog_owner)

  Αυτή η περίπτωση έχει καλυφθεί στην μετατροπή από Datalog σε OWL. Ωστόσο δεν είναι γνωστό εάν υπάρχουν και άλλες τέτοιες περιπτώσεις. Επίσης το owl:thing δεν γίνεται αντιληπτό στην μετατροπή από OWL σε Datalog. Οπότε η εύρεση του συνόλου περιπτώσεων που μετατρέπεται από την μία γλώσσα στην άλλη το owl:thing / _ αποτελεί πρόβλημα.

- Ένα ακόμα πρόβλημα της μετατροπής Datalog σε OWL είναι ο έλεγχος αν οι μεταβλητές στους κανόνες ακολουθούν το format που δόθηκε στην παράγραφο 8.3. Προφανώς οι συνδυασμοί μεταβλητών στους κανόνες είναι πολλοί και τα κατηγορήματα μπορούν να δοθούν με διαφορετική σειρά. Η ταξινόμηση των κατηγορημάτων ήταν μία ιδέα που λύνει τι πρόβλημα της σειράς των κατηγορημάτων αλλά δεν γίνεται κανένας έλεγχος για το αν τα κατηγορήματα με τις δοθείσες μεταβλητές μπορούν να μετατραπούν σε OWL.

- Υπάρχουν και άλλοι έλεγχοι που δεν έχουν υλοποιηθεί ακόμα. Ο κυριότερος είναι ο έλεγχος των δηλώσεων των κλάσεων, ιδιοτήτων και ατόμων στον μεταγλωττιστή OWL σε Datalog. Για την ώρα το πρόγραμμα αγνοεί τις δηλώσεις όπως και τα σχόλια. Επίσης δεν ελέγχει τα prefixes στην αρχή των προγραμμάτων. Ο έλεγχος των παραπάνω θα απαιτούσε την χρήση ενός ακόμα visitor που στην πράξη θα ελέγχει όλο το έγγραφο πριν το μεταγλωττίσει.

- Ένα τρίτο πρόβλημα είναι ότι η Datalog έχει πολλές επεκτάσεις οι οποίες δεν χρησιμοποιήθηκαν στα πλαίσια της πτυχιακής. Πολλές από αυτές θα μπορούσαν ίσως να μετατραπούν σε OWL. Η χρήση επεκτάσεων όμως στην Datalog την απομακρύνει από την λογική των φράσεων Horn και την φέρνει πιο κοντά στην λογική πρώτης τάξης κάτι που βλάπτει την ίδια την βάση της. Οπότε πρέπει να αποφασιστεί αν κάτι τέτοιο είναι ορθό και σκόπιμο να γίνει.

- Ένα τέταρτο πρόβλημα έγκειται στην μετατροπή n-ary κατηγορημάτων σε binary. Στα πλαίσια της παρούσας πτυχιακής έχουμε μετατρέψει κανόνες με ένα κατηγόρημα στο σώμα του κανόνα καθώς κανόνες με δύο κατηγορήματα απλά, συνήθως, δεν μεταγλωττίζονται. Ωστόσο δεν είναι ακόμα γνωστό πόσες και ποιες μεταβλητές μπορεί να περιέχει αυτό το κατηγόρημα. Επίσης ίσως υπάρχει πρόβλημα έκφρασης αν το κατηγόρημα της κεφαλής η του κανόνα έχει δύο ίδιες μεταβλητές.

- Τέλος, πρέπει να ελεγχθεί και ο τρόπος υλοποίησης αναδρομής από Datalog σε OWL. Μέχρι στιγμής μεταγλωττίζονται προγράμματα της μορφής:

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

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

όπου ο πρώτος κανόνας αποτελεί την αναδρομική βάση και ο δεύτερος το αναδρομικό βήμα. Το πρόγραμμα:

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

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

μεταφράζεται όμως με τον ίδιο τρόπο καθώς τα δύο προγράμματα είναι ισοδύναμα.

Ωστόσο δεν είναι γνωστό εάν η μετάφραση προγραμμάτων που η αναδρομική βάση έχει διαφορετικά κατηγορήματα από το αναδρομικό βήμα μεταφράζεται σε OWL, πχ εάν το πρόγραμμα

grandparent(X,Y):-parent(X,Y),parent(Y,Z).

ancestor(X,Y) :- grandparent(X,Y).

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

μπορεί να μετατραπεί από Datalog σε OWL.

Στην περίπτωση αυτή, ίσως θα ήταν σκόπιμο να χρησιμοποιηθούν fold/unfold τεχνικές στο αρχικό πρόγραμμα

ΑΝΑΦΟΡΕΣ

[1] Wikipedia, First-Order Logic; https://en.wikipedia.org/wiki/First-order_logic. [Προσπελάστηκε 27/10/19]

[2] Manolis Koubarakis, Η Λογική Πρώτης Τάξης; http://cgi.di.uoa.gr/~ys02/dialekseis2013/fol-syntax1spp.pdf. [Προσπελάστηκε 27/10/19]

[3] Wikipedia, Horn Clause; https://en.wikipedia.org/wiki/Horn_clause. [Προσπελάστηκε 27/10/19]

[4] Wikipedia, Datalog; https://en.wikipedia.org/wiki/Datalog. [Προσπελάστηκε 27/10/19]

[5] Wikipedia, Web Ontology Language; https://en.wikipedia.org/wiki/Web_Ontology_Language [Προσπελάστηκε 27/10/19]

[6] Wikipedia, Semantic Web; https://en.wikipedia.org/wiki/Semantic_Web. [Προσπελάστηκε 27/10/19]

[7] Wikipedia, Description Logic; https://en.wikipedia.org/wiki/Description_logic. [Προσπελάστηκε 27/10/19]

[8] Manolis Koubarakis, Introduction to Description Logics; http://cgi.di.uoa.gr/~pms509/lectures/dl-intro.pdf. [Προσπελάστηκε 27/10/19]

[9] Wikipedia, Java; https://en.wikipedia.org/wiki/Java_(programming_language). [Προσπελάστηκε 27/10/19]

[10] Βικιπέδια, Λήμμα Μεταγλωττιστής; https://el.wikipedia.org/wiki/Μεταγλωττιστής. [Προσπελάστηκε 27/10/19]

[11] Yannis Smaragdakis, Compilers, University Of Athens; http://cgi.di.uoa.gr/~thp06/lectures.html. [Προσπελάστηκε 27/10/19]

[12] Wikipedia, JavaCC; https://en.wikipedia.org/wiki/JavaCC. [Προσπελάστηκε 27/10/19]

[13] B.Grosof, I.Horrocks, R.Volz, S.Decker, “Description Logic Programs: Combining Logic Programs with Description Logic”, In Proc. of WWW 2003, Budapest, Hungary, May 2003, pp. 48-57. ACM, 2003.

[14] JavaCC - The Java Parser Generator; https://javacc.org/. [Προσπελάστηκε 27/10/19]

[15] JTB: The Java Tree Builder Homepage; http://compilers.cs.ucla.edu/jtb/. [Προσπελάστηκε 27/10/19]

[16] S. Ceri, G. Gottlob, L. Tanca, “What You Always Wanted to Know About Datalog (And Never Dared to Ask)”, *IEEE Transactions of Knowledge and Data Engineering*, Vol. 1, No. 1, March 1989.

[17] Manolis Koubarakis, An Introduction to OWL 2; http://cgi.di.uoa.gr/~pms547/lectures/introduction-to-owl2-1spp.pdf [Προσπελάστηκε 27/10/19]

[18] W3C, OWL Web Ontology Language: Overview, World Wide Web Consortium (W3C) Recommendation, February 2004; https://www.w3.org/TR/owl-features/. [Προσπελάστηκε 27/10/19]

[19] W3C, OWL Web Ontology Language: Guide, World Wide Web Consortium (W3C) Recommendation, February 2004; https://www.w3.org/TR/owl-guide/. [Προσπελάστηκε 27/10/19]

[20] Dahchour, Mohamed & Pirotte, Alain. (2002). “The Semantics of Reifying n-ary Relationships as Classes”, *ICEIS 2002 - Proceedings of the 4th International Conference on Enterprise Information Systems*, 2. 580-586.

[21] Bacchus, Fahiem & Chen, Xinguang & van Beek, Peter & Walsh, Toby. (2002), “Binary vs. non-binary constraints. Artificial Intelligence”, 14. 1-37. 10.1016/S0004-3702(02)00210-2.

[22] Bacchus, Fahiem & van Beek, Peter. (1998), “On the Conversion between Non-Binary and Binary Constraint Satisfaction Problems”.

[23] Codd, E. (1970), “A Relational Model of Data for Large Shared Data Banks” *Commun*, ACM, 2013. 377-387. 10.1007/978-3-642-48354-7_4.

[24] Wikipedia, Löwenheim–Skolem theorem; https://en.wikipedia.org/wiki/L%C3%B6wenheim%E2%80%93Skolem_theorem. [Προσπελάστηκε 27/10/19]

[25] Wikipedia, Gödel's completeness theorem; https://en.wikipedia.org/wiki/G%C3%B6del%27s_completeness_theorem. [Προσπελάστηκε 27/10/19]

[26] Berners-Lee, Tim; Fischetti, Mark (1999). *Weaving the Web*. HarperSanFrancisco. chapter 12. ISBN 978-0-06-251587-2.