ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Efficient OWL Ontology Classification

Αποδοτική Αναγνώριση Υπονοούμενων Ιεραρχικών Σχέσεων σε OWL Οντολογίες

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΟΠΑΝΟΣ ΒΕΛΙΣΑΡΙΟΣ
ΜΑΡΑΝΤΗΣ ΧΡΗΣΤΟΣ

Επιβλέπων : Ιωάννης Βασιλείου
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2014

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Efficient OWL Ontology Classification

Αποδοτική Αναγνώριση Υπονοούμενων Ιεραρχικών Σχέσεων σε OWL Οντολογίες

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΟΠΑΝΟΣ ΒΕΛΙΣΑΡΙΟΣ
ΜΑΡΑΝΤΗΣ ΧΡΗΣΤΟΣ

Επιβλέπων : Ιωάννης Βασιλείου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 12η Μαρτίου 2014.

.....	.....	.....
Ιωάννης Βασιλείου	Νεκτάριος Κοζύρης	Γιώργος Στάμου
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επίκ. Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2014

…….

Κόπανος Βελισάριος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

…….

Μαράντης Π. Χρήστος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Στόχος της παρούσας διπλωματικής εργασίας είναι η υλοποίηση ενός συστήματος αποθήκευσης και διαχείρισης οντολογιών, ικανού για τον υπολογισμό όλων των σχέσεων ιεραρχίας που υπονοούνται μεταξύ των εννοιών (κλάσεων) μιας οντολογίας με εκφραστικότητα περιορισμένη στο τμήμα (profile) $\mathcal{EL}$ της γλώσσας αναπαράστασης OWL. Βάση της υλοποίησής μας είναι ο συνδυασμός της τεχνική της Δομικής Υπαγωγής, μιας τεχνικής συλλογιστικής ανάλυσης βασισμένης στην επαναληπτική εφαρμογή κανόνων, με ένα Σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS) για την ταξινόμηση του TBox οντολογιών, με την εξ ολοκλήρου παράλειψη της Μηχανής Συλλογιστικής Ανάλυσης.

Σε αντίθεση με το σύστημά μας τα περισσότερα συστήματα συλλογιστικής ανάλυσης χρησιμοποιούν Μηχανές Συλλογιστικές Ανάλυσης, οι οποίες λειτουργούν αποκλειστικά στην κύρια μνήμη. Επειδή οι υπάρχουσες πραγματικές οντολογίες μεγάλου όγκου που χρησιμοποιούνται αυτή τη στιγμή έχουν πολύ μεγάλα και πολύπλοκα TBoxes, τα προαναφερθέντα συστήματα δεν επιδεικνύουν καλή κλιμάκωση όσο ο όγκος του TBox αυξάνει, ενώ χρειάζονται και καταναλώνουν πολύ κύρια μνήμη. Το σύστημά μας αποφεύγει τα παραπάνω προβλήματα, επιτελώντας την ίδια λειτουργία, αλλά σε μικρότερους χρόνους και με καλύτερη εκμετάλλευση της κύριας μνήμης.

Η λειτουργία του μπορεί να περιγραφεί με την χρήση τριών διακριτών σταδίων. Αρχικά επιτελείται η φόρτωση και κανονικοποίηση της οντολογίας (μετατροπή του TBox σε επιτρεπόμενους τύπους αξιωμάτων), ακολουθεί η ταξινόμησή της (υπολογισμός μεταβατικού κλεισίματος της οντολογίας) και τέλος ο υπολογισμός του transitive reduction (παραγωγή ταξονομίας που περιέχει μόνο άμεσες σχέσεις υπαγωγής μεταξύ ονομασμένων κλάσεων της οντολογίας).

Η υλοποίηση του συστήματός μας έγινε στα πλαίσια επέκτασης του συστήματος DBRS που έχει αναπτυχθεί στο εργαστήριο, χρησιμοποιώντας την γλώσσα προγραμματισμού Java και στην παρούσα έκδοσή του, το Σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων PostgreSQL.

Λέξεις-Κλειδιά : Σημασιολογικός Ιστός, Μεταδεδομένα, Οντολογία, Περιγραφική Λογική, Δομική Υπαγωγή, OWL, $\mathcal{EL}$, $\mathcal{ELH}$, Κανονικοποίηση, Ταξινόμηση, Transitive Reduction, Σχεσιακό Σύστημα Βάσεων Δεδομένων, TBox, Ταξονομία, Κύρια Μνήμη, Επαγωγικοί Κανόνες, Αξιώματα

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

The purpose of this paper is the development of an ontology store and management system, capable of computing all the implied hierarchy relations between the concepts (classes) of an ontology with expressiveness limited to the $\mathcal{EL}$ profile of the knowledge representation language OWL. The foundation of our work is the combination of the technique known as Structural Subsumption, a reasoning technique based on the iterative application of rules, with a Relational Database Management System, for the classification of ontology TBoxes, while entirely omitting the use of a Reasoner.

Unlike our system most Reasoning Systems use Reasoners, which work exclusively in the main memory. Because today's large real-world ontologies have very large and complicated TBoxes, the aforementioned systems don't show good scaling (in relation with the size of the TBox) and they consume too much main memory. Our system avoids these problems and manages to complete the same task, but faster and with better usage of the main memory.

Its function can be described with the usage of three distinct stages. At the first stage we load and normalize the ontology (TBox conversion into permitted types of axioms), at the second stage we classify it (transitive closure of the ontology) and then, at the final third stage, we produce its transitive reduction (a taxonomy which contains only direct subsumption relationships between named classes of the ontology).

Our system was developed as an extension to the DBRS system which was developed in the lab, using the programming language Java and, in its current version, the Relational Database Management System PostgreSQL.

Keywords : Semantic Web, Metadata, Ontology, Description Logic, Structural Subsumption, OWL, $\mathcal{EL}$, $\mathcal{ELH}$, Normalization, Classification, Transitive Reduction, Relational Database Management Systems, TBox, Taxonomy, Main Memory, Completion Rules, Axioms

Η σελίδα αυτή είναι σκόπιμα λευκή.

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο Εργαστήριο Συστημάτων Βάσεων Γνώσεων και Δεδομένων (ΕΣΒΓΔ) του Εθνικού Μετσόβιου Πολυτεχνείου και μας έδωσε την ευκαιρία να ασχοληθούμε με μερικά πολύ ενδιαφέροντα θέματα που προκύπτουν από την αυξανόμενη χρήση των μεταδεδομένων και την ανάπτυξη του Σημασιολογικού Ιστού. Σε αυτό το σημείο θα θέλαμε να ευχαριστήσουμε θερμά όσους συνέβαλαν στην εκπόνηση της παρούσας διπλωματικής εργασίας και ιδιαιτέρως τους καθηγητές Ι.Βασιλείου και Τ.Σελλή που μας έδωσαν την ευκαιρία να ασχοληθούμε με τα συγκεκριμένα θέματα, καθώς και τον υποψήφιο διδάκτορα Ι.Λιαγούρη για την πολύτιμη βοήθειά του και την πολύ καλή συνεργασία που είχαμε κατά την εκπόνηση της διπλωματικής.

Πίνακας Περιεχομένων

1

Εισαγωγή

Στο κεφάλαιο αυτό γίνεται μία εισαγωγή στους όρους Σημασιολογικός Ιστός (ΣΙ) και μεταδεδομένα (metadata) και παρουσιάζονται κάποια σημαντικά θέματα που προκύπτουν από την χρήση τους. Οι δύο αυτές έννοιες κρίνονται απαραίτητες για την κατανόηση του περιβάλλοντος ανάπτυξης της διπλωματικής μας. Επιπλέον παρουσιάζεται το αντικείμενο της παρούσας διπλωματικής εργασίας και τα θέματα που μελετήθηκαν κατά την διάρκεια της εκπόνησής της, καθώς και η οργάνωση του κειμένου που ακολουθεί.

1.1 Σημασιολογικός Ιστός και Μεταδεδομένα

Ο όρος Σημασιολογικός Ιστός (Semantic Web) συναντάται στις αρχές της δεκαετίας του 1990 και αναφέρεται στο όραμα εμπλουτισμού του υπάρχοντος Συντακτικού Ιστού (Syntactic Web) με σημασιολογική πληροφορία (semantics), δηλαδή με πληροφορία που περιγράφει τα ίδια τα δεδομένα που υπάρχουν στο διαδίκτυο και οργανώνονται, παρουσιάζονται και μεταφέρονται μέσω αυτού. Ουσιαστικός στόχος του Σημασιολογικού Ιστού είναι η εξέλιξη του σημερινού Ιστού, που κατακλύζεται από αρχεία που δεν έχουν μια κοινή, συγκεκριμένη δομή, και η μετατροπή του σε έναν ιστό όπου οι χρήστες θα μπορούν να βρίσκουν, να μοιράζονται και συνδυάζουν πληροφορία πολύ πιο εύκολα.

Το βασικό πρόβλημα είναι ότι οι χρήστες του σημερινού Ιστού μπορούν να επιτελέσουν τις παραπάνω λειτουργίες, αντίθετα από τους υπολογιστές, οι οποίοι δεν μπορούν, τουλάχιστον χωρίς την ανθρώπινη παρέμβαση και καθοδήγηση. Η χρήση λοιπόν της σημασιολογικής πληροφορίας, γνωστής και ως μεταδεδομένα, αποσκοπεί στο να καταστήσει τους πόρους του διαδικτύου (web resources) προσπελάσιμους από αυτοματοποιημένες διαδικασίες, δηλαδή από αλγοριθμικές διαδικασίες (software) που δεν απαιτούν καθόλου ή τουλάχιστον περιορίζουν σε ένα βαθμό την «ανθρώπινη παρέμβαση». Ο Σημασιολογικός Ιστός είναι δηλαδή στην ουσία ένας εξελιγμένος ιστός, όπου οι υπολογιστές θα μπορούν να «καταλαβαίνουν» και να ανταποκρίνονται σε πολύπλοκα ανθρώπινα αιτήματα βασιζόμενοι στην σημασία τους, οπότε αυτόματα θα μπορούν να επιτελούν πολύ περισσότερες και πιο πολύπλοκες λειτουργίες.

Η χρήση όμως των μεταδεδομένων στον Παγκόσμιο Ιστό (World Wide Web – WWW) εγείρει διάφορα ζητήματα. Ένα από τα βασικότερα προβλήματα είναι η ανάγκη ολοκληρωμένης διαχείρισης μεταδεδομένων πολύ μεγάλου όγκου. Αν αναλογιστούμε ότι ο Παγκόσμιος Ιστός (World Wide Web – WWW) περιέχει δισεκατομμύρια σελίδες ή ότι η οντολογία (ontology) SNOMED CT περιέχει 370.000 ονόματα κλάσεων, τότε εύκολα συνειδητοποιούμε ότι ένα αυτοματοποιημένο σύστημα θα πρέπει πραγματικά να αντιμετωπίσει τεράστιες εισόδους δεδομένων. Επιπλέον ο όγκος και η πολυπλοκότητα των μεταδεδομένων αυξάνεται συνεχώς με αποτέλεσμα ακόμα και την αδυναμία συλλογιστικής ανάλυσής[1] τους και πολλές φορές ακόμα και την απλή φόρτωσή τους (loading) στην κύρια μνήμη (main memory). Το παραπάνω πρόβλημα, σε συνδυασμό με την δυσκολία που αντιμετωπίζεται στην κατάτμηση και την επιμέρους ανάλυση των σημασιολογικών σχέσεων, οδηγούν στην ανάγκη ανάπτυξης τεχνικών που θα εκμεταλλεύονται μηχανισμούς δευτερεύουσας μνήμης (second storage mechanisms), έτσι ώστε να ξεπεραστούν τα απαγορευτικά όρια που παρουσιάζονται.

Επιπλέον απαραίτητη είναι και η περιγραφή των σημασιολογικών σχημάτων μέσω κατάλληλων τυπικών φορμαλισμών, καθιστώντας έτσι δυνατή την ανάλυσή τους από υπολογιστές. Οι τυπικοί αυτοί φορμαλισμοί θα πρέπει να συνοδεύονται και από τις αντίστοιχες τυπικές γλώσσες, οι οποίες θα πρέπει να μπορούν να «διαβαστούν» από υπολογιστές και να διαθέτουν σύνταξη συμβατή με τα ισχύοντα πρότυπα του Παγκόσμιου Ιστού (π.χ. XML), έτσι ώστε να μπορούν εύκολα να ενσωματωθούν σε αυτά ή απλώς να συνδυαστούν μαζί τους.

[1] Οι αλγόριθμοι συλλογιστικής ανάλυσης λειτουργούν αποκλειστικά στην κύρια μνήμη και αποσκοπούν στον έλεγχο της συνέπειας (consistency) των σημασιολογικών σχέσεων και στην εξαγωγή νέας πληροφορίας, δηλαδή νέων σχέσεων και χαρακτηριστικών, που δε δηλώνεται ρητά στο αρχικό σύνολο γνώσης, αλλά προκύπτει από αυτό μέσω μιας λογικής επαγωγής (logical inference).

Ένα ακόμα ζήτημα είναι η ανάπτυξη αλγορίθμων συλλογιστικής ανάλυσης ικανών να επεξεργαστούν αποδοτικά μεταδεδομένα πολύ υψηλής εκφραστικότητας. Οι χρόνοι ανάλυσης, λαμβάνοντας υπόψη την χρονική καθυστέρηση της μεταφοράς των δεδομένων πάνω στα πρωτόκολλα του διαδικτύου, θα πρέπει να είναι αποδεκτοί από τον χρήστη. Επιπρόσθετα, η πλήρης ανάλυση (sound and complete analysis) υψηλής εκφραστικότητας μεταδεδομένων είναι απαραίτητη για να μπορούν οι αλγόριθμοι να διαχειριστούν όλα τα χαρακτηριστικά και τις περίπλοκες σχέσεις μεταξύ των δεδομένων του Παγκόσμιου Ιστού.

Επιπλέον προβλήματα όπως η ασάφεια, η ασυνέπεια και η παραπλάνηση στον ορισμό εννοιών, καθώς και η αβεβαιότητα σχετικά με τις τιμές τους, απαιτεί τον συνδυασμό πολλών διαφορετικών τεχνικών για την αντιμετώπισή τους, όπως η ασαφής λογική (fuzzy logic), η αναιρέσιμη συλλογιστική (defeasible reasoning), η παρασυνεπής λογική (paraconsistent logic) και η κρυπτογραφία (cryptography).

1.2 Αντικείμενο Διπλωματικής Εργασίας

Σκοπός της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός και η ανάπτυξη ενός συστήματος υπολογισμού *όλων* των σχέσεων ιεραρχίας που υπονοούνται μεταξύ των εννοιών (κλάσεων) μιας οντολογίας με εκφραστικότητα περιορισμένη στο τμήμα (profile) $\mathcal{EL}$ της γλώσσας αναπαράστασης OWL.

Ο λόγος για τον οποίο περιορίσαμε την ανάλυσή μας στο υποσύνολο $\mathcal{EL}$ της γλώσσας OWL είναι διττός. Πρώτον, στο συγκεκριμένο τμήμα εκφραστικότητας οι διαδικασίες συλλογιστικής ανάλυσης απαιτούν *πολυωνυμικό χρόνο* σε σχέση με τον αριθμό των αρχικών (explicit) αξιωμάτων της οντολογίας, εξασφαλίζοντας έτσι την ευκολία (tractability) που απαιτείται για τη διαχείριση δεδομένων μεγάλου όγκου. Δεύτερον, η πλειοψηφία των μεγάλων σε όγκο ιατρικών οντολογιών στις οποίες ο υπολογισμός των σχέσεων ιεραρχίας είναι ένα υπαρκτό πρόβλημα, μπορούν να εκφραστούν (εξολοκλήρου ή τουλάχιστον σε μεγάλο βαθμό) με το συγκεκριμένο υποσύνολο της γλώσσας OWL.

Κεντρική ιδέα της υλοποίησης αποτελεί ο συνδυασμός μιας τεχνικής συλλογιστικής ανάλυσης με ένα Σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS), καθιστώντας έτσι δυνατή την ταξινόμηση του Tbox οντολογιών χωρίς τη μεσολάβηση κάποιας Μηχανής Συλλογιστικής Ανάλυσης. Σκοπός μας είναι η ολοκληρωμένη διαχείριση σημασιολογικής πληροφορίας εκφραστικότητας $\mathcal{EL}$ (πληρότητα του συστήματος) και η όσο το δυνατόν καλύτερη απόδοσή του, τόσο από άποψη χρόνου, όσο και κυρίως από άποψη χρήσης της κύριας μνήμης.

Η τεχνική συλλογιστικής ανάλυσης που επιλέξαμε είναι η τεχνική της Δομικής Υπαγωγής, μια τεχνική εξαγωγής υπονοούμενων σχέσεων βασισμένη σε κανόνες (rules), η οποία έχει αποδειχθεί ότι αποτελεί έναν ορθό και πλήρη αλγόριθμο ταξινόμησης μιας οντολογίας για μια σειρά από τμήματα της Περιγραφικής Λογικής και διαιρείται σε δύο διακριτά στάδια. Αρχικά, απαιτείται μια προεπεξεργασία της οντολογίας μέσω της λεγόμενης διαδικασίας *κανονικοποίησης* (normalization) των αξιωμάτων της. Σε αυτό το στάδιο τα αξιώματα της οντολογίας «αποσυντίθενται» σε

ένα σύνολο κανονικοποιημένων μορφών έτσι ώστε να αποτελέσουν τις αρχικές σχέσεις βάσει των οποίων θα υπολογιστεί η πλήρης ιεραρχία των κλάσεων στο επόμενο στάδιο, γνωστό και ως *ταξινόμηση* (classification). Το τελευταίο περιλαμβάνει την εξαντλητική (fix-point) εφαρμογή ενός αριθμού επαγωγικών κανόνων, οι οποίοι εξάγουν τις υπονοούμενες σχέσεις βασιζόμενοι στη σημασιολογία των αξιωμάτων της οντολογίας.

Η υλοποίηση έγινε στα πλαίσια επέκτασης του συστήματος DBRS[2] που έχει αναπτυχθεί στο εργαστήριο και το οποίο συνδυάζει ένα σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων (PostgreSQL) με μια Μηχανή Συλλογιστικής Ανάλυσης (Pellet Reasoner) προκειμένου να διαχειρίζεται αποδοτικά οντολογίες μεγάλου όγκου εκφρασμένες σε OWL.

Μετά την προσθήκη του υποσυστήματος που δημιουργήσαμε, το DBRS μπορεί πλέον να υπολογίζει το πλήρες σύνολο των έγκυρων σχέσεων ιεραρχίας μεταξύ των κλάσεων μιας OWL $\mathcal{EL}$ οντολογίας με χρήση μόνο του DBMS, παρακάμπτοντας δηλαδή τη Μηχανή Συλλογιστικής Ανάλυσης, και χρησιμοποιώντας έναν περισσότερο αποδοτικό αλγόριθμο στον οποίο το μέγεθος της μνήμης που παραχωρείται μπορεί να παραμετροποιηθεί από τον χρήστη.

1.3 Θέματα με τα οποία ασχοληθήκαμε

Τα θέματα με τα οποία ασχοληθήκαμε στα πλαίσια της διπλωματικής εργασίας συνοψίζονται στα παρακάτω:

- Μελετήσαμε και αναλύσαμε την Περιγραφική Λογική (Description Logic – DL) , μια οικογένεια γλωσσών αναπαράστασης γνώσης, που αποσκοπεί τόσο στην μοντελοποίηση της, όσο και στην εύκολη ανάλυσή της.
- Μελετήσαμε και περιγράψαμε τρία είδη αλγορίθμων συλλογιστικής ανάλυσης στην DL, τους Tableau αλγόριθμους, τους αλγόριθμους βασισμένους σε Datalog και τους αλγόριθμους βασισμένους στην μέθοδο της Δομικής Υπαγωγής.
- Μελετήσαμε και παρουσιάσαμε την μέθοδο της Δομικής Υπαγωγής σε τέσσερις διαφορετικές γλώσσες αναπαράστασης ($\mathcal{FL}$, $\mathcal{ALEN}$, $\mathcal{EL}$ και $\mathcal{Horn\ SHIQ}$). Εμβαθύναμε ιδιαίτερα στην γλώσσα $\mathcal{EL}$ αναλύοντας τέσσερις διαφορετικές επεκτάσεις της.
- Μελετήσαμε και περιγράψαμε την οικογένεια γλωσσών αναπαράστασης γνώσης για οντολογίες OWL 2, καθώς και τρία διαφορετικά profiles αυτής.
- Μελετήσαμε και αναλύσαμε τις σημαντικότερες εργασίες που ασχολούνται με την ταξινόμηση των εννοιών (κλάσεων) μιας οντολογίας και των οποίων οι υλοποιήσεις βασίζονται κατά κύριο λόγο στη μέθοδο της Δομικής Υπαγωγής.
- Μελετήσαμε και παρουσιάσαμε τις λειτουργίες του συστήματος DBRS στο οποίο ενσωματώσαμε το πρόγραμμά μας ως υποσύστημα.

[2] http://www.dbnet.ece.ntua.gr/pubs/details.php?id=1523&clang=0

- Μελετήσαμε εις βάθος την λειτουργία του αντικειμενο-σχεσιακού Συστήματος Βάσεων Δεδομένων PostgreSQL, με το οποίο υλοποιήσαμε την βάση δεδομένων του συστήματός μας.
- Αναπτύξαμε αλγόριθμους για απόδοση μοναδικών αναγνωριστικών και διαχείρισης ισοδύναμων αντικειμένων.
- Αναπτύξαμε αλγόριθμους που υλοποιούν την τεχνική της Δομικής Υπαγωγής για οντολογίες εκφραστικότητας $\mathcal{ELH}$, δηλαδή αλγόριθμους που κανονικοποιούν και ταξινομούν την οντολογία που το σύστημά μας δέχεται ως είσοδο.
- Ελέγξαμε την απόδοση του συστήματός μας με την χρήση 6 οντολογιών διαφορετικού όγκου και χαρακτηριστικών.
- Προτείναμε, βάση της εμπειρίας μας, μελλοντικές επεκτάσεις και βελτιώσεις του συστήματός μας.

1.4 Οργάνωση κειμένου

Η παρούσα διπλωματική εργασία οργανώνεται στα παρακάτω κεφάλαια:

- Στο Κεφάλαιο 2 παρουσιάζεται το γενικό θεωρητικό υπόβαθρο και έννοιες που κρίνονται απαραίτητες για την κατανόηση της παρούσας διπλωματικής.
- Στο Κεφάλαιο 3 παρουσιάζονται εργασίες σχετικές με την παρούσα διπλωματική που βασίζονται κατά κύριο λόγο στην μέθοδο της Δομικής Υπαγωγής. Παρουσιάζονται και αναλύονται τα χαρακτηριστικά κάθε συστήματος με απώτερο στόχο την αιτιολόγηση και αποτίμηση της δικής μας προσέγγισης.
- Στο Κεφάλαιο 4 αναλύονται οι βασικές απαιτήσεις που ικανοποιεί το σύστημα DBRS, στο οποίο ενσωματώνεται το σύστημά μας. Το DBRS χωρίζεται σε διακριτά υποσυστήματα και για κάθε ένα από αυτά γίνεται μια σύντομη περιγραφή της λειτουργίας του, καθώς και του τρόπου με τον οποίο αλληλεπιδρά με τα υπόλοιπα.
- Στο Κεφάλαιο 5 αναφέρονται οι εφαρμογές που υλοποιούν το σύστημά μας. Περιγράφονται μόνο οι εφαρμογές του δικού μας συστήματος και όχι ολόκληρου του DBRS. Στο τέλος του κεφαλαίου παρατίθεται ένα αναλυτικό παράδειγμα εκτέλεσης του συστήματός μας για ένα υποτυπώδες TBox.
- Στο Κεφάλαιο 6 παρουσιάζεται η υλοποίηση του συστήματος που δημιουργήσαμε. Αρχικά παρατίθενται οι πιο βασικοί αλγόριθμοι που αναπτύξαμε και αναφέρονται οι πλατφόρμες και τα προγραμματιστικά εργαλεία που χρησιμοποιήσαμε. Στη συνέχεια δίνονται τα βήματα εγκατάστασης του συστήματος και τέλος αναλύονται οι κλάσεις του πηγιαίου κώδικα (source code) και τα ευρετήρια που χρησιμοποιήθηκαν.
- Στο Κεφάλαιο 7 παρουσιάζονται συγκεντρωτικά αποτελέσματα σχετικά με την απόδοση του συστήματός μας. Εξετάζεται συγκεκριμένα η επίδραση του

μεγέθους της οντολογίας και της αύξησης της χρήσης της κύριας μνήμης στην χρόνο εκτέλεσης του συστήματος.

- Στο Κεφάλαιο 8 γίνεται μία σύνοψη της εργασίας και παρουσιάζονται τα συμπεράσματα αυτής. Προτείνονται πιθανές μελλοντικές επεκτάσεις και βελτιώσεις του συστήματός μας.
- Στο Κεφάλαιο 9 παρατίθεται η βιβλιογραφία.

2

Θεωρητικό Υπόβαθρο

Στο κεφάλαιο αυτό δίνεται η περιγραφή εννοιών που αποτελούν το θεωρητικό υπόβαθρο της παρούσας εργασίας και που απαιτούνται για την κατανόησή της. Ήδη, στην εισαγωγή, έχουμε αναφερθεί στους όρους Σημασιολογικός Ιστός (ΣΙ) και μεταδεδομένα. Εδώ θα ασχοληθούμε με την έννοια της οντολογίας (ontology) και με μια οικογένεια λογικών φορμαλισμών που ονομάζεται Περιγραφική Λογική (Description Logic - DL). Θα εστιάσουμε το ενδιαφέρον μας σε μεθόδους συλλογιστικής ανάλυσης (reasoning) των μεταδεδομένων, δίνοντας έμφαση στον επικρατέστερο Tableau αλγόριθμο, καθώς και σε τεχνικές συνδυασμού τους με DBMSs. Η τελευταία ενότητα αφορά την τυπική γλώσσα OWL που χρησιμοποιείται για την περιγραφή οντολογιών. Ο εξοικειωμένος αναγνώστης μπορεί να παραβλέψει αυτό το κεφάλαιο.

2.1 Η έννοια της οντολογίας

Η προσθήκη των μεταδεδομένων στο υπάρχον δικτυακό οικοδόμημα προϋποθέτει την οργάνωσή τους με τέτοιο τρόπο που να καθιστά τη διαχείρισή τους αποτελεσματική. Κλειδί στην επίτευξη αυτού του εγχειρήματος αποτελεί η χρήση των οντολογιών. Σαφής ορισμός για την γενικότερη έννοια της οντολογίας δεν υπάρχει και, μάλιστα, ο όρος διαφοροποιείται αρκετά ανάλογα με τον επιστημονικό τομέα στον οποίο χρησιμοποιείται. Ωστόσο, ένας ορισμός ο οποίος μπορεί να συνοψίσει όλη την λειτουργικότητα της έννοιας της οντολογίας και έχει γίνει ευρέως αποδεκτός είναι του Tom Gruber[3].

Μια οντολογία είναι μια τυπική (formal), κατηγορηματική (explicit) προδιαγραφή μιας διαμοιρασμένης (shared) εννοιολογικής αναπαράστασης (conceptualization).

Στον παραπάνω ορισμό ο όρος εννοιολογική αναπαράσταση (conceptualization) αναφέρεται σε ένα αφηρημένο μοντέλο φαινομένων του κόσμου στο οποίο έχουν προσδιοριστεί οι έννοιες που σχετίζονται με αυτόν, ο όρος κατηγορηματική (explicit) σημαίνει ότι το είδος των εννοιών που χρησιμοποιούνται και οι περιορισμοί που αφορούν την χρήση αυτών των εννοιών είναι προσδιορισμένα με σαφήνεια, ο όρος αυστηρή (formal) αναφέρεται στο ότι η οντολογία πρέπει να είναι μηχανικά αναγνώσιμη και ο όρος διαμοιρασμένη (shared) αναφέρεται στο ότι η οντολογία πρέπει να αποτυπώνει γνώση κοινής αποδοχής στα πλαίσια μιας κοινότητας.

Αν προσπαθήσουμε να συμπυκνώσουμε τον τρόπο με τον οποίο αντιλαμβανόμαστε τις οντολογίες στα πλαίσια του ΣΙ, τότε καταλήγουμε στο εξής: Τυπική περιγραφή ενός συνόλου πληροφοριών και των συσχετίσεων μεταξύ τους. Κάθε οντολογία μπορεί να θεωρηθεί ως σύνθεση δύο βασικών μερών:

Ένα λεξιλόγιο (intensional knowledge) που αποτελείται από ονόματα εννοιών (concepts) και σχέσεων (relationships). Χρησιμοποιείται για να περιγράψει την πληροφορία ή αλλιώς τον «κόσμο» που μοντελοποιούμε και διαφοροποιείται ανάλογα με αυτόν[4]. Στην ορολογία της Περιγραφικής Λογικής, όπως θα δούμε παρακάτω, το τμήμα αυτό ονομάζεται TBox (Terminology Box).

Ένα σύνολο επιπλέον γνώσης (extensional knowledge) σχετικά με τον «κόσμο», που περιλαμβάνει δηλώσεις/ισχυρισμούς (assertions). Οι δηλώσεις αντιστοιχίζουν άτομα (individuals) σε έννοιες και ζεύγη ατόμων ή ζεύγη ατόμου-σταθεράς (literal) σε σχέσεις. Αναφέρονται συχνά στη βιβλιογραφία και ως

[3] Αμερικανός πρωτοπόρος ερευνητής στα πεδίο της αναπαράστασης γνώσης και της μηχανικής οντολογιών. http://en.wikipedia.org/wiki/Tom_Gruber

[4] Για παράδειγμα, η μοντελοποίηση μιας ανθρώπινης οικογένειας απαιτεί διαφορετικές έννοιες και σχέσεις απ' ότι η μοντελοποίηση της γνώσης μας σχετικά με τις υπάρχουσες ποικιλίες κρασιών. Υπό αυτή την έννοια, κάθε οντολογία έχει το δικό της λεξιλόγιο.

στιγμιότυπα (instances). Στην Περιγραφική Λογική, το τμήμα αυτό ονομάζεται ABox (Assertion Box) και σημειώνουμε ότι σε μια οντολογία μπορεί να απουσιάζει τελείως.

Τα άτομα της οντολογίας αποτελούν, στην ουσία, τα αντικείμενα που θέλουμε να διαχειριστούμε και συχνά είναι αναγνωριστικά (URIs) πόρων του διαδικτύου. Οι έννοιες του λεξιλογίου ισοδυναμούν με σύνολα ατόμων που μοιράζονται ένα τουλάχιστον κοινό χαρακτηριστικό και αναφέρονται συχνά ως κλάσεις (classes). Οι σχέσεις αντιστοιχούν σε σύνολα από ζεύγη ατόμων (ή ζεύγη ατόμου-σταθεράς) και ονομάζονται ιδιότητες (properties), επειδή ακριβώς προσδίδουν ιδιότητες στα άτομα συνδέοντάς τα μεταξύ τους (ή με κάποια σταθερά). Στην Περιγραφική Λογική, ονομάζονται ρόλοι (roles).

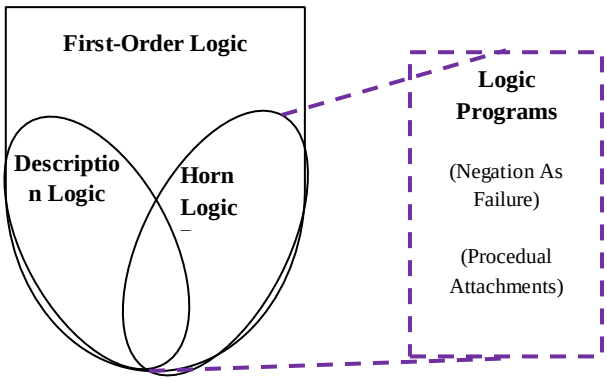
Οι οντολογίες χαρακτηρίζονται ως ο τυπικός (formal) προσδιορισμός των μεταδεδομένων ο οποίος εξασφαλίζει μια «κοινή αντίληψη» της περιοχής που μας ενδιαφέρει και παρέχει, δια της «φύσης» του, τη δυνατότητα συλλογιστικής ανάλυσης (reasoning), τόσο για την εξαγωγή νέων (υπονοούμενων) σχέσεων όσο και για τον έλεγχο της ισχύος ήδη υπαρχουσών (εύρεση αντιφάσεων). Η έννοια της τυπικότητας, όσον αφορά στον προσδιορισμό της μεταπληροφορίας, είναι καθοριστική για τη δυνατότητα των υπολογιστών να την αναλύσουν και άρα να διαχειριστούν αυτόματα και την ίδια την πληροφορία.

Η κατασκευή οντολογιών που μπορούν να «διαβαστούν» από υπολογιστές απαιτεί προφανώς την ύπαρξη τυπικών γλωσσών που θα χρησιμοποιούνται αποκλειστικά γι' αυτό το σκοπό και θα ικανοποιούν μια σειρά κριτηρίων. Ανάμεσα σ' αυτά είναι η ευκολία στη χρήση, η επαρκής εκφραστικότητα (expressivity), η δυνατότητα για συλλογιστική ανάλυση και φυσικά η συμβατότητα με τις ήδη υπάρχουσες τεχνολογίες του παγκόσμιου ιστού (π.χ. HTML, XML). Θα επανέλθουμε στα ζητήματα αυτά στις επόμενες ενότητες.

2.2 Περιγραφική Λογική (Description Logic – DL)

Η Περιγραφική Λογική (Description Logic - DL)[5] είναι μια από τις θεωρίες αναπαράστασης γνώσης. Η εκφραστικότητά της, το τι δηλαδή μπορεί να περιγράψει, περιορίζεται σε ένα τμήμα μιας άλλης θεωρίας που ονομάζεται Λογική Πρώτης Τάξης (First Order Logic - FOL). Μια τρίτη θεωρία αναπαράστασης γνώσης είναι ο Λογικός Προγραμματισμός (Logic Programs - LP) ο οποίος επίσης «μοιράζεται ένα κοινό κομμάτι» με τη Λογική Πρώτης Τάξης, γνωστό ως Λογική Horn (Horn Logic). Η έκταση της εκφραστικότητας των διαφόρων θεωριών αναπαράστασης γνώσης φαίνεται στο παρακάτω σχήμα.

[5] Συχνά ο όρος συναντάται και στον πληθυντικό (Description Logics) επειδή ακριβώς μπορεί να θεωρηθεί ως σύνολο λογικών φορμαλισμών (γλωσσών αναπαράστασης) που εντάσσονται στη θεωρία της Περιγραφικής Λογικής, αλλά έχουν διαφορετικές δυνατότητες ως προς την εκφραστικότητα της γνώσης που μπορούν να μοντελοποιήσουν.



Σχήμα 2.1. Γραφική αναπαράσταση εκφραστικότητας λογικών φορμαλισμών

Κάθε μια από τις προαναφερθείσες θεωρίες έχει και διαφορετική «φιλοσοφία» που αποτυπώνεται και στη σύνταξή της. Στη FOL η αναπαράσταση γίνεται με κατηγορήματα (predicates), μεταβλητές (variables) και σταθερές (constants), ενώ στον LP έχουμε κανόνες αιτίου-αιτιατού. Από την άλλη, η «φιλοσοφία» της DL είναι περισσότερο ανθρωποκεντρική (human-centered) και μοιάζει με το αντικειμενοστραφές μοντέλο προγραμματισμού (object oriented programming - OOP). Η μοντελοποίηση στην DL γίνεται με έννοιες, ρόλους, άτομα και σταθερές (literals), όπως αυτά παρουσιάστηκαν στην προηγούμενη ενότητα.

Η DL, όπως κάθε θεωρία αναπαράστασης γνώσης, διαθέτει τρία βασικά στοιχεία:

Λεξιλόγιο (Vocabulary)
Απαρτίζεται από ονόματα εννοιών (κλάσεων) και ρόλων. Οι έννοιες χρησιμοποιούνται για να ομαδοποιήσουν άτομα με κοινά χαρακτηριστικά και οι ρόλοι για να εκφράσουν συσχετίσεις μεταξύ τους ή μεταξύ αυτών και σταθερών. Όπως αναφέρθηκε και στην Ενότητα 2.1, το λεξιλόγιο διαφοροποιείται ανάλογα με τη γνώση που κάθε φορά μοντελοποιείται.

Συντακτικό (Syntax)
Βασίζεται στους κατασκευαστές (constructors) και σε σύμβολα που εκφράζουν σχέσεις (π.χ. ιεραρχικές) μεταξύ στοιχείων του λεξιλογίου ή/και ατόμων. Οι κατασκευαστές δηλώνουν μια λειτουργία (operation) μεταξύ δύο ή περισσοτέρων εννοιών ή ρόλων (π.χ. ένωση). Ο συνδυασμός τους με τα σύμβολα του συντακτικού και τα στοιχεία του λεξιλογίου οδηγεί στην κατασκευή των αξιωμάτων (axioms). Η γνώση που μοντελοποιούμε με DL είναι, στην ουσία, ένα σύνολο αξιωμάτων.

Σημασιολογία (Semantics)
Αφορά στην ερμηνεία των στοιχείων του λεξιλογίου, των κατασκευαστών και των συμβόλων του συντακτικού. Η σημασιολογία στην DL είναι συνολοθεωρητική: Μια

έννοια ερμηνεύεται ως σύνολο από αντικείμενα, ένας ρόλος ως σύνολο από ζεύγη αντικειμένων και ένα άτομο ως αντικείμενο.

Ο όρος μοντελοποιημένη γνώση, που χρησιμοποιήσαμε στα προηγούμενα, αναφέρεται στις θεωρίες αναπαράστασης γνώσης ως Βάση Γνώσης (Knowledge Base - KB) και αποτελεί θεμελιώδη έννοια την οποία θα χρησιμοποιούμε στο εξής.

Βάση Γνώσης (KB) ονομάζουμε ένα σύνολο γνώσης που περιγράφεται (μοντελοποιείται) με χρήση ενός τυπικού φορμαλισμού ή αλλιώς μιας γλώσσας αναπαράστασης (description language). Τα βασικά δομικά στοιχεία μιας KB είναι οι ατομικές έννοιες (atomic concepts), οι ατομικοί ρόλοι (atomic roles) και τα άτομα (individuals). Σύνθετες εκφράσεις (complex descriptions) μπορούν να κατασκευαστούν από αυτά χρησιμοποιώντας τους κατασκευαστές (constructors) που προσφέρει η εκάστοτε γλώσσα αναπαράστασης.

Η βασική γλώσσα αναπαράστασης, την οποία επεκτείνουν όλες οι άλλες, είναι η *AL* (Attributive Language) που περιγράφουμε στη συνέχεια. Συμβολίζουμε με *A, B* ατομικές έννοιες, με *r*, r_i (i=1,2,….) ατομικούς ρόλους, με *C*, C_i (i=1,2,….), *D* σύνθετες εκφράσεις και με *a*, *b*, *c* άτομα. Οι σύνθετες εκφράσεις στην *AL* σχηματίζονται βάσει του παρακάτω συντακτικού κανόνα:

$C, D \rightarrow A \mid \top \mid \bot \mid \neg A \mid C \sqcap D \mid \forall r.C \mid \exists r.\top$, όπου

$\top$ *Καθολική έννοια* (Universal concept). Περιλαμβάνει όλα τα άτομα της KB

$\bot$ *Κενή έννοια* (Bottom concept). Δεν περιλαμβάνει κανένα άτομο της KB.

$\neg A$ *Ατομική άρνηση* (Atomic negation). Η έννοια που περιλαμβάνει όλα εκείνα τα άτομα της KB που δεν ανήκουν στην ατομική έννοια A και μόνο αυτά.

$C \sqcap D$ *Τομή* (Intersection) δύο σύνθετων εννοιών. Η έννοια που περιλαμβάνει μόνο τα άτομα εκείνα της KB που ανήκουν και στις δύο αρχικές έννοιες *C* και *D*.

$\forall r.C$ *Περιορισμός τιμής* (Value restriction). Η έννοια που περιλαμβάνει όλα εκείνα τα άτομα της KB που συνδέονται μέσω του ατομικού ρόλου *r* μόνο με άτομα που ανήκουν στην σύνθετη έννοια *C*.

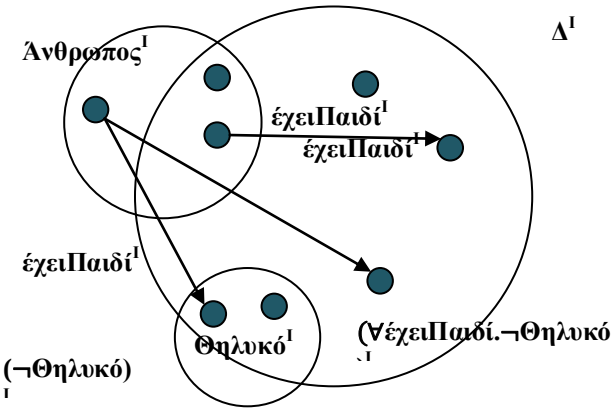
$\exists r.\top$ Περιορισμένη υπαρξιακή ποσοτικοποίηση (Limited existential quantification). Η έννοια που περιλαμβάνει όλα εκείνα τα άτομα της KB που συνδέονται μέσω του ατομικού ρόλου *r* με ένα τουλάχιστον άτομο της καθολικής έννοιας, δηλαδή της KB.

Οι επεξηγήσεις που δώσαμε για κάθε ένα από τα στοιχεία-έννοιες που συναντάμε στο συντακτικό κανόνα της *AL* αποτελούν στην ουσία μια ερμηνεία της σημασίας τους, δηλαδή μια απόδοση της σημασιολογίας τους με φυσική γλώσσα. Προκειμένου όμως να ορίσουμε αυστηρά αυτή τη σημασιολογία, εισάγουμε την έννοια των *διερμηνειών* (interpretations). Μια διερμηνεία *I* αποτελείται από μια δομή $(\Delta^I , \cdot^I)$, όπου Δ^I το πεδίο της διερμηνείας (domain of interpretation) και $\cdot^I$ μια συνάρτηση αντιστοίχησης[6]. Η συνάρτηση αυτή αντιστοιχίζει σε κάθε ατομική έννοια *A* ένα σύνολο $A^I \subseteq \Delta^I$ και σε κάθε ατομικό ρόλο *r* μια δυαδική σχέση $r^I \subseteq \Delta^I \times \Delta^I$. Επεκτείνεται στις σύνθετες εκφράσεις της *AL* βάσει των παρακάτω:

[6] Αυτός ο τυπικός ορισμός της σημασιολογίας είναι γνωστός στη βιβλιογραφία ως Tarski-style.

$\top^I = \Delta^I$
$\bot^I = \emptyset$
$(\neg A)^I = \Delta^I \setminus A^I$
$(C \sqcap D)^I = C^I \cap D^I$
$(\forall r.C)^I = \{a \in \Delta^I \mid \forall b.\, (a, b) \in r^I \rightarrow b \in C^I\}$
$(\exists r.\top)^I = \{a \in \Delta^I \mid \exists b.\, (a, b) \in r^I\}$

Όπως γίνεται αντιληπτό κάθε *AL* έννοια ερμηνεύεται ως ένα υποσύνολο του Δ^I. Για παράδειγμα η έννοια $\top$ ερμηνεύεται ως το σύνολο το οποίο περιέχει όλα τα αντικείμενα του χώρου ερμηνείας, ενώ η έννοια $\bot$ ερμηνεύεται ως το κενό σύνολο, το οποίο και δικαιολογεί την ονομασία που τους έχουμε προσδώσει. Εν συνεχεία η $C \sqcap D$ ερμηνεύεται ως το σύνολο το οποίο προκύπτει από την τομή των ερμηνειών των εννοιών *C* και *D*. Επιπρόσθετα, η ερμηνεία της έννοιας $\forall r.C$ περιέχει το σύνολο των αντικειμένων του Δ^I τα οποία αν συμμετέχουν στο ρόλο r^I με κάποιο άλλο αντικείμενο, τότε το αντικείμενο αυτό ανήκει στην ερμηνεία της έννοιας *C* δηλαδή στο σύνολο C^I. Είναι πολύ σημαντικό στο σημείο αυτό να τονίσουμε ότι ένα αντικείμενο ανήκει στην ερμηνεία της έννοιας $\forall r.C$ ακόμα και αν δε σχετίζεται μέσω της σχέσης r^I με κανένα άλλο αντικείμενο της οντολογίας (open world). Έτσι λοιπόν, στο σύνηθες παράδειγμα με τις έννοιες μιας οικογένειας έχουμε ότι και η ερμηνεία της έννοιας Άνθρωπος⊓∀έχειΠαιδί.¬Θηλυκό αποτελεί ένα σύνολο του Δ^I. Το σύνολο αυτό περιέχει τα αντικείμενα του Δ^I τα οποία ανήκουν ταυτόχρονα στην ερμηνεία της έννοιας Άνθρωπος, δηλαδή στο σύνολο Άνθρωπος^I, και αν συμμετέχουν στο ρόλο έχειΠαιδί^I με κάποιο άλλο αντικείμενο, τότε το αντικείμενο αυτό δεν ανήκει στο σύνολο αυτό που εμείς έχουμε αποδώσει ως ερμηνεία της έννοιας Θηλυκό. Στο παρακάτω σχήμα φαίνεται διαισθητικά η ερμηνεία της παραπάνω έννοιας.



Σχήμα 2.2 Γραφική αναπαράσταση της ερμηνείας της έννοιας Άνθρωπος⊓∀έχειΠαιδί.¬Θηλυκό

Κάθε ΚΒ που βασίζεται σε DL διαιρείται σε δύο κύρια μέρη: TBox και ABox[7]. Οι όροι μας είναι ήδη γνωστοί από την προηγούμενη ενότητα που αφορούσε στις οντολογίες. Εδώ θα τους περιγράψουμε ως σύνολα αξιωμάτων από την οπτική γωνία της Περιγραφικής Λογικής.

- *TBox* (Terminology Box): Είναι εκείνο το σύνολο των αξιωμάτων της ΚΒ που αφορούν στην ορολογία, δηλαδή στο λεξιλόγιο που χρησιμοποιείται. Σε περίπτωση εκφραστικότητας *AL* [8], το TBox περιλαμβάνει δύο ειδών αξιώματα:

  - *Αξιώματα υπαγωγής εννοιών* (Concept inclusion axioms)

    Όπως υποδηλώνει και το όνομά τους, εκφράζουν σχέσεις ιεραρχίας μεταξύ κλάσεων της ΚΒ και έχουν τη μορφή $C \sqsubseteq D$. Ένα αξίωμα υπαγωγής $C \sqsubseteq D$ δηλώνει ότι τα άτομα που ανήκουν στην έννοια-υπαγωγέας (subsumee) C ανήκουν, επίσης, και στην έννοια-υπαγόμενος (subsumer) D, αλλά όχι το αντίστροφο. Η αυστηρή σημασιολογία των αξιωμάτων αυτού του είδους ορίζεται ως εξής:

    Μια διερμηνεία I ικανοποιεί το αξίωμα $C \sqsubseteq D$ ανν $C^I \subseteq D^I$

  - *Αξιώματα Ισοδυναμίας εννοιών* (Concept equality axioms)[9]

    Εκφράζουν σχέσεις ισοδυναμίας μεταξύ κλάσεων της ΚΒ και έχουν τη μορφή $C \equiv D$. Ένα αξίωμα ισοδυναμίας $C \equiv D$ δηλώνει ότι τα άτομα που ανήκουν στην έννοια C ανήκουν επίσης και στην έννοια D και αντιστρόφως. Η αυστηρή σημασιολογία των αξιωμάτων αυτού του είδους ορίζεται ως εξής:

    Μια διερμηνεία I ικανοποιεί το αξίωμα $C \equiv D$ ανν $C^I \equiv D^I$

    Αυτομάτως καταλαβαίνουμε από τη σημασιολογία τους ότι τα αξιώματα ισοδυναμίας μπορούν να εκφραστούν και ως αξιώματα υπαγωγής. Στην ουσία, κάθε αξίωμα ισοδυναμίας εκφράζεται με δύο αξιώματα υπαγωγής ως εξής[10]:

    $$C \equiv D \leftrightarrow C \sqsubseteq D \text{ και } D \sqsubseteq C$$

---

[7] Η συγγένεια του όρου Βάση Γνώσης με αυτόν της Οντολογίας είναι κάτι παραπάνω από προφανής. Εκείνο που θέλουμε να σημειώσουμε εδώ είναι ότι, όταν (στα πλαίσια του ΣΙ) αναφερόμαστε σε οντολογίες, εννοούμε Βάσεις Γνώσεις (KBs) που έχουν «γραφτεί» με χρήση μιας τυπικής γλώσσας όπως οι RDF και OWL που θα δούμε στις Ενότητες 2.3 και 2.4.

[8] Το εύρος της γνώσης μιας ΚΒ δηλώνεται με το όνομα της γλώσσας αναπαράστασης που χρησιμοποιείται.

[9] Τα αξιώματα αυτά, όταν έχουν τη μορφή Όνομα έννοιας ≡ Σύνθετη έκφραση, αναφέρονται και ως αξιώματα ορισμού (definition axioms), θεωρώντας ότι ο ορισμός της έννοιας του πρώτου μέλους είναι η σύνθετη έκφραση του δεύτερου.

[10] Αυτό ισχύει μόνο αν στο TBox δεν έχουμε κύκλους (cycles), δηλαδή το όνομα μιας έννοιας δεν υπάρχει στον ορισμό της όπως π.χ. στο αξίωμα: Άνθρωπος ≡ Ζώο Π ∃Έχει_Πατέρα.Άνθρωπος.

- *ABox* (Assertion Box): Είναι εκείνο το σύνολο των αξιωμάτων που αφορούν στις δηλώσεις σχετικά με τα άτομα της KB. Σε περίπτωση εκφραστικότητας *AL*, το *ABox* περιλαμβάνει τα εξής είδη αξιωμάτων:

  - *Αξιώματα δηλώσεων εννοιών* (Concept assertion axioms)

    Εκφράζουν αντιστοιχήσεις ατόμων της KB σε έννοιες και έχουν τη μορφή $a : C$. Ένα αξίωμα $a : C$ δηλώνει ότι το άτομο *a* είναι τύπου *C*. Η αυστηρή σημασιολογία των αξιωμάτων αυτού του είδους ορίζεται ως εξής:

    Μια διερμηνεία *I* ικανοποιεί το αξίωμα $a : C$ ανν $a^I \in C^I$

  - *Αξιώματα δηλώσεων ρόλων* (Role assertion axioms)

    Εκφράζουν αντιστοιχήσεις ζευγών ατόμων σε ατομικούς ρόλους και έχουν τη μορφή $(a , b) : r$, όπου a υποκείμενο (subject) και b αντικείμενο (object) της δήλωσης. Ένα αξίωμα (a , b) : r δηλώνει ότι τα άτομα *a* και *b* συνδέονται μεταξύ τους μέσω της σχέσης *r*. Η αυστηρή σημασιολογία των αξιωμάτων αυτού του είδους ορίζεται ως εξής:

    Μια διερμηνεία *I* ικανοποιεί το αξίωμα $(a , b) : r$ ανν $(a^I , b^I) \in r^I$

Συνοψίζοντας, μια διερμηνεία *I* που ικανοποιεί όλα τα αξιώματα του TBox, με τον τρόπο που ορίσαμε παραπάνω, λέμε ότι ικανοποιεί το TBox της KB και αποτελεί ένα μοντέλο (model) του. Αν επίσης ικανοποιεί κατ' αντιστοιχία και το ABox, τότε ονομάζεται μοντέλο της Βάσης Γνώσης**.**

Όπως αναφέραμε στα προηγούμενα, η εκφραστικότητα μιας γλώσσας αναπαράστασης εξαρτάται από τους κατασκευαστές που αυτή παρέχει. Προσθέτοντας λοιπόν νέους κατασκευαστές στην βασική γλώσσα *AL*, παίρνουμε γλώσσες μεγαλύτερης εκφραστικότητας. Παραδείγματα τέτοιων κατασκευαστών είναι:

- *Ένωση εννοιών* (Concept union)

  Δηλώνεται με το γράμμα *U*. Η ένωση δύο σύνθετων εννοιών συμβολίζεται με $C \sqcup D$ και αποτελεί την έννοια εκείνη η οποία περιλαμβάνει όλα τα άτομα που ανήκουν είτε στην *C* είτε στην *D* είτε και στις δύο έννοιες μαζί. Η σημασιολογία της ορίζεται ως εξής:

  $$(C \sqcup D)^I \leftrightarrow C^I \cup D^I$$

- *Πλήρης υπαρξιακή ποσοτικοποίηση* (Full existential quantification)

  Δηλώνεται με το γράμμα *E*. Η πλήρης υπαρξιακή ποσοτικοποίηση συμβολίζεται με $\exists r.C$ και αποτελεί την έννοια εκείνη η οποία περιλαμβάνει όλα εκείνα τα άτομα που συνδέονται, μέσω του ατομικού ρόλου *r*, με ένα τουλάχιστον άτομο της σύνθετης έννοιας *C*. Η σημασιολογία της ορίζεται ως εξής:

  $$(\exists r.C)^I = \{a \in \Delta^I \mid \exists b. (a , b) \in r^I \wedge b \in C^I\}$$

- *Περιορισμός αριθμού* (Number restriction)[11]

Δηλώνεται με το γράμμα *N*. Συμβολίζεται με $\geq nr$ και με $\leq nr$. Ο περιορισμός $\geq nr$ αποτελεί την έννοια που περιλαμβάνει όλα εκείνα τα άτομα που μετέχουν το λιγότερο (at least) n φορές στις δηλώσεις του ατομικού ρόλου *r*. Κατ' αντιστοιχία, ο περιορισμός $\leq nr$ αποτελεί την έννοια που περιλαμβάνει όλα εκείνα τα άτομα που μετέχουν το πολύ (at most) n φορές στις δηλώσεις του ατομικού ρόλου *r*. Η σημασιολογία τους ορίζεται ως εξής:

$$(\geq nr)^I = \{a \in \Delta^I \mid \#\{b \mid (a, b) \in r^I\} \geq n\}$$
$$(\leq nr)^I = \{a \in \Delta^I \mid \#\{b \mid (a, b) \in r^I\} \leq n\}$$

όπου ο συμβολισμός **#{ … }** δηλώνει την πληθικότητα των μελών ενός συνόλου (cardinality of set). Ένας ρόλος για τον οποίο ισχύει μέγιστος περιορισμός αριθμού $\leq nr$ με n=1 ονομάζεται *λειτουργικός ρόλος* (functional role) και η ύπαρξή του συμβολίζεται με *F*.

- *Σύνθετη άρνηση* (Complex negation)[12]

Δηλώνεται με το γράμμα C και συμβολίζεται με $\neg C$. Η σύνθετη άρνηση $\neg C$ είναι η σύνθετη εκείνη έννοια που περιλαμβάνει όλα τα άτομα της KB που δεν ανήκουν στην έννοια C. Η σημασιολογία της ορίζεται ως εξής:

$$(\neg C)^I = \Delta^I \setminus C^I$$

Επεκτείνοντας την $\mathcal{AL}$ με οποιοδήποτε υποσύνολο των παραπάνω κατασκευαστών, παίρνουμε μια νέα γλώσσα αναπαράστασης (μεγαλύτερης εκφραστικότητας από την αρχική) που ανήκει στη λεγόμενη οικογένεια $\mathcal{AL}$-γλωσσών. Για παράδειγμα, επεκτείνοντας την $\mathcal{AL}$ με περιορισμό αριθμού, παίρνουμε τη γλώσσα $\mathcal{ALN}$. Εκείνο που αξίζει να σημειώσουμε εδώ είναι ότι ο συνδυασμός δύο κατασκευαστών μπορεί να προσφέρει εκφραστικότητα ίδια με αυτή ενός άλλου κατασκευαστή. Χαρακτηριστική περίπτωση αποτελεί η σύνθετη άρνηση που, όταν προστεθεί στη γλώσσα $\mathcal{AL}$, την εξοπλίζει με ένωση εννοιών, αφού όπως γνωρίζουμε από τη θεωρία συνόλων $\neg(C^I \cap D^I) = (\neg C)^I \cup (\neg D)^I$ επομένως $\neg(C \sqcap D) = (\neg C) \sqcup (\neg D)$. Έτσι, γράφουμε $\mathcal{ALC}$ και όχι $\mathcal{ALE}$.

Άλλοι κατασκευαστές που προσδίδουν επιπλέον εκφραστικότητα είναι οι:

- *Ποιοτικός περιορισμός αριθμού* (Qualified number restriction)[13]

Δηλώνεται με το γράμμα *Q* και συμβολίζεται με $\geq nr.C$ και $\leq nr.C$. Ο περιορισμός $\geq nr.C$ αποτελεί την έννοια που περιλαμβάνει όλα εκείνα τα άτομα που μετέχουν ως υποκείμενα το λιγότερο (at least) n φορές στις δηλώσεις του ατομικού ρόλου *r*, έχοντας ως αντικείμενο άτομο της έννοιας *C*. Κατ' αντιστοιχία, ο περιορισμός $\leq nr.C$ αποτελεί την έννοια που περιλαμβάνει όλα εκείνα τα άτομα που μετέχουν ως υποκείμενα το πολύ (at most) n φορές στις δηλώσεις του

[11] Αναφέρεται συχνά και ως περιορισμός μεγέθους συνόλου (cardinality restriction)
[12] Αναφέρεται συχνά και ως πλήρης άρνηση (full negation)
[13] Αναφέρεται συχνά και ως ποιοτικός περιορισμός μεγέθους συνόλου (qualified cardinality restriction)

ατομικού ρόλου *r*, έχοντας ως αντικείμενο άτομο της έννοιας *C*. Η σημασιολογία τους ορίζεται ως εξής:

$$(\geq nr.C)^I = \{a \in \Delta^I \mid \#\{b \mid (a, b) \in r^I \wedge b \in C^I\} \geq n\}$$
$$(\leq nr.C)^I = \{a \in \Delta^I \mid \#\{b \mid (a, b) \in r^I \wedge b \in C^I\} \leq n\}$$

- *Μεταβατικός ρόλος* (Transitive role)

Συμβολίζεται με *Tra(r)* και, συνήθως, στη βιβλιογραφία η ύπαρξή του υποδηλώνεται με το γράμμα *S* που αποτελεί σύντμηση των $\mathcal{AL}$, *C* και μεταβατικών ρόλων. Ο *Tra(r)* είναι εκείνος ο ρόλος του οποίου οι δηλώσεις έχουν μεταβατική ιδιότητα, δηλαδή μπορεί να σχηματίζουν μια «αλυσίδα» ή, όπως συχνά ονομάζεται, ένα μεταβατικό κλείσιμο (Transitive Closure - TC). Η σημασιολογία του μεταβατικού ρόλου ορίζεται ως εξής:

$$(Tra(r))^I = \{a,b,c \in \Delta^I \mid \{(a,b),(b,c)\} \subseteq r^I \rightarrow (a,c) \in r^I\}$$

- *Αντίστροφος ρόλος* (Inverse role)

Δηλώνεται με το γράμμα *I* και συμβολίζεται με r^-. Ο r^- είναι εκείνος ο ρόλος που περιλαμβάνει ακριβώς τα αντίστροφα ζεύγη ατόμων σε σχέση με τον ατομικό ρόλο *r*. Η σημασιολογία του r^- ορίζεται ως εξής:

$$(r^-)^I = \{a,b \in \Delta^I \mid (a,b) \in r^I \leftrightarrow (b,a) \in r^{-I}\}$$

Ένας ρόλος r που είναι αντίστροφος με τον εαυτό του ονομάζεται συμμετρικός ρόλος (symmetric role).

- *Κλάσεις από Συγκεκριμένα Άτομα* (Nominals)

Δηλώνονται με το γράμμα *O* και συμβολίζονται με {*a*,*b*,…}, όπου *a*,*b*,… άτομα της KB. Χρησιμεύουν στον ορισμό απαριθμημένων εννοιών (enumerated concepts), δηλαδή εννοιών που ορίζονται ως ένα σύνολο συγκεκριμένων ατόμων της KB. Η σημασιολογία τους ορίζεται ως εξής: $(\{a\})^I = \{a^I\}$.

Σε εκφραστικές γλώσσες αναπαράστασης συναντάμε και άλλους κατασκευαστές όπως ένωση ρόλων (role union), τομή ρόλων (role intersection), ανακλαστικό ρόλο (reflexive role), μη ανακλαστικό ρόλο (irreflexive role) κ.α. Δεν έχουμε σκοπό να αναλύσουμε εξαντλητικά όλες τις περιπτώσεις. Εκείνο, όμως, που αξίζει να σημειώσουμε εδώ είναι ένα ενδιαφέρον είδος αξιωμάτων που δεν αναφέραμε μέχρι τώρα καθότι δεν υπάρχει στην $\mathcal{AL}$. Ο λόγος για τα αξιώματα υπαγωγής ρόλων (role inclusion axioms), η ύπαρξη των οποίων σε μια KB δηλώνεται με το γράμμα *H*. Τα αξιώματα αυτά έχουν τη μορφή $r_1 \sqsubseteq r_2$, όπου r_1 και r_2 ατομικοί ρόλοι. Ένα αξίωμα $r_1 \sqsubseteq r_2$ δηλώνει ότι τα ζεύγη ατόμων του ρόλου r_1 είναι υποσύνολο των ζευγών του ρόλου r_2, αλλά όχι το αντίστροφο. Η σημασιολογία των αξιωμάτων αυτού του είδους ορίζεται ως εξής:

Μια διερμηνεία *I* ικανοποιεί το αξίωμα $r_1 \sqsubseteq r_2$ αν $r_1^I \subseteq r_2^I$

Κατ' αντιστοιχία, ορίζονται και αξιώματα ισοδυναμίας ρόλων (role equivalence) με τον τρόπο που περιγράψαμε και για τα αξιώματα ισοδυναμίας των εννοιών στην $\mathcal{AL}$.

Πολλά συστήματα διαχείρισης οντολογιών, όπως και το DBRS, χειρίζονται γλώσσες αναπαράστασης που υποστηρίζουν και *τύπους δεδομένων* (datatypes), όπως αυτοί ορίζονται αυστηρά με χρήση της XML. Η ύπαρξη τύπων δεδομένων και *τιμών δεδομένων* (literals) δηλώνεται με το γράμμα $^{(D)}$ και συνοδεύεται από ρόλους τύπων δεδομένων (datatype roles), οι οποίοι συνδέουν άτομα με σταθερές εκφράζοντας μια σχέση μεταξύ τω δύο.

Στον επόμενο πίνακα συνοψίζονται οι διάφοροι συμβολισμοί και οι αντίστοιχες εκφραστικότητες που συναντάμε στην Περιγραφική Λογική.

Συμβολισμός	**Εκφραστικότητα - Κατασκευαστές**
$\mathcal{AL}$	Ατομική άρνηση Τομή εννοιών Περιορισμός τιμής Περιορισμένη υπαρξιακή ποσοτικοποίηση
$\mathcal{FL}^-$	$\mathcal{AL}$ χωρίς ατομική άρνηση
$\mathcal{FL}_0$	$\mathcal{FL}^-$ χωρίς περιορ. υπαρξιακή ποσοτικοποίηση
U	Ένωση εννοιών
E	Πλήρης υπαρξιακή ποσοτικοποίηση
N	Περιορισμοί αριθμών
C	Σύνθετη άρνηση
Q	Ποιοτικοί περιορισμοί αριθμών
S	$\mathcal{AL}$, C και μεταβατικοί ρόλοι
I	Αντίστροφοι ρόλοι
O	Απαριθμητικά
H	Ιεραρχία ατομικών ρόλων
R	Ανακλαστικοί ρόλοι Μη ανακλαστικοί ρόλοι Ασύμβατοι ρόλοι Αξιώματα υπαγωγής σύνθετων ρόλων
F	Λειτουργικοί ρόλοι

(D)	Τιμές δεδομένων Τύποι δεδομένων Ρόλοι τύπου δεδομένων

Πίνακας 2.1 Σύμβολα εκφραστικότητας στην DL

2.3 Συλλογιστική ανάλυση στην Περιγραφική Λογική

Όπως καταστήσαμε σαφές και στα προηγούμενα, η Περιγραφική Λογική δεν αποσκοπεί μόνο στη μοντελοποίηση της γνώσης, αλλά και στην εύκολη ανάλυσή της. Η ανάλυση αυτή περιλαμβάνει δύο σκέλη. Το πρώτο αφορά στον έλεγχο της ισχύος των αξιωμάτων της KB, ενώ το δεύτερο στην εξαγωγή νέων αξιωμάτων που προκύπτουν από τα υπάρχοντα μέσω μιας αλγοριθμικής διαδικασίας.

Οι επαγωγικές διαδικασίες (inference procedures) σε μια KB εκφραστικότητας *AL* διακρίνονται σε:

- *TBox επαγωγικές διαδικασίες*

Πρόκειται για επαγωγικές διαδικασίες που σχετίζονται με την ορολογία της KB και διακρίνονται στις ακόλουθες:

  - *Ικανοποιησιμότητα* (Satisfiability)

    Μια έννοια είναι ικανοποιήσιμη αν υπάρχει ένα μη κενό σύνολο από άτομα της KB τα οποία ανήκουν σε αυτή. Πιο αυστηρά, μια έννοια *C* είναι ικανοποιήσιμη, δεδομένου ενός TBox *T*, αν υπάρχει ένα μοντέλο *I* του *T* τέτοιο ώστε $C^I \neq Ø$. Σε αυτή την περίπτωση, η διερμηνεία *I* ονομάζεται και μοντέλο της *C*. Η ικανοποιησιμότητα μιας έννοιας *C* μπορεί να αναχθεί σε εύρεση της σχέσης υπαγωγής (βλ. αμέσως μετά): $\perp \sqsubseteq C$.

  - *Υπαγωγή* (Subsumption)

    Μια έννοια *C* υπάγεται σε μια έννοια *D* αν όλα τα άτομα που ανήκουν στη *C* ανήκουν και στη *D*, αλλά όχι το αντίστροφο. Πιο αυστηρά, μια έννοια *C* υπάγεται μιας έννοιας *D* αν, δεδομένου ενός TBox *T*, υπάρχει ένα μοντέλο *I* για το οποίο ισχύει $C^I \subseteq D^I$. Αν η υπαγωγή των εννοιών *C* και *D* δηλώνεται ρητά στο *T* ή μπορεί να εξαχθεί από αυτό με συλλογιστική, τότε γράφουμε: $T \models C \sqsubseteq D$.

  - *Ισοδυναμία* (Equivalence)

    Μια έννοια *C* είναι ισοδύναμη με μια έννοια *D* αν όλα τα άτομα που ανήκουν στη *C* ανήκουν και στη *D* και αντίστροφα. Πιο αυστηρά, μια έννοια *C* είναι ισοδύναμη με μια έννοια *D* αν, δεδομένου ενός TBox *T*, υπάρχει ένα

μοντέλο I για το οποίο ισχύει $C^I \equiv D^I$. Βάσει των όσων έχουμε πει μέχρι τώρα, η εύρεση σχέσης ισοδυναμίας μπορεί να αναχθεί σε εύρεση σχέσεων υπαγωγής. Αν η ισοδυναμία των εννοιών C και D δηλώνεται ρητά στο T ή μπορεί να εξαχθεί από αυτό με συλλογιστική, τότε γράφουμε: $T \models C \equiv D$.

- *Ασυμβατότητα* (Disjointness)

Μια έννοια C είναι ασύμβατη με μια έννοια D αν δεν υπάρχουν άτομα που να ανήκουν και στις δύο μαζί. Πιο αυστηρά, μια έννοια C είναι ασύμβατη με μια έννοια D αν, δεδομένου ενός TBox T, υπάρχει ένα μοντέλο I για το οποίο ισχύει $C^I \cap D^I = \emptyset$. Επομένως, είναι φανερό ότι η εύρεση σχέσης ασυμβατότητας μπορεί να αναχθεί σε εύρεση της σχέσης υπαγωγής: $(C \sqcap D) \sqsubseteq \bot$.

Έως τώρα, είδαμε ότι οι επαγωγικές διαδικασίες του TBox μπορούν να αναχθούν όλες σε εύρεση σχέσεων υπαγωγής. Όμως, ένα ενδιαφέρον στοιχείο της Περιγραφικής Λογικής είναι ότι όλες μπορούν επίσης να αναχθούν σε εύρεση ικανοποιησιμότητας, δεδομένου ότι επιτρέπεται σύνθετη άρνηση (C). Υπό αυτές τις συνθήκες, δεδομένου ενός TBox T, ισχύουν:

Η έννοια C υπάγεται στην έννοια D αν $C \sqcap \neg D$ μη ικανοποιήσιμη
Οι έννοιες C και D είναι ισοδύναμες αν $C \sqcap \neg D$ και $\neg C \sqcap D$ μη ικανοποιήσιμες
Οι έννοιες C και D είναι ασύμβατες αν $C \sqcap D$ μη ικανοποιήσιμη

- *ABox επαγωγικές διαδικασίες*

Πρόκειται για επαγωγικές διαδικασίες που σχετίζονται με τις δηλώσεις της KB και διακρίνονται στις ακόλουθες:

- *Συνέπεια* (Consistency)

Ένα ABox είναι συνεπές αν, δεδομένου ενός TBox, δε διαθέτει αξιώματα που προκαλούν αντιφάσεις (contradictions). Πιο αυστηρά, ένα ABox A είναι συνεπές αν, δεδομένου ενός TBox T, υπάρχει ένα μοντέλο I που ικανοποιεί όλα τα αξιώματα του A.

- *Συνεπαγωγή* (Entailment)

Ένα ABox A συνεπάγεται ένα αξίωμα αν κάθε διερμηνεία I που ικανοποιεί το A, ικανοποιεί και το αξίωμα αυτό. Στην περίπτωση αυτή, για παράδειγμα σε μια δήλωση έννοιας $a : C$, γράφουμε $A \models a : C$.

Κατ' αντιστοιχία με τις αναγωγές στο TBox, η συνεπαγωγή σε ένα ABox μπορεί να αναχθεί σε συνέπεια του ABox ως εξής:

$A \models a : C$ αν $A \cup \{a: \neg C\}$ μη συνεπές

Από το τελευταίο, αντιλαμβανόμαστε ότι όλες οι επαγωγικές διαδικασίες του TBox μπορούν να αναχθούν εύκολα σε συνέπεια ABox. Για παράδειγμα, δυο σύνθετες έννοιες *C* και *D* είναι ασύμβατες αν, δεδομένου ενός TBox *T* και ενός υποτιθέμενου[14] ατόμου a, για κάθε διερμηνεία *I* του ABox *A,* το *A* U {a : C ⊓ D} είναι μη συνεπές. Την ιδέα αυτή, δηλαδή την αναγωγή όλων των επαγωγικών διαδικασιών σε εύρεση συνέπειας ABox, χρησιμοποιεί ο ευρύτατα διαδεδομένος Tableau αλγόριθμος που θα δούμε στην Παράγραφο.

2.3.1 Tableau αλγόριθμοι

Γενικά, οι Tableau αλγόριθμοι για την Περιγραφική Λογική (DL) προσπαθούν να αποδείξουν την ικανοποιησιμότητα μιας έννοιας C παράγοντας ένα μοντέλο αυτής, δηλαδή μια ερμηνεία I στην οποία το σύνολο που αντιστοιχεί στην έννοια C δεν είναι κενό ($C^I \neq Ø$). Tableau είναι ένας γράφος που αντιπροσωπεύει ένα μοντέλο όπου οι κόμβοι, που αναπαριστούνται με x, αντιστοιχούν σε άτομα και οι ακμές, που αναπαριστούνται από το όνομα του ρόλου (π.χ. r) αντιστοιχούν σε ατομικούς ρόλους. Κάθε κόμβος του tableau ανατίθεται σε ένα σύνολο εννοιών που αναπαριστούνται με L(x). Αυτές είναι οι έννοιες στις οποίες ανήκει ο κόμβος στο συγκεκριμένο μοντέλο, ή με άλλα λόγια, στη συγκεκριμένη *ολοκλήρωση* (completion), και ανανεώνονται σταδιακά κατά την κατασκευή του μοντέλου.

Μια ενδιαφέρουσα ιδιότητα των κατασκευασμένων σε DL μοντέλων είναι το ότι είναι δέντρα (και όχι γράφοι). Με άλλα λόγια, εάν μια DL έννοια είναι ικανοποιήσιμη, κάτι το οποίο ισχύει για κάθε έννοια μιας $\mathcal{ELH}$ οντολογίας, τότε *όλα* τα δυνατά της μοντέλα είναι της μορφής δέντρου. Αυτή είναι η καλούμενη ιδιότητα των μοντέλων-δέντρων(tree-model property) είναι ο κύριος λόγος που τα βασισμένα σε DL συστήματα συμπεριφέρονται καλά σε πρακτικές εφαρμογές.

Η δημιουργία των προαναφερθέντων μοντέλων σε σχήμα δέντρου επιτυγχάνεται μέσω μιας επαναληπτικής διαδικασίας όπου ένα σύνολο *συμπερασματικών κανόνων* (inference rules), οι οποίοι διαφέρουν ανάλογα με την εκφραστικότητα της οντολογίας, δηλαδή των κατασκευαστών που υποστηρίζονται, εφαρμόζονται ο ένας μετά τον άλλον σε όλα τα αξιώματα της οντολογίας, συμπεριλαμβάνοντας και εκείνα που παράχθηκαν σε όλα τα προηγούμενα βήματα του αλγορίθμου, μέχρις ότου (i) κανένας κανόνας δεν μπορεί να εφαρμοστεί πλέον ή (ii) βρεθεί μία αντίφαση στο μοντέλο.

Στην γενική περίπτωση, και αν παραλείψουμε τις διάφορες τεχνικές βελτιστοποίησης, η ιεραρχία της οντολογίας δημιουργείται αφού ελεγχθεί η σχέση υπαγωγής μεταξύ κάθε ζευγαριού από ονοματισμένες κλάσεις ορισμένες στο TBox, που σημαίνει ότι εάν έχουμε *n* ονόματα κλάσεων, τότε χρειαζόμαστε $O(n^2)$ συνολικούς ελέγχους. Στις βασισμένες σε Tableau διαδικασίες, κάθε έλεγχος υπαγωγής της μορφής $C_1 \sqsubseteq C_2$ μεταφράζεται σε δύο ελέγχους ικανοποιησιμότητας για τις περιγραφές εννοιών (i) $C_1 \sqcap \neg C_2$ και (ii) $C_2 \sqcap \neg C_1$ αντίστοιχα.

---

[14] Υποτιθέμενο με την έννοια ότι δεν υπάρχει εξαρχής στο ABox της KB, αλλά εισάγεται προσωρινά προκειμένου να αποδειχθεί το ζητούμενο.

Στο πλαίσιο αυτό, αν η πρώτη περιγραφή δεν είναι ικανοποιήσιμη για κάθε δυνατό μοντέλο της οντολογίας, τότε το $C_1 \sqsubseteq C_2$ ισχύει. Με το ίδιο σκεπτικό, αν η δεύτερη περιγραφή δεν είναι ικανοποιήσιμη σε κάθε δυνατό μοντέλο, τότε το $C_2 \sqsubseteq C_1$ ισχύει. Τέλος, στην περίπτωση που και οι δύο περιγραφές δεν είναι ικανοποιήσιμες σε κάθε δυνατό μοντέλο της οντολογίας, τότε οι δύο κλάσεις είναι ισοδύναμες, δηλαδή $C_1 \equiv C_2$.

Για να μπορέσουμε να δώσουμε έμφαση στον τρόπο με τον οποίο οι βασισμένοι σε Tableau αλγόριθμοι πραγματοποιούν την διαδικασία ταξινόμησης, πρέπει πρώτα να εισαγάγουμε τους ακόλουθους συμπερασματικούς κανόνες:

- $\sqcap$-κανόνας: αν η έννοια $C_1 \sqcap C_2 \in L(x)$ & $\{C_1 , C_2\} \notin L(x)$, τότε $L(x) = L(x) \cup \{C_1 , C_2\}$
- $\sqcup$- κανόνας: αν η έννοια $C_1 \sqcup C_2 \in L(x)$ & $\{C_1 , C_2\} \cap L(x)$, τότε $L(x) = L(x) \cup \{C_1\}$ ή $L(x) = L(x) \cup \{C_2\}$
- $\exists$- κανόνας: αν η έννοια $\exists r.C \in L(x)$ & $\nexists(y, r_1 \sqsubseteq r)$ τέτοια ώστε $r_1(x , y)$ & $C \in L(y)$, τότε δημιουργείστε καινούργιο κόμβο y με $L(y) = \{C\}$ και προσθέστε τον ισχυρισμό ρόλου (role assertion) r(x , y) στο μοντέλο
- $\forall$- κανόνας: αν η έννοια $\forall r.C \in L(x)$ & $\exists(y, r_1 \sqsubseteq r)$ τέτοια ώστε $r_1(x, y)$ & $\{C\} \notin L(y)$, τότε $L(y) = L(y) \cup \{C\}$

Προφανώς οι κανόνες αυτοί εφαρμόζονται στο σύνολο εννοιών που έχουν ανατεθεί σε κάθε άτομο (κόμβο) του Tableau.

Εκτός από το γεγονός ότι οι βασισμένοι σε Tableau αλγόριθμοι έχουν πολυπλοκότητα εκθετική στην χειρότερη περίπτωση (είναι διαδικασίες διάψευσης-refutation procedures και έτσι παράγουν μη ντετερμινιστικά συμπεράσματα ακόμα και για αρκετά απλά TBoxes), το κύριο μειονέκτημά τους είναι ότι *καταναλώνουν πολύ μνήμη*. Ακόμα και αν το μέγεθος του αρχικού TBox χωράει στην κύρια μνήμη, το μέγεθος του κατασκευασμένου μοντέλου μπορεί γίνει απαγορευτικά μεγάλο (κυρίως λόγω των "φρέσκων" ατόμων που εισάγονται από τον $\exists$-κανόνα) και να αποτραπεί έτσι η εφαρμογή των συμπερασματικών κανόνων από το να φτάσει σε ένα σταθερό σημείο(fix-point) ή από το να αναγνωρίσει μία αντίφαση.

Οι Tableau βασισμένες υλοποιήσεις έχουν σαν στόχο εκφραστικές OWL οντολογίες (ειδικά τα μη βατά κομμάτια της γλώσσας) όπου μπορούν να προκύψουν ασυνέπειες στο μοντέλο.

2.3.1.1 Παράδειγμα Εκτέλεσης

Έστω ότι έχουμε μια $\mathcal{ELH}$ οντολογία με το ακόλουθο TBox που αποτελείται από δύο αξιώματα κλάσης και ένα αξίωμα ιδιότητας.

$$\text{MuscularOrgan} \equiv \text{Organ} \sqcap \exists \text{isPartOf.MuscularSystem} \quad (1)$$

$$\text{Heart} \sqsubseteq \text{Organ} \sqcap \exists \text{belongsTo.}(\text{MuscularSystem} \sqcap \text{CirculatorySystem}) \quad (2)$$

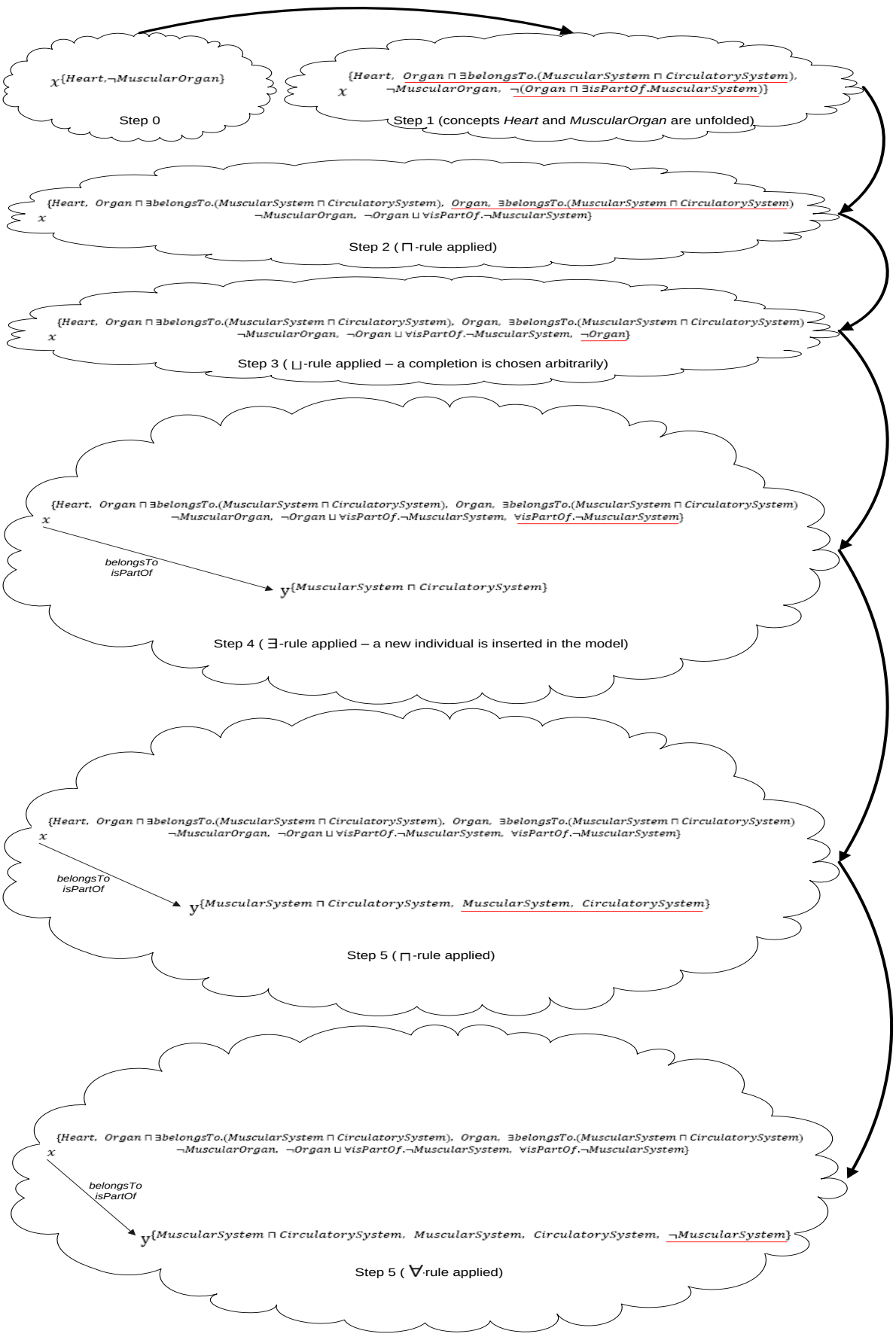
$$\text{belongsTo} \sqsubseteq \text{isPartOf} \quad (3)$$

Στη φυσική γλώσσα, τα παραπάνω τρία αξιώματα σημαίνουν τα εξής:

- Το πρώτο αξίωμα δείχνει ότι "το MuscularOrgan είναι ένα Organ που είναι επίσης μέρος του MuscularSystem".
- Το δεύτερο αξίωμα δείχνει ότι "η Heart είναι ένα Organ που ανήκει και στο MuscularSystem και στο CirculatorySystem".
- Το τρίτο αξίωμα δείχνει ότι "αν δύο άτομα συνδέονται μέσω της ιδιότητας belongsTo τότε συνδέονται επίσης με την ιδιότητα isPartOf".

Για την κατανόηση του αλγορίθμου tableau ας υποθέσουμε ότι πρέπει να ελέγξουμε την σχέση υπαγωγής Heart $\sqsubseteq$ MuscularOrgan. Τα βήματα του βασισμένου σε Tableau αλγορίθμου απεικονίζονται στο Σχήμα 2.3. Οι έννοιες που προστέθηκαν σε κάθε βήμα της ολοκλήρωσης είναι οι υπογραμμισμένες με κόκκινο χρώμα.

Στο Step 3, ο μη ντετερμινιστικός $\sqcup$-κανόνας διαλέγει μία δυνατή ολοκλήρωση όπου ο κόμβος x ανήκει τόσο στην κλάση Organ όσο και στο συμπλήρωμά της. Αυτό είναι προφανώς μία αντίφαση, οπότε ο αλγόριθμος γυρίζει πίσω(backtracks) και συνεχίζει με μία εναλλακτική ολοκλήρωση που δεν παράγει αντίφαση (τουλάχιστον όχι σε αυτό το βήμα). Τέλος, στο βήμα 5, ο $\forall$-κανόνας αναγκάζει τον κόμβο y να γίνει μέλος τόσο της κλάσης MuscularOrgan όσο και του συμπληρώματός της, οπότε αντιμετωπίζεται και πάλι αντίφαση και εφόσον ο αλγόριθμος δεν μπορεί να γυρίσει αυτή τη φορά πίσω σε ένα εναλλακτικό μοντέλο (δεν υπάρχουν άλλα δυνατά μοντέλα), η διαδικασία τερματίζει και η περιγραφή έννοιας $C_1 \sqcap \neg C_2$ αποδεικνύεται ότι είναι μη ικανοποιήσιμη σε κάθε μοντέλο του δοσμένου TBox. Ας σημειωθεί ότι αυτό δεν σημαίνει ότι το $C_1 \sqsubseteq C_2$ ισχύει. Δεν είμαστε σίγουροι ότι το C_1 υπάγεται στο C_2 εκτός αν ελέγξουμε επίσης για το $C_2 \sqsubseteq C_1$ κατά τον ίδιο τρόπο. Ο έλεγχος αυτός παραλείπεται εδώ λόγω έλλειψης χώρου.



Σχήμα 2.3 Ο αλγόριθμος Tableau για το παράδειγμα εκτέλεσης

2.3.2 Αλγόριθμοι Βασισμένοι σε Datalog

Μία εναλλακτική λύση στο πρόβλημα ταξινόμησης μιας οντολογίας είναι η μετάφραση των αξιωμάτων της οντολογίας σε ένα Datalog πρόγραμμα και η εκτέλεση της συλλογιστικής με χρήση μίας Datalog μηχανής. Ακολουθώντας αυτή την προσέγγιση, και σε περίπτωση που η οντολογία που θέλουμε να ταξινομήσουμε είναι μεγάλη σε μέγεθος, που σημαίνει ότι μπορεί να είναι δύσκολη η διαχείριση του TBox αποκλειστικά στην κύρια μνήμη, μπορεί να χρησιμοποιήσουμε επίσης τεχνικές προσανατολισμένες στον δίσκο (disk-oriented techniques) και βελτιστοποιήσεις στον χώρο των Επαγωγικών Συστημάτων Βάσεων Δεδομένων (Deductive Database Systems) έτσι ώστε να παρέχουμε καλύτερα κλιμακώσιμους (scalable) αλγορίθμους συλλογιστικής.

DL Syntax	**Datalog Syntax**	**Meaning**
Ατομικές Έννοιες (Atomic Concepts) A	P(X)	μια ατομική έννοια αντιστοιχεί σε ένα κατηγόρημα P. Άτομα τύπου A αντιστοιχούνται σε σταθερές που ικανοποιούν το κατηγόρημα P.
Ατομικοί Ρόλοι (Atomic Roles) r	R(X,Y)	ένας ατομικός ρόλος αντιστοιχεί σε ένα κατηγόρημα R δύο μεταβλητών. Άτομα που συνδέονται μέσω του ρόλου r αντιστοιχούνται σε σταθερές που ικανοποιούν το κατηγόρημα R.
Τομή Εννοιών (Concept Intersection) $C_1 \sqcap C_2$	$P_1(X) \Lambda P_2(X)$	άτομα που ανήκουν σε δύο ή παραπάνω έννοιες αντιστοιχούνται σε σταθερές που ικανοποιούν όλα τα αντίστοιχα κατηγορήματα
Υπαρξιακός Περιορισμός (Existential Restriction) ∃r.C	$R(X,Y) \Lambda P_c(Y)$	άτομα της έννοιας ∃r.C αντιστοιχούνται σε σταθερές που ικανοποιούν το κατηγόρημα R μαζί με σταθερές που ικανοποιούν επίσης το κατηγόρημα P_c. Το κατηγόρημα P_c αντιστοιχεί στην έννοια C, ενώ το κατηγόρημα R αντιστοιχεί στον ρόλο r.
Υπαγωγή Κλάσης (Class Subsumption) $C_1 \sqsubseteq C_2$	$P_2(X) \leftarrow P_1(X)$	τετριμμένο
Υπαγωγή Ρόλου (Role Subsumption) $r_1 \sqsubseteq r_2$	$R_2(X,Y) \leftarrow R_1(X,Y)$	τετριμμένο

Ισχυρισμός Κλάσης (Class Assertion) C(a)	P(a)	τετριμμένο
Ισχυρισμός Ρόλου (Role Assertion) r(a,b)	R(a,b)	τετριμμένο

Πίνακας 2.2 Αντιστοιχίες από DL σε Datalog για το κομμάτι της $\mathcal{ELH}$

Στον Πίνακα 2.2 απεικονίζονται, για τους σκοπούς του παραδείγματός μας, οι αντιστοιχίες μεταξύ μίας $\mathcal{ELH}$ οντολογίας (εκφρασμένης σε DL) και των αντίστοιχων datalog κανόνων.

Έχοντας μεταφράσει μία $\mathcal{ELH}$ οντολογία (όποτε δυνατόν – δείτε παρακάτω) σε ένα datalog πρόγραμμα, μπορούμε εύκολα να αποφασίσουμε για την σχέση υπαγωγής μεταξύ δύο εννοιών[15], έστω C_1 και C_2, με τον ακόλουθο τρόπο. Έστω P_1 και P_2 τα κατηγορήματα που αντιστοιχούν στις έννοιες C_1 και C_2 αντίστοιχα. Θέλουμε να ελέγξουμε αν η C_1 υπάγεται στην C_2. Τα βήματα της διαδικασίας είναι:

- Ορίζουμε μια σταθερά a η οποία ικανοποιεί το κατηγόρημα P_1, δηλαδή εισάγουμε ένα νέο γεγονός $P_1(a)$
- Αξιολογούμε το ερώτημα $P_2(a)$?
- Ορίζουμε μια σταθερά b που ικανοποιεί το κατηγόρημα P_2, δηλαδή εισάγουμε ένα νέο γεγονός $P_2(b)$
- Αξιολογούμε το ερώτημα $P_1(b)$?

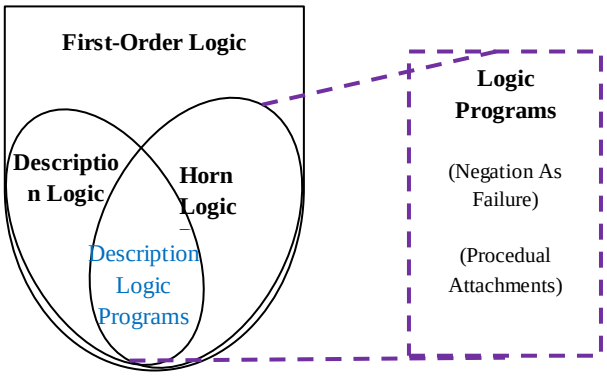
Προφανώς, σε περίπτωση που το ερώτημα $P_2(a)$? δώσει θετική απάντηση και το ερώτημα $P_1(b)$? δώσει αρνητική, τότε η έννοια C_1 υπάγεται στην έννοια C_2, δηλαδή το $C_1 \sqsubseteq C_2$ ισχύει.

Η Datalog βασίζεται στην Horn Λογική (Horn Logic) η οποία, ακριβώς όπως η Περιγραφική Λογική, είναι ένα αποφασήσιμο (decidable) υποσύνολο της Λογικής Πρώτης Τάξης(First-Order Logic). Όπως φαίνεται Σχήμα 2.4, τα αντίστοιχα κομμάτια έχουν ένα "κοινό μέρος", που ονομάζεται Προγράμματα Περιγραφικής Λογικής (Description Logic Programs), στα οποία μπορεί να οριστεί μια αντιστοιχία ένα-προς-ένα μεταξύ των δύο φορμαλισμών. Δυστυχώς, έξω από αυτήν την "περιοχή", η προαναφερθείσα αντιστοίχιση δεν είναι απλή (αν και εφικτή).

Πίσω στο παράδειγμα εκτέλεσης που ορίστηκε στην ενότητα 2.3.1.1, μπορούμε εύκολα να διαπιστώσουμε ότι το αξίωμα MuscularOrgan ⊒ Organ ⊓ ∃isPartOf.MuscularSystem μπορεί να μεταφραστεί στον κανόνα MuscularOrgan(X) ← Organ(X) Λ PartOf(X,Y) Λ MuscularSystem(Y), αλλά το αντίστροφο, δηλαδή το αξίωμα MuscularOrgan ⊑ Organ ⊓ ∃isPartOf.MuscularSystem, δεν μπορεί να μεταφραστεί σε έναν ασφαλή κανόνα horn (safe horn rule), και ως εκ τούτου δεν

[15] Η σχέση υπαγωγής μεταξύ δύο ρόλων μπορεί εύκολα να αναγνωριστεί μέσω μιας απλής επανάληψης (recursion).

μπορεί να εκφραστεί στη datalog. Με άλλα λόγια, το πρώτο TBox αξίωμα του παραδείγματος εκτέλεσης (και το δεύτερο επίσης) δεν μπορούν να μεταφραστούν σε έναν έγκυρο κανόνα για μια datalog μηχανή.



Σχήμα 2.4 Λογικοί Φορμαλισμοί

Μια πρόσφατη δουλειά εισάγει μια επέκταση της datalog, που ονομάζεται $Datalog_1\mp$, η οποία καλύπτει το $\mathcal{EL}$ κομμάτι της DL και έτσι μπορεί να χρησιμοποιηθεί για την ταξινόμηση μιας $\mathcal{ELH}$ οντολογίας. Παρόλα αυτά, η προσέγγιση που παρουσιάζεται δεν έχει υλοποιηθεί ως ένα συγκεκριμένο σύστημα.

Ένα πλεονέκτημα της βασισμένης σε datalog προσέγγισης σε σχέση με τους βασισμένους σε Tableau αλγορίθμους είναι ότι δεν περιέχουν μη ντετερμενιστικά συμπεράσματα και έτσι έχουν πολυωνυμική στην χειρότερη περίπτωση πολυπλοκότητα σε σχέση με το μέγεθος των δεδομένων.

Επειδή έχουν σχεδιαστεί για την επίτευξη πολυωνυμικής στην χειρότερη περίπτωση πολυπλοκότητας, οι βασισμένοι σε datalog αλγόριθμοι μπορούν να εφαρμοστούν μόνο σε ορισμένα περιορισμένα κομμάτια της Περιγραφικής Λογικής. Πέρα από αυτά τα κομμάτια, η δημιουργία προγραμμάτων datalog που να είναι ισοδύναμα με τις οντολογίες που δίνονται ως είσοδοι απαιτεί περίπλοκες λογικές μετατροπές. Για παράδειγμα, πιο εκφραστικές OWL οντολογίες, π.χ. αυτές που υποστηρίζουν την ένωση (U) εννοιών, μπορούν να αντιστοιχιστούν μόνο σε μία διαζευκτική επέκταση της datalog, δηλαδή σε προγράμματα datalog που περιέχουν κανόνες με μία η παραπάνω προτάσεις ένωσης (union clauses) στις κεφαλές τους.

2.3.3 Αλγόριθμοι Βασισμένοι στη Δομική Υπαγωγή

Μία εναλλακτική πρόταση για τη λύση του προβλήματος της ταξινόμησης είναι η εφαρμογή αλγορίθμων βασισμένων στην τεχνική της Δομικής Υπαγωγής (structural subsumption). Με απλά λόγια, οι αλγόριθμοι που ανήκουν σε αυτή την οικογένεια εκτελούν την ταξινόμηση της οντολογίας ταυτόχρονα εφαρμόζοντας επαναληπτικά, μέχρι να φτάσουν σε ένα συγκεκριμένο σημείο, ένα σύνολο από ορθούς και πλήρεις συμπερασματικούς κανόνες (inference rules) που καθορίζονται ανάλογα με τη δομή των αξιωμάτων που υπάρχουν στο Tbox. Γενικά, οι αλγόριθμοι δομικής υπαγωγής δεν εφαρμόζονται στο αρχικό TBox αλλά σε μία

κανονικοποιημένη εκδοχή του, η οποία προκύπτει μετά από μία διαδικασία κανονικοποίησης (normalization).

Στα επόμενα υποκεφάλαια θα παρουσιαστούν μία σειρά από αλγορίθμους ταξινόμησης βασισμένους στην τεχνική της Δομικής Υπαγωγής για διάφορες γλώσσες αναπαράστασης.

2.3.3.1 Δομική Υπαγωγή στην οικογένεια $\mathcal{FL}^-$

Σε πρώτη φάση θα μελετήσουμε έναν αλγόριθμο υπαγωγής βασισμένο στην τεχνική της Δομικής Υπαγωγής για την γλώσσα αναπαράστασης $\mathcal{FL}^-$. Δεδομένου ότι με N_C συμβολίζεται το σύνολο των ατομικών εννοιών, με N_R το σύνολο των ατομικών ρόλων η σύνταξη και C,D είναι σύνθετες έννοιες, η σύνταξη και η σημασιολογία της γλώσσας αναπαράστασης $\mathcal{FL}^-$ συνοψίζεται στον παρακάτω πίνακα.

Κατασκευαστής	**Σύνταξη**	**Σημασιολογία**
TOP – καθολική έννοια	$\top$	Δ^I
Ατομική έννοια $P \subseteq N_C$	P	$P^I \subseteq \Delta^I$
Τομή	$C \sqcap D$	$C^I \sqcap D^I$
Περιορισμένη υπαρξιακή ποσοτικοποίηση, $r \in N_R$	$\exists r.\top$	$\{ a \in \Delta^I \mid \exists\, b: (a,b) \in r^I \}$
Περιορισμός τιμής, $r \in N_R$	$\forall\, r.C$	$\{ a \in \Delta^I \mid \forall\, b: (a,b) \in r^I \Lambda y \in C^I \}$

Πίνακας 2.3 Σύνταξη και σημασιολογία της $\mathcal{FL}^-$

Ο αλγόριθμος λειτουργεί σε 2 φάσεις. Τα αρχικά αξιώματα μετατρέπονται στα λογικά ισοδύναμα αυτών σε κανονική μορφή ακολουθώντας τους παρακάτω κανόνες:

- Όλες οι φωλιασμένες τομές ενώνονται σε γραμμικό χρόνο

$$A \sqcap (B \sqcap C) \sim> A \sqcap B \sqcap C$$

- Όλες οι τομές περιορισμών τιμής ενοποιούνται

$$\forall\, r.C \sqcap \forall\, r.D \sim> \forall\, r.\, (C \sqcap D)$$

Μετά την κανονικοποίηση όλες οι έννοιες θα έχουν παρασταθεί ως αλυσίδα τομών. Έστω ότι $C = C_1 \sqcap \ldots \sqcap C_n$ και $D = D_1 \sqcap \ldots \sqcap D_m$ έννοιες όπως έχουν προκύψει από την κανονικοποίηση. Ο αλγόριθμος βασισμένος στην τεχνική της Δομικής Υπαγωγής SUBS?[*C,D*] (δηλαδή $D \sqsubseteq C$) επιστρέφει TRUE αν και μόνο αν για κάθε C_i ισχύουν:

- Αν το C_i είναι ατομική έννοια ή περιορισμένη υπαρξιακή ποσοτικοποίηση $\exists r.\top$, τότε υπάρχει ένα D_j τέτοιο ώστε $C_i = D_j$.
- Αν το C_i είναι μία σύνθετη έννοια της μορφής $\forall r.C'$, τότε υπάρχει ένα $\forall r.D'$ (με ίδια ατομική έννοια *r*) τέτοιο ώστε SUBS?[C', D'] να είναι TRUE

Έχει αποδειχθεί ότι ο παραπάνω αλγόριθμος είναι tractable (σε πολυωνυμικό χρόνο) και συγκεκριμένα μπορεί να ολοκληρωθεί σε $O(n^2)$ όπου n είναι το μήκος της μεγαλύτερης παραμέτρου.

Για την κατανόηση του παραπάνω αλγορίθμου θα χρησιμοποιήσουμε ένα παράδειγμα εκτέλεσης. Ας θεωρήσουμε την παρακάτω οντολογία $\mathcal{FL}^-$ η οποία στην φυσική γλώσσα θα μπορούσε να περιγραφεί ως εξής:

- "Πατέρας ενήλικων παιδιών είναι ο άνδρας που έχει τουλάχιστον ένα παιδί και όλα του τα παιδιά είναι ενήλικα"
- "Πατέρας ενήλικων αγοριών είναι ο άνδρας που έχει τουλάχιστον ένα παιδί, όλα του τα παιδιά είναι ενήλικα και όλα τα παιδιά του είναι αγόρια"

Με βάση τον φορμαλισμό της $\mathcal{FL}^-$ η παραπάνω οντολογία θα μπορούσε να περιγραφεί ως εξής:

$$Fatherwithadultchildren = Male \sqcap \exists hasChild.\top \sqcap \forall hasChild.Adult \quad (1)$$

$$Fatherwithadultboys = Male \sqcap \exists hasChild.\top \sqcap \forall hasChild.Adult \sqcap \forall hasChild.Male \quad (2)$$

Μετά την εφαρμογή των κανόνων της κανονικοποίησης το αξίωμα (2) έχει μετατραπεί στο ακόλουθο.

$$Fatherwithadultboys = Male \sqcap \exists hasChild.\top \sqcap \forall hasChild.(Adult \sqcap Male) \quad (3)$$

Ο αλγόριθμος της υπαγωγής που περιγράφεται παραπάνω επιστρέφει TRUE στο ερώτημα SUBS?[C (= *Fatherwithadultchildren*), D (= *Fatherwithadultboys*)] εφόσον:

$C_1 = D_1$ (=*Male*) και $C_2 = D_2$ (=$\sqcap\exists hasChild.\top$) και SUBS?[$C'_3$(=*Adult*),$D'_3$ (=*Adult*$\sqcap$*Male*)]=TRUE

Ένας παρόμοιος αλγόριθμος έχει ισχύ και για την πιο εκτεταμένη περιγραφική λογική $\mathcal{FL}$, η οποία περιέχει επιπλέον εκφραστικότητα εξαιτίας της ύπαρξης του κατασκευαστή της πλήρους υπαρξιακής ποσοτικοποίησης ($\exists r.C$ - $\{ x \in \Delta^I \mid \exists y: (x,y) \in r^I \wedge y \in C^I\}$) αλλά δεν είναι πλέον tractable. Έχει αποδειχθεί ότι το πρόβλημα της υπαγωγής στην $\mathcal{FL}$ είναι co-NP-complete.

2.3.3.2 Δομική Υπαγωγή στην οικογένεια $\mathcal{ALEN}$

Μέσα στο επόμενο κεφάλαιο θα περιγράψουμε την τεχνική της Δομικής Υπαγωγής στην γλώσσα αναπαράστασης $\mathcal{ALEN}$, και κατ' επέκταση και στις υπογλώσσες αυτής $\mathcal{ALN}$ και $\mathcal{ALE}$. Κύριο χαρακτηριστικό της γλώσσας αναπαράστασης $\mathcal{ALEN}$ είναι η ταυτόχρονη ύπαρξη τόσο των κατασκευαστών πλήρους υπαρξιακής ποσοτικοποίησης και περιορισμού αριθμού όσο και των κατασκευαστών της τομής, της ατομικής άρνησης και του περιορισμού τιμής.

Δεδομένου ότι με N_C συμβολίζεται το σύνολο των ατομικών εννοιών, με N_R το σύνολο των ατομικών ρόλων η σύνταξη και C,D είναι σύνθετες έννοιες, η σύνταξη και η σημασιολογία της γλώσσας αναπαράστασης $\mathcal{ALEN}$ (και των υπογλωσσών $\mathcal{ALN}$ και $\mathcal{ALE}$) συνοψίζεται στους παρακάτω πίνακες.

Κατασκευαστής	**Σύνταξη**	$\mathcal{ALEN}$	$\mathcal{ALE}$	$\mathcal{ALN}$
TOP – καθολική έννοια	$\top$	x	x	
BOTTOM – κενή έννοια	$\bot$	x	x	x
Ατομική έννοια $P \in N_C$	P	x	x	x
Ατομική άρνηση, $P \in N_C$	$\neg P$	x	x	x
Τομή	$C \sqcap D$	x	x	x
Πλήρης υπαρξιακή ποσοτικοποίηση, $r \in N_R$	$\exists r.C$	x	x	
Περιορισμός τιμής, $r \in N_R$	$\forall r.C$	x	x	
Περιορισμός αριθμού , $r \in N_R$, $n \in N$	$(\geq nr)$	x		x
Περιορισμός αριθμού , $r \in N_R$, $n \in N$	$(\leq nr)$	x		x

Πίνακας 2.4 Σύνταξη της περιγραφής εννοιών

Σύνταξη	**Σημασιολογία**
$\top$	Δ^I
$\bot$	$\emptyset$
P	$P^I \subseteq \Delta^I$
$\neg P$	$\Delta^I \setminus P^I$
$C \sqcap D$	$C^I \sqcap D^I$
$\exists r.C$	$\{ a \in \Delta^I \mid \exists\, b: (a,b) \in r^I \wedge b \in C^I \}$
$\forall\, r.C$	$\{ a \in \Delta^I \mid \forall\, b: (a,b) \in r^I \wedge b \in C^I \}$
$(\geq nr)$, $r \in N_R$, $n \in N$	$\{a \in \Delta^I \mid \#\{ b: (a,b) \in r^I \} \geq n \}$
$(\leq nr)$, $r \in N_R$, $n \in N$	$\{ a \in \Delta^I \mid \#\{ b: (a,b) \in r^I \} \leq n \}$

Πίνακας 2.5 Η σημασιολογία της περιγραφής εννοιών

Όπως έχει ήδη επισημανθεί, για την εφαρμογή ενός αλγορίθμου βασισμένου στην τεχνική της Δομικής Υπαγωγής, είναι συνήθως αναγκαίο η μετατροπή των αρχικών αξιωμάτων της οντολογίας σε μία ισοδύναμη κανονική μορφή. Και στην περίπτωση της γλώσσας αναπαράστασης $\mathcal{ALEN}$ χρειάζεται να προηγηθεί μία

διαδικασία κανονικοποίησης. Η περιγραφή μιας $\mathcal{ALEN}$ – σύνθετης έννοιας C είναι σε κανονική μορφή όταν όλες οι τομές $C_1 \sqcap \ldots \sqcap C_n$ που εμφανίζονται στο C πληρούν τις ακόλουθες προϋποθέσεις:

- Περιέχουν το πολύ ένα περιορισμό αριθμού της μορφής $(\geq nr)$
- Περιέχουν το πολύ ένα περιορισμό αριθμού της μορφής $(\leq nr)$
- Περιέχουν το πολύ ένα περιορισμό τιμής της μορφής $\forall\, r.C$

Για να επιτευχθεί η κανονική μορφή εφαρμόζονται επαναληπτικά και εξαντλητικά σε γραμμικό χρόνο οι παρακάτω κανόνες:

- $(\geq mr) \sqcap (\geq nr) \rightarrow (\geq nr)$ εάν $n \geq m$,
- $(\leq mr) \sqcap (\leq nr) \rightarrow (\leq nr)$ εάν $n \leq m$, και
- $\forall\, r.C \sqcap \forall\, r.C \rightarrow \forall\, r.(C \sqcap D)$

Σε αυτό το σημείο είναι χρήσιμη η εισαγωγή μιας σειράς από συντομεύσεις που θα διευκολύνουν την κωδικοποίηση του αλγορίθμου της Δομικής Υπαγωγής. Με την προϋπόθεση ότι η σύνθετη έννοια C είναι σε κανονική μορφή μπορούμε να αναπαραστήσουμε ως:

- prim(C) το σύνολο των ατομικών εννοιών που εμφανίζονται στο υψηλότερο επίπεδο του C
- $\min_r(C) := max\{k \mid C \sqsubseteq (\geq kr)\}$ (Σημείωση: είναι πάντα πεπερασμένο)[16]
- $\max_r(C) := min\{k \mid C \sqsubseteq (\leq kr)\}$ (Σημείωση: αν δεν υπάρχει τέτοιο k τότε $\max_r(C) := \infty$)
- $\text{val}_r(C) := C'$ εάν υπάρχει περιορισμός τιμής της μορφής $\forall\, r.C'$ στο υψηλότερο επίπεδο του C, αλλιώς $\text{val}_R(C) := \top$
- $\text{exr}_r(C) := \{ C' \mid$ εάν υπάρχουν $\exists r.C'$ στο υψηλότερο επίπεδο του $C\}$
- π ένα υπαρξιακό διαχώρισμα (existential partition) του συνόλου $\text{exr}_r(C) = \{C_1,\ldots,C_m\}$ τέτοιο ώστε να ισχύει α. Η πληθικότητα του π είναι $n=\min\{ \max_r(C), \mid \text{exr}_r(C)|\}$ και $\beta.(\sqcap_{C' \in S} C') \sqcap \text{val}_r(C) \not\equiv \bot$ για όλα τα $S \in \pi$.[17]
- $\Gamma_r(C)$ το σύνολο των υπαρξιακών διαχωρισμάτων του C
- $\text{exr}_r(C)^{\pi} := \{(\sqcap_{C' \in S} C') \mid S \in \pi\}$
- $\kappa_r(C) := \min_r(\forall r.\, \text{val}_r(C) \sqcap (\sqcap_{C' \in \text{exr}(C)} \exists\, r.C'))$
- $\text{exr}_r(C)^{*} := U_{\pi \in \Gamma r(C)} \text{exr}_r(C)^{\pi}$

[16] Το minr(C) θα πρέπει να τονιστεί ότι δεν προκύπτει αποκλειστικά από τους περιορισμούς αριθμών (≥mr) απλώς η ύπαρξη αυτού διευκολύνει τη διαδικασία και μειώνει την πολυπλοκότητα του υπολογισμού (στη χειρότερη περίπτωση υπολογίζονται σε χρόνο πολυωνυμικό ως προς το μέγεθος του C). Αν υπάρχει (≥mr) στο C για τον υπολογισμό του minr(C) θα εξεταστούν τα k για τα οποία ισχύει $m \leq k \leq \max\{m, |\text{exr}(C)|\}$ σε διαφορετική περίπτωση για τα k όπου $0 \quad k \leq |\text{exr}(C)|$. Π.χ. για το $C=\exists r.(A \sqcap B) \sqcap \exists r.(\neg A.\sqcap B) \sqcap (\geq 1r)$ έχουμε ότι minr(C)=2.

[17] Διαχώρισμα ενός συνόλου S είναι ένα σύνολο {S1,…, Sn} από μη κενά υποσύνολα Si ⊆ S του S τέτοια ώστε τα Si είναι ανα ζέυγη disjoint και S=S1 U … U Sn.

Με την βοήθεια των παραπάνω συμβολισμών πλέον είναι εφικτή η κωδικοποίηση ενός θεωρήματος βασισμένου πάνω στην τεχνική της Δομικής Υπαγωγής, ο οποίος δίνει την απάντηση για το πρόβλημα υπαγωγής 2 σύνθετων $\mathcal{ALEN}$ εννοιών.

Έστω C,D οι περιγραφές δύο $\mathcal{ALEN}$ – σύνθετων εννοιών με $\text{exr}_r(C) = \{C_1,\ldots,C_m\}$. Τότε $C \sqsubseteq D$ αν και μόνο αν $C \equiv \bot$ και $D \equiv \top$, ή ισχύουν τα ακόλουθα:

- $\text{prim}(D) \subseteq \text{prim}(C)$
- $\max_r(C) \leq \max_r(D)$
- $\min_r(C) \geq \min_r(D)$
- για κάθε $D' \in \text{exr}_r(D)$ ισχύει ότι
  - $\text{exr}_r(C) = \varnothing$, $\min_r(C) \geq 1$ και $\text{val}_r(C) \sqsubseteq D'$ ή
  - $\text{exr}_r(C) \neq \varnothing$ και για κάθε $\pi \in \Gamma_r(C)$ υπάρχει $C' \in \text{exr}_r(C)^{\pi}$ τέτοιο ώστε $C' \sqcap \text{val}_r(C) \sqsubseteq D'$
- εάν $\text{val}_r(D) \not\equiv \top$, τότε
  - $\max_r(C) = 0$ ή
  - $\kappa_r(C) < \max_r(C)$ και $\text{val}_r(C) \sqsubseteq \text{val}_r(D)$ ή
  - $0 < \kappa_r(C) = \max_r(C)$ $\text{val}_r(C) \sqcap C' \sqsubseteq \text{val}_r(D)$ για όλα τα $C' \in \text{exr}_r(C)^*$

Έχει αποδειχτεί ότι το παραπάνω θεώρημα δίνει ορθές και πλήρεις απαντήσεις στο πρόβλημα της υπαγωγής δύο σύνθετων $\mathcal{ALEN}$ εννοιών και θα μπορούσε να αποτελέσει την βάση για την δημιουργία ενός πρακτικού αλγορίθμου δομικής υπαγωγής.

2.3.3.3 Δομική Υπαγωγή στην οικογένεια $\mathcal{EL}$

Πρόσφατα, έχει αποδειχθεί ότι μία μικρή σχετικά περιγραφική λογική η $\mathcal{EL}$, η οποία υποστηρίζει την τομή και την πλήρη υπαρξιακή ποσοτικοποίηση, έχει αξιοσημείωτες αλγοριθμικές επιδόσεις στην ταξινόμηση εννοιών. Η περιγραφική λογική $\mathcal{EL}$ συνδέεται με το profile OWL 2 $\mathcal{EL}$ της OWL 2, στο οποίο τα σημαντικότερα προβλήματα συλλογιστικής αποφασίζονται σε πολυωνυμικό χρόνο (tractable). Επομένως το πρόβλημα της υπαγωγής στην $\mathcal{EL}$ παραμένει tractable τόσο με κυκλικό όσο και με ακυκλικό TBox όσο και με την παρουσία GCIs (general concept inclusion - γενικευμένες υπαγωγές εννοιών) σε αντίθεση με άλλες γλώσσες αναπαραστάσεις όπως η $\mathcal{FL}_0$.

Στη συνέχεια θα εξετάσουμε διαφόρους αλγορίθμους βασισμένους στην τεχνική της Δομικής Υπαγωγής για διάφορες επεκτάσεις της $\mathcal{EL}$ ξεκινώντας από τις απλές ($\mathcal{ELH}$) και καταλήγοντας σε πιο σύνθετες ($\mathcal{EL}$++).

2.3.3.3.1 Δομική Υπαγωγή στην $\mathcal{EL}\mathcal{H}$

Η σύνταξη και η σημασιολογία της $\mathcal{EL}\mathcal{H}$ δεδομένου της ύπαρξης ενός συνόλου ατομικών εννοιών N_C και ενός συνόλου ατομικών ρόλων N_R φαίνονται στους παρακάτω πίνακες τόσο για τις έννοιες όσο και για τα αξιώματα.

Έννοιες	**Σύνταξη**	**Σημασιολογία**
TOP – καθολική έννοια	$\top$	Δ^I
Ατομική έννοια $P \in N_C$	P	$P^I \sqsubseteq \Delta^I$
Τομή	$C \sqcap D$	$C^I \sqcap D^I$
Πλήρης υπαρξιακή ποσοτικοποίηση, $r \in N_R$	$\exists r.C$	$\{ a \in \Delta^I \mid \exists\, b: (a,b) \in r^I \wedge b \in C^I \}$

Πίνακας 2.6 Σύνταξη και σημασιολογία των εννοιών της $\mathcal{EL}\mathcal{H}$

Αξιώματα	**Σύνταξη**	**Σημασιολογία**
Υπαγωγή εννοιών (concept inclusion)	$C \sqsubseteq D$	$C^I \subseteq D^I$
Υπαγωγή ρόλων (role inclusion)	$r \sqsubseteq s$	$r^I \subseteq s^I$

Πίνακας 2.7 Σύνταξη και σημασιολογία των αξιωμάτων της $\mathcal{EL}\mathcal{H}$

Και στην περίπτωση των $\mathcal{EL}$ οντολογιών είναι απαραίτητη μια διαδικασία κανονικοποίησης καθώς ο αλγόριθμος βασισμένος στην Δομική Υπαγωγή εφαρμόζεται σε μία κανονικοποιημένη μορφή του αρχικού TBox, η διαδικασία κανονικοποίησης ***αναθέτει νέα ονόματα στις σύνθετες περιγραφές εννοιών*** που εφαρμόζονται στα αξιώματα, και έχει ως αποτέλεσμα ένα καινούργιο TBox T όπου κάθε αξίωμα έχει μία από τις ακόλουθες μορφές:

- $A \sqsubseteq B$
- $A_1 \sqcap A_2 \sqsubseteq B$
- $A \sqsubseteq \exists r.B$
- $\exists r.A \sqsubseteq B$
- $r \sqsubseteq s$

Ας σημειωθεί ότι το βήμα κανονικοποίησης απαιτεί πολυωνυμικό χρόνο σε σχέση με το μέγεθος του αρχικού TBox. Παίρνοντας υπόψη αυτούς τους πέντε τύπους κανονικοποιημένων αξιωμάτων, οι συμπερασματικοί κανόνες για το $\mathcal{EL}\mathcal{H}$ κομμάτι της Περιγραφικής Λογικής είναι οι ακόλουθοι:

- *Αν* A ∈ T *τότε* T = T U {A ⊑ A , A ⊑ ⊤}
- *Αν* A ⊑ B ∈ T *και* B ⊑ C ∈ T, *τότε* T = T U {A ⊑ C}
- *Αν* A ⊑ B ∈ T *και* A ⊑ C ∈ T *και* B ⊓ C ⊑ D ∈ T, *τότε* T = T U {A ⊑ D}
- *Αν* A ⊑ B ∈ T *και* B ⊑ ∃r.C ∈ T, *τότε* T = T U {A ⊑ ∃r.C}
- *Αν* A ⊑ ∃r.B ∈ T *και* r ⊑ s ∈ T, *τότε* T = T U {A ⊑ ∃s.B}
- *Αν* A ⊑ ∃r.B ∈ T *και* B ⊑ C ∈ T *και* ∃r.C ⊑ D ∈ T, *τότε* T = T U {A⊑ D}

Ο κανόνας 1 εφαρμόζεται μόνο μία φορά, στην αρχή κάθε διαδικασίας, ενώ οι εναπομείναντες κανόνες εφαρμόζονται επαναληπτικά σε όλα τα αξιώματα, συμπεριλαμβανομένων αυτών που παράχθηκαν σε προηγούμενα βήματα του αλγορίθμου TBox T, δηλαδή μέχρι το T να μην αλλάζει μετά από μία πλήρη εφαρμογή κάθε συμπερασματικού κανόνα σε όλα τα αξιώματα που περιλαμβάνει στο τρέχον βήμα (fix-point).

Για να γίνει πιο κατανοητό το πως δουλεύει ο αλγόριθμος, θα χρησιμοποιήσουμε το παράδειγμα εκτέλεσης που ορίστηκε στην ενότητα 2.3.1.1. Αρχικά, η διαδικασία κανονικοποίησης αλλάζει το αρχικό TBox στο ακόλουθο:

MuscularOrgan ⊑ Organ	*(από το 1^{ο} αξίωμα)*
MuscularOrgan ⊑ ∃isPartOf.MuscularSystem	*(από το 1^{ο} αξίωμα)*
Organ ⊓ ∃isPartOf.MuscularSystem ⊑ MuscularOrgan	*(από το 1^{ο} αξίωμα)*
Heart ⊑ Organ	*(από το 2^{ο} αξίωμα)*
Heart ⊑ ∃belongsTo.MCSystem	*(από το 2^{ο} αξίωμα)*
MuscularSystem ⊓ CirculatorySystem ⊑ MCSystem	*(από το 2^{ο} αξίωμα)*
MCSystem ⊑ MuscularSystem	*(από το 2^{ο} αξίωμα)*
MCSystem ⊑ CirculatorySystem	*(από το 2^{ο} αξίωμα)*
belongsTo ⊑ isPartOf	*(3^{ο} αξίωμα)*

Στη συνέχεια, οι συμπερασματικοί κανόνες εφαρμόζονται στο T το οποίο σταδιακά αυξάνεται με τα ακόλουθα αξιώματα (αποτελέσματα του πρώτου κανόνα είναι):

Heart ⊑ ∃belongsTo.MuscularSystem	*(παράχθηκε από τον 4^{ο} κανόνα)*
Heart ⊑ ∃isPartOf.MuscularSystem	*(παράχθηκε από τον 5^{ο} κανόνα)*
Heart ⊑ MuscularOrgan	*(παράχθηκε από τον 3^{ο} κανόνα)*

Τα παραπάνω αξιώματα προστίθενται στο T το ένα μετά το άλλο. Συγκεκριμένα, το πρώτο αξίωμα πάει σαν είσοδος στον 5^{ο} κανόνα για να παραχθεί το δεύτερο αξίωμα, ενώ το δεύτερο αξίωμα πάει σαν είσοδος στον 3^{ο} κανόνα για να παραχθεί το τρίτο. Ας σημειωθεί, ότι σε αυτή την περίπτωση, μόνο μία από τις τρεις σχέσεις υπαγωγής που παίρνουμε σαν έξοδο παράγεται κατά την δεύτερη φάση (Heart ⊑ MuscularOrgan). Τα άλλα δύο αξιώματα (MuscularOrgan ⊑ Organ, Heart ⊑ Organ) παράγονται κατά την κανονικοποίηση του TBox.

2.3.3.3.2 Δομική Υπαγωγή στην $\mathcal{EL}^+$

Η περιγραφική λογική $\mathcal{EL}^+$ αποτελεί μία επέκταση της $\mathcal{EL}$ με την προσθήκη αξιωμάτων της μορφής γενικευμένων υπαγωγών εννοιών (general concept inclusion – GCIs) και επιπλέον εκφραστικότητα στα αξιώματα υπαγωγής ρόλων. Η σύνταξη και η σημασιολογία τόσο των εννοιών όσο και των αξιωμάτων της $\mathcal{EL}^+$ φαίνεται στους παρακάτω πίνακες.

Έννοιες	**Σύνταξη**	**Σημασιολογία**
TOP – καθολική έννοια	$\top$	Δ^I
Ατομική έννοια $P \in N_C$	P	$P^I \subseteq \Delta^I$
Τομή	$C \sqcap D$	$C^I \sqcap D^I$
Πλήρης υπαρξιακή ποσοτικοποίηση, $r \in N_R$	$\exists r.C$	$\{ x \in \Delta^I \mid \exists y: (x,y) \in r^I \wedge y \in C^I \}$

Πίνακας 2.8 Σύνταξη και σημασιολογία των εννοιών της $\mathcal{EL}^+$

Γενικευμένη υπαγωγή εννοιών (general concept inclusion-GCI)[18]	$C \sqsubseteq D$	$C^I \subseteq D^I$
Υπαγωγή ρόλων (role inclusion - RI)[19]	$r_1 \circ \ldots \circ r_n \sqsubseteq s$	$r_1{}^I \circ \ldots \circ r_n{}^I \subseteq s^I$

Πίνακας 2.9 Σύνταξη και σημασιολογία των αξιωμάτων της $\mathcal{EL}^+$

Μία $\mathcal{EL}^+$ οντολογία αποτελείται από ένα πεπερασμένο σύνολο αξιωμάτων γενικευμένης υπαγωγής εννοιών και αξιωμάτων υπαγωγής ρόλων. Το πρόβλημα που επιζητά λύση είναι το πρόβλημα της ταξινόμησης (classification), δηλαδή ο υπολογισμός όλης της ιεραρχίας των σχέσεων υπαγωγής των ατομικών εννοιών που

[18] Η κύρια χρήση των GCIs στην $\mathcal{EL}^+$ είναι ο ορισμός ατομικών εννοιών με όρους περιγραφής σύνθετων εννοιών.Για αυτό το λόγο εισάγονται μία σειρά από όροι: (α)Ο ορισμός έννοιας (concept definition) $A \equiv C$ ως συνδυασμό των GCIs $A \sqsubseteq C$ και $C \sqsubseteq A$ με το A να αντιστοιχεί σε ατομική έννοια. Το C περιγράφει τις αναγκαίες και ικανές συνθήκες για να είναι κάτι στιγμιότυπο του A. (β)Ο ορισμός πρωταρχικής έννοιας (primitive concept definition) $A \sqsubseteq C$ με A ατομική έννοια, ο οποίος αναφέρεται στις αναγκαίες αλλά όχι ικανές συνθήκες για να είναι κάτι στιγμιότυπο του A.

[19] Υπαγωγές ρόλων της μορφής $r \sqsubseteq s$ αποκαλούνται ιεραρχίες ρόλων (role hierarchies). Η μεταβατικότητα (transitivity) των ρόλων μπορεί να εκφραστεί μέσω του $ror \sqsubseteq r$. Με τα RIs μπορούν επίσης να εκφραστούν κανόνες right - identity στη μορφή $ros \sqsubseteq r$.

περιλαμβάνονται σε μία οντολογία. Τη λύση σε αυτό το πρόβλημα το δίνει ο αλγόριθμος που ακολουθεί βασισμένος στην τεχνική της Δομικής Υπαγωγής.
Όπως και στις προηγούμενες περιπτώσεις το πρώτο στάδιο του αλγορίθμου περιλαμβάνει την μετατροπή του TBox σε κανονική μορφή μέσω μιας διαδικασίας κανονικοποίησης που γίνεται σε χρόνο γραμμικό ως προς το μέγεθος του TBox. Στο καινούργιο TBox T κάθε αξίωμα έχει μία από τις ακόλουθες μορφές:

- $A \sqsubseteq B$
- $A_1 \sqcap A_2 \sqsubseteq B$
- $A \sqsubseteq \exists r.B$
- $\exists r.A \sqsubseteq B$
- $r \sqsubseteq s$
- $r \circ s \sqsubseteq t$

Αντίστοιχα, οι συμπερασματικοί κανόνες για το $\mathcal{EL}^+$ κομμάτι της Περιγραφικής Λογικής οι οποίοι εκτελούνται επαναληπτικά (με εξαίρεση τον κανόνα 1) και εξαντλητικά έως ότου δεν προκύπτει καμία αλλαγή στο TBox T είναι οι ακόλουθοι:

- *Αν* $A \in T$, *τότε* $T = T \cup \{A \sqsubseteq A, A \sqsubseteq T\}$
- *Αν* $A \sqsubseteq B \in T$ *και* $B \sqsubseteq C \in T$, *τότε* $T = T \cup \{A \sqsubseteq C\}$
- *Αν* $A \sqsubseteq B \in T$ *και* $A \sqsubseteq C \in T$ *και* $B \sqcap C \sqsubseteq D \in T$, *τότε* $T = T \cup \{A \sqsubseteq D\}$
- *Αν* $A \sqsubseteq B \in T$ *και* $B \sqsubseteq \exists r.C \in T$, *τότε* $T = T \cup \{A \sqsubseteq \exists r.C\}$
- *Αν* $A \sqsubseteq \exists r.B \in T$ *και* $r \sqsubseteq s \in T$, *τότε* $T = T \cup \{A \sqsubseteq \exists s.B\}$
- *Αν* $A \sqsubseteq \exists r.B \in T$ *και* $B \sqsubseteq C \in T$ *και* $\exists r.C \sqsubseteq D \in T$, *τότε* $T = T \cup \{A \sqsubseteq D\}$
- *Αν* $A \sqsubseteq \exists r.B \in T$ *και* $B \sqsubseteq \exists s.C \in T$ *και* $r \circ s \sqsubseteq t \in T$, *τότε* $T = T \cup \{A \sqsubseteq \exists t.C\}$

2.3.3.3.3 Δομική Υπαγωγή στην $\mathcal{EL}^{++}$

Η περιγραφική λογική $\mathcal{EL}^{++}$ αποτελεί μια περισσότερη εκφραστική επέκταση της $\mathcal{EL}$ στην οποία έχουν προστεθεί οι κατασκευαστές της κενής έννοιας (BOTTOM – concept), των ονομαστικών εννοιών (nominals), μία περιορισμένη εκδοχή των συμπαγών πεδίων ορισμού (concrete domain) και μία αυστηρή μορφή σύνθεσης ρόλων. Η σύνταξη και η σημασιολογία τόσο των εννοιών όσο και των αξιωμάτων της $\mathcal{EL}^{++}$, δεδομένου της ύπαρξης δεδομένου της ύπαρξης ενός συνόλου ατομικών εννοιών N_C, ενός συνόλου ατομικών ρόλων N_R και πιθανόν ενός συνόλου ατομικών ονομάτων (individuals) N_I δίνεται στους ακόλουθους πίνακες.

Έννοιες	**Σύνταξη**	**Σημασιολογία**
TOP – καθολική έννοια	$\top$	Δ^I
BOTTOM – κενή έννοια[20]	$\bot$	$\emptyset$
Ονομαστικές έννοιες Nominals[21]	{a}	$\{a^I\}$
Ατομική έννοια $P \in N_C$	P	$P^I \subseteq \Delta^I$
Τομή	$C \sqcap D$	$C^I \sqcap D^I$
Πλήρης υπαρξιακή ποσοτικοποίηση, $r \in N_R$	$\exists r.C$	$\{ a \in \Delta^I \mid \exists\, b: (a,b) \in r^I \wedge b \in C^I\}$
Συμπαγές πεδίο ορισμού Concrete domain[22]	$p(f_{1,\ldots},f_k)$ για $f \in P^{Dj}$	$\{ a \in \Delta^I \mid \exists\, b_1,\ldots b_k \in \Delta^{Dj}$: $f_i^I(a) = b_i$ για $1 \le i \le k$ $\wedge(b_1,\ldots,b_k) \in p^{Dj}\}$

Πίνακας 2.10 Σύνταξη και σημασιολογία των εννοιών της $\mathcal{EL}^{++}$

Αξιώματα	**Σύνταξη**	**Σημασιολογία**
Γενικευμένη υπαγωγή εννοιών (general concept inclusion- GCI)	$C \sqsubseteq D$	$C^I \subseteq D^I$
Υπαγωγή ρόλων (role inclusion - RI)[23]	$r_1 \circ \ldots \circ r_n \sqsubseteq s$	$r_1{}^I \circ \ldots \circ r_n{}^I \subseteq s^I$

Πίνακας 2.11 Σύνταξη και σημασιολογία των αξιωμάτων της $\mathcal{EL}^{++}$

Πριν από την παρουσίαση του αλγορίθμου (τόσο του αρχικού σταδίου της κανονικοποίησης όσο και του σταδίου της εφαρμογής των κανόνων συμπερασμού) είναι σκόπιμο να γίνουν μια σειρά από παρατηρήσεις και ορισμοί συμβολισμών που είναι αναγκαίοι για την κατανόηση όσων ακολουθούν.

- Αντί για την χρήση TBox γίνεται χρήση ενός παρόμοιου λεξιλογίου που ονομάζεται CBox (constraint box – σύνολο περιορισμών) και αποτελείται από ένα πεπερασμένο σύνολο αξιωμάτων γενικευμένης υπαγωγής εννοιών (GCIs) και υπαγωγής ρόλων (RIs).

[20] Το bottom concept σε συνδυασμό με GSIs μπορεί να χρησιμοποιηθεί για να εκφράσει disjointness σύνθετων περιγραφών εννοιών π.χ. $C \sqcap D \sqsubseteq \bot$ δηλώνει ότι τα C και D είναι ξένα μεταξύ τους (disjoint).

[21] Η αξίωση μοναδικού ονόματος για τα individual names μπορεί να δηλωθεί γράφοντας $\{a\} \sqcap \{b\} \sqsubseteq \bot$ για όλα τα individuals a,b.

[22] Ο constructor αυτός παρέχει μία διεπαφή (interface), που επιτρέπει την αναφορά σε strings και ακεραίους.

[23] Ισχύει ότι και στη παραπομπή 17

- Δεδομένου ενός CBox C, χρησιμοποιούμε τον συμβολισμό BC_C για να δηλώσουμε το μικρότερο δυνατό σύνολο περιγραφής εννοιών που περιέχει τα εξής:
  - την έννοια top $\top$.
  - όλες τις ατομικές έννοιες που χρησιμοποιούνται στο C.
  - όλες τις περιγραφές εννοιών της μορφής {a} ή $p(f_1,...,f_k)$ που εμφανίζονται στο C.
- Ορίζουμε R_C το σύνολο όλων των ρόλων που εμφανίζονται στο C.
- Ορίζεται ο συμβολισμός $\sim>_R \subseteq BC_C \times BC_C$ ως εξής:
  Ισχύει $C \sim>_R D$ ανν υπάρχουν $C_1,\ldots,C_k \in BC_C$ τέτοια ώστε
  - $C_1 = C$ ή $C_1 = \{b\}$ για κάποιο ατομικό όνομα (individual) b
  - Υπάρχει $r_j(C_j,C_{j+1})$ για κάποιο $r_j \in R_C$
  - $C_k = D$
- Ορίζεται ο συμβολισμός $con_j(\Gamma)$ για ένα σύνολο Γ από περιγραφές εννοιών $\mathcal{EL}^{++}(D_1,...D_n)$ και $1 \leq j \leq n$ ως εξής:

$$con_j(\Gamma) := \bigwedge_{p(f1,\ldots\ldots,fk) \in \Gamma \text{ with } p \in PDj} p(f_1,\ldots\ldots,f_k)$$

Το πρώτο στάδιο του αλγορίθμου περιλαμβάνει την μετατροπή του CBox σε κανονική μορφή μέσω μιας διαδικασίας κανονικοποίησης που γίνεται σε χρόνο γραμμικό ως προς το μέγεθος του CBox. Ένα CBox C είναι σε κανονική μορφή αν ισχύουν τα παρακάτω:

- Όλα τα GCIs έχουν μία από τις παρακάτω μορφές, όπου $C_1,C_2 \in BC_C$ και $D \in BC_C \cup \{\bot\}$
  - $C_1 \sqsubseteq D$
  - $C_1 \sqcap C_2 \sqsubseteq D$
  - $C_1 \sqsubseteq \exists r.C_2$
  - $\exists r.C_1 \sqsubseteq D$
- Όλα τα RIs είναι της μορφής $r \sqsubseteq s$ ή $r_1 \circ r_2 \sqsubseteq s$

Για να μετατραπεί το CBox σε κανονική μορφή εφαρμόζονται επαναληπτικά και εξαντλητικά σε γραμμικό χρόνο οι παρακάτω κανόνες κανονικοποίησης όπου C', $D' \notin BC_C$ και u, A νέες ατομικές έννοιες και ρόλοι που εισάγονται στο BC_C.

1. $r_1 \circ \ldots\ldots \circ r_k \sqsubseteq s \rightarrow \{ r_1 \circ \ldots\ldots \circ r_{k-1} \sqsubseteq u, u \circ r_k \sqsubseteq s\}$
2. $C \sqcap D' \sqsubseteq E \rightarrow \{ D' \sqsubseteq A , C \sqcap A \sqsubseteq E \}$
3. $\exists r.C' \sqsubseteq D \rightarrow \{ C' \sqsubseteq A , \exists r.A \sqsubseteq D \}$
4. $\bot \sqsubseteq D \rightarrow \emptyset$
5. $C' \sqsubseteq D' \rightarrow \{ C' \sqsubseteq A , A \sqsubseteq D' \}$
6. $B \sqsubseteq \exists r.C' \rightarrow \{ B \sqsubseteq \exists r.A , A \sqsubseteq C' \}$
7. $B \sqsubseteq C \sqcap D \rightarrow \{ B \sqsubseteq C , B \sqsubseteq D \}$

Έχει αποδειχθεί ότι για να μην χαθεί η γραμμικότητα της κανονικοποίησης είναι απαραίτητο οι παραπάνω κανόνες να εφαρμόζονται τμηματικά εξαντλητικά,

δηλαδή αρχικά εξαντλητική εφαρμογή των κανόνων 1 έως 4 και στη συνέχεια 5 έως 7.

Σε δεύτερη φάση, οι συμπερασματικοί κανόνες για το $\mathcal{EL}^{++}$ κομμάτι της Περιγραφικής Λογικής οι οποίοι επαναλαμβάνονται επαναληπτικά (με εξαίρεση τον κανόνα 1) και εξαντλητικά έως ότου δεν προκύπτει καμία αλλαγή στο CBox C είναι οι ακόλουθοι:

- *Αν* $A \sqsubseteq C$, *τότε* $C = C \cup \{A \sqsubseteq A, A \sqsubseteq \top\}$
- *Αν* $A \sqsubseteq B \in C$ *και* $B \sqsubseteq D \in C$, *τότε* $C = C \cup \{A \sqsubseteq D\}$
- *Αν* $A \sqsubseteq B \in C$ *και* $A \sqsubseteq D \in C$ *και* $B \sqcap D \sqsubseteq E \in C$, *τότε* $C = C \cup \{A \sqsubseteq E\}$
- *Αν* $A \sqsubseteq B \in C$ *και* $B \sqsubseteq \exists r.D \in C$, *τότε* $C = C \cup \{A \sqsubseteq \exists r.D\}$
- *Αν* $A \sqsubseteq \exists r.B \in C$ *και* $B \sqsubseteq D \in C$ *και* $\exists r.D \sqsubseteq E \in C$, *τότε* $C = C \cup \{A \sqsubseteq E\}$
- *Αν* $A \sqsubseteq \exists r.B \in C$ *και* $B \sqsubseteq \bot \in C$, *τότε* $C = C \cup \{A \sqsubseteq \bot\}$
- Αν $\{a\} \sqsubseteq A \in C$ και $\{a\} \sqsubseteq B \in C$ και $A \rightsquigarrow_R B$, τότε $C = C \cup \{A \sqsubseteq B\}$
- Αν το $con_j(S(A))$[24]) είναι μη ικανοποιήσιμο[25] D_j, τότε $C = C \cup \{A \sqsubseteq \bot\}$
- Αν από τον συμπερασμό[26] του $con_j(S(A))$ στο D_j συνεπάγεται το $p(f_{1,\dots},f_k) \in BC_C$, τότε $C = C \cup \{A \sqsubseteq p(f_{1,\dots},f_k)\}$
- Αν $A \sqsubseteq p(f_{1,\dots},f_k) \in C$ και $A \sqsubseteq p'(f'_{1,\dots},f'_k) \in C$ και $p \in P^{Dj}$ και $p' \in P^{Dl}$ και $j \neq l$ και $f_s = f'_t$ για κάποιο s,t, *τότε* $C = C \cup \{A \sqsubseteq \bot\}$
- *Αν* $A \sqsubseteq \exists r.B \in C$ *και* $r \sqsubseteq s \in C$, *τότε* $C = C \cup \{A \sqsubseteq \exists s.B\}$
- *Αν* $A \sqsubseteq \exists r.B \in T$ *και* $B \sqsubseteq \exists s.C \in T$ *και* $ro\ s \sqsubseteq t \in T$, *τότε* $T = T \cup \{A \sqsubseteq \exists s.C\}$

Ο αλγόριθμος ταξινόμησης που έχει βασιστεί στους παραπάνω συμπερασματικούς κανόνες είναι πολυωνυμικός καθώς έχει αποδειχθεί ότι σε ένα κανονικοποιημένο CBox οι κανόνες μπορούν να εφαρμοστούν μόνο ένα πολυωνυμικό αριθμό φορών και η εφαρμογή κάθε κανόνα γίνεται σε πολυωνυμικό χρόνο.

2.3.3.4 Δομική Υπαγωγή στην Horn-ShiQ

Το τελευταίο υποτμήμα της περιγραφικής λογικής για το οποίο θα μελετήσουμε την τεχνική της Δομικής Υπαγωγής είναι ένα tractable υποτμήμα της περιγραφικής λογικής $\mathcal{SHIQ}$ γνωστό ως H σύνταξη και η σημασιολογία τόσο των εννοιών όσο και των αξιωμάτων της $\mathcal{SHIQ}$, δεδομένου της ύπαρξης δεδομένου της ύπαρξης ενός συνόλου ατομικών εννοιών N_C, ενός συνόλου ατομικών ρόλων N_R και πιθανόν ενός συνόλου ατομικών ονομάτων (individuals) N_I δίνεται στους ακόλουθους πίνακες.

[24] Το S(A) είναι το σύνολο με όλες τις υπερκλάσεις του A. π.χ. Αν $A \sqsubseteq B$, $A \sqsubseteq D$ τότε $S(A) = \{B, C\}$

Έννοιες	**Σύνταξη**	**Σημασιολογία**
TOP – καθολική έννοια	$\top$	Δ^I
BOTTOM – κενή έννοια	$\bot$	$\emptyset$
Ατομική έννοια $P \in N_C$	P	$P^I \subseteq \Delta^I$
Σύνθετη άρνηση, $C \in N_C$	$\neg C$	$\Delta^I \setminus C^I$
Τομή εννοιών	$C \sqcap D$	$C^I \cap D^I$
Ένωση εννοιών	$C \sqcup D$	$C^I \cup D^I$
Πλήρης υπαρξιακή ποσοτικοποίηση, $r \in N_R$	$\exists r.C$	$\{ a \in \Delta^I \mid \exists\, b: (a,b) \in r^I \wedge b \in C^I \}$
Περιορισμός τιμής, $r \in N_R$	$\forall\, r.C$	$\{ a \in \Delta^I \mid \forall\, b: (a,b) \in r^I \wedge b \in C^I \}$
Ποιοτικός περιορισμός αριθμού – ελάχιστη πληθικότητα, $r \in N_R$, $n \in N$	$(\geq nr.C)$	$\{ a \in \Delta^I \mid \#\{ b: (a,b) \in r^I \wedge b \in C^I \} \geq n \}$
Ποιοτικός περιορισμός αριθμού – μέγιστη πληθικότητα, $r \in N_R$, $n \in N$	$(\leq nr.C)$	$\{ a \in \Delta^I \mid \#\{ b: (a,b) \in r^I \wedge b \in C^I \} \leq n \}$

Πίνακας 2.12 Σύνταξη και σημασιολογία των εννοιών της $\mathcal{SHIQ}$

Αξιώματα	**Σύνταξη**	**Σημασιολογία**
Γενικευμένη υπαγωγή εννοιών (general concept inclusion- GCI)	$C \sqsubseteq D$	$C^I \subseteq D^I$
Υπαγωγή ρόλων (role inclusion - RI)	$r \sqsubseteq s$	$r^I \subseteq s^I$
Μεταβατικός ρόλος (role transitivity)	Tra(r)	$r^I \circ r^I \subseteq r^I$
Αντίστροφος ρόλος (inverse role)	r^-	$\{a,b \in \Delta^I \mid (a,b) \in r^I \leftrightarrow (b,a) \in r^{-I}\}$

Πίνακας 2.13 Σύνταξη και σημασιολογία των αξιωμάτων της $\mathcal{SHIQ}$

Για να ορίσουμε τον *Horn* υποτμήμα της $\mathcal{SHIQ}$ θα πρέπει να αναφερθούμε συνοπτικά στην απόδοση πολικότητας μιας $\mathcal{SHIQ}$ έννοιας. Οι θετικές και αρνητικές πολικότητες των $\mathcal{SHIQ}$ εννοιών ορίζονται ως εξής :

- Το C εμφανίζεται θετικά στο C
- Το C εμφανίζεται θετικά (αρνητικά) στο $\neg C_-$, $C_+ \sqcap D_+$, $C_+ \sqcup D_+$, $\exists r.C_+$, $\forall r.C_+$, $\geq nr.C_+$, $\leq mr.C_-$ και $C_- \sqsubseteq D_+$, εάν το C εμφανίζεται θετικά (αρνητικά) στο C_+ ή στο D_+, ή εμφανίζεται αρνητικά (θετικά) στο C_-.

Μία έννοια C εμφανίζεται θετικά (αρνητικά) σε μια οντολογία O, εάν το C εμφανίζεται θετικά (αρνητικά) σε ένα αξίωμα της O. Είναι επομένως δυνατό για ένα concept να εμφανίζεται και θετικά και αρνητικά σε ένα αξίωμα ή μια οντολογία.

Μία $\mathcal{SHIQ}$ οντολογία O είναι $\mathcal{Horn}$, εάν :

- Δεν εμφανίζεται θετικά στην O κανένα concept της μορφής $C \sqcup D$ ή $\leq mr.C$ με m>1.
- Δεν εμφανίζεται αρνητικά στην O κανένα concept της μορφής $\neg C, \forall\ r.C, \geq nr.C$ με n>1 , ή $\leq mr.C$.

Πριν από την εφαρμογή του αλγορίθμου ταξινόμησης βασισμένο στην τεχνική της Δομικής Υπαγωγής είναι απαραίτητη η πραγματοποίηση μιας προεργασίας πάνω στην $\mathcal{Horn\ SHIQ}$ οντολογία αντίστοιχη με την διαδικασία της κανονικοποίησης που έχουμε συναντήσει και σε άλλες περιγραφικές λογικές.

Το πρώτο στάδιο αυτής της προεργασίας ονομάζεται δομικός μετασχηματισμός (structural transformation) και χρησιμοποιείται για την απλοποίηση των αξιωμάτων της οντολογίας, διατηρώντας όμως την δομή της. Δεδομένης μια $\mathcal{Horn\ SHIQ}$ οντολογία O, για κάθε υποέννοια C στην O εισάγουμε μία νέα ατομική έννοια A_C και ορίζουμε μια συνάρτηση st(C) ως :

- st(A) = A , st($\top$) =$\top$, st($\bot$) = $\bot$
- st($\neg C$) = $\neg A_C$
- st($C \sqcap D$) = st($A_C \sqcap A_D$) , st($C \sqcup D$) = st($A_C \sqcup A_D$)
- st($\exists r.C$) = $\exists r.A_C$, st($\forall r.C$) = $\forall r.A_C$
- st($\geq nr.C$) = $\geq nr.A_C$, st($\leq mr.C$) = $\leq mr.A_C$

Το αποτέλεσμα της εφαρμογής του δομικού μετασχηματισμού στην O είναι η δημιουργία σε πολυωνυμικό χρόνο μιας νέας οντολογία O', η οποία είναι λογικά αντίστοιχη της αρχικής και περιέχει όλα τα role inclusions και role transivity axioms της O, με την προσθήκη των παρακάτω αξιωμάτων υπαγωγής εννοιών:

- $A_C \sqsubseteq$ st(C_+) για κάθε C που εμφανίζεται θετικά στην O
- st(C_-) $\sqsubseteq A_C$ για κάθε C που εμφανίζεται αρνητικά στην O
- $A_C \sqsubseteq A_D$ για κάθε concept inclusion $C \sqsubseteq D \in O$

Το δεύτερο στάδιο της απαραίτητης προεργασίας είναι μία μορφή κανονικοποίησης της οντολογίας που έχει προκύψει από την εφαρμογή του δομικού μετασχηματισμού ώστε να περιέχει αποκλειστικά αξιώματα της παρακάτω μορφής.

- $\prod A_i \sqsubseteq C$
- $r_1 \sqsubseteq r_2$
- Tra(r)

όπου $\prod A_i$ μία τομή ατομικών εννοιών (πιθανώς και κενή) και C μια απλή έννοια (simple concept)[27]

[27] Μία έννοια C ονομάζεται απλή (simple) εάν είναι της μορφής $\bot$, A, $\exists r.A$, $\forall r.A$ και $\leq 1 s.A$.

Για να επιτευχθεί η κανονική μορφή που επιθυμούμε πρέπει να εφαρμόσουμε μια σειρά από κανόνες κανονικοποίησης στην οντολογία O' που προκύπτει από το στάδιο του δομικού μετασχηματισμού. Από τη στιγμή που η αρχική οντολογίας O ήταν *Horn SHIQ* ,το C_+ , το οποίο εμπλέκεται μέσω της συνάρτησης st() με τα νέα αξιώματα της O', μπορεί να είναι μόνο της μορφής $\top$, $\bot$, A, $\neg C$, $C \sqcap D$, $\exists r.C$, $\forall r.C$, $\geq ns.C$ και $\leq 1r.C$, και το C_- αντίστοιχα μπορεί να είναι μόνο της μορφής $\top$, $\bot$, A, $C \sqcap D$, $C \sqcup D$, $\exists r.C$ και $\geq 1s.C$.

Επομένως, τα αξιώματα υπαγωγής της οντολογίας O' της μορφής $A \sqsubseteq \text{st}(C_+)$ τα οποία δεν πληρούν τις προϋποθέσεις κανονικοποίησης, μαζί με τον αντίστοιχο κανόνα που θα πρέπει να εφαρμοστεί ώστε να έρθουν σε κανονική μορφή είναι τα εξής:

- $A \sqsubseteq \text{st}(\neg C) = \neg A_C \rightarrow A \sqcap A_C \sqsubseteq \bot$
- $A \sqsubseteq \text{st}(\geq ns.C) = \geq ns.A_C \rightarrow A \sqsubseteq \exists s.B_i$, $B_i \sqsubseteq A_C$, $1 \leq i \leq n$, $B_i \sqcap B_j \sqsubseteq \bot$, $1 \leq i \leq j \leq n$, όπου τα B_i είναι καινούργια atomic concepts.

Αντίστοιχα, αξιώματα υπαγωγής της οντολογίας O' της μορφής $\text{st}(C_-) \sqsubseteq A$ τα οποία δεν πληρούν τις προϋποθέσεις κανονικοποίησης, μαζί με τον αντίστοιχο κανόνα που θα πρέπει να εφαρμοστεί ώστε να έρθουν σε κανονική μορφή είναι τα εξής:

- $\text{st}(C \sqcup D) = A_C \sqcup A_D \sqsubseteq A \rightarrow A_C \sqsubseteq A, A_D \sqsubseteq A$
- $\text{st}(\exists r.C) = \exists r.A_C \sqsubseteq A \rightarrow A_C \sqsubseteq \forall r^-.A$
- $\text{st}(\geq 1s.C) = \geq 1s.A_C \sqsubseteq A \rightarrow A_C \sqsubseteq \forall s^-.A$

Το τελευταίο στάδιο της διαδικασίας της προεργασίας αποτελείται από μία τεχνική για την απαλοιφή των αξιωμάτων μεταβατικών ρόλων. Τα μεταβατικά αξιώματα που υπάρχουν στην κανονικοποιημένη οντολογία μπορούν να αλληλεπιδράσουν μόνο με αξιώματα της μορφής $\prod A_i \sqsubseteq \forall r.B$ μέσω απλών αξιωμάτων υπαγωγής ρόλων. Επομένως αυτό το τελευταίο στάδιο της προεργασίας εισάγει μια νέα τριπλέτα αξιωμάτων για κάθε αξίωμα της μορφής $\prod A_i \sqsubseteq \forall r.B$ για το οποία υπάρχει και ένας μεταβατικό ρόλος t (Tra(t)) ο οποίος είναι υπορόλος του r. Η τριπλέτα αυτή όπου B^t είναι μία νέα ατομική έννοια είναι η εξής:

- $\prod A_i \sqsubseteq \forall t.B^t$
- $B^t \sqsubseteq \forall t.B^t$
- $B^t \sqsubseteq B$

Μετά την ολοκλήρωση της απαραίτητης προεργασίας μιας *Horn* SHIQ οντολογίας μπορούμε να εφαρμόσουμε εξαντλητικά και επαναληπτικά τους συμπερασματικούς κανόνες βασισμένους στην τεχνική της Δομικής Υπαγωγής που δίνονται παρακάτω. Στους παρακάτω κανόνες τα M και N αντιστοιχούν σε τομές ατομικών εννοιών, τα A, A_i, B σε ατομικές έννοιες και το C σε απλές (simple) έννοιες.

- *Αν* $M \in O$, *τότε* $O = O \cup \{M \sqsubseteq T\}$
- *Αν* $M \sqcap A \in O$, *τότε* $O = O \cup \{M \sqcap A \sqsubseteq A\}$
- *Αν* $M \sqsubseteq A_i \in O$ *και* $\prod A_i \sqsubseteq C \sqsubseteq O$, *τότε* $O = O \cup \{M \sqsubseteq C\}$

- *Αν* $M \sqsubseteq \exists r.N \in O$ *και* $N \sqsubseteq \bot \in O$, *τότε* $O = O \cup \{ M \sqsubseteq \bot \}$
- *Αν* $M \sqsubseteq \exists r_1.N \in O$, $M \sqsubseteq \forall r_2.A \in O$ *και* $r_1 \sqsubseteq r_2 \in O$, *τότε* $O = O \cup \{ M \sqsubseteq \exists r_1.(N \sqcap A)\}$
- *Αν* $M \sqsubseteq \exists r_1.N \in O$, $N \sqsubseteq \forall r_2.A \in O$ *και* $r_1 \sqsubseteq r_2^- \in O$, *τότε* $O = O \cup \{ M \sqsubseteq A\}$
- *Αν* $M \sqsubseteq \exists r_1.N_1 \in O$, $N_1 \sqsubseteq B \in O$, $r_1 \sqsubseteq s \in O$, $M \sqsubseteq \exists r_2.N_2 \in O$, $N_2 \sqsubseteq B \in O$, $r_2 \sqsubseteq s \in O$ *και* $M \sqsubseteq \leq 1 s.B$, *τότε* $O = O \cup \{ M \sqsubseteq \exists r_1.(N_1 \sqcap N_2)\}$
- *Αν* $M \sqsubseteq \exists r_1.N_1 \in O$, $N_1 \sqsubseteq \exists r_2.(N_2 \sqcap A) \in O$, $r_1 \sqsubseteq s^- \in O$, $M \sqsubseteq B \in O$, $N_2 \sqcap A \sqsubseteq B \in O$, $r_2 \sqsubseteq s \in O$ *και* $N_1 \sqsubseteq \leq 1 s.B$, *τότε* $O = O \cup \{ M \sqsubseteq A, M \sqsubseteq \exists r_2^-.N_1\}$

Η τεχνική της Δομικής Υπαγωγής στο *Horn SHIQ* τμήμα της περιγραφικής λογικής δεν είναι πολυωνυμικός, όπως συμβαίνει με το προηγούμενα υποτμήματα που μελετήσαμε, αλλά έχει αποδειχθεί ότι ο αλγόριθμος της απόφασης για την υπαγωγή εννοιών είναι ExpTime- complete.

2.4 Η γλώσσα OWL 2

Οι έρευνες που οδήγησαν στην κατασκευή της αρχικής γλώσσας OWL (Web Ontology Language) προήλθαν από μια αδιαμφισβήτητη διαπίστωση. Τα υπάρχοντα μοντέλα μέχρι εκείνη την στιγμή, όπως το RDFS, είχαν περιορισμένη εκφραστικότητα. Ομάδες εργασίας του World Wide Web Consortium (W3C) αναγνώρισαν μια σειρά από περιπτώσεις όπου δεν επιδείκνυαν επάρκεια και έπρεπε να επεκταθούν. Οι σημαντικότερες από αυτές είναι:

- Λογικός συνδυασμός κλάσεων (Boolean combination of classes)
  Δε μπορούμε να ορίσουμε κλάσεις ως συνδυασμό ήδη υπαρχόντων κλάσεων. (π.χ. Άνθρωπος ≡ Άντρας ⊔ Γυναίκα)
- Ασυμβατότητα κλάσεων (Disjointness of classes)
  Δε μπορούμε να πούμε ότι οι κλάσεις Άντρας και Γυναίκα είναι ασύμβατες.
- Ειδικά χαρακτηριστικά ιδιοτήτων (special characteristics of properties)
  Δε μπορούμε να δηλώσουμε μεταβατικές ιδιότητες (π.χ. Μεγαλύτερος_από), αντίστροφες ιδιότητες (π.χ. Είναι_μέρος_του και Έχει_μέρος) και λειτουργικές ιδιότητες (π.χ. Έχει_μητέρα).
- Τοπικό εύρος ιδιοτήτων (Local scope of properties)
  Δε μπορούμε να πούμε ότι το εύρος (range) της ιδιότητας Έχει_παιδί είναι Άτομο, όταν αναφέρεται σε ανθρώπους, ή Ελέφαντας, όταν αναφέρεται σε ελέφαντες.
- Περιορισμοί μεγέθους συνόλων (Cardinality restrictions)
  Δε μπορούμε να εκφράσουμε περιορισμούς όσον αφορά στον αριθμό (μέγιστο ή ελάχιστο) των διακριτών ατόμων που μπορούν να συσχετιστούν μέσω μιας ιδιότητας (π.χ. ένα κατάστημα μπορεί να έχει το πολύ 30 υπαλλήλους).
- Ταυτότητα/Διαφορετικότητα ατόμων (Equality/Inequality of individuals)

Δε μπορούμε να πούμε ότι δύο άτομα με διαφορετικά ονόματα είναι τα ίδια ή να δηλώσουμε ρητά ότι είναι διαφορετικά.

Η υπέρβαση των παραπάνω μειονεκτημάτων και η ταυτόχρονη προσπάθεια για εξασφάλιση αποδοτικής συλλογιστικής ανάλυσης οδήγησαν σε μια πιο «δυνατή» γλώσσα κατασκευής οντολογιών. Αρχικά και για να «κληροδοτηθούν» τα πλεονεκτήματα του RDF/S, προτάθηκαν οι DAML-ONT και OIL. Από αυτές προέκυψε η DAML+OIL [DAML] που με τη σειρά της αποτέλεσε τη βάση για την υλοποίηση της OWL.

Η OWL χρησιμοποιεί σε μεγάλο βαθμό το RDF/S και μπορούμε να πούμε ότι κατά κάποιο τρόπο το επεκτείνει. Αυτό γιατί η OWL, αν και έχει XML-like σύνταξη όπως ο «πρόγονός» της, αυτή είναι σαφώς βελτιωμένη και συνοδεύεται από ένα γραφικό μέρος που υιοθετεί συμβάσεις της UML (Unified Modeling Language) [UML] γλώσσας. Παρέχει ένα σύνολο κατασκευαστών (constructors) όπως τομή (conjuction), ένωση (disjunction), άρνηση (negation), υπαρξιακή ποσοτικοποίηση (existential quantification), περιορισμούς αριθμών (cardinality restrictions) κ.α. Η παραπομπή στην ορολογία της Περιγραφικής Λογικής είναι προφανής, γι' αυτό και στη βιβλιογραφία η OWL χαρακτηρίζεται ως DL-based.

Η OWL διαιρούταν σε τρεις γλώσσες

- OWL Full, η οποία αναφερόταν σ' ολόκληρη τη γλώσσα.
- OWL DL, η οποία ήταν το μέρος της OWL που ήταν αποφασίσιμο (decidable in finite time) και πρόσφερε πλήρη (sound & complete) συλλογιστική ανάλυση αντιστοιχώντας στην εκφραστικότητα $SHOIN^{(D)}$ της Περιγραφικής Λογικής
- OWL Lite, η οποία ήταν το τμήμα της OWL με τη χαμηλότερη εκφραστικότητα και κατά συνέπεια την ευκολότερη συλλογιστική ανάλυση αντιστοιχώντας στο τμήμα $SHIF^{(D)}$ της Περιγραφικής Λογικής.

Μετά την πρώτη παρουσίαση της OWL το 2004, το 2009 προτάθηκε από το W3C η επέκταση της γλώσσας OWL με ονομασία OWL 2 στην οποία προστέθηκαν νέα χαρακτηριστικά όπως η δυνατότητα με το ίδιο URI να ορίζονται τόσο κλάσεις όσο και άτομα και ρόλοι, η διερμηνεία να βασίζεται στο περιεχόμενο και η εισαγωγή μιας απλής μορφής meta–modelling. Αντίστοιχα με τις γλώσσες της OWL, στην OWL 2 ορίζονται οι γλώσσες OWL 2 Full και OWL 2 DL, ενώ δεν υπάρχει αντίστοιχη της Lite.

Όπως γίνεται εύκολα αντιληπτό από όσα προηγήθηκαν, η γλώσσα OWL 2 χαρακτηρίζεται από υψηλή εκφραστικότητα , τόσο υπολογιστική όσο και για τους χρήστες, καθιστώντας δύσκολη την χρήση της για την υλοποίηση σύνθετων εφαρμογών. Για την αντιμετώπιση αυτού του χαρακτηριστικού σχεδιάστηκαν τα επιπλέον profiles αυτής της γλώσσας ως προσεγγιστικά υποτμήματα της OWL 2 κατάλληλα για μια ποικιλία εφαρμογών πέρα από τις ήδη υπάρχουσες γλώσσες της OWL 2. Στα OWL 2 Profiles θυσιάζεται ένα κομμάτι της εκφραστικότητας της OWL 2 με αντάλλαγμα την αύξηση της απόδοσης της συλλογιστικής διαδικασίας και την βελτίωση της υπολογιστικής δυναμικότητας.

Υπάρχουν πολλά διαφορετικά υποτμήματα της OWL 2 που έχουν καλές υπολογιστικές ικανότητες αλλά ως OWL 2 Profiles έχουν επιλεγεί μόνο τρία από αυτά τα οποία είχαν ήδη μία πρωταρχική κοινότητα χρηστών. Τα τρία αυτά profiles είναι ανεξάρτητα μεταξύ τους, δηλαδή κανένα δεν είναι υποτμήμα κάποιου άλλου. Τα κύριο χαρακτηριστικό και των 3 profiles είναι ότι είναι tractable, δηλαδή μπορούν να αποφανθούν για συλλογιστικά ερωτήματα σε πολυωνυμικό χρόνο, και ντετερμινιστικά, δηλαδή δεν απαιτούν επιλογές προβλέψεις (guessing) ή πισωγυρισμάτων (backtracking) στην εκτέλεση των ερωτημάτων.

Συνοπτικά τα τρία OWL 2 profiles θα εξεταστούν στα επόμενα υποκεφάλαια. Πριν συμβεί αυτό όμως, θα ασχοληθούμε με την κοινή σύνταξη των οντοτήτων (entities), των σταθερών (literals – συνήθως συμβολοσειρές και ακέραιοι) και των ατόμων (individuals) και για τα 3 profiles της OWL 2. Σε κάθε profile θα αναφερόμαστε μόνο στα τμήματα της OWL 2 τα οποία τα οποία διαφοροποιούνται σε σχέση με το OWL 2 Specification[28]

2.4.1 Γενικά χαρακτηριστικά της OWL 2

Οι οντότητες (entities) είναι τα βασικά δομικά στοιχεία μιας OWL 2 οντολογίας και ορίζουν το λεξιλόγιο (vocabulary) της οντολογίας. Κάθε οντότητα είναι συνδεδεμένη μοναδικά με ένα IRI ενώ διαχωρίζονται στις παρακάτω κατηγορίες ανάλογα με τον σκοπό και τον τρόπο που έχουν οριστεί.

- *Κλάσεις* (Classes)

Οι κλάσεις αντιπροσωπεύουν ένα σύνολο από άτομα (individuals). Για παράδειγμα η κλάση a:Child αντιπροσωπεύει το σύνολο όλων των παιδιών. Στην OWL 2 υπάρχουν δύο προκαθορισμένες κλάσεις με IRIs owl:Thing[29] και owl:Nothing που αντιπροσωπεύουν το σύνολο όλων των ατόμων και το άδειο σύνολο αντίστοιχα.

- *Τύποι Δεδομένων* (Datatypes)

Τα datatypes αντιπροσωπεύουν ένα σύνολο από τιμές δεδομένων (data types) όπως είναι οι συμβολοσειρές και οι ακέραιοι. Για παράδειγμα το datatype xsd:integer υποδηλώνει το σύνολο των ακεραίων, το xsd:dateTime το σύνολο των ημερομηνιών και το owl:real το σύνολο των πραγματικών. Αντίστοιχα datatypes υπάρχουν για μια σειρά από τιμές δεδομένων. Συγκεντρωτικά τα πιο συνηθισμένα datatypes εμφανίζονται στον παρακάτω πίνακα.

[28] OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax

[29] Το πρόθεμα συμβολίζει το αντίστοιχο IRI που έχει γίνει ο ορισμός. Τα πιο συνηθισμένα είναι τα εξής:
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs: http://www.w3.org/2000/01/rdf-schema#
xsd: http://www.w3.org/2001/XMLSchema#
owl: http://www.w3.org/2002/07/owl#

Πραγματικοί, δεκαδικοί και ακέραιοι αριθμοί			
owl:real *xsd:nonNegativeInteger*	*owl:rational* *xsd:negativeInteger*	*xsd:decimal* *xsd:long*	*xsd:integer* *xsd:int*
xsd:nonPositiveInteger	*xsd:positiveInteger*	*xsd:short*	*xsd:byte*
xsd:unsignedLong	*xsd:unsignedInt*	*xsd:unsignedShort*	*xsd:unsignedByte*
Float		*xsd:double*	*xsd:float*
Συμβολοσειρές			
xsd:string *xsd:Name*	*xsd:normalizedString* *xsd:NCName*	*xsd:token* *xsd:NMTOKEN*	*xsd:language*
Boolean		*xsd:boolean*	
Binary Data		*xsd:hexBinary*	*xsd:base64Binary*
IRIs		*xsd:anyURI*	
Χρονικά στιγμιότυπα		*xsd:dateTime*	*xsd:dateTimeStamp*

Πίνακας 2.14 Συνηθισμένα Datatypes στην OWL 2

- *Ιδιότητες αντικειμένων* (Object Properties)

Τα object properties συνδέουν ζεύγη ατόμων (individuals). Για παράδειγμα το object property a:parentOf μπορεί να συνδέσει 2 άτομα μεταξύ τους. Στην OWL 2 υπάρχουν δύο προκαθορισμένα object properties με IRIs owl:topObjectProperty και owl:bottomObjectProperty που συνδέει όλα τα πιθανά ζεύγη ατόμων και δεν συνδέει κανένα ζεύγος ατόμων αντίστοιχα.

- *Ιδιότητες δεδομένων* (Data Properties)

Τα data properties συνδέουν ένα άτομο (individual) με μία σταθερά (literal). Για παράδειγμα το data property a:hasName μπορεί να συνδέσει ένα άτομο (a:John) με ένα string (π.χ. "John Papadopoulos") . Στην OWL 2 υπάρχουν δύο προκαθορισμένα data properties με IRIs owl:topDataProperty και owl:bottomDataProperty που συνδέει όλα τα πιθανά άτομα με όλες τις σταθερές και δεν συνδέει κανένα άτομο με σταθερά αντίστοιχα.

- *Ιδιότητες σχολίων* (Annotation Properties)

Τα annotation properties μπορούν να χρησιμοποιηθούν για να προσθέσουν ένα σχόλιο σε μία οντολογία, σε ένα IRI ή σε ένα αξίωμα. Για παράδειγμα το annotation property rdfs:comment μπορεί να προσθέσει ένα σχόλιο σε ένα συγκεκριμένο άτομο. Στην OWL 2 υπάρχουν μια σειρά από προκαθορισμένα annotation properties από σχετικά απλά όπως το rdfs:comment που προσθέτει ένα σχόλιο σε ένα IRI και το rdfs:label που προσθέτει ένα αντιληπτό από τον άνθρωπο όνομα σε ένα IRI έως πιο σύνθετα όπως το owl:incompatibleWith που

καθορίζει το IRI από μία προηγούμενη έκδοση της οντολογίας που δεν είναι συμβατή με την τρέχουσα έκδοση αυτής.

Εκτός από οντότητες στην OWL 2 υπάρχουν και άτομα (individuals). Υπάρχουν δύο τύποι individuals στην OWL 2, τα ονοματισμένα άτομα (Named Individuals) και τα ανώνυμα (Anonymous Individuals). Ένα παράδειγμα named individual είναι το a:John ενώ ένα παράδειγμα ανώνυμου ατόμου με τη αντίστοιχη σύνταξη είναι το _:a1.

Επιπλέον, στην OWL 2 υπάρχουν και οι σταθερές (literals) μέσω των οποίων γίνεται η αναπαράσταση τιμών δεδομένων όπως ένα συγκεκριμένο string ή οι ακέραιοι. Κάθε σταθερά αποτελείται από ένα lexical form, που είναι ένα string, και ένα datatype και η σύνταξη του είναι lexicalForm^^datatypeIRI. Για παράδειγμα η σταθερά-literal "1"^^xsd:integer αντιπροσωπεύει τον ακέραιο 1. Στην περίπτωση που το datatype είναι τύπου rdf:PlainLiteral τότε οι σταθερές μπορούν να χρησιμοποιηθούν ως συντομογραφίες, δηλαδή το "abc" μπορεί να χρησιμοποιηθεί μέσα στην οντολογία ως συντομογραφία του ορισμού σταθεράς "abc@"^^rdf:PlainLiteral.

Συνήθως σε μία OWL 2 οντολογία είναι απαραίτητο κάθε οντότητα να ορίζεται και αυτό είναι εφικτό μέσα από έναν τύπο αξιώματος γνωστό ως Declaration. Η σύνταξη του ορισμού μιας οντότητας είναι η εξής:

Declaration := 'Declaration' '(' **axiomAnnotations Entity** ')'
Entity :=
'Class' '(' **Class** ')' |
'Datatype' '(' **Datatype** ')' |
'ObjectProperty' '(' **ObjectProperty** ')' |
'AnnotationProperty' '(' **AnnotationProperty** ')' |
'NamedIndividual' '(' **NamedIndividual** ')'

Για παράδειγμα η κλάση άνθρωπος ορίζεται ως Declaration (Class (a:Person)) ενώ ένα άτομο ορίζεται ως Declaration (NamedIndividuals (a:John)).

Τέλος, στην OWL 2 υπάρχουν μια σειρά από αξιώματα ισχυρισμών που αφορούν τα άτομα (individual). Το πιο σημαντικό από αυτά τα αξιώματα είναι το αξίωμα ισχυρισμού κλάσης (ClassAssertion) μέσω του οποίου συνδέεται ένα άτομο με μία κλάση (π.χ. ClassAssertion (a:Person a:John)) η σύνταξη του οποίου είναι η εξής:

ClassAssertion := 'ClassAssertion' '(' **ClassExpression Individual** ')'

2.4.2 OWL 2 EL

Το profile αυτό, βασισμένο στην περιγραφική λογική $\mathcal{EL}$++ από την οποία πήρε και το όνομα του, είναι κατάλληλο για εφαρμογές που χρησιμοποιούν

οντολογίες με μεγάλο αριθμό ιδιοτήτων και κλάσεων, όπως η πολύ μεγάλη βιοϊατρική οντολογία SNOMED-CT. Η OWL 2 $\mathcal{EL}$ μπορεί να αιχμαλωτίσει την εκφραστική δύναμη που χρησιμοποιείται από τέτοιας κλίμακας οντολογίες και να αποφασίσει για τα βασικά προβλήματα συλλογιστική (συνέπεια οντολογίας - ontology consistency, υπαγωγής εκφράσεων κλάσεων - class expression subsumption, έλεγχος στιγμιοτύπου - instance checking) σε χρόνο πολυωνυμικό ως προς το μέγεθος της οντολογίας. Η συλλογιστική ανάλυση (reasoning) στην OWL 2 $\mathcal{EL}$ είναι PTIME – complete.

Στην OWL 2 $\mathcal{EL}$ οι οντότητες (entities) και τα οι τύποι δεδομένων (datatypes) ορίζονται όπως έχουν παρουσιαστεί στο προηγούμενο κεφάλαιο με ένα επιπλέον περιορισμό. Για να επιτυγχάνονται οι επιθυμητές υπολογιστικές ιδιότητες τα υποστηριζόμενα datatypes από την OWL 2 $\mathcal{EL}$ θα πρέπει να είναι τέτοια ώστε η τομή του πεδίου τιμών δύο από αυτών να είναι είτε κενή είτε πεπερασμένη. Επομένως στην OWL 2 $\mathcal{EL}$ δεν επιτρέπεται για προφανείς λόγους η χρήση των ακόλουθων datatypes (xsd:double, xsd:float, xsd:nonPositiveInteger, xsd:positiveInteger, xsd:negativeInteger, xsd:long, xsd:int, xsd:short, xsd:byte, xsd:unsignedLong , xsd:unsignedInt, xsd:unsignedByte, xsd:unsignedShort, xsd:language και xsd:boolean). Επίσης, η OWL 2 $\mathcal{EL}$ δεν υποστηρίζει ανώνυμα άτομα (anonymous individuals).

Για να είναι αποδοτική το reasoning στο profile OWL 2 $\mathcal{EL}$ το σύνολο των υποστηριζόμενων εκφράσεων που σχετίζονται με τις κλάσεις (ClassExpression) είναι το ακόλουθο και το καθένα από αυτά περιγράφεται αναλυτικά στη συνέχεια :

ClassExpression :=
Class | **ObjectIntersectionOf** | **ObjectOneOf** |
ObjectSomeValuesFrom | **ObjectHasValue** | **ObjectHasSelf** |
DataSomeValuesFrom | **DataHasValue**

- **Class** (παραπομπή στο 2.4.1)

- **ObjectIntersectionOf** := 'ObjectIntersectionOf' '(' **ClassExpression** **ClassExpression** { **ClassExpression** } ')'

Αντιπροσωπεύει την τομή δύο ή περισσότερων εκφράσεων κλάσεων και περιέχει όλα τα άτομα που είναι στιγμιότυπα όλων των εκφράσεων κλάσεων που περιέχονται στα ορίσματα. Στην περιγραφική λογική DL αντιστοιχεί με το $A \sqcap B \sqcap C$

Για παράδειγμα έχουμε την κλάση a:Man με στιγμιότυπα a:John και a:Alex ["Ο Γιάννης είναι άνδρας" "Ο Αλέξης είναι άνδρας"][30] και την κλάση a:Married που περιέχει επίσης το στιγμιότυπο a:John ["Ο Γιάννης είναι παντρεμένος"] τότε το ObjectIntersectionOf (a:Man a:Married) περιέχει το στιγμιότυπο a:John ["Ο Γιάννης είναι ένα άτομο το οποίο είναι άντρας και παντρεμένος"].

[30] Με τον συμβολισμό [ ] θα αναφέρουμε το αντίστοιχο παράδειγμα στην φυσική γλώσσα.

- **ObjectOneOf :**= ′ObjectOneOf′ ′(′ **Individual** ′)′

Γενικά στην OWL 2 η έκφραση ObjectOneOf χρησιμοποιείται για την απαρίθμηση ατόμων[31]. Συγκεκριμένα όμως στο profile OWL 2 $\mathcal{EL}$ επιτρέπεται μόνο ένα άτομο. Για παράδειγμα το ObjectOneOf (a:John) είναι μία απαρίθμηση με μόνο ένα άτομο.

- **ObjectSomeValuesFrom :**= ′ObjectSomeValuesFrom′ ′(′ **ObjectPropertyExpression ClassExpression** ′)′

Αντιπροσωπεύει την πλήρη υπαρξιακή ποσοτικοποίηση της περιγραφικής λογικής ($\exists r.C$). Αποτελείται από δύο ορίσματα, μία έκφραση ιδιοτήτων αντικειμένων και μία έκφραση κλάσης, και περιέχει όλα τα άτομα που συνδέονται μέσω της ιδιότητας αντικειμένων με ένα άτομο, το οποίο είναι στιγμιότυπο της έκφρασης κλάσης που δίνεται ως όρισμα.

Για παράδειγμα έχουμε την ιδιότητα αντικειμένων a:fatherOf που συνδέει τα άτομα a:John και a:Alex [“Ο Γιάννης είναι ο πατέρας του Αλέξη”] και έχουμε ότι το άτομο a:Alex είναι στιγμιότυπο της κλάσης a:Man [“Ο Αλέξης είναι άνδρας”] τότε το ObjectSomeValuesFrom(a:fatherOf a:Man) περιέχει το στιγμιότυπο a:John [“Ο Γιάννης είναι ένα άτομο για το οποίο ισχύει ότι είναι πατέρας κάποιου, ο οποίος είναι άνδρας”].

- **ObjectHasValue :**= ′ObjectHasValue′ ′(′ **ObjectPropertyExpression Individual** ′)′

Η συγκεκριμένη έκφραση αποτελεί μια συντακτική συντόμευση για την έκφραση κλάσης ObjectSomeValuesFrom (ObjectPropertyExpression ObjectOneOf (a)), δηλαδή περιέχει όλα τα άτομα που συνδέονται μέσω της ιδιότητας αντικειμένων με ένα άτομο.

Για παράδειγμα έχουμε την ιδιότητα αντικειμένων a:fatherOf που συνδέει τα άτομα a:John και a:Alex [“Ο Γιάννης είναι ο πατέρας του Αλέξη”] τότε το ObjectHasValue(a:fatherOf a:Alex) περιέχει το άτομο a:John [“Ο Γιάννης είναι ένα άτομο για το οποίο ισχύει ότι είναι πατέρας κάποιου, ο οποίος είναι ο Αλέξης”].

- **ObjectHasSelf :**= ′ObjectHasSelf′ ′(′ **ObjectPropertyExpression** ′)′

Η έκφραση αυτή περιέχει όλα τα άτομα που είναι συνδεδεμένα μέσω μιας συγκεκριμένης ιδιότητας αντικειμένων, η οποία δίνεται ως όρισμα, με τον εαυτό τους.

Για παράδειγμα έχουμε την ιδιότητα αντικειμένων a:likes που συνδέει τα άτομα a:John με a:Alex και a:John με a:John [“Ο Γιάννης συμπαθεί τον Αλέξη” “Ο Γιάννης συμπαθεί τον εαυτό του”] τότε το ObjectHasSelf(a:likes) περιέχει το

[31] **ObjectOneOf :**= ′ObjectOneOf′ ′(′ **Individual** {**Individual**} ′)′ (π.χ. ObjectOneOf (a:John a:Alex))

άτομο a:John [“Ο Γιάννης είναι ένα άτομο για το οποίο ισχύει ότι συμπαθεί τον εαυτό του”].

- **DataSomeValuesFrom :=** 'DataSomeValuesFrom' '(' **DataPropertyExpression DataRange** ')'

Η συγκεκριμένη υπαρξιακή έκφραση κλάσης δέχεται δύο ορίσματα, μία έκφραση ιδιοτήτων δεδομένων και μια έκφραση εύρους δεδομένων. Τα επιτρεπόμενα εύρη δεδομένων για την OWL 2 $\mathcal{EL}$ θα οριστούν στην συνέχεια αλλά το συνηθέστερο εύρος δεδομένων που χρησιμοποιείται είναι ένα απλό datatype. Η έκφραση αυτή περιλαμβάνει όλα τα άτομα που συνδέονται μέσω μιας ιδιότητας δεδομένων με ένα επιτρεπόμενο εύρος δεδομένων (συνήθως ένα datatype) που δίνεται ως όρισμα.

Για παράδειγμα έχουμε την ιδιότητα δεδομένων a:hasName που συνδέει το άτομο a:John με το string “John Papadopoulos” [“Το άτομο Γιάννης έχει όνομα Γιάννης Παπαδόπουλος”] τότε η έκφραση DataSomeValuesFrom(a:hasName “John Papadopoulos”) περιέχει το άτομο a:John [“Ο Γιάννης είναι ένα άτομο για το οποίο ισχύει ότι έχει το όνομα Γιάννης Παπαδόπουλος”]

- **DataHasValue :=** 'DataHasValue' '(' **DataPropertyExpression Literal** ')'

Η συγκεκριμένη έκφραση κλάσης μπορεί να θεωρηθεί ως μια συντόμευση της έκφρασης DataSomeValuesFrom στην περίπτωση που το δεύτερο όρισμα είναι μία σταθερά. Επομένως, η έκφραση αυτή περιλαμβάνει όλα τα άτομα που συνδέονται μέσω μιας ιδιότητας δεδομένων με μία σταθερά που δίνεται ως όρισμα.

Για παράδειγμα έχουμε την ιδιότητα δεδομένων a:hasAge που συνδέει το άτομο a:John με το literal “23”^^xsd:integer [“Το άτομο Γιάννης είναι 23 ετών”] τότε η έκφραση DataHasValue(a:hasAge “23”^^xsd:integer) περιέχει το άτομο a:John [“Ο Γιάννης είναι ένα άτομο για το οποίο ισχύει ότι έχει ηλικία ίση με 23”]

Εκτός από τις υποστηριζόμενες εκφράσεις κλάσεων το profile OWL 2 $\mathcal{EL}$ υποστηρίζει μια σειρά από εκφράσεις εύρους δεδομένων (Data Ranges) οι οποίες αναλύονται ακολούθως.

DataRange := **Datatype** | **DataIntersectionOf** | **DataOneOf**

- **Datatype** (παραπομπή στο 2.4.1)

- **DataIntersectionOf :=** 'DataIntersectionOf' '(' **DataRange DataRange** { **DataRange** } ')'

Αντιπροσωπεύει την τομή δύο ή περισσότερων εκφράσεων εύρους δεδομένων και επιτρέπει όλα τα δεδομένα που είναι επιτρεπτά σε όλα τα εύρη δεδομένων που περιέχονται στα ορίσματα.

Για παράδειγμα το DataIntersectionOf (xsd:nonNegativeInteger xsd:nonPositiveInteger)[32] περιέχει το 0.

- **DataOneOf :**= 'DataOneOf' '(' **Literal** ')'

Γενικά στην OWL 2 η έκφραση DataOneOf χρησιμοποιείται για την απαρίθμηση σταθερών[33]. Συγκεκριμένα όμως στο profile OWL 2 $\mathcal{EL}$ επιτρέπεται μόνο μία σταθερά. Για παράδειγμα το DataOneOf ("23"^^xsd:integer) είναι μία απαρίθμηση με μόνο μία σταθερά - literal.

Στη συνέχεια θα εξετάσουμε τους επιτρεπόμενους τύπους αξιωμάτων στο profile OWL 2 $\mathcal{EL}$. Τα αξιώματα διακρίνονται σε τρεις κατηγορίες, ανάλογα με τον τύπο των εκφράσεων που εμπλέκονται σε αυτά, τα αξιώματα κλάσεων (ClassAxiom), τα αξιώματα ιδιοτήτων αντικειμένων (ObjectPropertyAxiom) και τα αξιώματα ιδιοτήτων δεδομένων (DataPropertyAxiom).

Τα επιτρεπόμενα αξιώματα κλάσεων συνοψίζονται στο παρακάτω σχήμα και το καθένα από αυτά αναλύεται στη συνέχεια.

ClassAxiom := **SubClassOf** | **EquivalentClasses** | **DisjointClasses**

- **SubClassOf** := 'SubClassOf' '(' **subClassExpression** **superClassExpression** ')'
  subClassExpression := **ClassExpression**
  superClassExpression := **ClassExpression**
  Το συγκεκριμένο αξίωμα κλάσης είναι το βασικότερο και συνηθέστερο αξίωμα στην περιγραφή μιας οντολογίας με τον φορμαλισμό OWL, είναι αντίστοιχο με την σχέση υπαγωγής ($A \sqsubseteq B$) της περιγραφικής λογικής και χρησιμοποιείται για την δημιουργία μιας ιεραρχίας κλάσεων. Στην περιγραφική λογική θα μπορούσαμε να γράψουμε subClassExpression $\sqsubseteq$ superClassExpression. Σημασιολογικά το αξίωμα αυτό δηλώνει ότι όλα τα στιγμιότυπα της υποκλάσης είναι και στιγμιότυπα της υπερκλάσης.
  Για παράδειγμα το αξίωμα SubClassOf (a:Baby a:Child) στην φυσική γλώσσα μπορεί να μεταφραστεί ως [" Όποιο άτομο είναι μωρό είναι και παιδί"] ενώ το αξίωμα SubClassOf (a:Parent ObjectSomeValuesFrom (a:hasChild a:Child)) αντιστοιχεί στο [" Όποιο άτομο είναι γονιός έχει και την ιδιότητα ότι έχει τουλάχιστον ένα παιδί"].

- **EquivalentClasses :**= 'EquivalentClasses' '(' **ClassExpression** **ClassExpression** {**ClassExpression**} ')'

[32] Το παράδειγμα είναι εποπτικό καθώς τα συγκεκριμένα datatypes δεν είναι επιτρεπτά στην OWL 2 $\mathcal{EL}$.

[33] **DataOneOf :**= 'DataOneOf' '(' **Literal** {**Literal**} ')' (π.χ. DataOneOf ("John" "23"^^xsd:integer))

Το συγκεκριμένο αξίωμα υλοποιεί τη σχέση ισοδυναμίας της περιγραφικής λογικής ($A \equiv B$), θέτει δύο ή περισσότερες εκφράσεις κλάσεων ισοδύναμες και σημασιολογικά δηλώνει ότι τα άτομα που είναι στιγμιότυπα της μιας έκφρασης κλάσης είναι σίγουρα στιγμιότυπα και των άλλων.
Για παράδειγμα το αξίωμα EquivalentClasses(a:Boy ObjectIntersectionOf (a:Child a:Man)) αντιστοιχεί στο [“Το αγόρι είναι ένα αρσενικό παιδί”].

- **DisjointClasses :=** ′DisjointClasses′ ′(′ **ClassExpression ClassExpression** {**ClassExpression**}′)′

Με το συγκεκριμένο αξίωμα δύο ή περισσότερες εκφράσεις κλάσεων τίθενται ξένες μεταξύ τους, δηλαδή σημασιολογικά δηλώνει ότι κανένα άτομο δεν μπορεί να είναι ταυτόχρονα στιγμιότυπο σε δύο από τις εκφράσεις κλάσεων που δίνονται ως ορίσματα.
Για παράδειγμα το αξίωμα DisjointClasses(a:Boy a:Girl) αντιστοιχεί στο [“ Ένα άτομο δεν μπορεί να είναι ταυτόχρονα και αγόρι και κορίτσι”]

Τα επιτρεπόμενα αξιώματα ιδιοτήτων αντικειμένων συνοψίζονται στο παρακάτω σχήμα και αναλύονται στη συνέχεια.

ObjectPropertyAxiom :=
EquivalentObjectProperties | **SubObjectPropertyO**f |
ObjectPropertyDomain | **ObjectPropertyRange** |
ReflexiveObjectProperty | **TransitiveObjectProperty**

- **EquivalentObjectProperties :=** ′EquivalentObjectProperties′ ′(′ **ObjectPropertyExpression ObjectPropertyExpression** {**ObjectPropertyExpression**} ′)′

Το αξίωμα αυτό δηλώνει ότι δύο ή περισσότερες εκφράσεις ιδιότητας αντικειμένων είναι σημασιολογικά ισοδύναμες ή αλλιώς ότι αν δύο άτομα συνδέονται μεταξύ τους με μια ιδιότητα αντικειμένων τότε θα συνδέονται και με τις υπόλοιπες. Το αξίωμα αυτό είναι μια συντακτική συντόμευση των αξιωμάτων SubObjectPropertyOf (OPE_1 OPE_2) και SubObjectPropertyOf (OPE_2 OPE_1) .
Για παράδειγμα το αξίωμα EquivalentObjectProperties(a:hasBoy a:hasMaleChild) αντιστοιχεί στο [“Το να έχει κάποιος ένα αγόρι είναι το ίδιο με το να έχει ένα αρσενικό παιδί”].

- **SubObjectPropertyOf :=** ′SubObjectPropertyOf′ ′(′ **subObjectPropertyExpression superObjectPropertyExpression** ′)′

subObjectPropertyExpression := ObjectPropertyExpression | **propertyExpressionChain**
propertyExpressionChain := ′ObjectPropertyChain ′ ′(′ **ObjectPropertyExpression ObjectPropertyExpression** {**ObjectPropertyExpression**} ′)′

superObjectPropertyExpression := **ObjectPropertyExpression**
Το συγκεκριμένο αξίωμα είναι αντίστοιχο με το αξίωμα υπαγωγής ρόλων της περιγραφικής λογικής ($r \sqsubseteq s$) και υποδηλώνει ότι η μια έκφραση ιδιότητας αντικειμένων είναι υποέκφραση ιδιότητας της άλλης. Σημασιολογικά το αξίωμα αυτό δηλώνει ότι αν ένα άτομο x συνδέεται μέσω της υποέκφρασης ιδιότητας αντικειμένων με ένα άτομο y τότε συνδέεται με το y και μέσω της υπερέκφρασης ιδιότητας αντικειμένων. Μία ειδική περίπτωση είναι όταν η υποέκφραση ιδιότητας δεν είναι μια απλή έκφραση ιδιότητας αντικειμένων αλλά είναι μία αλυσίδα ιδιοτήτων. Σε αυτή την περίπτωση σημασιολογικά το αξίωμα δηλώνει ότι αν ένα άτομο x συνδέεται μέσω της αλυσίδας με το άτομο y τότε συνδέεται και μέσω της υπερέκφρασης ιδιότητας.
Για παράδειγμα το αξίωμα SubObjectPropertyOf (a:hasDog a:hasPet) αντιστοιχεί στο [“ Όποιο άτομο έχει σκυλί συνεπάγεται ότι έχει και κατοικίδιο”] ενώ το αξίωμα SubObjectPropertyOf (ObjectPropertyChain (a:hasMother a:hasSister) a:hasAunt) αντιστοιχεί στο [“ Όποιο άτομο έχει μητέρα η οποία έχει αδελφή συνεπάγεται ότι αυτό το άτομο έχει θεία”].

- **ObjectPropertyDomain :**= ′ObjectPropertyDomain′ ′(′ **ObjectPropertyExpression ClassExpression** ′)′

Το συγκεκριμένο αξίωμα ορίζει το πεδίο ορισμού (domain) μιας έκφρασης ιδιότητας αντικειμένων, δηλαδή θέτει έναν περιορισμό (constraint) στις επιτρεπόμενες εκφράσεις κλάσεων που μπορούν να συνδεθούν μέσω μιας έκφρασης ιδιότητας αντικειμένων με άλλες εκφράσεις κλάσεων. Επομένως, σημασιολογικά εάν ένα άτομο x συνδέεται μέσω της έκφρασης ιδιότητας αντικειμένων που δίνεται ως όρισμα με ένα οποιοδήποτε άλλο άτομο τότε αυτό το άτομο x είναι στιγμιότυπο της δεδομένης έκφρασης κλάσεως. Το αξίωμα αυτό είναι μια συντακτική συντόμευση του αξιώματος SubClassOf (ObjectSomeValuesFrom (OPE owl:Thing) CE).
Για παράδειγμα το αξίωμα ObjectPropertyDomain (a:hasDog a:Person) αντιστοιχεί στο ισοδύναμο στη φυσική γλώσσα [“ Όποιο άτομο έχει σκυλί πρέπει να είναι και άνθρωπος”].

- **ObjectPropertyRange :**= ′ObjectPropertyRange′ ′(′ **ObjectPropertyExpression ClassExpression** ′)′

Το αξίωμα αυτό είναι αντίστοιχο με το προηγούμενο με την διαφορά ότι ορίζεται το πεδίο εμβέλειας (range) μιας έκφρασης ιδιότητας αντικειμένων, δηλαδή θέτει έναν περιορισμό (constraint) στις επιτρεπόμενες εκφράσεις κλάσεων, οι οποίες μέσω μιας έκφρασης ιδιότητας αντικειμένων μπορούν να συνδεθούν από άλλες εκφράσεις κλάσεων. Επομένως, σημασιολογικά εάν ένα οποιοδήποτε άτομο συνδέεται μέσω της έκφρασης ιδιότητας αντικειμένων που δίνεται ως όρισμα με ένα άτομο x τότε αυτό το άτομο x είναι στιγμιότυπο της δεδομένης έκφρασης κλάσεως. Το αξίωμα αυτό είναι μια συντακτική συντόμευση του αξιώματος SubClassOf (owl:Thing ObjectAllValuesFrom (OPE CE)).

Για παράδειγμα το αξίωμα ObjectPropertyRange (a:hasDog a:Dog) αντιστοιχεί στο ισοδύναμο στη φυσική γλώσσα [“ Κάποιος μπορεί να έχει σκυλί μόνο κάτι που είναι σκυλί”].

- **ReflexiveObjectProperty** := ′ReflexiveObjectProperty′ ′(′ **ObjectPropertyExpression** ′)′

Το συγκεκριμένο αξίωμα υποδηλώνει ότι μια έκφραση ιδιότητας αντικειμένων είναι ανακλαστική (reflexive), δηλαδή σημασιολογικά ότι όλα τα άτομα συνδέονται με τον εαυτό τους μέσω της συγκεκριμένης έκφρασης ιδιότητας αντικειμένων. Το αξίωμα αυτό είναι μια συντακτική συντόμευση του αξιώματος SubClassOf (owl:Thing ObjectHasSelf (OPE)).
Για παράδειγμα το αξίωμα ReflexiveObjectProperty (a:knows) αντιστοιχεί στο [“ Κάθε άτομο γνωρίζει τον εαυτό του”].

- **TransitiveObjectProperty** := ′TransitiveObjectProperty′ ′(′ **ObjectPropertyExpression** ′)′

Το συγκεκριμένο αξίωμα υποδηλώνει ότι μια έκφραση ιδιότητας αντικειμένων είναι μεταβατική (transitive), δηλαδή σημασιολογικά ότι εάν ένα άτομο x συνδέεται μέσω της έκφρασης ιδιότητας αντικειμένων με το άτομο y και το άτομο y συνδέεται με το άτομο z μέσω της ίδιας έκφρασης ιδιότητας αντικειμένων τότε και το άτομο x θα συνδέεται με το άτομο μέσω της συγκεκριμένης έκφρασης ιδιότητας αντικειμένων. Το αξίωμα αυτό είναι μια συντακτική συντόμευση του αξιώματος SubObjectPropertyOf (ObjectPropertyChain (OPE OPE) OPE).
Για παράδειγμα το αξίωμα TransitiveObjectProperty (a:ancestorOf) αντιστοιχεί στο [“ Κάθε άτομο γνωρίζει τον εαυτό του”].

Τέλος, τα επιτρεπόμενα αξιώματα ιδιοτήτων δεδομένων (data property axiom) συνοψίζονται παρακάτω και αναλύονται στη συνέχεια.

DataPropertyAxiom :=
SubDataPropertyOf | **EquivalentDataProperties** |
DataPropertyDomain | **DataPropertyRange** | **FunctionalDataProperty**

- **SubDataPropertyOf** := ′SubDataPropertyOf′ ′(′ **subDataPropertyExpression superDataPropertyExpression** ′)′

subDataPropertyExpression := **DataPropertyExpression**
superDataPropertyExpression := **DataPropertyExpression**
Το συγκεκριμένο αξίωμα υποδηλώνει ότι η μια έκφραση ιδιότητας δεδομένων είναι υποέκφραση ιδιότητας της άλλης. Σημασιολογικά το αξίωμα αυτό δηλώνει ότι αν ένα άτομο x συνδέεται μέσω της υποέκφρασης ιδιότητας δεδομένων με μία σταθερά (literal) y τότε συνδέεται με το y και μέσω της υπερέκφρασης ιδιότητας δεδομένων.

Για παράδειγμα το αξίωμα SubDataPropertyOf (a:hasLastName a:hasName) αντιστοιχεί στο [“ Όποιο άτομο έχει ένα επίθετο συνεπάγεται ότι έχει και ένα όνομα”] .

- **EquivalentDataProperties** := ′EquivalentDataProperties′ ′(′ **DataPropertyExpression DataPropertyExpression** {**DataPropertyExpression**} ′)′

Το αξίωμα αυτό δηλώνει ότι δύο ή περισσότερες εκφράσεις ιδιότητας δεδομένων είναι σημασιολογικά ισοδύναμες ή αλλιώς ότι αν ένα άτομο συνδέεται με μία σταθερά (literal) y μέσω μιας έκφρασης ιδιότητας δεδομένων τότε θα συνδέεται και μέσω των υπολοίπων. Το αξίωμα αυτό είναι μια συντακτική συντόμευση των αξιωμάτων SubDataPropertyOf (DPE_1 DPE_2) και SubDataPropertyOf (DPE_2 DPE_1) .
Για παράδειγμα το αξίωμα EquivalentDataProperties(a:hasName a:hatNamen) δηλώνει ότι είναι ισοδύναμο ένα άτομο να συνδεθεί με το όνομα του μέσω μια ιδιότητας δεδομένων είτε εκφρασμένη στα Αγγλικά είτε στα Γερμανικά.

- **DataPropertyDomain** := ′DataPropertyDomain′ ′(′**DataPropertyExpression ClassExpression**′)′

Το συγκεκριμένο αξίωμα ορίζει το πεδίο ορισμού (domain) μιας έκφρασης ιδιότητας δεδομένων, δηλαδή θέτει έναν περιορισμό (constraint) στις επιτρεπόμενες εκφράσεις κλάσεων που μπορούν να συνδεθούν μέσω μιας έκφρασης ιδιότητας δεδομένων δεδομένων με οποιαδήποτε σταθερά (literal) . Επομένως, σημασιολογικά εάν ένα άτομο x συνδέεται μέσω της έκφρασης ιδιότητας δεδομένων που δίνεται ως όρισμα με ένα οποιοδήποτε literal τότε αυτό το άτομο x είναι στιγμιότυπο της δεδομένης έκφρασης κλάσεως. Το αξίωμα αυτό είναι μια συντακτική συντόμευση του αξιώματος SubClassOf (DataSomeValuesFrom (DPE rdfs:Literal) CE).
Για παράδειγμα το αξίωμα DataPropertyDomain (a:hasName a:Person) αντιστοιχεί στο ισοδύναμο στη φυσική γλώσσα [“ Όποιο άτομο έχει όνομα πρέπει να είναι και άνθρωπος”].

- **DataPropertyRange** := ′DataPropertyRange′ ′(′ **DataPropertyExpression DataRange** ′)′

Το αξίωμα αυτό είναι αντίστοιχο με το προηγούμενο με την διαφορά ότι ορίζεται το πεδίο εμβέλειας (range) μιας έκφρασης ιδιότητας δεδομένων, δηλαδή θέτει έναν περιορισμό (constraint) στα επιτρεπόμενα εύρη δεδομένων, τα οποία μέσω μιας έκφρασης ιδιότητας δεδομένων μπορούν να συνδεθούν με κάποιες εκφράσεις κλάσεων. Επομένως, σημασιολογικά εάν ένα οποιοδήποτε άτομο συνδέεται μέσω της έκφρασης ιδιότητας δεδομένων που δίνεται ως όρισμα με μία σταθερά (literal) x τότε αυτό η σταθερά x ανήκει στο δεδομένο εύρος δεδομένων. Το αξίωμα αυτό είναι μια συντακτική συντόμευση του αξιώματος SubClassOf (owl:Thing DataAllValuesFrom (DPE DR)).

Για παράδειγμα το αξίωμα DataPropertyRange (a:hasName xsd:string) αντιστοιχεί στο ισοδύναμο στη φυσική γλώσσα [“ Κάποιος μπορεί να έχει όνομα μόνο κάτι που είναι τύπου συμβολοσειράς”].

- **FunctionalDataProperty** := 'FunctionalDataProperty' '(' **DataPropertyExpression** ')'

Το συγκεκριμένο αξίωμα υποδηλώνει ότι μια έκφραση ιδιότητας δεδομένων είναι λειτουργική (functional), δηλαδή σημασιολογικά ότι για όλα τα άτομα μπορεί να υπάρξει μέχρι μία το πολύ σταθερά (literal) με την οποία συνδέονται μέσω της έκφρασης ιδιότητας δεδομένων. Το αξίωμα αυτό είναι μια συντακτική συντόμευση του αξιώματος SubClassOf (owl:Thing DataMaxCardinality (1 DPE)).
Για παράδειγμα το αξίωμα FunctionalDataProperty (a:hasAge) αντιστοιχεί στο [“ Κάθε άτομο έχει το πολύ μία ηλικία”].

2.4.3 OWL 2 QL

Το profile OWL 2 QL απευθύνεται σε εφαρμογές που χρησιμοποιούν πολύ μεγάλους όγκους δεδομένων και η απάντηση ερωτημάτων (query answering) είναι το σημαντικότερο πρόβλημα συλλογιστικής. Το ακρωνύμιο QL αντανακλά το γεγονός ότι η απάντηση ερωτημάτων σε αυτό το profile μπορεί να πραγματοποιηθεί ξαναγράφοντας ερωτήματα σε μία τυπική σχεσιακή γλώσσα ερωτημάτων (Query Language).

Στο profile OWL 2 QL, η συνδετική απάντηση ερωτήματος (conjunctive query answering) μπορεί να εφαρμοστούν με συμβατικά συστήματα σχεσιακών βάσεων δεδομένων. Χρησιμοποιώντας μια κατάλληλη τεχνική συλλογιστικής, μια ορθή και πλήρης διαδικασία απάντησης ερωτημάτων μπορεί να εκτελεστεί ακόμα και σε LOGSPACE χρόνο με σεβασμό στο μέγεθος των δεδομένων (ισχυρισμούς). Όπως και στην OWL 2 EL, αλγόριθμοι πολυωνυμικού χρόνου αλγόριθμοι μπορούν να χρησιμοποιηθούν για τα λοιπά προβλήματα συλλογιστικής όπως η συνέπεια οντολογία (ontology consistency) και η υπαγωγή εκφράσεων κλάσεων (class expression subsumption).

Επιπλέον, βασίζεται στην DL-Lite οικογένεια της περιγραφικής λογικής και η εκφραστική δύναμη του profile είναι αναγκαστικά αρκετά περιορισμένη, αν και περιλαμβάνει τα περισσότερα από τα κύρια χαρακτηριστικά των εννοιολογικών μοντέλων (conceptual model), όπως τα UML διαγράμματα κλάσεων και τα ER διαγράμματα.

Στο OWL 2 QL οι οντότητες (entities) και οι τύποι δεδομένων (datatypes) ορίζονται όπως έχουν παρουσιαστεί στην ενότητα 2.4.2 για το OWL 2 EL με τους ίδιους περιορισμούς σχετικά με τα υποστηριζόμενα datatypes και τη μη υποστήριξη ανώνυμων ατόμων (anonymous individuals).

Κύριο χαρακτηριστικό του OWL 2 *QL* και σημαντική διαφοροποίηση σε σχέση με το OWL 2 *EL* είναι η ύπαρξη περιορισμών σχετικά με τους τύπους των υποστηριζόμενων εκφράσεων που σχετίζονται με τις κλάσεις (ClassExpression) ανάλογα με το που εμπλέκονται αυτές οι εκφράσεις κλάσεων στα αξιώματα υπαγωγής. Για αυτό το λόγο ορίζουμε τις εκφράσεις υποκλάσεων (subClassExpression) και τις εκφράσεις υπερκλάσεων (superClassExpression) αντίστοιχα.

Για τις εκφράσεις υποκλάσεων (subClassExpression) οι υποστηριζόμενες εκφράσεις συνοψίζονται στο ακόλουθο σχήμα και περιγράφονται αναλυτικά στη συνέχεια.

subClassExpression :=
Class |
SubObjectSomeValuesFrom | **DataSomeValuesFrom**

- **Class** (βλ. 2.4.1)

- **DataSomeValuesFrom** **:=** 'DataSomeValuesFrom' '(' **DataPropertyExpression DataRange** ')' (βλ. **DataSomeValuesFrom** στην 2.4.2)
- **subObjectSomeValuesFrom** **:=** 'ObjectSomeValuesFrom' '(' **ObjectPropertyExpression** *owl:Thing* ')'
  Αντίστοιχο με το **ObjectSomeValuesFrom** που ορίστηκε στην ενότητα 2.4.2 με τη διαφορά ότι το όρισμα της έκφρασης κλάσης περιορίζεται στην παράμετρο *owl:Thing*. Επομένως αντιπροσωπεύει την υπαρξιακή ποσοτικοποίηση της περιγραφικής λογικής ($\exists r.\top$) και περιέχει όλα τα άτομα που συνδέονται μέσω της ιδιότητας αντικειμένων με ένα τουλάχιστον άτομο.
  Για παράδειγμα έχουμε την ιδιότητα αντικειμένων a:fatherOf που συνδέει τα άτομα a:John και a:Alex ["Ο Γιάννης είναι ο πατέρας του Αλέξη"] και έχουμε ότι το άτομο a:Alex είναι στιγμιότυπο της κλάσης a:Man ["Ο Αλέξης είναι άνδρας"] τότε το ObjectSomeValuesFrom(a:fatherOf owl:Thing) περιέχει το στιγμιότυπο a:John ["Ο Γιάννης είναι ένα άτομο για το οποίο ισχύει ότι είναι πατέρας κάποιου"].

Για τις εκφράσεις υπερκλάσεων (superClassExpression) οι υποστηριζόμενες εκφράσεις συνοψίζονται στο ακόλουθο σχήμα και περιγράφονται αναλυτικά στη συνέχεια.

superClassExpression :=
Class |
superObjectIntersectionOf | **superObjectComplementOf** |
superObjectSomeValuesFrom | **DataSomeValuesFrom**

- **Class** (βλ. 2.4.1)

- **superObjectIntersectionOf** := 'ObjectIntersectionOf' '(' **superClassExpression superClassExpression** { **superClassExpression** } ')' (βλ. **ObjectIntersectionOf** στην 2.4.2)

- **superObjectComplementOf** := 'ObjectComplementOf' '(' **ClassExpression** ')'

Η συγκεκριμένη έκφραση κλάσης περιέχει όλα τα άτομα (individual) που δεν είναι στιγμιότυπα της έκφρασης κλάσης που δίνεται ως όρισμα, δηλαδή αντιστοιχεί στη έννοια του συμπληρώματος.
Για παράδειγμα έχουμε τις κλάσεις a:Man και a:Woman, οι οποίες είναι ξένες μεταξύ τους. Αν το a:John είναι στιγμιότυπο της κλάσης a:Man ["Ο Γιάννης είναι άνδρας"] τότε το ObjectComplementOf (a:Woman) περιέχει το στιγμιότυπο a:John. ["Ο Γιάννης είναι ένα άτομο το όποιο δεν μπορεί να είναι γυναίκα"].

- **superObjectSomeValuesFrom** := 'ObjectSomeValuesFrom' '(' **ObjectPropertyExpression superClassExpression** ')' (βλ. **ObjectSomeValuesFrom** στην 2.4.2)

- **DataSomeValuesFrom** := 'DataSomeValuesFrom' '(' **DataPropertyExpression DataRange** ')' (βλ. **DataSomeValuesFrom** στην 2.4.2)

Εκτός από τις υποστηριζόμενες εκφράσεις κλάσεων το profile OWL 2 *QL* υποστηρίζει μια σειρά από εκφράσεις εύρους δεδομένων (Data Ranges) οι οποίες αναλύονται ακολούθως.

DataRange := **Datatype** | **DataIntersectionOf**

- **Datatype** (βλ. στην 2.4.1)

- **DataIntersectionOf** := 'DataIntersectionOf' '(' **DataRange DataRange** { **DataRange** } ')' (βλ. **DataIntersectionOf** στην 2.4.2)

Στη συνέχεια θα εξετάσουμε τους επιτρεπόμενους τύπους αξιωμάτων στο profile OWL 2 *QL*. Τα αξιώματα διακρίνονται σε τρεις κατηγορίες, ανάλογα με τον τύπο των εκφράσεων που εμπλέκονται σε αυτά, τα αξιώματα κλάσεων (ClassAxiom), τα αξιώματα ιδιοτήτων αντικειμένων (ObjectPropertyAxiom) και τα αξιώματα ιδιοτήτων δεδομένων (DataPropertyAxiom).

Τα επιτρεπόμενα αξιώματα κλάσεων συνοψίζονται στο παρακάτω σχήμα και το καθένα από αυτά αναλύεται στη συνέχεια.

ClassAxiom := **SubClassOf** | **EquivalentClasses** | **DisjointClasses**

- **SubClassOf** := 'SubClassOf' '(' **subClassExpression superClassExpression**')' (βλ. **SubClassOf** στην 2.4.2)

- **EquivalentClasses** := 'EquivalentClasses' '(' **subClassExpression subClassExpression** {**subClassExpression**} ')'

Αντίστοιχα με τα αξιώματα **EquivalentClasses** που ορίστηκαν στην 2.4.2 με την διαφορά ότι στα αξιώματα ισοδυναμίας στην OWL 2 *QL* μπορούν να εμπλακούν μόνο εκφράσεις κλάσεων με τους περιορισμούς που ορίστηκαν για τις υποκλάσεις.

- **DisjointClasses** := 'DisjointClasses' '(' **subClassExpression subClassExpression** {**subClassExpression**}')'

Αντίστοιχα με τα αξιώματα **DisjointClasses** που ορίστηκαν στην 2.4.2 με την διαφορά ότι στην OWL 2 *QL* μπορούν να εμπλακούν μόνο εκφράσεις κλάσεων με τους περιορισμούς που ορίστηκαν για τις υποκλάσεις.

Τα επιτρεπόμενα αξιώματα ιδιοτήτων αντικειμένων συνοψίζονται στο παρακάτω σχήμα και αναλύονται στη συνέχεια.

ObjectPropertyAxiom :=
SubObjectPropertyOf | **EquivalentObjectProperties** |
DisjointObjectProperties | **InverseObjectProperties** |
ObjectPropertyDomain | **ObjectPropertyRange** |
ReflexiveObjectProperty |
SymmetricObjectProperty | **AsymmetricObjectProperty**

- **EquivalentObjectProperties** := 'EquivalentObjectProperties' '(' **ObjectPropertyExpression ObjectPropertyExpression** {**ObjectPropertyExpression**} ')' (βλ. **EquivalentObjectProperties** στην 2.4.2)

- **SubObjectPropertyOf** := 'SubObjectPropertyOf' '(' **ObjectPropertyExpression ObjectPropertyExpression** ')'

Αντίστοιχα με τα αξιώματα **SubObjectPropertyOf** που ορίστηκαν στην 2.4.2 με την διαφορά ότι στα αξιώματα υπαγωγής ιδιοτήτων αντικειμένων στην OWL 2 *QL* δεν μπορούν να χρησιμοποιηθούν αλυσίδες αντικειμένων παρά μόνο εκφράσεις ιδιοτήτων αντικειμένων.

- **DisjointObjectProperties** := 'DisjointObjectProperties' '(' **ObjectPropertyExpression ObjectPropertyExpression** {**ObjectPropertyExpression**} ')'

Το αξίωμα αυτό δηλώνει ότι δύο ή περισσότερες εκφράσεις ιδιότητας αντικειμένων είναι σημασιολογικά ξένες ή αλλιώς ότι αν δύο άτομα συνδέονται μεταξύ τους με μια ιδιότητα αντικειμένων τότε δεν θα μπορούν να συνδέονται και με κάποια από τις υπόλοιπες.
Για παράδειγμα το αξίωμα DisjointObjectProperties(a:hasFather a:hasMother) αντιστοιχεί στο ["Δεν μπορεί ένα άτομο να είναι ταυτόχρονα και πατέρας και μητέρα ενός άλλου ατόμου"].

- **InverseObjectProperties** := 'InverseObjectProperties' '(' **ObjectPropertyExpression ObjectPropertyExpression** ')'

Το αξίωμα αυτό δηλώνει ότι οι εκφράσεις ιδιότητας αντικειμένων που δίνονται ως ορίσματα είναι σημασιολογικά αντίστροφες ή αλλιώς ότι αν ένα άτομο x συνδέεται μέσω την μίας ιδιότητας με ένα άτομο y , τότε το y συνδέεται μέσω τις άλλης ιδιότητας με το άτομο x.
Για παράδειγμα το αξίωμα InverseObjectProperties(a:hasFather a:fatherOf) αντιστοιχεί στο ["Αν ένα άτομο x έχει πατέρα ένα άτομο y τότε το y είναι πατέρας του ατόμου x"].

- **ObjectPropertyDomain** := 'ObjectPropertyDomain' '(' **ObjectPropertyExpression superClassExpression** ')'

Αντίστοιχα με τα αξιώματα **ObjectPropertyDomain** που ορίστηκαν στην 2.4.2 με την διαφορά ότι στην OWL 2 *QL* μπορούν να εμπλακούν μόνο εκφράσεις κλάσεων με τους περιορισμούς που ορίστηκαν για τις υπερκλάσεις (superClassExpression).

- **ObjectPropertyRange** := 'ObjectPropertyRange' '(' **ObjectPropertyExpression superClassExpression** ')'

Αντίστοιχα με τα αξιώματα **ObjectPropertyRange** που ορίστηκαν στην 2.4.2 με την διαφορά ότι στην OWL 2 *QL* μπορούν να εμπλακούν μόνο εκφράσεις κλάσεων με τους περιορισμούς που ορίστηκαν για τις υπερκλάσεις (superClassExpression).

- **ReflexiveObjectProperty** := 'ReflexiveObjectProperty' '(' **ObjectPropertyExpression** ')' (βλ. **ReflexiveObjectProperty** στην 2.4.2)

- **SymmetricObjectProperty** := 'SymmetricObjectProperty' '(' **ObjectPropertyExpression** ')'

Το συγκεκριμένο αξίωμα υποδηλώνει ότι μια έκφραση ιδιότητας αντικειμένων είναι συμμετρική (symmetric), δηλαδή σημασιολογικά ότι εάν ένα άτομο x συνδέεται μέσω της έκφρασης ιδιότητας αντικειμένων με το άτομο y τότε και το

άτομο x θα συνδέεται με το άτομο y μέσω της συγκεκριμένης έκφρασης ιδιότητας αντικειμένων. Το αξίωμα αυτό είναι μια συντακτική συντόμευση του αξιώματος SubObjectPropertyOf (OPE ObjectInverseOf (OPE)).
Για παράδειγμα το αξίωμα SymmetricObjectProperty (a:hasFriend) αντιστοιχεί στο [“ Αν το άτομο x έχει φίλο το άτομο y, τότε και το άτομο y έχει φίλο το άτομο x”].

- **AsymmetricObjectProperty** := 'AsymmetricObjectProperty' '('**ObjectPropertyExpression**')'

Το συγκεκριμένο αξίωμα υποδηλώνει ότι μια έκφραση ιδιότητας αντικειμένων είναι μη συμμετρική (asymmetric), δηλαδή σημασιολογικά ότι εάν ένα άτομο x συνδέεται μέσω της έκφρασης ιδιότητας αντικειμένων με το άτομο y τότε το άτομο x δεν μπορεί να συνδέεται με το άτομο y μέσω της συγκεκριμένης έκφρασης ιδιότητας αντικειμένων.
Για παράδειγμα το αξίωμα AsymmetricObjectProperty (a:parentOf) αντιστοιχεί στο [“ Αν το άτομο x είναι γονέας του ατόμου y, τότε το άτομο y δεν μπορεί να είναι γονέας του ατόμου x”].

Τέλος, τα επιτρεπόμενα αξιώματα ιδιοτήτων δεδομένων (data property axiom) συνοψίζονται παρακάτω και αναλύονται στη συνέχεια.

DataPropertyAxiom :=
SubDataPropertyOf | **EquivalentDataProperties** | **DisjointDataProperties** | **DataPropertyDomain** | **DataPropertyRange**

- **SubDataPropertyOf** := 'SubDataPropertyOf' '(' **subDataPropertyExpression superDataPropertyExpression** ')'
subDataPropertyExpression := **DataPropertyExpression**
superDataPropertyExpression := **DataPropertyExpression**
(βλ. **SubDataPropertyOf** στην 2.4.2)

- **EquivalentDataProperties** := 'EquivalentDataProperties' '(' **DataPropertyExpression DataPropertyExpression** {**DataPropertyExpression**} ')'
(βλ. **SubDataPropertyOf** στην 2.4.2)

- **DisjointDataProperties** := 'DisjointDataProperties' '(' **DataPropertyExpression DataPropertyExpression** {**DataPropertyExpression**} ')'

Το αξίωμα αυτό δηλώνει ότι δύο ή περισσότερες εκφράσεις ιδιότητας δεδομένων είναι σημασιολογικά ξένες ή αλλιώς ότι αν ένα άτομο συνδέεται με μία σταθερά

(literal) y μέσω μιας έκφρασης ιδιότητας δεδομένων τότε δεν θα μπορεί να συνδέεται και μέσω των υπολοίπων.
Για παράδειγμα το αξίωμα DisjointDataProperties(a:hasName a:hasAge) δηλώνει ότι ένα άτομο δεν μπορεί να έχει το ίδιο όνομα και την ίδια ηλικία.

- **DataPropertyDomain** := 'DataPropertyDomain' '('**DataPropertyExpression superClassExpression**')'

Αντίστοιχα με τα αξιώματα **DataPropertyDomain** που ορίστηκαν στην 2.4.2 με την διαφορά ότι στην OWL 2 $\mathcal{QL}$ μπορούν να εμπλακούν μόνο εκφράσεις κλάσεων με τους περιορισμούς που ορίστηκαν για τις υπερκλάσεις (superClassExpression).

- **DataPropertyRange** := 'DataPropertyRange' '(' **DataPropertyExpression DataRange** ')'
  (βλ. **SubDataPropertyOf** στην 2.4.2)

2.4.4 OWL 2 RL

Το τρίτο και τελευταίο profile που θα μελετήσουμε είναι το OWL 2 $\mathcal{RL}$, το οποίο απευθύνεται σε εφαρμογές που απαιτούν κλιμακούμενη συλλογιστική ανάλυση (scalable reasoning) χωρίς να θυσιάζουν αρκετή εκφραστική δύναμη. Το ακρωνύμιο $\mathcal{RL}$ αντανακλά το γεγονός ότι η συλλογιστική ανάλυση σε αυτό το προφίλ μπορεί να γίνει με τη χρήση μια τυπικής γλώσσας κανόνων (Rule Language) μέσω συλλογιστικών μηχανών βασισμένων σε κανόνες.

Τα σημαντικότερα προβλήματα συλλογιστικής μπορούν να επιλυθούν σε πολυωνυμικό χρόνο με σεβασμό στο μέγεθος της οντολογίας, για παράδειγμα η τυπική συλλογιστική ανάλυση σε OWL 2 $\mathcal{RL}$ είναι PTIME-complete.

Στο OWL 2 $\mathcal{RL}$ οι οντότητες (entities) και οι τύποι δεδομένων (datatypes) ορίζονται όπως έχουν παρουσιαστεί στην ενότητα 2.4.1 με εξαίρεση την υποστήριξη των παρακάτω datatypes (*owl:topObjectProperty*, *owl:bottomObjectProperty*, *owl:topDataProperty* , *owl:bottomDataProperty*, *owl:real* και *owl:rational*). Επίσης στα αξιώματα ισχυρισμών (ClassAssertions) μπορούν να συμμετέχουν μόνο εκφράσεις κλάσεων με τους περιορισμούς (που θα παρουσιαστούν στη συνέχεια) για τις εκφράσεις υπερκλάσεων.

Όπως και στο OWL 2 $\mathcal{QL}$, έτσι και στο OWL 2 $\mathcal{RL}$ υπάρχουν περιορισμοί σχετικά με τους τύπους των υποστηριζόμενων εκφράσεων που σχετίζονται με τις κλάσεις (ClassExpression) ανάλογα με το που εμπλέκονται αυτές οι εκφράσεις κλάσεων στα αξιώματα. Για αυτό το λόγο ορίζουμε τις εκφράσεις υποκλάσεων (subClassExpression), τις εκφράσεις υπερκλάσεων (superClassExpression) και τις εκφράσεις κλάσεων ισοδυναμίας (equivClassExpression) αντίστοιχα.

Για τις εκφράσεις υποκλάσεων (subClassExpression) οι υποστηριζόμενες εκφράσεις συνοψίζονται στο ακόλουθο σχήμα και περιγράφονται αναλυτικά στη συνέχεια.

subClassExpression :=
Class other than *owl:Thing* |
subObjectIntersectionOf | **subObjectUnionOf** | **ObjectOneOf** |
subObjectSomeValuesFrom | **ObjectHasValue** |
DataSomeValuesFrom | **DataHasValue**

- **Class** (βλ. 2.4.1)

- **subObjectIntersectionOf** := 'ObjectIntersectionOf' '(' **subClassExpression subClassExpression** { **subClassExpression** } ')'

Αντίστοιχα με τα αξιώματα **ObjectIntersectionOf** που ορίστηκαν στην 2.4.2 με την διαφορά ότι στα αξιώματα ισοδυναμίας στην OWL 2 $\mathcal{RL}$ μπορούν να εμπλακούν μόνο εκφράσεις κλάσεων με τους περιορισμούς που ορίστηκαν για τις υποκλάσεις.

- **subObjectUnionOf** := 'ObjectUnionOf' '(' **subClassExpression subClassExpression** { **subClassExpression** } ')'

Αντιπροσωπεύει την ένωση δύο ή περισσότερων εκφράσεων υποκλάσεων και περιέχει όλα τα άτομα που είναι στιγμιότυπα τουλάχιστον σε μία έκφραση υποκλάσεων που περιέχονται στα ορίσματα. Στην περιγραφική λογική $\mathcal{DL}$ αντιστοιχεί με το $A \sqcup B \sqcup C$
Για παράδειγμα έχουμε την κλάση a:Man με στιγμιότυπο a:John["Ο Γιάννης είναι άνδρας"] και την κλάση a:Woman που περιέχει επίσης το στιγμιότυπο a:Mary ["Η Μαίρη είναι γυναίκα"] τότε το ObjectUnionOf (a:Man a:WomanMarried) περιέχει τα στιγμιότυπα a:John και a:Mary ["Ο Γιάννης και η Μαίρη είναι άτομα τα οποία είναι είτε άντρας είτε γυναίκα"].

- **ObjectOneOf** := 'ObjectOneOf' '(' **Individual** ')' (βλ. **ObjectOneOf** στην 2.4.2)

- **subObjectSomeValuesFrom** := 'ObjectSomeValuesFrom' '(' **ObjectPropertyExpression** *owl:Thing* ')'

subObjectSomeValuesFrom := 'ObjectSomeValuesFrom' '(' **ObjectPropertyExpression subClassExpression** ')' (βλ. **subObjectSomeValuesFrom** στην 2.4.3 και **ObjectSomeValuesFrom** στην 2.4.2 για υποκλάσεις)

- **ObjectHasValue** := 'ObjectHasValue' '(' **ObjectPropertyExpression Individual** ')'

(βλ. **ObjectHasValue** στην 2.4.2)

- **DataSomeValuesFrom** := 'DataSomeValuesFrom' '(' **DataPropertyExpression DataRange** ')' (βλ. **DataSomeValuesFrom** στην 2.4.2)

- **DataHasValue** := 'DataHasValue' '(' **DataPropertyExpression Literal** ')' (βλ. **ObjectHasValue** στην 2.4.2)

Για τις εκφράσεις υπερκλάσεων (superClassExpression) οι υποστηριζόμενες εκφράσεις συνοψίζονται στο ακόλουθο σχήμα και περιγράφονται αναλυτικά στη συνέχεια.

superClassExpression :=
Class other than *owl:Thing* |
superObjectIntersectionOf | **superObjectComplementOf** |
superObjectAllValuesFrom | **ObjectHasValue** | **superObjectMaxCardinality** |
DataAllValuesFrom | **DataHasValue** | **superMaxCardinality**

- **Class** (βλ. 2.4.1)

- **superObjectIntersectionOf** := 'ObjectIntersectionOf' '(' **superClassExpression superClassExpression** { **superClassExpression** } ')' (βλ. **ObjectIntersectionOf** στην 2.4.2)

- **superObjectComplementOf** := 'ObjectComplementOf' '(' **superClassExpression** ')'
(βλ. **superObjectComplementOf** στην 2.4.3)

- **superObjectAllValuesFrom** := 'ObjectAllValuesFrom' '(' **ObjectPropertyExpression superClassExpression** ')'

Αντιπροσωπεύει τον περιορισμό τιμής της περιγραφικής λογικής ($\forall r.C$). Αποτελείται από δύο ορίσματα, μία έκφραση ιδιοτήτων αντικειμένων και μία έκφραση υπερκλάσης, και περιέχει όλα τα άτομα που συνδέονται μέσω της ιδιότητας αντικειμένων μόνο με άτομα, τα οποία είναι στιγμιότυπα της έκφρασης υπερκλάσης που δίνεται ως όρισμα.
Για παράδειγμα έχουμε την ιδιότητα αντικειμένων a:hasPet που συνδέει τα άτομα a:John και a:Jack και a:John και a:Fox [“Ο Γιάννης έχει κατοικίδιο τον Jack” ,“Ο Γιάννης έχει κατοικίδιο τον Fox”] και έχουμε ότι τα άτομα a:Jack και a:Fox είναι στιγμιότυπα της κλάσης a:Dog [“Ο Jack είναι σκύλος” , “Ο Fox είναι σκύλος”] τότε το ObjectAllValuesFrom(a:hasPet a:Dog) περιέχει το στιγμιότυπο a:John

[“Ο Γιάννης είναι ένα άτομο για το οποίο ισχύει ότι όλα τα κατοικίδια που έχει είναι σκυλιά”].

- **ObjectHasValue :=** ′ObjectHasValue′ ′(′ **ObjectPropertyExpression Individual** ′)′
  (βλ. **ObjectHasValue** στην 2.4.2)

- **superObjectMaxCardinality** :=
‘ObjectMaxCardinality’ ‘(‘ **zeroOrOne ObjectPropertyExpression** [**subClassExpression** ‘)’ |
‘ObjectMaxCardinality’ ‘(‘ **zeroOrOne ObjectPropertyExpression** *owl:Thing* ‘)’
Αντιπροσωπεύει τόσο τον περιορισμό αριθμού (≤*nr*) όσο και τον ποιοτικό περιορισμού αριθμού (≤*nr.C*)της περιγραφικής λογικής με n=0 ή 1. Αποτελείται από τρία ορίσματα, τον πληθικό αριθμό, μία έκφραση ιδιοτήτων αντικειμένων και μία έκφραση υποκλάσης είτε το *owl:Thing*, και περιέχει όλα τα άτομα που συνδέονται μέσω της ιδιότητας αντικειμένων με κανένα ή το πολύ 1 άτομο. Αν υπάρχει και το τρίτο όρισμα θα πρέπει επιπλέον τα άτομα αυτά να είναι στιγμιότυπα της έκφρασης υποκλάσης που δίνεται ως όρισμα.
Για παράδειγμα έχουμε την ιδιότητα αντικειμένων a:hasPet που συνδέει τα άτομα a:John και a:Jack [“Ο Γιάννης έχει κατοικίδιο τον Jack] και έχουμε ότι το άτομο a:Jack είναι στιγμιότυπο της κλάσης a:Dog [“Ο Jack είναι σκύλος”] τότε το ObjectMaxCardinality(1 a:hasPet a:Dog) περιέχει το στιγμιότυπο a:John [“Ο Γιάννης είναι ένα άτομο για το οποίο ισχύει ότι έχει το πολύ 1 κατοικίδιο το οποίο είναι σκυλί”].

- **DataAllValuesFrom :=** ′DataAllValuesFrom′ ′(′ **DataPropertyExpression DataRange** ′)′

Η συγκεκριμένη καθολική έκφραση κλάσης δέχεται δύο ορίσματα, μία έκφραση ιδιοτήτων δεδομένων και μια έκφραση εύρους δεδομένων, που συνήθως είναι ένα datatype. Η έκφραση αυτή περιλαμβάνει όλα τα άτομα που συνδέονται μέσω μιας ιδιότητας δεδομένων αποκλειστικά με ένα επιτρεπόμενο εύρος δεδομένων (συνήθως ένα datatype) που δίνεται ως όρισμα.
Για παράδειγμα έχουμε την ιδιότητα δεδομένων a:hasUserID που συνδέει ένα άτομο με τα UserID που χρησιμοποιεί, τα οποίο μπορεί να είναι είτε ένα string είτε ένα integer τότε η έκφραση DataAllValuesFrom(a:hasUserID xsd:integer) περιέχει όλα τα άτομα που όλα τους τα UserID είναι ακέραιοι.

- **DataHasValue :=** ′DataHasValue′ ′(′ **DataPropertyExpression Literal** ′)′
  (βλ. **DataHasValue** στην 2.4.2)

- **superDataMaxCardinality** := 'DataMaxCardinality' '(' **zeroOrOne DataPropertyExpression [DataRange]** ')'

Αποτελείται από τρία ορίσματα, τον πληθικό αριθμό, μία έκφραση ιδιοτήτων δεδομένων και μία έκφραση εύρους δεδομένων (rdfs:Literal αν απουσιάζει), και περιέχει όλα τα άτομα που συνδέονται μέσω της ιδιότητας δεδομένων με κανένα ή το πολύ 1 τύπο δεδομένων που ορίζονται στο εύρος.
Για παράδειγμα έχουμε την ιδιότητα αντικειμένων a:hasName που συνδέει το άτομο a:John με ένα δεδομένο τύπου string τότε το DataMaxCardinality(1 a:hasName) περιέχει το στιγμιότυπο a:John ["Ο Γιάννης είναι ένα άτομο για το οποίο ισχύει ότι έχει το πολύ ένα όνομα"].

Εκτός από τις υποστηριζόμενες εκφράσεις κλάσεων το profile OWL 2 *RL* υποστηρίζει μια σειρά από εκφράσεις εύρους δεδομένων (Data Ranges) οι οποίες αναλύονται ακολούθως.

DataRange := **Datatype** | **DataIntersectionOf**

- **Datatype** (βλ. στην 2.4.1)

- **DataIntersectionOf** := 'DataIntersectionOf' '(' **DataRange DataRange** { **DataRange** } ')'
(βλ. **DataIntersectionOf** στην 2.4.2)

Στη συνέχεια θα εξετάσουμε τους επιτρεπόμενους τύπους αξιωμάτων στο profile OWL 2 *RL*. Τα επιτρεπόμενα αξιώματα κλάσεων (ClassAxioms) συνοψίζονται στο παρακάτω σχήμα και το καθένα από αυτά αναλύεται στη συνέχεια.

ClassAxiom := **SubClassOf** | **EquivalentClasses** | **DisjointClasses**

- **SubClassOf** := 'SubClassOf' '(' **subClassExpression superClassExpression**')'
(βλ. **SubClassOf** στην 2.4.2)

- **EquivalentClasses** := 'EquivalentClasses' '(' **subClassExpression subClassExpression** {**subClassExpression**} ')'

Αντίστοιχα με τα αξιώματα **EquivalentClasses** που ορίστηκαν στην 2.4.2 με την διαφορά ότι στα αξιώματα ισοδυναμίας στην OWL 2 *QL* μπορούν να εμπλακούν μόνο εκφράσεις κλάσεων με τους περιορισμούς που ορίστηκαν για τις υποκλάσεις.

- **DisjointClasses** := 'DisjointClasses' '(' **subClassExpression subClassExpression** {**subClassExpression**}')'

Αντίστοιχα με τα αξιώματα **DisjointClasses** που ορίστηκαν στην 2.4.2 με την διαφορά ότι στην OWL 2 *QL* μπορούν να εμπλακούν μόνο εκφράσεις κλάσεων με τους περιορισμούς που ορίστηκαν για τις υποκλάσεις.

Τα επιτρεπόμενα αξιώματα ιδιοτήτων (Object Property Axioms) αντικειμένων συνοψίζονται στο παρακάτω σχήμα και αναλύονται στη συνέχεια.

ObjectPropertyAxiom :=
SubObjectPropertyOf | **EquivalentObjectProperties** |
DisjointObjectProperties | **InverseObjectProperties** |
ObjectPropertyDomain | **ObjectPropertyRange** |
FunctionalObjectProperty | **InverseFunctionalObjectProperty** |
IrreflexiveObjectProperty |
SymmetricObjectProperty | **AsymmetricObjectProperty** |
TransitiveObjectProperty

- **SubObjectPropertyOf** (βλ. **SubObjectPropertyOf** στην 2.4.2)

- **EquivalentObjectProperties** := 'EquivalentObjectProperties' '(' **ObjectPropertyExpression** **ObjectPropertyExpression** {**ObjectPropertyExpression**} ')'
(βλ. **EquivalentObjectProperties** στην 2.4.2)

- **DisjointObjectProperties** := 'DisjointObjectProperties' '(' **ObjectPropertyExpression** **ObjectPropertyExpression** {**ObjectPropertyExpression**} ')'
(βλ. **EquivalentObjectProperties** στην 2.4.3)

- **InverseObjectProperties** := 'InverseObjectProperties' '(' **ObjectPropertyExpression ObjectPropertyExpression** ')'
(βλ. **InverseObjectProperties** στην 2.4.3)

- **ObjectPropertyDomain** := 'ObjectPropertyDomain' '(' **ObjectPropertyExpression superClassExpression** ')'

Αντίστοιχα με τα αξιώματα **ObjectPropertyDomain** που ορίστηκαν στην 2.4.2 με την διαφορά ότι στην OWL 2 *RL* μπορούν να εμπλακούν μόνο εκφράσεις κλάσεων με τους περιορισμούς που ορίστηκαν για τις υπερκλάσεις (superClassExpression).

- **ObjectPropertyRange** := 'ObjectPropertyRange' '(' **ObjectPropertyExpression superClassExpression** ')'

Αντίστοιχα με τα αξιώματα **ObjectPropertyRange** που ορίστηκαν στην 2.4.2 με την διαφορά ότι στην OWL 2 $\mathcal{RL}$ μπορούν να εμπλακούν μόνο εκφράσεις κλάσεων με τους περιορισμούς που ορίστηκαν για τις υπερκλάσεις (superClassExpression).

- **FunctionalObjectProperty** := 'FunctionalObjectProperty' '(' **ObjectPropertyExpression** ')'

Το συγκεκριμένο αξίωμα υποδηλώνει ότι μια έκφραση ιδιότητας αντικειμένων είναι λειτουργική (functional), δηλαδή σημασιολογικά ότι για κάθε άτομο x μπορεί να υπάρχει το πολύ ένα άτομο y με το οποίο να συνδέεται μέσω της έκφρασης ιδιότητας αντικειμένων. Το αξίωμα αυτό είναι μια συντακτική συντόμευση του αξιώματος SubClassOf (owl:Thing ObjectMaxCardinality (1 OPE)).
Για παράδειγμα το αξίωμα FunctionalObjectProperty (a:hasFather) αντιστοιχεί στο ["Κάθε άτομο μπορεί να έχει το πολύ έναν πατέρα"].

- **InverseFunctionalObjectProperty** := 'InverseFunctionalObjectProperty' '(' **ObjectPropertyExpression** ')'

Το συγκεκριμένο αξίωμα υποδηλώνει ότι μια έκφραση ιδιότητας αντικειμένων είναι αντίστροφα λειτουργική (inverse-functional), δηλαδή σημασιολογικά ότι για κάθε άτομο x μπορεί να υπάρχει το πολύ ένα άτομο y το οποίο να συνδέεται μέσω της έκφρασης ιδιότητας αντικειμένων με το x. Το αξίωμα αυτό είναι μια συντακτική συντόμευση του αξιώματος SubClassOf (owl:Thing ObjectMaxCardinality (1 ObjectInverseOf(OPE))).
Για παράδειγμα το αξίωμα FunctionalObjectProperty (a:fatherOf).

- **IrreflexiveObjectProperty** := 'FunctionalObjectProperty' '(' **ObjectPropertyExpression** ')'

Το συγκεκριμένο αξίωμα υποδηλώνει ότι μια έκφραση ιδιότητας αντικειμένων είναι μη ανακλαστική (irreflexive), δηλαδή σημασιολογικά ότι κανένα άτομο δεν μπορεί να συνδέεται με τον εαυτό τους μέσω της συγκεκριμένης έκφρασης ιδιότητας αντικειμένων. Το αξίωμα αυτό είναι μια συντακτική συντόμευση του αξιώματος SubClassOf (ObjectHasSelf (OPE) owl:Nothing).
Για παράδειγμα το αξίωμα ReflexiveObjectProperty (a:hasFather) αντιστοιχεί στο [" Κανένα άτομο δεν μπορεί να έχει πατέρα τον εαυτό του"].

- **SymmetricObjectProperty** := 'SymmetricObjectProperty' '(' **ObjectPropertyExpression** ')'
  (βλ. **SymmetricObjectProperty** στην 2.4.3)

- **AsymmetricObjectProperty** :='AsymmetricObjectProperty' '('**ObjectPropertyExpression**')'
  (βλ. **SymmetricObjectProperty** στην 2.4.3)

- **TransitiveObjectProperty** := 'TransitiveObjectProperty' '(' **ObjectPropertyExpression** ')'
  (βλ. **TransitiveObjectProperty** στην 2.4.2)

2.4.5 Συγκεντρωτικό Υλικό για την OWL 2

Ολοκληρώνοντας την αναφορά μας στην γλώσσα αναπαράστασης γνώσης OWL 2 μπορούμε να συνοψίσουμε στους παρακάτω πίνακες τη σύνταξη και τη σημασιολογία μέσω της διερμηνείας όλων των κατασκευαστών της γλώσσας που συναντήσαμε στα profiles καθώς και κάποιων κατασκευαστών που δεν συναντήσαμε σε κανένα profile αλλά περιλαμβάνονται στην OWL 2[34].

ObjectInverseOf(OP)	$\{ (x , y) \mid (y , x) \in (OP)^{OP} \}$
DataIntersectionOf(DR_1 ... DR_n)	$(DR_1)^{DT} \cap ... \cap (DR_n)^{DT}$
DataUnionOf(DR_1 ... DR_n)	$(DR_1)^{DT} \cup ... \cup (DR_n)^{DT}$
DataComplementOf(DR)	$(\Delta_D)^n \setminus (DR)^{DT}$ where n is the arity of DR
DataOneOf(lt_1 ... lt_n)	$\{ (lt_1)^{LT} , ... , (lt_n)^{LT} \}$
DatatypeRestriction(DT F_1 lt_1 ... F_n lt_n)	$(DT)^{DT} \cap (F_1 , lt_1)^{FA} \cap ... \cap (F_n , lt_n)^{FA}$
ObjectIntersectionOf(CE_1 ... CE_n)	$(CE_1)^{C} \cap ... \cap (CE_n)^{C}$
ObjectUnionOf(CE_1 ... CE_n)	$(CE_1)^{C} \cup ... \cup (CE_n)^{C}$
ObjectComplementOf(CE)	$\Delta_I \setminus (CE)^{C}$
ObjectOneOf(a_1 ... a_n)	$\{ (a_1)^{I} , ... , (a_n)^{I} \}$
ObjectSomeValuesFrom(OPE CE)	$\{ x \mid \exists y : (x, y) \in (OPE)^{OP}$ and $y \in (CE)^{C} \}$
ObjectAllValuesFrom(OPE CE)	$\{ x \mid \forall y : (x, y) \in (OPE)^{OP}$ implies $y \in (CE)^{C} \}$
ObjectHasValue(OPE a)	$\{ x \mid (x , (a)^{I}) \in (OPE)^{OP} \}$
ObjectHasSelf(OPE)	$\{ x \mid (x , x) \in (OPE)^{OP} \}$
ObjectMinCardinality(n OPE)	$\{ x \mid \#\{ y \mid (x , y) \in (OPE)^{OP} \} \geq n \}$
ObjectMaxCardinality(n OPE)	$\{ x \mid \#\{ y \mid (x , y) \in (OPE)^{OP} \} \leq n \}$
ObjectExactCardinality(n OPE)	$\{ x \mid \#\{ y \mid (x , y) \in (OPE)^{OP} \} = n \}$
ObjectMinCardinality(n OPE CE)	$\{ x \mid \#\{ y \mid (x , y) \in (OPE)^{OP}$ and $y \in (CE)^{C} \} \geq n \}$
ObjectMaxCardinality(n OPE CE)	$\{ x \mid \#\{ y \mid (x , y) \in (OPE)^{OP}$ and $y \in (CE)^{C} \} \leq n \}$

[34] Δεν κρίνεται σκόπιμη για τις ανάγκες τις εργασίας η αναλυτική παρουσίαση αυτών των κατασκευαστών.

ObjectExactCardinality(n OPE CE)	$\{ x \mid \#\{ y \mid (x, y) \in (OPE)^{OP} \text{ and } y \in (CE)^{C} \} = n \}$
DataSomeValuesFrom(DPE_1 ... DPE_n DR)	$\{ x \mid \exists\, y_1, \ldots, y_n : (x, y_k) \in (DPE_k)^{DP}$ for each $1 \leq k \leq n$ and $(y_1, \ldots, y_n) \in (DR)^{DT} \}$
DataAllValuesFrom(DPE_1 ... DPE_n DR)	$\{ x \mid \forall\, y_1, \ldots, y_n : (x, y_k) \in (DPE_k)^{DP}$ for each $1 \leq k \leq n$ imply $(y_1, \ldots, y_n) \in (DR)^{DT} \}$
DataHasValue(DPE lt)	$\{ x \mid (x, (lt)^{LT}) \in (DPE)^{DP} \}$
DataMinCardinality(n DPE)	$\{ x \mid \#\{ y \mid (x, y) \in (DPE)^{DP} \} \geq n \}$
DataMaxCardinality(n DPE)	$\{ x \mid \#\{ y \mid (x, y) \in (DPE)^{DP} \} \leq n \}$
DataExactCardinality(n DPE)	$\{ x \mid \#\{ y \mid (x, y) \in (DPE)^{DP} \} = n \}$
DataMinCardinality(n DPE DR)	$\{ x \mid \#\{ y \mid (x, y) \in (DPE)^{DP} \text{ and } y \in (DR)^{DT} \} \geq n \}$
DataMaxCardinality(n DPE DR)	$\{ x \mid \#\{ y \mid (x, y) \in (DPE)^{DP} \text{ and } y \in (DR)^{DT} \} \leq n \}$
DataExactCardinality(n DPE DR)	$\{ x \mid \#\{ y \mid (x, y) \in (DPE)^{DP} \text{ and } y \in (DR)^{DT} \} = n \}$
SubClassOf(CE_1 CE_2)	$(CE_1)^{C} \subseteq (CE_2)^{C}$
EquivalentClasses(CE_1 ... CE_n)	$(CE_j)^{C} = (CE_k)^{C}$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$
DisjointClasses(CE_1 ... CE_n)	$(CE_j)^{C} \cap (CE_k)^{C} = \emptyset$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$
DisjointUnion(C CE_1 ... CE_n)	$(C)^{C} = (CE_1)^{C} \cup \ldots \cup (CE_n)^{C}$ and $(CE_j)^{C} \cap (CE_k)^{C} = \emptyset$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$
SubObjectPropertyOf(OPE_1 OPE_2)	$(OPE_1)^{OP} \subseteq (OPE_2)^{OP}$
SubObjectPropertyOf(ObjectPropertyChain(OPE_1 ... OPE_n) OPE)	$\forall\, y_0, \ldots, y_n : (y_0, y_1) \in (OPE_1)^{OP}$ and ... and $(y_{n-1}, y_n) \in (OPE_n)^{OP}$ imply $(y_0, y_n) \in (OPE)^{OP}$
EquivalentObjectProperties(OPE_1 ... OPE_n)	$(OPE_j)^{OP} = (OPE_k)^{OP}$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$
DisjointObjectProperties(OPE_1 ... OPE_n)	$(OPE_j)^{OP} \cap (OPE_k)^{OP} = \emptyset$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$
ObjectPropertyDomain(OPE CE)	$\forall\, x, y : (x, y) \in (OPE)^{OP}$ implies $x \in (CE)^{C}$
ObjectPropertyRange(OPE CE)	$\forall\, x, y : (x, y) \in (OPE)^{OP}$ implies $y \in (CE)^{C}$
InverseObjectProperties(OPE_1 OPE_2)	$(OPE_1)^{OP} = \{ (x, y) \mid (y, x) \in (OPE_2)^{OP} \}$

FunctionalObjectProperty(OPE)	$\forall x, y_1, y_2$: $(x, y_1) \in (OPE)^{OP}$ and $(x, y_2) \in (OPE)^{OP}$ imply $y_1 = y_2$
InverseFunctionalObjectProperty(OPE)	$\forall x_1, x_2, y$: $(x_1, y) \in (OPE)^{OP}$ and $(x_2, y) \in (OPE)^{OP}$ imply $x_1 = x_2$
ReflexiveObjectProperty(OPE)	$\forall x : x \in \Delta_I$ implies $(x, x) \in (OPE)^{OP}$
IrreflexiveObjectProperty(OPE)	$\forall x : x \in \Delta_I$ implies $(x, x) \notin (OPE)^{OP}$
SymmetricObjectProperty(OPE)	$\forall x, y$: $(x, y) \in (OPE)^{OP}$ implies $(y, x) \in (OPE)^{OP}$
AsymmetricObjectProperty(OPE)	$\forall x, y$: $(x, y) \in (OPE)^{OP}$ implies $(y, x) \notin (OPE)^{OP}$
TransitiveObjectProperty(OPE)	$\forall x, y, z$: $(x, y) \in (OPE)^{OP}$ and $(y, z) \in (OPE)^{OP}$ imply $(x, z) \in (OPE)^{OP}$
SubDataPropertyOf(DPE_1 DPE_2)	$(DPE_1)^{DP} \subseteq (DPE_2)^{DP}$
EquivalentDataProperties(DPE_1 ... DPE_n)	$(DPE_j)^{DP} = (DPE_k)^{DP}$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$
DisjointDataProperties(DPE_1 ... DPE_n)	$(DPE_j)^{DP} \cap (DPE_k)^{DP} = \emptyset$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$
DataPropertyDomain(DPE CE)	$\forall x, y$: $(x, y) \in (DPE)^{DP}$ implies $x \in (CE)^{C}$
DataPropertyRange(DPE DR)	$\forall x, y$: $(x, y) \in (DPE)^{DP}$ implies $y \in (DR)^{DT}$
FunctionalDataProperty(DPE)	$\forall x, y_1, y_2$: $(x, y_1) \in (DPE)^{DP}$ and $(x, y_2) \in (DPE)^{DP}$ imply $y_1 = y_2$
SameIndividual(a_1 ... a_n)	$(a_j)^I = (a_k)^I$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$
DifferentIndividuals(a_1 ... a_n)	$(a_j)^I \neq (a_k)^I$ for each $1 \leq j \leq n$ and each $1 \leq k \leq n$ such that $j \neq k$
ClassAssertion(CE a)	$(a)^I \in (CE)^{C}$
ObjectPropertyAssertion(OPE a_1 a_2)	$((a_1)^I, (a_2)^I) \in (OPE)^{OP}$
NegativeObjectPropertyAssertion(OPE a_1 a_2)	$((a_1)^I, (a_2)^I) \notin (OPE)^{OP}$
DataPropertyAssertion(DPE a lt)	$((a)^I, (lt)^{LT}) \in (DPE)^{DP}$
NegativeDataPropertyAssertion(DPE a lt)	$((a)^I, (lt)^{LT}) \notin (DPE)^{DP}$

3

Σχετικές εργασίες

Στο κεφάλαιο αυτό παρουσιάζονται οι σημαντικότερες εργασίες που ασχολούνται με την ταξινόμηση (classification) των εννοιών (κλάσεων) μιας οντολογίας (Terminology Box – Tbox) και των οποίων οι υλοποιήσεις βασίζονται κατά κύριο λόγο στη μέθοδο της Δομικής Υπαγωγής (Structural Subsumption). Περιγράφονται διάφορες τεχνικές και καλύπτονται διαφορετικά εύρη εκφραστικότητας. Στην πρώτη ενότητα του κεφαλαίου παρουσιάζονται τα εργαλεία ταξινόμησης CEL classifier, CB reasoner και DB reasoner. Στην ενότητα δύο παρουσιάζονται μέθοδοι βασισμένες στην γλώσσα Datalog, στην ενότητα τρία συνδυαστικές μέθοδοι, ενώ στο τέλος του κεφαλαίου υπάρχει ένας συγκεντρωτικός πίνακας για την σύγκριση των συστημάτων. Επιχειρείται μια παρουσίαση των χαρακτηριστικών του κάθε συστήματος με απώτερο στόχο την αιτιολόγηση και αποτίμηση της προσέγγισης που εμείς ακολουθήσαμε. Συνεπώς η μελέτη αυτού του κεφαλαίου συνιστάται για την αξιολόγηση των όσων ακολουθούν στα επόμενα κεφάλαια.

3.1 Εργαλεία ταξινόμησης

Στην ενότητα αυτή παρουσιάζονται τα τρία σημαντικότερα συστήματα ταξινόμησης που βασίζονται στην μέθοδο της Δομικής Υπαγωγής, δηλαδή στον υπολογισμό όλων των σχέσεων υπαγωγής μεταξύ των ατομικών εννοιών της οντολογίας μέσω της επαναληπτικής εφαρμογής κανόνων στα αρχικά αξιώματα. Για το κάθε σύστημα παρατίθενται αρχικά τα ιδιαίτερα χαρακτηριστικά του, έπειτα οι λεπτομέρειες της υλοποίησής του και τέλος παρουσιάζονται πειραματικά αποτελέσματα.

3.1.1 CEL classifier

Ο ταξινομητής CEL χρησιμοποιείται για την ταξινόμηση οντολογιών εκφραστικότητας $\mathcal{EL}^+$. Η γλώσσα προγραμματισμού που χρησιμοποιείται είναι η Lisp και η διαδικασία λειτουργεί στην μνήμη (memory-based).

Στην εκφραστικότητα $\mathcal{EL}^+$ οι περιγραφές των εννοιών επιτυγχάνονται με την βοήθεια ενός συνόλου κατασκευαστών (constructors), ξεκινώντας με ένα σύνολο N_C ονομάτων εννοιών και ένα σύνολο N_R ονομάτων ρόλων. Οι περιγραφές $\mathcal{EL}^+$ εννοιών γίνονται με τους κατασκευαστές που παρατηρούνται στο πάνω μέρος του πίνακα του Σχήματος 3.1. Μια $\mathcal{EL}^+$ οντολογία είναι ένα σύνολο από υπαγωγές γενικών εννοιών (general concept inclusions – GCIs) και υπαγωγές ρόλων (role inclusions - RIs), των οποίων η σύνταξη φαίνεται στο κάτω μέρος του πίνακα του Σχήματος 3.1.

Name	Syntax	Semantics
top	$\top$	Δ^I
conjuction	$C \sqcap D$	$C^I \cap D^I$
existential restriction	$\exists r.C$	$\{x \in \Delta^I \mid \exists y \in \Delta^I : (x,y) \in r^I \wedge y \in C^I$
general concept inclusion	$C \sqsubseteq D$	$C^I \subseteq D^I$
role inclusion	$r_1 \circ \cdots\cdots r_n \sqsubseteq s$	$r_1^I \circ \cdots\cdots r_n^I \subseteq s^I$

Σχήμα 3.1. Σύνταξη και Σημασιολογία της $\mathcal{EL}^+$

Η σημασιολογία της $\mathcal{EL}^+$ ορίζεται μέσω των ερμηνειών (interpretations) $I=(\Delta^I, \cdot^I)$, όπου το πεδίο Δ^I είναι ένα μη-κενό σύνολο από άτομα, και η συνάρτηση ερμηνείας $\cdot^I$ αντιστοιχεί κάθε όνομα έννοιας $A \in N_C$ σε ένα υποσύνολο A^I του Δ^I και κάθε όνομα ρόλου $r \in N_R$ σε μια δυαδική σχέση r^I στο Δ^I. Μια ερμηνεία I είναι ένα μοντέλο της οντολογίας O αν για κάθε υπαγωγή στην O, οι συνθήκες που δίνονται στην στήλη σημασιολογία του Σχήματος 3.1 ισχύουν.

Δίνονται οι παρακάτω ορισμοί για την κατανόηση της $\mathcal{EL}^+$:

- Ορισμός έννοιας (concept definition) $A \equiv C$, με A όνομα έννοιας, είναι μια γενίκευση που αντιστοιχεί στις δύο υπαγωγές γενικών εννοιών $A \sqsubseteq C$ και $C \sqsubseteq A$. Το C περιγράφει ουσιαστικά τις ικανές και αναγκαίες συνθήκες για να είναι ένα στιγμιότυπο του A.
- Υπαγωγές γενικών εννοιών της μορφής $A \subseteq C$, με το A όνομα έννοιας, ονομάζονται ορισμοί πρωτογενών εννοιών (primitive concept definition). Δίνουν μόνο αναγκαίες (αλλά όχι ικανές) συνθήκες για ένα στιγμιότυπο του A.
- Γενικευμένο Tbox (general Tbox) καλείται ένα μετρήσιμο σύνολο από υπαγωγές γενικών εννοιών.
- Tbox καλείται ένα μετρήσιμο σύνολο από ορισμούς (πιθανώς πρωτογενών) εννοιών με μοναδικές τις αριστερές πλευρές τους.
- Πρωτογενές Tbox (primitive Tbox) καλείται αυτό που περιέχει μόνο ορισμούς πρωτογενών εννοιών.
- Ακυκλικό Tbox (acyclic Tbox) λέγεται αυτό για το οποίο δεν υπάρχουν ονόματα εννοιών $A_0, \ldots\ldots, A_{n-1}$ τέτοια ώστε το $A_{(i+1) \bmod n}$ να εμφανίζεται στην δεξιά πλευρά του ορισμού έννοιας του A_i, για κάθε $i<n$.
- Ιεραρχίες ρόλων (role hierarchies) καλούνται υπαγωγές ρόλων της μορφής $r \subseteq s$.
- Η μεταβατικότητα (transitivity) ενός ρόλου μπορεί να εκφραστεί γράφοντας $r \circ r \subseteq r$.
- Τέλος, οι υπαγωγές ρόλων (RIs) μπορούν να εκφράσουν κανόνες δεξιάς-ταυτότητας (right-identity rules) $r \circ s \subseteq r$.

Για την ταξινόμηση της οντολογίας το σύστημα την μετατρέπει πρώτα σε κανονική μορφή (normal form), που απαιτεί όλες οι υπαγωγές γενικών εννοιών και οι υπαγωγές ρόλων να είναι μιας εκ των μορφών που παρουσιάζονται στην αριστερή πλευρά του Σχήματος 3.2. Με την δημιουργία νέων ονομάτων εννοιών και ρόλων και ξαναγράφοντας την οντολογία σύμφωνα με κάποιους κανόνες, κάθε οντολογία O μπορεί να μεταμορφωθεί σε μια κανονικοποιημένη, τέτοια ώστε να διατηρούνται όλες οι υπαγωγές μεταξύ ονομάτων εννοιών. Η κανονικοποίηση μπορεί να επιτευχθεί σε γραμμικό χρόνο, παράγοντας μια οντολογία με μέγεθος γραμμικά ανάλογο με το μέγεθος της αρχικής οντολογίας.

Normal Form	Completion Rules
$A \sqsubseteq B$	**CR1** If $A \in S(X)$, $A \sqsubseteq B \in O$, and $B \notin S(X)$ then $S(X) \coloneqq S(X) \cup \{B\}$
$A_1 \sqcap A_2 \sqsubseteq B$	**CR2** If $A_1, A_2 \in S(X)$, $A_1 \sqcap A_2 \sqsubseteq B \in O$, and $B \notin S(X)$ then $S(X) \coloneqq S(X) \cup \{B\}$
$A \sqsubseteq \exists r.B$	**CR3** If $A \in S(X)$, $A \sqsubseteq \exists r.B \in O$, and $(X,B) \notin R(r)$ then $R(r) \coloneqq R(r) \cup \{(X,B)\}$
$\exists r.A \sqsubseteq B$	**CR4** If $(X,Y) \in R(r)$, $A \in S(Y)$, $\exists r.A \sqsubseteq B \in O$, and $B \notin S(X)$ then $S(X) \coloneqq S(X) \cup \{B\}$
$r \sqsubseteq s$	**CR5** If $(X,Y) \in R(r)$, $r \sqsubseteq s \in O$, and $(X,Y) \notin R(s)$ then $R(s) \coloneqq R(s) \cup \{(X,Y)\}$
$r \circ s \sqsubseteq t$	**CR6** If $(X,Y) \in R(r)$, $(Y,Z) \in R(s)$, $r \circ s \sqsubseteq t \in O$, and $(X,Z) \notin R(t)$ then $R(t) \coloneqq R(t) \cup \{(X,Z)\}$

Σχήμα 3.2. Κανονική Μορφή και Κανόνες Ολοκλήρωσης

Στη συνέχεια θεωρείται ότι η οντολογία O που δέχεται ο CEL ως είσοδο είναι σε κανονικοποιημένη μορφή. Έστω ότι NC_O είναι το σύνολο των ονομάτων εννοιών που εμφανίζονται στην O συμπεριλαμβανομένης και της καθολικής έννοιας T, και NR_O είναι το σύνολο των ονομάτων ρόλων που εμφανίζονται στην O. Το σύστημα υπολογίζει:

- μία αντιστοίχηση S που αναθέτει σε κάθε στοιχείο του NC_O ένα υποσύνολο του NC_O.
- μία αντιστοίχηση R που αναθέτει σε κάθε στοιχείο του NR_O μία δυαδική σχέση στο NC_O .

Η λογική είναι ότι οι αντιστοιχήσεις αυτές κάνουν τις υπονοούμενες (implicit) σχέσεις ρητές (explicit) με την έννοια ότι $B \in S(A)$ υπονοεί $A \sqsubseteq_O B$, και $(A,B) \in R(r)$ υπονοεί $A \sqsubseteq \exists r.B$. Οι αντιστοιχήσεις αρχικοποιούνται θέτοντας $S(A) \coloneqq \{A,T\}$ για κάθε $A \in NC_O$, και $R(r) := \emptyset$ για κάθε $r \in NR_O$. Στη συνέχεια τα σύνολα S(A) και R(r) επεκτείνονται μέσω της εφαρμογής των κανόνων ολοκλήρωσης που παρουσιάζονται στο δεξιό μέρος του Σχήματος 3.2, μέχρι να μην μπορεί να εφαρμοστεί κανένας περαιτέρω. Η διαδικασία έχει αποδειχθεί ότι είναι ορθή και πλήρης, δηλαδή μετά τον τερματισμό έχουμε ότι $B \in S(A)$ εάν και μόνο εάν $A \sqsubseteq_O B$, για όλα τα ονόματα εννοιών A,B που εμφανίζονται στην οντολογία O. Η διαδικασία τερματίζει σε χρόνο πολυωνυμικό σε σχέση με το μέγεθος της οντολογίας που δόθηκε ως είσοδος.
Για εποπτικούς λόγους παρουσιάζονται οι κανόνες ολοκλήρωσης του Σχήματος 3.2 σε ποιο κατανοητή μορφή στο Σχήμα 3.3.

$$\mathbf{CR1}\ \frac{X \sqsubseteq A}{X \sqsubseteq B} : A \sqsubseteq B \in O \qquad \boldsymbol{CR2}\ \frac{X \sqsubseteq A_1\ X \sqsubseteq A_2}{X \sqsubseteq B} :\ A_1 \sqcap A_2 \sqsubseteq B \in O$$

$$\mathbf{CR3}\ \frac{X \sqsubseteq A}{X \sqsubseteq \exists r.B} : A \sqsubseteq \exists r.B \in O \qquad \boldsymbol{CR4}\ \frac{X \sqsubseteq \exists r.Y\ \ Y \sqsubseteq A}{X \sqsubseteq B} : \exists r.A \sqsubseteq B \in O$$

$$\boldsymbol{CR5}\ \frac{X \sqsubseteq \exists r.Y}{X \sqsubseteq \exists s.Y} : r \sqsubseteq s \in O \qquad \boldsymbol{CR6}\ \frac{X \sqsubseteq \exists r.Y\ \ Y \sqsubseteq \exists s.Z}{X \sqsubseteq \exists t.Z} : \mathrm{r} \circ \mathrm{s} \sqsubseteq \mathrm{t} \in \mathrm{O}$$

Σχήμα 3.3. Κανόνες Ολοκλήρωσης

Για τον καθορισμό της σειράς εφαρμογής των κανόνων ολοκλήρωσης χρησιμοποιείται μια σειρά από ουρές, μια για κάθε έννοια που εμφανίζεται στην οντολογία. Ουσιαστικά οι ουρές αυτές περιέχουν τις αλλαγές στην δομή των δεδομένων (δηλαδή στα σύνολα S(A) και R(r)) που δεν έχουν γίνει ακόμα. Οι πιθανές εγγραφές των ουρών είναι της μορφής:

$$\mathrm{B},\ \mathrm{B} \rightarrow \mathrm{B'},\ \text{και}\ \exists \mathrm{r.B}$$

με B και B' ονόματα εννοιών, και r όνομα ρόλου. Η εγγραφή B∈queue(A) σημαίνει ότι το όνομα έννοιας B πρέπει να προστεθεί στο S(A). Παρόμοια, $\mathrm{B} \rightarrow \mathrm{B'} \in$queue(A) σημαίνει ότι το B' πρέπει να προστεθεί στο S(A) εάν το S(A) περιέχει ήδη το B, και ∃r.B∈queue(A) σημαίνει ότι το (A,B) πρέπει να προστεθεί στο R(r). Το γεγονός ότι μία τέτοια πρόσθεση θέτει σε εφαρμογή άλλους κανόνες, θα αντιμετωπίζεται με την κατάλληλη επέκταση των ουρών μετά την πρόσθεση.

Για την κατανόηση του πως ακριβώς διαχειρίζονται οι ουρές, θα πρέπει να θεωρηθεί η (κανονικοποιημένη) οντολογία O που δίνεται ως είσοδος, ως μία αντιστοίχηση O' από έννοιες σε ένα σύνολο από εγγραφές ουρών ως εξής: για κάθε όνομα έννοιας A (συμπεριλαμβανομένης και της περίπτωσης A=T), O'(A) είναι το ελάχιστο σύνολο από εγγραφές ουρών τέτοιο ώστε

- αν A⊑B∈O, τότε B∈O'(A)
- αν $\mathrm{A} \sqcap \mathrm{A'} \sqsubseteq \mathrm{B} \in \mathrm{O}$ ή $\mathrm{A'} \sqcap \mathrm{A} \subseteq \mathrm{B} \in \mathrm{O}$, τότε $\mathrm{A'} \rightarrow \mathrm{B} \in \mathrm{O'(A)}$
- αν A⊑∃r.B∈O, τότε ∃r.B∈O'(A)

Αντίστοιχα, για κάθε ∃r.A, O'(∃r.A) είναι το ελάχιστο σύνολο από εγγραφές ουρών τέτοιο ώστε, αν ∃r.A⊑B∈O, τότε B∈O'(∃r.A).

Επειδή τα σύνολα S(A) έχουν αρχικοποιηθεί με {A.T}, οι ουρές queue(A) αρχικοποιούνται με O'(A)∪ O'(T) , δηλαδή προστίθενται στις ουρές οι άμεσες συνέπειες του ότι είναι στιγμιότυπα των A και T. Στη συνέχεια, παίρνονται επανειλημμένα (και άρα αφαιρούνται) εγγραφές από τις μη-κενές ουρές και επεξεργάζονται. Η επεξεργασία των ουρών συνεχίζεται έως ότου όλες οι ουρές να αδειάσουν. Σε κανένα σημείο της διαδικασίας δεν χρειάζεται να γίνει ο οποιοσδήποτε έλεγχος για το ποιος κανόνας είναι εφαρμόσιμος ή όχι.

Όπως προαναφέρθηκε, ο CEL διαβάζει την οντολογία, μετατρέπει κάθε ορισμό έννοιας σε ένα ζευγάρι από υπαγωγές γενικών εννοιών, κανονικοποιεί την οντολογία και αρχίζει την εφαρμογή της διαδικασίας που περιγράφθηκε.

Διαπιστώθηκε όμως κατά την διάρκεια των πειραμάτων ότι δεν μπορούσε να ταξινομήσει τις O^{SNOMED} και $O^{GO}{}_{def}$ (οι δύο αυτές οντολογίες περιγράφονται πιο κάτω), διότι οι δομές γίνονταν πολύ μεγάλες εξαιτίας των πάρα πολλών υπολογίσιμων σχέσεων υπαγωγής. Στην περίπτωση της $O^{GO}{}_{def}$ οι πολλές αυτές σχέσεις προέρχονταν από τις πολλές ισοδύναμες έννοιες (κάτι που δημιουργεί πρόβλημα αφού n ισοδύναμες έννοιες δίνουν n^2 σχέσεις υπαγωγής). Επιπλέον αν υπάρχει υπαγωγή μεταξύ δύο ισοδύναμων κλάσεων μεγέθους n και m αντίστοιχα, τότε πρέπει να αποθηκευτούν $n \cdot m$ σχέσεις υπαγωγής. Στην περίπτωση της O^{SNOMED}, οι υπερβολικά πολλές σχέσεις υπαγωγής εμφανίζονται λόγω των πολλών νέων εννοιών που εισάγονται κατά την διαδικασία της κανονικοποίησης. Οι παρατηρήσεις αυτές οδήγησαν στις εξής βελτιώσεις:

Αναγνώριση Συνωνύμων (Synonyms Identification). Συνώνυμα είναι τα ονόματα εννοιών A,B που έχουν δηλωθεί ρητά ως ισοδύναμα από έναν ορισμό έννοιας $A \equiv B$. Ο αλγόριθμος κρατάει μόνο έναν "αντιπρόσωπο" από τα συνώνυμα. Τα υπόλοιπα αποθηκεύονται έτσι ώστε να μπορούν να χρησιμοποιηθούν σε απαντήσεις σε ερωτήσεις υπαγωγής. Η αφαίρεση των συνωνύμων μειώνει κατά το μισό τα ονόματα εννοιών στην $O^{GO}{}_{def}$ και επιτρέπει την ταξινόμησή της από τον CEL.

Βελτιωμένη κανονική μορφή. Σκοπός είναι να μειωθούν τα επιπλέον ονόματα εννοιών που εισάγονται κατά την κανονικοποίηση. Αυτό επιτυγχάνεται με δύο τρόπους:
Πρώτον, σκοπός των νέων ονομάτων εννοιών που εισάγονται είναι το να αντικαταστήσουν σύνθετες περιγραφές εννοιών στις υπαγωγές γενικών εννοιών (GCIs). Η γενική ιδέα είναι ότι εισάγεται ένα όνομα έννοιας A σαν αντικαταστάτης της σύνθετης έννοιας C, και προστίθεται στην οντολογία το $A \sqsubseteq C$ ή το $C \sqsubseteq A$ ανάλογα με το αν το C ήταν στο αριστερό ή στο δεξί μέρος ενός GCI. Στην βελτιωμένη κανονικοποίηση οι αντίστοιχες με την C έννοιες παρακολουθούνται και έτσι αποφεύγεται η εισαγωγή πολλαπλών νέων εννοιών για την ίδια περιγραφή έννοιας.
Δεύτερον, στην βελτιωμένη κανονικοποίηση χρησιμοποιούνται τομές n-βαθμού στην αριστερή πλευρά των GCIs, αποφεύγοντας έτσι την εισαγωγή n-1 εννοιών για κάθε τέτοια τομή. Ο κανόνας ολοκλήρωσης CR2 αλλάζει ανάλογα:

$$\boldsymbol{CR2}\ If\ A_1, \ldots, A_n \in S(X), A_1 \sqcap \ldots . \sqcap A_n \sqsubseteq B \in O, and\ B \notin S(X)$$
$$then\ S(X) \coloneqq S(X) \cup \{B\}$$

Επιπλέον, εγγραφές των ουρών της μορφής $B \rightarrow B'$, αντικαθίστανται με τις εγγραφές B_1, ... , $B_n \rightarrow B'$. Οι αλλαγές αυτές μειώνουν τον αριθμό των νέων εισαγόμενων εννοιών κατά την διαδικασία κανονικοποίησης της O^{SNOMED} από 401.830 σε 114.658, επιτρέποντας την ταξινόμησή της.

Παρουσιάζονται στην συνέχεια κάποια πειράματα που έγιναν για να συγκριθεί ο CEL με τις μοντέρνες βασισμένες σε tableau μηχανές συλλογιστικής ανάλυσης. Τα πειράματα αυτά είναι βασισμένα σε τρεις μεγάλες βιο-ιατρικές οντολογίες: την Gene

Ontology (Go), την βάση ιατρικής γνώσης Galen (GALEN) και την Systematized Nomenclature of Medicine (SNOMED).

GO. Η Gene Ontology προσφέρει ένα λεξιλόγιο για την περιγραφή των ιδιοτήτων των γονιδίων και των προϊόντων γονιδίων. Περιέχει 16.803 έννοιες και ένα μεταβατικό ρόλο ("part-of"). Το Σχήμα 3.4 δείχνει ένα παράδειγμα περιγραφής μιας έννοιας στην GO.

[Term]
Id: GO : 0000133
name : polarisome
namespace : cellular_component
def : "Protein complex playing a role in determining cell polarity"
is_a : GO : 0043234 ! protein complex
relationship : part_of GO : 0005938 ! cell cortex
relationship : part_of GO : 0030427 ! site of polarized growth

Σχήμα 3.4. Περιγραφή της έννοιας Polarisome στην GO.

Η πιο απλή μετάφραση ορισμών εννοιών της GO σε μια EL^+ οντολογία είναι με την χρήση ορισμών πρωτογενών εννοιών. Έτσι η παραπάνω έννοια GO θα οριζότανε ως:

$$GO0000133 \sqsubseteq GO0043234 \sqcap \exists part\_of.GO0005938 \sqcap \exists part\_of.GO0030427$$

Η μετάφραση αυτή δίνει την O^{GO}_{prim} , ένα ακυκλικό, πρωτογενές TBox με μία υπαγωγή ρόλου. Παρόλα αυτά η O^{GO}_{prim} δεν είναι κατάλληλη για έλεγχο συστημάτων περιγραφικής λογικής, διότι είναι πολύ απλή. Ο υπολογισμός της ταξινόμησης θα σήμαινε απλά τον υπολογισμό του μεταβατικού κλεισίματος (transitive closure) των ρητών σχέσεων υπαγωγής.

Για την απόκτηση μιας οντολογίας βασισμένης στην GO και πιο δύσκολης στην ταξινόμηση, απαιτείται η χρήση ορισμών μη-πρωτογενών εννοιών. Έτσι το παραπάνω παράδειγμα γράφεται:

$$GO0000133 \equiv GO0043234 \sqcap \exists part\_of.GO0005938 \sqcap \exists part\_of.GO0030427$$

Το αποτέλεσμα είναι η O^{GO}_{def} , ένα ακυκλικό TBox με την ίδια υπαγωγή ρόλου όπως παραπάνω. Παρόλα αυτά η οντολογία που παράγεται είναι αρκετά ασυνήθιστη, αφού η υπόθεση ότι οι ορισμοί εννοιών παρέχουν και επαρκείς συνθήκες, πολλές έννοιες καταλήγουν ισοδύναμες, δηλαδή υπάγονται η μια στην άλλη.

GALEN. Σκοπός της GALEN είναι η διανομή και επαναχρησιμοποίηση ιατρικών δεδομένων. Περιέχει ιεραρχίες ρόλων, μεταβατικούς, λειτουργικούς και αντίστροφους ρόλους. Εφόσον η $\mathcal{EL}^+$ και ο CEL δεν υποστηρίζουν αντίστροφους και λειτουργικούς ρόλους, οι αντίστροφοι έχουν αφαιρεθεί και οι λειτουργικοί αντιμετωπίζονται ως απλοί. Η οντολογία που προκύπτει ονομάζεται O^{GALEN} και

περιέχει 1.214 υπαγωγές γενικών εννοιών, όπως και 2.041 ορισμούς πρωτογενών και 699 ορισμούς μη-πρωτογενών εννοιών. Αναφέρεται σε 413 ρόλους, εκ των οποίων οι 26 είναι μεταβατικοί, ενώ υπάρχουν επίσης 412 αξιώματα ιεραρχίας ρόλων.

SNOMED. Η SNOMED είναι μια τυποποίηση της ιατρικής ορολογίας που χρησιμοποιείται στις Ηνωμένες Πολιτείες και τη Μεγάλη Βρετανία. Περιέχει 379.691 έννοιες (38.719 ορισμούς εννοιών και 340.972 ορισμούς πρωτογενών εννοιών) και 52 ονόματα ρόλων. Δεν περιέχει μεταβατικούς ρόλους, αλλά 11 ιεραρχίες ρόλων και ένα κανόνα δεξιάς-ταυτότητας (right-identity rule) της μορφής $r \circ s \sqsubseteq r$. Εδώ χρησιμοποιείται μία πιο περιορισμένη έκδοση της O^{SNOMED} που περιέχει μόνο τους ορισμούς μη-πρωτογενών εννοιών και καλείται O^{SNOMED}_{core}.

Στο Σχήμα 3.5 συγκρίνεται ο CEL με δύο από τα πιο γνωστά βασισμένα σε tableau συστήματα συλλογιστικής ανάλυσης, τους FaCT (v2.32.17) και RACER (v1.7.6).

	O_{prim}^{GO}	O_{def}^{GO}	O^{GALEN}	O_{core}^{SNOMED}	O^{SNOMED}
No. of CDefs.	0	16.803	699	38.719	38.719
No. of PCDefs.	16.803	0	2.041	0	340.972
No. of GCIs	0	0	1.214	0	0
No. of role axioms	1	1	438	0	12
$\|CN_O\|$	16.806	16.806	2.740	53.234	379.691
$\|RN_O\|$	1	1	413	52	52
CEL	25	10.833	16	1.360	12.772
FaCT	117	11	19	unattainable	unattainable
RACER	36	unattainable	25	84.917	unattainable

Σχήμα 3.5. Αποτελέσματα Αξιολόγησης

3.1.2 DB reasoner

Η μηχανή συλλογιστικής ανάλυσης DB (DB reasoner) χρησιμοποιείται για την ταξινόμηση οντολογιών εκφραστικότητας $\mathcal{ELH}$. Η γλώσσα προγραμματισμού που χρησιμοποιείται είναι η ML και η διαδικασία λειτουργεί στον σκληρό δίσκο (disk-based).

Στον παρακάτω πίνακα παρουσιάζεται συνοπτικά η σύνταξη και η σημασιολογία της ELH. Το σύνολο των εννοιών της $\mathcal{ELH}$ ορίζεται χρησιμοποιώντας τους δύο κατασκευαστές που δίνονται στον πίνακα, την τομή (conjuction) και τον υπαρξιακό περιορισμό (existential restriction).

Name	Syntax	Semantics
Concepts:		
atomic concept	A	A^I (given)
top concept	$\top$	Δ^I
conjuction	$C \sqcap D$	$C^I \cap D^I$
existential restriction	$\exists r.C$	$\{x \in \Delta^I \mid \exists y \in \Delta^I : (x,y) \in r^I \wedge y \in C^I$
Axioms:		
concept inclusion	$C \sqsubseteq D$	$C^I \subseteq D^I$
role inclusion	$r \sqsubseteq s$	$r^I \subseteq s^I$

Πίνακας 3.1. Σύνταξη και Σημασιολογία της $\mathcal{EL}\mathcal{H}$

Το σύστημα αφού φορτώσει την οντολογία, σαν πρώτο βήμα απονείμει έναν ακέραιο αριθμό σε κάθε έννοια και ρόλο που παρουσιάζεται στην οντολογία. Αυτούς τους αριθμούς-ταυτότητες (ids) θα τους χρησιμοποιήσει για τον υπολογισμό των κανονικοποιημένων αξιωμάτων. Για την αναπαράσταση όλης της πληροφορίας για τις έννοιες είναι αρκετή η αποθήκευση, για κάθε έννοια, του κατασκευαστή της μαζί με την ταυτότητα της κάθε άμεσης υπο-έννοιας της. Έτσι όλη η πληροφορία για τις έννοιες και τους ρόλους της οντολογίας μπορεί να αναπαρασταθεί στην βάση με την χρήση τεσσάρων πινάκων (Σχήμα 3.6) : ένα για τις ατομικές έννοιες, ένα για τους ατομικούς ρόλους, ένα για τις τομές και ένα για τους υπαρξιακούς περιορισμούς.

at_conc	
name	id
MuscularOrgan	1
Organ	2
MuscularSystem	4
Heart	6
CirculatorySystem	10

at_role	
name	id
isPartOf	3
BelongsTo	8

conj					
A	⊓	B	pos	neg	id
2		5	√	√	1
4		10	√		11
2		12	√		13

exis					
∃	r.	B	pos	neg	id
	3	4	√	√	5
	8	11	√		12

Σχήμα 3.6. Παράδειγμα αποθήκευσης $\mathcal{EL}\mathcal{H}$ εννοιών και ρόλων σε βάση δεδομένων

Εκτός από τον αριθμό-ταυτότητα κάθε σύνθετης έννοιας, αποθηκεύεται επίσης το αν η έννοια εμφανίζεται θετικά ή αρνητικά στην οντολογία. Οι πολικότητες

των εννοιών χρησιμοποιούνται για τον σχηματισμό της κανονικοποιημένης οντολογίας, για την αποθήκευση της οποίας χρησιμοποιούνται πέντε πίνακες στην βάση δεδομένων: ένας για τις σχέσεις υπαγωγής μεταξύ εννοιών, ένας για τις σχέσεις υπαγωγής μεταξύ ρόλων, ένας για τις τομές που εμφανίζονται αρνητικά στην οντολογία, ένας για τους υπαρξιακούς περιορισμούς που εμφανίζονται αρνητικά στην οντολογία και ένας για τους υπαρξιακούς περιορισμούς που εμφανίζονται θετικά στην οντολογία. Εξαλείφονται έτσι οι σύνθετες έννοιες (complex concepts) χρησιμοποιώντας βοηθητικές ατομικές έννοιες και αξιώματα. Η οντολογία που προκύπτει θα περιέχει ως γνωστόν μόνο αξιώματα της μορφής :

$$A \sqsubseteq B,\ A \sqcap B \sqsubseteq C,\ A \sqsubseteq \exists r.B,\ \exists s.B \sqsubseteq C,\ r \sqsubseteq s$$

όπου τα A,B,C είναι ατομικές έννοιες και τα r,s είναι ατομικοί ρόλοι. Για παράδειγμα τα αποθηκευμένα αξιώματα του Σχήματος 3.6 δίνονται στο Σχήμα 3.7 αποθηκευμένα σε κανονική μορφή.

ax_c_incl		
A	⊑	B
1		2
1		5
11		4
11		10
13		2
13		12
6		13

ax_r_incl		
r	⊑	s
8		3

ax_conj				
A	⊓	B	⊑	C
2		5		1

ax_exis_pos				
A	⊑	∃	r.	B
5			3	4
12			8	11

ax_exis_neg				
∃	s.	B	⊑	C
	3	4		5

Σχήμα 3.7. Παράδειγμα αποθήκευσης $\mathcal{ELH}$ εννοιών και ρόλων σε βάση δεδομένων μετά από κανονικοποίηση

Επειδή το σύστημα DB σχεδιάστηκε για να διαχειρίζεται μεγάλου όγκου οντολογίες δεν είναι δυνατόν να φορτώνεται όλη η οντολογία στην μνήμη. Έτσι όλοι οι πίνακες πρέπει να κατασκευάζονται την στιγμή που διαβάζονται τα αξιώματα, κάτι που οδηγεί στο πρόβλημα ανάθεσης σε μια έννοια που έχει προσπελαστεί σε προηγούμενο αξίωμα του ίδιου αριθμού-ταυτότητα που της είχε ανατεθεί τότε. Όμως η συνεχής προσπέλαση της βάσης για την επίλυση αυτού του προβλήματος δεν είναι εφικτή, διότι οδηγεί σε πολύ μεγάλους χρόνους. Έτσι χρησιμοποιούνται προσωρινοί πίνακες στην μνήμη για την αποθήκευση των πρόσφατα εισαχθέντων εννοιών. Όταν μια έννοια εισάγεται στην βάση της δίνεται ένας αριθμός-ταυτότητα είτε έχει εισαχθεί στην βάση νωρίτερα είτε όχι, και αργότερα επαναφέρεται η μοναδικότητα μέσω μιας σειράς SQL ερωτημάτων.

Για την ταξινόμηση της κανονικοποιημένης $\mathcal{ELH}$ οντολογίας η διαδικασία εφαρμόζει τους κανόνες ολοκλήρωσης που παρουσιάζονται στο Σχήμα 3.8. Οι κανόνες αυτοί παράγουν αξιώματα της μορφής A⊑B και C⊑∃r.D που είναι λογικές συνέπειες των αξιωμάτων της οντολογίας O. Οι κανόνες IR1 και IR2 παράγουν τετριμμένα αξιώματα για $A \in N_C \cap sub(O)$. Οι εναπομείναντες εφαρμόζονται σε ήδη παραχθείσα αξιώματα και χρησιμοποιούν τα κανονικοποιημένα αξιώματα ως βοηθητικές συνθήκες. Οι κανόνες ολοκλήρωσης εφαρμόζονται έως ότου να μην μπορεί να παραχθεί κανένα νέο αξίωμα. Έχει αποδειχθεί ότι οι κανόνες IR1-CR5 είναι ορθοί και πλήρεις, δηλαδή μία σχέση υπαγωγής A⊑B περιέχεται από την O, μόνο εάν παράγεται από αυτούς τους κανόνες.

$$\boldsymbol{IR1}\ \frac{}{A \sqsubseteq A} \qquad \boldsymbol{IR2}\ \frac{}{A \sqsubseteq \top} \qquad \mathbf{CR1}\ \frac{A \sqsubseteq B}{A \sqsubseteq C} : B \sqsubseteq C \in O$$

$$\boldsymbol{CR2}\ \frac{A \sqsubseteq B \quad A \sqsubseteq C}{A \sqsubseteq D} : B \sqcap C \sqsubseteq D \in O \qquad \mathbf{CR3}\ \frac{A \sqsubseteq B}{A \sqsubseteq \exists r.C} : B \sqsubseteq \exists r.C \in O$$

$$\boldsymbol{CR4}\ \frac{A \sqsubseteq \exists r.B}{A \sqsubseteq \exists s.B} : r \sqsubseteq s \in O \qquad \boldsymbol{CR5}\ \frac{A \sqsubseteq \exists s.B \quad B \sqsubseteq C}{A \sqsubseteq D} : \exists s.C \sqsubseteq D \in O$$

Σχήμα 3.8. Οι Κανόνες Ολοκλήρωσης για την $\mathcal{ELH}$

Οι κανόνες ολοκλήρωσης του Σχήματος 3.8 εφαρμόζονται με την σειρά προτεραιότητας που αναγράφονται, δηλαδή εφαρμόζεται εξαντλητικά ο κανόνας CR1 πριν εφαρμοστεί ο κανόνας CR2 και εκτελείται κλείσιμο (closure) κατά τους CR1 και CR2 πριν εκτελεστούν οι εναπομείναντες κανόνες CR3 – CR5. Το κλείσιμο κατά τους κανόνες CR1 και CR2 της μορφής A⊑B για ένα συγκεκριμένο A μπορεί να υπολογιστεί ανεξάρτητα από άλλες υπαγωγές, καθώς οι προϋποθέσεις και τα συμπεράσματα αυτών των κανόνων μοιράζονται την ίδια έννοια A. Χρησιμοποιείται ένας προσωρινός πίνακας στην μνήμη για τον υπολογισμό του κλεισίματος για ένα συγκεκριμένο αριθμό από έννοιες A και έπειτα εισάγεται η ένωση των αποτελεσμάτων στον κυρίως πίνακα στην μνήμη.

Επειδή η άμεση εκτέλεση των CR3 - CR5 απαιτεί την εκτέλεση πολλών περίπλοκων ενώσεων (join) στην βάση, οι κανόνες αυτοί συνδυάζονται στον παρακάτω συμπερασματικό κανόνα με πολλαπλές προϋποθέσεις:

$$\frac{A \sqsubseteq B}{\exists r.A \sqsubseteq \exists s.B} : r \sqsubseteq s \in O, \exists r.A \in sub^{+}(O), \exists sub^{-}(O)$$

Ο κανόνας αυτός παράγει καινούργιες σχέσεις υπαγωγής μεταξύ υπαρξιακά περιορισμένων εννοιών στην οντολογία χρησιμοποιώντας σχέσεις υπαγωγής που έχουν παραχθεί σε προηγούμενα στάδια. Η κεντρική ιδέα είναι η συνεχής παραγωγή καινούργιων σχέσεων υπαγωγής, η προσθήκη τους στους πίνακες και η εκτέλεση αυξητικού κλεισίματος (incremental closure) κατά τους κανόνες CR1 και CR2.

Ο DB δεν παράγει ως έξοδο το σύνολο όλων των σχέσεων υπαγωγής μεταξύ όλων των υπο-εννοιών μιας οντολογίας, αλλά μία ταξονομία που περιέχει μόνο άμεσες σχέσεις υπαγωγής μεταξύ ατομικών εννοιών. Για τον υπολογισμό της

ταξονομίας θα πρέπει να μειωθεί μεταβατικά (transitively reduced) το σύνολο των σχέσεων υπαγωγής μεταξύ ατομικών εννοιών. Η μεταβατική μείωση μιας μεταβατικής σχέσης υπαγωγής μπορεί να επιτευχθεί με την εφαρμογή ενός βήματος μεταβατικού κλεισίματος και με την αναγνώριση των παραγόμενων σχέσεων ως «μη άμεσων», βρίσκοντας έτσι το εναπομείναν σύνολο των «άμεσων» σχέσεων υπαγωγής. Παρόλο που η διαδικασία αυτή μπορεί εύκολα να επιτευχθεί στις βάσεις δεδομένων με μόνο μία ένωση (join), αυτή η προσέγγιση δεν έχει καλή απόδοση διότι απαιτεί ένα μεγάλο αριθμό από ενημερώσεις (updates) στον δίσκο για την ενημέρωση των σχέσεων υπαγωγής ως μη άμεσων. Αντίθετα ο αριθμός των άμεσων σχέσεων υπαγωγής είναι πολύ μικρότερος. Για την επίλυση αυτού του προβλήματος χρησιμοποιείται ένας πίνακας στην μνήμη. Στον πίνακα αυτόν εισάγονται επανειλημμένα όλες οι σχέσεις υπαγωγής $A \sqsubseteq X$ για έναν δεσμευμένο αριθμό ατομικών εννοιών A, εκτελείται μεταβατική μείωση σύμφωνα με την μέθοδο που περιγράφεται παραπάνω και εξάγονται οι άμεσες σχέσεις υπαγωγής στην ταξονομία στον δίσκο.

Για την αποτίμηση της λειτουργίας της μηχανής συλλογιστικής ανάλυσης DB και για την σύγκρισή της με τις ήδη υπάρχουσες μηχανές συλλογιστικής ανάλυσης, εκτελέστηκε μια σειρά πειραμάτων με τέσσερεις μεγάλες βιο-ιατρικές οντολογίες διαφορετικών μεγεθών και πολυπλοκότητας. Η Gene Ontology (GO) και η National Cancer Institute (NCI) οντολογία, είναι αρκετά μεγάλες οντολογίες που περιέχουν αντίστοιχα 20.465 και 27.652 κλάσεις. Η οντολογία Galen⁻ που περιέχει 23.136 κλάσεις είναι παράγωγο της οντολογίας OpenGalen, με την αφαίρεση των λειτουργιών ρόλων (role functionalities), της μεταβατικότητας των ρόλων (role transitivity) και των αντίστροφων ρόλων (inverse roles). Η μεγαλύτερη οντολογία που χρησιμοποιήθηκε ήταν η Snomed, βασισμένη στην Snomed CT, που περιείχε 315.489 κλάσεις.

Στο Σχήμα 3.9 συγκρίνονται η μηχανή συλλογιστικής ανάλυσης που δημιουργήθηκε με τις ήδη υπάρχουσες μηχανές συλλογιστικής ανάλυσης CB, CEL, FaCT++ και Hermit, οι οποίες λειτουργούν στην μνήμη. Παρατηρείται ότι η επίδοση της μηχανής συλλογιστικής ανάλυσης DB είναι παραπλήσια με αυτή των στην μνήμη μηχανών συλλογιστικής ανάλυσης και, συγκεκριμένα, ξεπερνά τον CEL στην Galen⁻ και στην Snomed, τον FaCT++ στην Galen⁻ , και τον Hermit σε όλες τις οντολογίες.

Reasoner	NCI	GO	Galen⁻	Snomed
DB	35.51	20.36	100.86	1183.80
CB	7.64	1.23	3.36	45.17
CEL v.1.0	3.60	1.02	169.23	1302.18
FaCT++ v.1.3.0	4.60	10.50	-	965.84
HermiT v.0.9.3	70.23	92.76	-	-

Σχήμα 3.9. Σύγκριση με άλλες μηχανές συλλογιστικής ανάλυσης. Ο χρόνος είναι σε δευτερόλεπτα.

3.1.3 CB reasoner

Η μηχανή συλλογιστικής ανάλυσης CB (CB reasoner) χρησιμοποιείται για την ταξινόμηση οντολογιών εκφραστικότητας *Horn SHIQ*, δηλαδή *SHIQ* οντολογιών που μπορούν να μεταφραστούν στο *Horn* κομμάτι της λογικής πρώτης τάξης. . Η γλώσσα προγραμματισμού που χρησιμοποιείται είναι η Java και η διαδικασία λειτουργεί στην μνήμη (memory-based).

Το σύνολο των *SHIQ* εννοιών ορίζεται μέσω των κατασκευαστών που παρουσιάζονται στο Σχήμα 3.10. Ένας *SHIQ* ρόλος είναι είτε $r \in N_R$, είτε ένας αντίστροφος ρόλος (inverse role) r^- με $r \in N_R$.

Name	Syntax	Semantics
Concepts		
atomic concept	A	A^I (given)
top concept	$\top$	Δ^I
bottom concept	$\bot$	$\emptyset$
negation	$\neg C$	$\Delta^I \setminus C^I$
conjuction	$C \sqcap D$	$C^I \cap D^I$
disjunction	$C \sqcup D$	$C^I \cup D^I$
existential restriction	$\exists R.C$	$\{x \mid R^I(x,C^I) \neq \emptyset\}$
universal restriction	$\forall R.C$	$\{x \mid R^I(x,\Delta^I \setminus C^I) = \emptyset\}$
min cardinality	$\geq nS.C$	$\{x \| S^I(x,C^I) \| \geq n\}$
max cardinality	$\leq mS.C$	$\{x \| S^I(x,C^I) \| \leq m\}$
Axioms		
role inclusion	$R_1 \sqsubseteq R_2$	$R_1^I \subseteq R_2^I$
role transivity	Tra(R)	$R^I \circ R^I \subseteq R^I$
concept inclusion	$C \sqsubseteq D$	$C^I \subseteq D^I$

Σχήμα 3.10. Σύνταξη και Σημασιολογία της *SHIQ*

Παρουσιάζονται οι παρακάτω έννοιες:

- Μία οντολογία είναι ένα σύνολο από αξιώματα (axioms) όπως αυτά που περιγράφονται στον παραπάνω πίνακα.
- Ένας ρόλος λέγεται μεταβατικός (transitive) εάν $Tra(R) \in O$ ή $Tra(R^-) \in O$.
- $R_1 \sqsubseteq_O R_2$ είναι η μικρότερη μεταβατική αυτοπαθής σχέση (transitive reflexive relation) μεταξύ δύο ρόλων έτσι ώστε η σχέση $R_1 \sqsubseteq R_2 \in O$, να εννοεί ότι $R_1 \sqsubseteq_O R_2$ και $R_1^- \sqsubseteq_O R_2^-$.

- Είναι απαραίτητο για κάθε έννοια της μορφής $\geq$nS.C και $\leq$mS.C στην O, ο ρόλος S να είναι απλός (simple), δηλαδή το $R \sqsubseteq_O S$ να μην ισχύει για κανέναν μεταβατικό ρόλο R.
- Με $\|V\|$ συμβολίζεται ο αριθμός των στοιχείων του συνόλου (cardinality) $V \sqsubseteq \Delta^I$.
- Μία ερμηνεία (interpretation) I ικανοποιεί ένα αξίωμα α, εάν η αντίστοιχη συνθήκη στα δεξιά του παραπάνω πίνακα ισχύει.
- Η ερμηνεία I είναι μοντέλο (model) της O, εάν η I ικανοποιεί κάθε αξίωμα στην O.
- Το α είναι λογική συνέπεια της O, ή αλλιώς συνεπάγεται από την O, εάν κάθε μοντέλο της O ικανοποιεί το α.

Οι θετικές και αρνητικές πολικότητες των SHIQ εννοιών ορίζονται ως εξής :

- Η C εμφανίζεται θετικά στο C
- Η C εμφανίζεται θετικά (αρνητικά) στη $\neg C_-$, $C_+ \sqcap D_+$, $C_+ \sqcup D_+$, $\exists R.C_+$, $\forall R.C_+$, $\geq nS.C_+$, $\leq mS.C_-$ και $C_- \sqsubseteq D_+$, εάν η C εμφανίζεται θετικά (αρνητικά) στη C_+ ή στη D_+ , ή εμφανίζεται αρνητικά (θετικά) στη C.

Μία έννοια C εμφανίζεται θετικά (αρνητικά) σε μια οντολογία O, εάν η C εμφανίζεται θετικά (αρνητικά) σε ένα αξίωμα της O. Είναι επομένως δυνατό για μία έννοια να εμφανίζεται και θετικά και αρνητικά σε ένα αξίωμα ή μια οντολογία.

Μία *SHIQ* οντολογία O είναι *Horn*, εάν :

- Δεν εμφανίζεται θετικά στην O καμία έννοια της μορφής $C \sqcup D$ ή $\leq mR.C$ με $m>1$.
- Δεν εμφανίζεται αρνητικά στην O καμία έννοια της μορφής $\neg C$, $\forall R.C$, $\geq nR.C$ με $n>1$, ή $\leq mR.C$.

Ο δομικός μετασχηματισμός (structural transformation) χρησιμοποιείται για την απλοποίηση των αξιωμάτων της οντολογίας, διατηρώντας όμως την δομή της. Έχοντας δοσμένη μια *SHIQ* οντολογία O, για κάθε υπο-έννοια C στην O εισάγεται μία καινούργια ατομική έννοια A_C και ορίζεται μια συνάρτηση st(C) ως :

- $st(A) = A$, $st(\top) = \top$, $st(\bot) = \bot$
- $st(\neg C) = \neg A_C$
- $st(C \sqcap D) = st(A_C \sqcap A_D)$, $st(C \sqcup D) = st(A_C \sqcup A_D)$
- $st(\exists R.C) = \exists R.A_C$, $st(\forall R.C) = \forall R.A_C$
- $st(\geq nR.C) = \geq nR.A_C$, $\leq st(mR.C) = \leq mR.A_C$

Το αποτέλεσμα της εφαρμογής δομικού μετασχηματισμού στην O είναι η οντολογία O΄ που διατηρεί τις λογικές συνέπειες των αξιωμάτων της οντολογίας και περιέχει όλες τις υπαγωγές ρόλων και τα αξιώματα μεταβατικών ρόλων της O, με την προσθήκη των παρακάτω αξιωμάτων :

- $A_C \sqsubseteq st(C)$ για κάθε C που εμφανίζεται θετικά στην O
- $st(C) \sqsubseteq A_C$ για κάθε C που εμφανίζεται αρνητικά στην O
- $A_C \sqsubseteq A_D$ για κάθε υπαγωγή έννοιας $C \sqsubseteq D \in O$

Η $\mathcal{EL}^+$ επιτρέπει μόνο έννοιες που έχουν κατασκευαστεί από ατομικές έννοιες A και την καθολική έννοια T , χρησιμοποιώντας τομές $C \sqcap D$ και υπαρξιακούς

περιορισμούς ∃r.C, όπου το r είναι ατομικός ρόλος. Τα αξιώματα της $\mathcal{EL}^+$ μπορούν να είναι είτε υπαγωγές εννοιών C ⊑ D, είτε σύνθετες υπαγωγές ρόλων της μορφής r_1 o.....o $r_n \sqsubseteq s$, που ερμηνεύονται ως r_1^I o.....o $r_n^I \sqsubseteq s^I$, όπου ο είναι μία σύνθεση δυαδικών σχέσεων.

Θεωρείται ένα κοινό κομμάτι της $\mathcal{EL}^+$ και της $\mathcal{SHIQ}$ που καλείται $\mathcal{ELH}$ και επιτρέπει μόνο (απλές) υπαγωγές ρόλων της μορφής r ⊑ s. Εύκολα διαπιστώνεται ότι μια $\mathcal{ELH}$ οντολογία είναι και *Horn* $\mathcal{SHIQ}$ οντολογία, αφού δεν περιέχει concepts της μορφής ¬C, C ⊔ D, ∀R.C, ≥ nR.C, ≤ mR.C.

$$\boldsymbol{IR1}\ \frac{}{A \sqsubseteq A} \qquad \boldsymbol{IR2}\ \frac{}{A \sqsubseteq \top}$$

$$\mathbf{CR1}\ \frac{A \sqsubseteq B \quad B \sqsubseteq C \in O}{A \sqsubseteq C}$$

$$\mathbf{CR2}\ \frac{A \sqsubseteq B \quad A \sqsubseteq C \quad B \sqcap C \sqsubseteq D \in O}{A \sqsubseteq D}$$

$$\boldsymbol{CR3}\ \frac{A \sqsubseteq B \quad B \sqsubseteq \exists r.C \in O}{A \sqsubseteq \exists r.C}$$

$$\boldsymbol{CR4}\ \frac{A \sqsubseteq \exists r.B \quad r \sqsubseteq s \in O}{A \sqsubseteq \exists s.B}$$

$$\boldsymbol{CR5}\ \frac{A \sqsubseteq \exists r.B \quad B \sqsubseteq C \quad \exists r.C \sqsubseteq D \in O}{A \sqsubseteq D}$$

Σχήμα 3.11. Κανόνες Ολοκλήρωσης για μια κανονικοποιημένη $\mathcal{ELH}$ οντολογία

Η *Horn* $\mathcal{SHIQ}$ επεκτείνει την $\mathcal{ELH}$ επιτρέποντας αντίστροφους ρόλους (inverse roles), λειτουργικούς περιορισμούς (functionality restrictions) και θετικές εμφανίσεις καθολικών περιορισμών (universal restrictions). Οι κατασκευαστές αυτοί είχαν αποκλειστεί από τις $\mathcal{ELH}$ και $\mathcal{EL}^{++}$, αφού η προσθήκη τους οδηγεί από πολυωνυμικό σε εκθετικό χρόνο.

Οι θετικές εμφανίσεις καθολικών περιορισμών μπορούν να εκφραστούν μέσω αντίστροφων ρόλων χρησιμοποιώντας την ισότητα:

$$A \sqsubseteq \forall R.B \quad \Leftrightarrow \quad \exists R^{-}.A \sqsubseteq B \quad (1)$$

Η αύξηση στην πολυπλοκότητα εξαιτίας αυτών των δύο κατασκευαστών μπορεί να δικαιολογηθεί από δύο νέους τρόπους αλληλεπίδρασης που δημιουργούνται:

$$\frac{A \sqsubseteq \exists R.B \quad B \sqsubseteq \forall R^{-}.D}{A \sqsubseteq D} \qquad (2)$$

$$\frac{A \sqsubseteq \exists R.B \quad C \sqsubseteq \forall R.D}{A \sqcap C \sqsubseteq \exists R.(B \sqcap D)} \qquad (3)$$

Διαπιστώνεται εύκολα με την χρήση της ισότητας (1) ότι ο κανόνας (2) είναι στιγμιότυπο του CR5 του Σχήματος 3.11. Αντίθετα με όλους τους άλλους κανόνες, ο

κανόνας (3) παράγει αξιώματα της μορφής $\sqcap A_i \sqsubseteq \exists R.\sqcap B_j$, όπου $\sqcap A_i$ και $\sqcap B_j$ είναι αυθαίρετες τομές από ατομικές έννοιες. Ο αριθμός τέτοιων αξιωμάτων δεν είναι πλέον πολυωνυμικά φραγμένος, υπάρχουν όμως εκθετικά πολλά στην χειρότερη περίπτωση τέτοια αξιώματα, και η διαδικασία χρησιμοποιεί αυτή την ιδιότητα για να θέσει ένα εκθετικό πάνω όριο.

Όσον αφορά τους λειτουργικούς περιορισμούς, εφόσον για έναν λειτουργικό ρόλο S, το αξίωμα $C \sqsubseteq \exists S.D$ υπονοεί $C \sqsubseteq \forall S.D$, προκύπτουν οι επόμενοι δύο κανόνες ολοκλήρωσης, ανάλογοι των (3) και (2):

$$\frac{A \sqsubseteq \exists S.B \quad C \sqsubseteq \exists S.D \quad A \sqsubseteq \leq 1S.\top}{A \sqcap C \sqsubseteq \exists S.(B \sqcap D)} \qquad (4)$$

$$\frac{A \sqsubseteq \exists S.B \quad B \sqsubseteq \exists S^{-}.C \quad B \sqsubseteq \leq 1S^{-}.\top}{A \sqsubseteq C} \qquad (5)$$

Η αρχική *Horn SHIQ* οντολογία Ο μετασχηματίζεται στην οντολογία Ο΄που περιέχει μόνο αξιώματα της μορφής (n1) $\sqcap A_{(i)} \sqsubseteq C$ (n2) $R_1 \sqsubseteq R_2$ (n3) Tra(R), όπου $A_{(i)}$ ατομική έννοια, $R_{(i)}$ ρόλος και C είναι μία απλή έννοια, δηλαδή μια έννοια της μορφής ⊥, A, ∃R.A, ∀R.A ή ≤1S.A. Ο μετασχηματισμός αυτός διατηρεί τις συνεπαγωγές των αξιωμάτων α της Ο και μπορεί να επιτευχθεί σε πολυωνυμικό χρόνο θεωρώντας μοναδιαία κωδικοποίηση των αριθμών.

Συγκεκριμένα εφαρμόζοντας δομικό μετασχηματισμό στην Ο, προκύπτει η οντολογία Ο΄ που περιέχει μόνο αξιώματα της μορφής $A_1 \sqsubseteq A_2$, $A \sqsubseteq st(C_+)$ και $st(C_-) \sqsubseteq A$.Οι υπαγωγές εννοιών της μορφής $A \sqsubseteq st(C_+)$, μετασχηματίζονται στη μορφή (n1) ως εξής:

- $A \sqsubseteq st(\neg C) = \neg A_C \Rightarrow A \sqcap A_C \sqsubseteq \bot$;
- $A \sqsubseteq st(\geq nS.C) = \geq nS.A_C \Rightarrow A \sqsubseteq \exists S.B_i$, $B_i \sqsubseteq A_C$, $1 \leq i \leq n$, $B_i \sqcap B_j \subseteq \bot$, $1 \leq i \leq j \leq n$, όπου τα B_i είναι καινούργιες ατομικές έννοιες.

Οι υπαγωγές εννοιών της μορφής $st(C_-) \leq A$, μετασχηματίζονται στη μορφή (n1) ως εξής:

- $st(C \sqcup D) = A_C \sqcup A_D \sqsubseteq A \Rightarrow A_C \sqsubseteq A, A_D \sqsubseteq A$;
- $st(\exists R.C) = \exists R.A_C \sqsubseteq A \Rightarrow A_C \sqsubseteq \forall R^{-}.A$;
- $st(\geq 1S.C) = \geq 1S.A_C \sqsubseteq A \Rightarrow A_C \sqsubseteq \forall S^{-}.A$;

Μετά την κανονικοποίηση σκοπός είναι η εξάλειψη των μεταβατικών αξιωμάτων (transitivity axioms). Μεταβατικά αξιώματα της μορφής (n3) μπορούν να αλληλεπιδράσουν μόνο με αξιώματα $\sqcap A_i \sqsubseteq \forall R.B$ της μορφής (n1) μέσω υπαγωγών ρόλων της μορφής (n2). Ο μετασχηματισμός αυτός εισάγει τρία νέα αξιώματα (6) για κάθε αξίωμα $\sqcap A_i \sqsubseteq \forall R.B$ της μορφής (n1) και κάθε μεταβατικό υπό-ρόλο T του R, όπου B^T είναι μία νέα ατομική έννοια :

$$\sqcap A_i \sqsubseteq \forall T.B^T \quad B^T \sqsubseteq \forall T.B^T \quad B^T \sqsubseteq B \qquad (6)$$

Διαισθητικά, το B^T χρησιμοποιείται για την διάδοση του B σε όλα τα στοιχεία που μπορούν να προσεγγιστούν από στοιχεία στην τομή $\sqcap A_\iota$ μέσω μίας T-αλυσίδας.

Στη συνέχεια, για την επίτευξη της ταξινόμησης, εφαρμόζονται οι συμπερασματικοί κανόνες (inference rules) που παρουσιάζονται στο Σχήμα 3.12, παράγοντας αξιώματα της μορφής $M \sqsubseteq C$ και $M \sqsubseteq \exists R.N$, όπου τα M και N είναι τομές από ατομικές έννοιες και το C είναι μία απλή έννοια. Οι κανόνες I1 και I2 παράγουν τις αρχικές υπαγωγές εννοιών για αυθαίρετα M και A, οι οποίες μετά επεκτείνονται μέσω του κανόνα R1 χρησιμοποιώντας τα αξιώματα στην O. Ο κανόνας R2 διαδίδει την συνεπαγωγή της κενής έννοιας μέσω των υπαρξιακών περιορισμών. Οι κανόνες R3 και R4 ασχολούνται με τους καθολικούς περιορισμούς και γενικοποιούν τους κανόνες (3) και (2). Παρόμοια, οι κανόνες R5 και R6 ασχολούνται με τους λειτουργικούς περιορισμούς και γενικοποιούν τους κανόνες (4) και (5). Οι κανόνες R3-R6 θεωρούν ότι έχει υπολογιστεί το "κλείσιμο" $R_1 \sqsubseteq_O R_2$ των υπαγωγών ρόλων της μορφής (n2). Οι κανόνες I1-R6 εφαρμόζονται εξαντλητικά μέχρι να μην μπορούν να παραχθούν καινούργια αξιώματα. Η οντολογία O΄ που παράγεται ονομάζεται κορεσμός (saturation) της O.

$$\boldsymbol{I1}\ \frac{}{M \sqcap A \sqsubseteq A} \qquad \boldsymbol{I2}\ \frac{}{M \sqsubseteq \top}$$

$$\boldsymbol{R1}\ \frac{M \sqsubseteq A_i \quad \sqcap A_i \sqsubseteq C \in O}{M \sqsubseteq C} \qquad \boldsymbol{R2}\ \frac{M \sqsubseteq \exists R.N \quad N \sqsubseteq \bot}{M \sqsubseteq \bot}$$

$$\boldsymbol{R3}\ \frac{M \sqsubseteq \exists R_1.N \quad M \sqsubseteq \forall R_2.A \quad R_1 \sqsubseteq_O R_2}{M \sqsubseteq \exists R_1.(N \sqcap A)} \qquad \boldsymbol{R4}\ \frac{M \sqsubseteq \exists R_1.N \quad N \sqsubseteq \forall R_2.A \quad R_1 \sqsubseteq_O R_2^-}{M \sqsubseteq A}$$

$$\boldsymbol{R5}\ \frac{\begin{array}{c} M \sqsubseteq \exists R_1.N_1 \quad N_1 \sqsubseteq B \quad R_1 \sqsubseteq_O S \\ M \sqsubseteq \exists R_2.N_2 \quad N_2 \sqsubseteq B \quad R_2 \sqsubseteq_O S \\ M \sqsubseteq \leq 1S.B \end{array}}{M \sqsubseteq \exists R_1.(N_1 \sqcap N_2)} \qquad \boldsymbol{R6}\ \frac{\begin{array}{c} M \sqsubseteq \exists R_1.N_1 \quad N_1 \sqsubseteq \exists R_2.(N_2 \sqcap A) \quad R_1 \sqsubseteq_O S^- \\ M \sqsubseteq B \quad N_2 \sqcap A \sqsubseteq B \quad R_2 \sqsubseteq_O S \\ N_1 \sqsubseteq \leq 1S.B \end{array}}{M \sqsubseteq A \quad M \sqsubseteq \exists R_2^-.N_1}$$

Σχήμα 3.12. Συμπερασματικοί κανόνες για *Horn SHIQ* οντολογίες

Η διαδικασία είναι ορθή και πλήρης για την ταξινόμηση *Horn SHIQ* οντολογιών. Επιπλέον, εφόσον ο αριθμός των αξιωμάτων της μορφής $M \sqsubseteq C$ και $M \sqsubseteq \exists R.N$ είναι το πολύ εκθετικός στον αριθμό των ατομικών εννοιών, η διαδικασία εγγυάται τον τερματισμό της σε εκθετικό χρόνο. Επειδή ο υπολογισμός των υπαγωγών εννοιών στις *Horn SHIQ* οντολογίες είναι εκθετικός, η διαδικασία είναι υπολογιστικά βέλτιστη.

Στο Σχήμα 3.13 το σύστημα συλλογιστικής ανάλυσης CB συγκρίνεται με τους Fact++ v.1.2.1, Pellet v.1.5, HermiT v.0.9.3 και τον CEL v.1.0b. Οι οντολογίες που χρησιμοποιήθηκαν ήταν οι GO και NCI (20.465 και 27.652 κλάσεις αντίστοιχα), η Snomed (389.472 κλάσεις) και 4 εκδόσεις της Galen, η GalenA (2.748 κλάσεις), η GalenB (23.136 κλάσεις) και οι $GalenA^-$ και $GalenB^-$ (έχουν αφαιρεθεί η λειτουργικότητα και οι αντίστροφοι ρόλοι). Οι χρόνοι που αφορούν το CB αντιστοιχούν στο φόρτωμα, την προεπεξεργασία, την ταξινόμηση της οντολογίας, του

υπολογισμού του μεταβατικού κλεισίματος και της αλφαβητικής κατανομής της ιεραρχίας των εννοιών.

	FaCT++	Pellet	HermiT	CEL	CB
GO	15.24	72.02	199.52	1.84	1.17
NCI	6.05	26.47	169.47	5.76	3.57
GalenA$^-$	3166.74	133.25	91.98	3.27	0.26
GalenA	465.35	-	45.72	n/a	0.32
GalenB$^-$	-	-	-	189.12	4.07
GalenB	-	-	-	n/a	9.58
Snomed	650.37	-	-	1185.70	49.44

Σχήμα 3.13. Σύγκριση χρόνων ταξινόμησης (σε δευτερόλεπτα). Το "-" σημαίνει ότι η μηχανή συλλογιστικής ανάλυσης δεν κατάφερε να παράγει αποτέλεσμα. Το n/a σημαίνει ότι η δοκιμή δεν μπορούσε να γίνει.

3.2 Μέθοδοι βασισμένες στην γλώσσα Datalog

Εδώ παρουσιάζονται τρεις μέθοδοι ταξινόμησης οντολογιών κάνοντας χρήση της γλώσσας προγραμματισμού Datalog. Βασικό βήμα της ταξινόμησης για όλες τις διαδικασίες είναι η μετάφραση των αρχικών αξιωμάτων της οντολογίας σε Datalog. Η Datalog βασίζεται στην *Horn* Λογική (*Horn Logic*) η οποία, ακριβώς όπως η Περιγραφική Λογική, είναι ένα αποφασίσιμο (decidable) υποσύνολο της Λογικής Πρώτης Τάξης(First-Order Logic).Για τις δύο πρώτες μεθόδους που παρουσιάζονται δεν υπάρχει μέχρι στιγμής υλοποίηση, ενώ η τρίτη μέθοδος έχει υλοποιηθεί στο σύστημα Orel.

3.2.1 Μέθοδος των Cali, Gottlob και Lukasiewiez

Με την μέθοδο αυτή η οποία λειτουργεί στον σκληρό δίσκο , μπορεί να επιτευχθεί η ταξινόμηση οντολογιών εκφραστικότητας $\mathcal{DL}$-$Lite_{\mathcal{F}}$, $\mathcal{DL}$-$Lite_{\mathcal{R}}$, και $\mathcal{DL}$-$Lite_{\mathcal{A}}$, μέσω της μετάφρασης των αξιωμάτων της οντολογίας σε Datalog$^{\pm}$ (και συγκεκριμένα στις φρουρούμενη (guarded) Datalog$^{\pm}$ και γραμμική (linear) Datalog$^{\pm}$) που επεκτείνει την Datalog με υπαρξιακές ποσοτικοποιήσεις (existential quantifications) στις κεφαλές των κανόνων, και με κάποια άλλα γνωρίσματα. Παρακάτω παρουσιάζονται κάποια απαραίτητα εισαγωγικά στοιχεία για την κατανόηση της μεθόδου:

Βάσεις δεδομένων και Ερωτήματα (Queries). Θεωρούνται άπειρα σύνολα από σταθερές Δ (data constants) , μηδενικά Δ_N (nulls) και μεταβλητές X, όπου

διαφορετικές σταθερές αντιπροσωπεύουν διαφορετικές τιμές (μοναδικότητα – unique name assumption). Με **X** συμβολίζεται μία ακολουθία από μεταβλητές $X_1,...,X_k$ με $k \geq 0$. Θεωρείται επίσης ένα σχεσιακό σχήμα R (relational schema) που είναι ένα σύνολο από ονόματα σχέσεων – relation names (ή κατηγορήματα – predicates). Ένας όρος t (term) είναι μια σταθερά, μηδενικό ή μεταβλητή. Ένας ατομικός τύπος (atomic formula) έχει τη μορφή $P(t_1,\ldots t_n)$, όπου το P είναι κατηγόρημα και με pred(a) ή dom(a) συμβολίζεται το σύνολο των ορισμάτων του. Μια βάση δεδομένων D (στιγμιότυπο) του R είναι ένα σύνολο από ατομικούς τύπους με κατηγορήματα από το R και ορίσματα από το $\Delta \sqcup \Delta_N$. Ένα συζευκτικό ερώτημα (conjunctive query-CQ) στο R έχει τη μορφή $Q(\mathbf{X}) = \exists \mathbf{Y} \Phi(\mathbf{X},\mathbf{Y})$, όπου $\Phi(\mathbf{X},\mathbf{Y})$ είναι μία σύζευξη ατομικών τύπων με τις μεταβλητές **X** και **Y**. Ένα Boolean συζευκτικό ερώτημα (BCQ) είναι της μορφής Q(). Οι απαντήσεις στα CQs και τα BCQs ορίζονται μέσω ομομορφισμών (homomorphisms), που είναι αντιστοιχίσεις $\mu : \Delta \sqcup \Delta_N \sqcup X \Rightarrow \Delta \sqcup \Delta_N \sqcup X$ τέτοιες ώστε (i) $c \in \Delta$ υπονοεί $\mu(c)=c$ (ii) $c \in \Delta_N$ υπονοεί $\mu(c) \in \Delta \sqcap \Delta_N$ (iii)η μ επεκτείνεται και σε ατομικούς τύπους, σύνολα ατομικών τύπων και συζεύξεις ατομικών τύπων. Το σύνολο όλων των απαντήσεων σε ένα CQ, που συμβολίζεται με Q(D), είναι το σύνολο όλων των πλειάδων t (tuples) στο Δ για τις οποίες υπάρχει ομομορφισμός μ: $\mathbf{X} \sqcup \mathbf{Y} \Rightarrow \Delta \sqcup \Delta_N$ τέτοιος ώστε $\mu(\Phi(\mathbf{X},\mathbf{Y})) \sqsubseteq D$ και $\mu(\mathbf{X})=t$. Η απάντηση σε ένα BCQ Q() είναι Yes, και συμβολίζεται $D \models Q$, εάν και μόνο εάν $Q(D) \neq \emptyset$.

TGDs. Μια εξάρτηση παραγωγής πλειάδων (tuple-generating dependency – TGD) σ είναι ένας τύπος της μορφής $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y}) \Rightarrow \exists \mathbf{Z} \Psi(\mathbf{X},\mathbf{Y})$, όπου $\Phi(\mathbf{X},\mathbf{Y})$ και $\Psi(\mathbf{X},\mathbf{Y})$ είναι συζεύξεις ατόμων στο σχεσιακό σχήμα R και καλούνται σώμα και κεφαλή του σ (body(σ), head(σ)) αντίστοιχα. Η σ ικανοποιείται εάν και μόνο εάν όποτε υπάρχει ένας ομομορφισμός h που αντιστοιχεί άτομα του $\Phi(\mathbf{X},\mathbf{Y})$ σε άτομα της βάσης δεδομένων D, υπάρχει μια προέκταση h' του h που αντιστοιχεί άτομα του $\Psi(\mathbf{X},\mathbf{Y})$ σε άτομα της D. Για ένα σύνολο από TGDs Σ στο R και μια βάση δεδομένων D στο R, το σύνολο των μοντέλων (models) της D δοσμένου του Σ, που συμβολίζεται mods(Σ,D), είναι το σύνολο όλων των βάσεων δεδομένων B τέτοιων ώστε $B \models D \sqcup \Sigma$. Το σύνολο των απαντήσεων σε ένα CQ Q στην D δοσμένου του Σ, που συμβολίζεται ως ans(Q,Σ,D), είναι το σύνολο όλων των πλειάδων a τέτοιων ώστε $a \in Q(B)$ για κάθε $B \in mods(\Sigma,D)$. Η απάντηση σε ένα BCQ Q στην D δοσμένου του Σ είναι Yes, και συμβολίζεται με $D \sqcup \Sigma \models Q$, εάν και μόνο εάν $ans(Q,\Sigma,D) \neq \emptyset$.

Το TGD κυνηγητό. Το «κυνηγητό» (chase) είναι η διαδικασία (μαζί με την έξοδό της) επιδιόρθωσης μιας βάσης δεδομένων έτσι ώστε να ικανοποιεί ένα σύνολο εξαρτήσεων. Επιτυγχάνεται μέσω κανόνων, οι οποίοι χωρίζονται σε καθολικούς (oblivious) και περιορισμένους (restricted).

Ο (καθολικός) κανόνας του TGD κυνηγητού. Έστω μια βάση δεδομένων D, ένα σχεσιακό σχήμα R, και ένα TGD σ στο R της μορφής $\Phi(\mathbf{X},\mathbf{Y}) \Rightarrow \exists \mathbf{Z} \Psi(\mathbf{X},\mathbf{Y})$. Τότε το σ είναι εφαρμόσιμο (applicable) στην D εάν υπάρχει ένας ομομορφισμός h που αντιστοιχεί τα άτομα του $\Phi(\mathbf{X},\mathbf{Y})$ στα άτομα της D. Έστω ότι το σ είναι εφαρμόσιμο

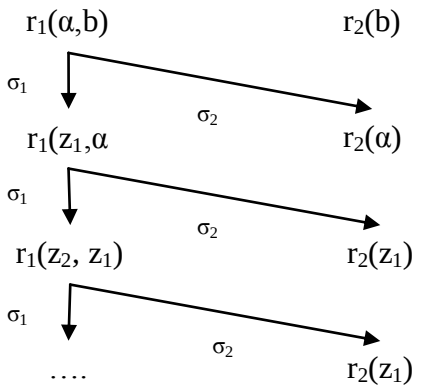
και ο ομομορφισμός h_1 επεκτείνει τον h ως εξής: για κάθε $X_i \in \mathbf{X}$, $h_1(X_i) = h(X_i)$ και κάθε $Z_j \in \mathbf{Z}$, $h_1(Z_j) = z_j$, όπου z_j είναι ένα νέο μηδενικό, δηλαδή $z_j \in \Delta_N$, το z_j δεν εμφανίζεται στην D και ακολουθεί λεξικογραφικά όλα τα υπόλοιπα μηδενικά που έχουν ήδη εισαχθεί. Η εφαρμογή του σ προσθέτει στην D το άτομο $h_1(\Psi(\mathbf{X},\mathbf{Y}))$ εάν δεν υπάρχει ήδη.

Επίπεδο (level) ατόμου σε ένα TGD κυνηγητό. Έστω D η αρχική βάση δεδομένων. Τα άτομα στην D έχουν επίπεδο 0. Έστω ότι εφαρμόζεται ένα TGD όμοιο με τον παραπάνω κανόνα στην βάση. Εάν το άτομο με το υψηλότερο επίπεδο ανάμεσα σε αυτά στον $h_1(\Phi(\mathbf{X},\mathbf{Y}))$ έχει επίπεδο k, τότε το προστιθέμενο $h_1(\Psi(\mathbf{X},\mathbf{Y}))$ έχει επίπεδο k+1. Ο αλγόριθμος για το κυνηγητό εφαρμόζει εξαντλητικά τον κανόνα με προτεραιότητα κατά πλάτος (breadth-first), που οδηγεί σε ένα κυνηγητό για τα Σ και D, που συμβολίζεται chase(Σ,D). Το κυνηγητό για επίπεδο μέχρι $k \geq 0$, που συμβολίζεται $\text{chase}^k(\Sigma,D)$, είναι το σύνολο όλων των ατόμων στο chase(Σ,D) με επίπεδο το πολύ k. Το κυνηγητό σχετικό με τα TGDs είναι ένα καθολικό μοντέλο (universal model), δηλαδή για κάθε $B \in \text{mods}(\Sigma,D)$ υπάρχει ένας ομομορφισμός που αντιστοιχεί το chase(Σ,D) στην B. Αυτό υπονοεί Q(chase(Σ,D)) = ans(Q,Σ,D).

Μέσω των παρακάτω ορίζεται η φρουρούμενη Datalog:

Ένα TGD σ είναι φρουρούμενο (guarded) εάν και μόνο εάν περιέχει ένα άτομο στο σώμα του που περιέχει όλες τις καθολικά ορισμένες μεταβλητές του σ. Το πιο αριστερά τέτοιο άτομο του σ είναι το άτομο φρουρός (guard atom), ενώ τα άτομα του σ που δεν είναι φρουροί ονομάζονται παράπλευρα άτομα (side atoms). Για παράδειγμα το TGD $r(X,Y),s(Y,X,Z) \Rightarrow \exists W s(Z,X,W)$ φρουρείται από τον φρουρό s(Y,X,Z).

Ο γράφος του κυνηγητού (chase graph) είναι ο κατευθυνόμενος γράφος που αποτελείται από το chase(Σ,Δ) ως το σύνολο των κόμβων του και έχει ακμή από το a στο b εάν και μόνο εάν το b αποκτάται από το a και ίσως άλλα άτομα από μια μόνο εφαρμογή ενός TGD $\sigma \in \Sigma$. Το δάσος του φρουρούμενου κυνηγητού (guarded chase forest) είναι το κομμάτι του γράφου για τα Σ και D που περιέχει μόνο όλα τα άτομα που έχουν επισημανθεί ως φρουροί και τα παιδιά τους. Το υπόδεντρο του a, που συμβολίζεται subtree(a), είναι ο γράφος που περιέχει μόνο τα παιδιά του a. Ο τύπος του a, που συμβολίζεται type(a) είναι το σύνολο όλων των ατόμων b στο chase(Σ,D) που περιέχουν μόνο σταθερές ή μηδενικά από το a ως ορίσματα. Για παράδειγμα έστω ότι σε ένα στιγμιότυπο $D=\{r_1(a,b),r_2(b)\}$ εφαρμόζονται τα TGDs σ_1: $r_1(X,Y),r_2(Y) \Rightarrow \exists Z r_1(Z,X)$ και σ_2: $r_1(X,Y) \Rightarrow r_2(X)$, τότε το δάσος του φρουρούμενου κυνηγητού για τα $\{\sigma_1,\sigma_2\}$ και D θα είναι:



Δοσμένου ενός συνόλου $S \sqsubseteq \Delta \sqcup \Delta_N$, δύο σύνολα ατόμων A_1 και A_2 λέγονται S-ισομορφικά (S-isomorphic) εάν και μόνο έαν υπάρχει μια αμφιμονοσήμαντη αντιστοιχία β: $A_1 \sqcup \text{dom}(A_1) \Rightarrow A_2 \sqcup \text{dom}(A_2)$ τέτοια ώστε (i) το β και το β^{-1} είναι ομομορφισμοί και (ii)$\beta(c)=c=\beta^{-1}(c)$ για κάθε $c \in S$. Δύο άτομα α_1 και α_2 είναι S-ισομορφικά εάν και μόνο εάν τα $\{\alpha_1\}$ και $\{\alpha_2\}$ είναι S-ισομορφικά.

Ισχύουν τα παρακάτω:

1. Έστω S ένα σύνολο από σταθερές και μηδενικά, και a_1, a_2 άτομα από το chase(Σ,D) με S-ισομορφικούς τύπους. Τότε το υπόδεντρο του a_1 είναι S-ισομορφικό ως προς το a_2.
2. Έστω w ο μέγιστος αριθμός κατηγορημάτων στο R, $\delta=(2w)^w \cdot 2^{(2w)w \cdot |R|}$, $a \in$ chase(Σ,D), και P έαν σύνολο από ζευγάρια (b,S) που το καθένα αποτελείται από ένα άτομο b και ένα τύπο S από άτομα c με ορίσματα από το a και νέα μηδενικά. Αν $|P|>\delta$, τότε το P περιέχει τουλάχιστον δύο dom(a)-ισομορφικά ζευγάρια.
3. Έστω ότι το a είναι φρουρός στο γράφο του κυνηγητού για τα Σ και D, και $b \in$ type(a). Τότε, depth(b)≤ depth(a)+k, όπου το k εξαρτάται μόνο από το Σ.
4. Έστω Q ένα BCQ. Αν υπάρχει ένας ομομορφισμός h που αντιστοιχεί το Q στο chase(Σ,D), τότε υπάρχει ένας ομομορφισμός λ που αντιστοιχεί το Q στο g-chase^k(Σ,D), όπου το k εξαρτάται μόνο από τα Q και R και g-chase^k(Σ,D) είναι το φρουρούμενο κυνηγητό επιπέδου μέχρι $k \geq 0$.
5. Το Σ έχει την ιδιότητα του φραγμένου βάθους φρουρού (bounded guard-depth property – BGDP),εάν και μόνο εάν, για κάθε βάση D για το R και για κάθε BCQ Q, όποτε υπάρχει ένας ομομορφισμός μ που αντιστοιχεί το Q στο chase(Σ,D), τότε υπάρχει ένας ομομορφισμός λ τέτοιος ώστε όλοι οι πρόγονοι του λ(Q) στον γράφο του κυνηγητού για τα Σ και D περιέχονται στο g-chase^{vg}(Σ,D), όπου το vg εξαρτάται μόνο από τα Q και R.

Μέσω των (3) και (4) αποδεικνύεται ότι τα φρουρούμενα TGDs έχουν την BGDP ιδιότητα. Άρα, η απάντηση σε BCQ μέσω φρουρούμενων TGDs γίνεται σε πολυωνυμικό χρόνο, εφόσον από τον ομομορφισμό λ στα (4) και (5) Q(g-chase^k(Σ,D)) = Q(chase(Σ,D)) = ans(Q,Σ,D). Μπορεί να επιτευχθεί και σε γραμμικό χρόνο αν το Q είναι ατομικό.

Έχοντας ορίσει την φρουρούμενη Datalog, μπορεί να οριστεί και η γραμμική Datalog:

Ένα TGD είναι γραμμικό (linear) εάν και μόνο εάν περιέχει ένα και μοναδικό άτομο στο σώμα του. Ένα σύνολο από TGDs Σ έχει την ιδιότητα του φραγμένου βάθους παραγωγής (bounded derivation-depth property-BDDP), εάν και μόνο εάν όποτε D⊔Σ|= Q, τότε $chase^{vd}$ (Σ,D) |= Q, όπου το $\vee g$ εξαρτάται μόνο από τα Q και R.

Προφανώς, στην περίπτωση των γραμμικών TGDs, για κάθε a∈chase(Σ,D), το υπόδεντρο του a καθορίζεται μόνο από το a αντί του type(a). Επομένως για ένα μοναδικό άτομο, το βάθος του συμπίπτει με τον αριθμό που χρειάζεται να εφαρμοστεί ο κανόνας του TGD κυνηγητού μέχρι αυτό να παραχθεί. Έτσι, ως άμεση συνέπεια του ότι τα φρουρούμενα TGDs έχουν την BGDP ιδιότητα, και τα γραμμικά TGDs έχουν την BDDP ιδιότητα.

Ισχύει το θεώρημα: Έστω ένα σχεσιακό σχήμα R, ένα σύνολο Σ από TGDs στο R, μια βάση δεδομένων D στο R, και Q ένα BCQ στο R. Αν το Σ έχει την BDDP ιδιότητα, τότε το Q είναι επανεγράψιμο πρώτης τάξης (FO-rewritable).

Εφόσον αποδείχτηκε πριν ότι τα γραμμικά TGDs έχουν αυτή την ιδιότητα, τα BCQs είναι επανεγράψιμα πρώτης τάξης στην γραμμική περίπτωση.

Οποιαδήποτε από τις $\mathcal{DL}$-$Lite_{\mathcal{F}}$, $\mathcal{DL}$-$Lite_{\mathcal{R}}$, και $\mathcal{DL}$-$Lite_{\mathcal{A}}$ μπορεί να μεταφραστεί στην $Datalog_0^{\pm}$, που είναι η γραμμική $Datalog^{\pm}$ που ορίστηκε παραπάνω, με την προσθήκη αρνητικών περιορισμών (negative constraints) και μη-συγκρουόμενων κλειδιών (non-conflicting keys). Από εκεί και πέρα μπορούμε εύκολα να αποφασίσουμε για την σχέση υπαγωγής μεταξύ δύο εννοιών, έστω C_1 και C_2, με τον ακόλουθο τρόπο. Έστω P_1 και P_2 τα κατηγορήματα που αντιστοιχούν στις έννοιες C_1 και C_2 αντίστοιχα. Θέλουμε να ελέγξουμε αν η C_1 υπάγεται στην C_2. Τα βήματα της διαδικασίας είναι:

- Ορίζουμε μια σταθερά a η οποία ικανοποιεί το κατηγόρημα P_1, δηλαδή εισάγουμε ένα νέο γεγονός $P_1(a)$
- Αξιολογούμε το ερώτημα $P_2(a)$?
- Ορίζουμε μια σταθερά b που ικανοποιεί το κατηγόρημα P_2, δηλαδή εισάγουμε ένα νέο γεγονός $P_2(b)$
- Αξιολογούμε το ερώτημα $P_1(b)$?

Προφανώς, σε περίπτωση που το ερώτημα $P_2(a)$? δώσει θετική απάντηση και το ερώτημα $P_1(b)$? δώσει αρνητική, τότε η έννοια C_1 υπάγεται στην έννοια C_2, δηλαδή το $C_1 \sqsubseteq C_2$ ισχύει.

Έχοντας για παράδειγμα τα σύνολα:

$$\mathbf{A}=\{\text{Scientist, Article, ConPaper, JouPaper}\}$$
$$\mathbf{R_A}=\{\text{has Author, hasFirstAuthor, isAuthorOf}\}$$
$$\mathbf{I}=\{i_1,i_2\}$$

τα παρακάτω αξιώματα (αριστερά του⇨) μετατρέπονται σε TGDs ως εξής:

ConPaper ⊑ Article	⇨	ConPaper(X) → Article(X)
JouPaper ⊑ Article	⇨	JouPaper(X) → Article(X)
ConPaper ⊑ ¬JouPaper	⇨	ConPaper(X) → ¬JouPaper(X)
Scientist ⊑ ∃isAuthorOf	⇨	Scientist(X) → ∃ZisAuthorOf(X,Z)
∃isAuthorOf ⊑ Scientist	⇨	isAuthorOf(X,Y) → Scientist(X)
$isAuthorOf^{-}$ ⊑ Article	⇨	isAuthorOf(Y,X) → Article(X)
$isAuthorOf^{-}$ ⊑ hasAuthor	⇨	isAuthorOf(Y,X) → has Author(X,Y)
$hasAuthor^{-}$ ⊑ isAuthorOf	⇨	hasAuthor(Y,X) → isAuthorOf(X,Y)
(funct hasFirstAuthor)	⇨	hasFirstAuthor(X,Y), hasFirstAuthor(X,Y') → Y=Y'
Scientist(i_1)	⇨	Scientist(i_1)
isAuthorOf(i_1,i_2)	⇨	isAuthorOf(i_1,i_2)
Article(i_2)	⇨	Article(i_2)

3.2.2 Μέθοδος Krötzsch

Η μέθοδος αυτή λειτουργεί στον σκληρό δίσκο και είναι κατάλληλη για τόσο για την ταξινόμηση, όσο και για τον έλεγχο στιγμιότυπων οντολογιών της περιγραφικής λογικής *SROEL(X)*, η οποία περιέχει τα κύρια χαρακτηριστικά της *OWL EL*.

Η περιγραφική λογική *SROEL(X)* βασίζεται στα σύνολα $\mathbf{N_I}$ ατόμων, $\mathbf{N_C}$ εννοιών και $\mathbf{N_R}$ ρόλων. Το σύνολο C από *SROEL(X)* έννοιες δίνεται ως:

$$\mathbf{C} ::= \top|\bot|\ \mathbf{N_C}\ |\mathbf{C}\sqcap\mathbf{C}|\exists\mathbf{N_R}.\mathbf{C}|\exists\mathbf{N_R}.\mathrm{Self}|\{\mathbf{N_I}\}$$

Οι ονοματικές έννοιες (nominals) {α} αντιπροσωπεύουν μονοσύνολα. Ένα αξίωμα *SROEL(X)* μπορεί να είναι ένας ισχυρισμός (assertion) C(a) ή R(a,b), μια υπαγωγή έννοιας C⊑D, ή μια υπαγωγή ρόλων που έχει μία από τις μορφές R ⊑ K, R o S ⊑ K, C x D ⊑ K, R ⊑ C xD, όπου τα C,D∈**C**, τα R,S,K∈ $\mathbf{N_R}$, και τα α,β∈ $\mathbf{N_I}$. Ο καθολικός ρόλος (universal role) U ορίζεται ως ⊤ x ⊤⊑U, ενώ ο κενός ρόλος N ως ∃N.⊤ ⊑ ⊥. Βάσεις γνώσεων (knowledge bases-KB)είναι σύνολα από *SROEL(X)* αξιώματα που ικανοποιούν τις ιδιότητες απλότητας των ρόλων (simplicity of roles) και των αποδεκτών περιορισμών εύρους (admissibility of roles).

Παρόμοια με τις παραγωγικές βάσεις δεδομένων (deductive databases) τα κατηγορήματα χωρίζονται σε άμεσα (extensional – EDB) και σε έμμεσα (intensional - IDB). Τα EDB χρησιμοποιούνται μόνο σε γεγονότα (facts) ή στα προθέματα (σώματα-bodies) των κανόνων, ενώ τα IDB χρησιμοποιούνται χωρίς περιορισμούς.

Όπως προαναφέρθηκε η μέθοδος που παρουσιάζεται εδώ είναι κατάλληλη και για έλεγχο στιγμιότυπου, δηλαδή έλεγχος του αν το C(α) συνεπάγεται για C ∈ $\mathbf{N_C}$ και a ∈ $\mathbf{N_I}$. Αυτό επιτυγχάνεται μέσω παραγωγικών κανόνων (deduction rules), που

χρησιμοποιούνται για την παραγωγή λογικών συμπερασμάτων και μπορούν να θεωρηθούν ως ένα μέσο εξαγωγής νέων σχέσεων από ήδη υπάρχουσες. Οι κανόνες αυτοί δημιουργούνται ως κανόνες Datalog που χρησιμοποιούν χαρακτηριστικά σύμβολα (signature symbols) στην θέση των όρων. Για παράδειγμα το αξίωμα $A \sqsubseteq B$ μπορεί να αναπαρασταθεί ως subclassOf(A,B) και να εισαχθεί ο κανόνας subclassOf(x,y)∧subclassOf(y,z)→ subclassOf(x,z). Οπότε τα αρχικά αξιώματα μετατρέπονται σε σύνολα από Datalog EDB γεγονότα.

Μετά την μετάφραση των αξιωμάτων της οντολογίας (η οποία θεωρείται ότι έχει αρχικά κανονικοποιηθεί) σε Datalog γεγονότα, όλα τα συμπεράσματα αυτών των γεγονότων μπορούν να υπολογιστούν μέσω των παραγωγικών κανόνων με μια προσέγγιση από κάτω προς τα πάνω (bottom-up) και όλες οι συνεπαγωγές μπορούν να ελεγχθούν χωρίς παραπάνω επαναληπτικούς υπολογισμούς. Η υλοποίηση που γίνεται εδώ θεωρεί μόνο *SROEL(X)* αξιώματα που έχουν μία εκ των μορφών που παρουσιάζονται στο Σχήμα 3.14 (αριστερά του ↦). Για μια κανονικοποιημένη βάση γνώσης KB, ορίζεται μια Datalog θεωρία $P(KB):=P_{inst} \sqcup \{I_{inst}(\alpha)|\alpha \in KB\} \sqcup \{I_{inst}(s)|s \in \mathbf{N_I} \sqcup \mathbf{N_C} \sqcup \mathbf{N_R}\}$, όπου είναι οι παραγωγικοί κανόνες στο Σχήμα 3.15. Προκύπτει έτσι ο υπολογισμός παραγωγής (derivation calculus) K_{inst} : ο K_{inst} παράγει ένα γεγονός C(α) από την KB αν η P(KB) περιέχει το isa(α,C).

Τα IDB κατηγορήματα isa, spo και self στην P_{inst} αντιστοιχούν σε αξιώματα του ABox για ατομικές έννοιες, ρόλους και έννοιες ∃R.Self αντίστοιχα. Ο κανόνας 1 είναι ο εναρκτήριος κανόνας που είναι υπεύθυνος για την παραγωγή των πρώτων isa γεγονότων. Ο κανόνας 2 εκφράζει την μόνη περίπτωση όπου αυτοπαθή (reflexive) spo γεγονότα οδηγούν σε self γεγονότα. Οι κανόνες από τον 3 μέχρι τον 24 εκφράζουν τις αναμενόμενες παραγωγές (expected derivations) για κάθε ένα από τους τύπους των αξιωμάτων όπως έχουν κωδικοποιηθεί από τα EDB κατηγορήματα. Ο κανόνας 4 ελέγχει για ασυνέπειες, ενώ οι κανόνες 9 και 10 χρησιμοποιούν βοηθητικές σταθερές $e^{A \sqsubseteq \exists R.B}$ για την διαχείριση των υπαρξιακών, όπου μια τέτοια σταθερά αντιπροσωπεύει την κλάση όλων των διάδοχων ρόλων που παράγονται από το αξίωμα από το οποίο προέρχεται. Οι εναπομείναντες κανόνες 25 μέχρι 27 κωδικοποιούν την συλλογιστική ανάλυση της ισότητας (equality reasoning) που είναι σχετική στην περίπτωση των ονοματικών εννοιών όπου οι δηλώσεις isa(α,b) κωδικοποιούν την ισότητα των α και b.

Η κανονικοποίηση των αξιωμάτων και ο υπολογισμός του I_{inst} επιτυγχάνονται σε γραμμικό χρόνο, ενώ ο χρόνος για την συλλογιστική ανάλυση είναι πολυωνυμικός σε σχέση με το μέγεθος των γεγονότων. Η υλοποίηση δεν υποστηρίζει την διαχείριση των *OWL EL* κλειδιών (keys) και των ιδιοτήτων των τύπων δεδομένων (datatype properties).

$C(\alpha)$	$\mapsto$	$SubClass(\alpha,C)$	$R(\alpha,b)$	$\mapsto$	$Ex^{-}(\alpha,R,b,b)$
$A \sqsubseteq C$	$\mapsto$	$SubClass(A,C)$	$A \sqcap B \sqsubseteq C$	$\mapsto$	$Conj^{-}(A,B,C)$
$\top \sqsubseteq C$	$\mapsto$	$Top(C)$	$A \sqsubseteq \exists R.Self$	$\mapsto$	$Self^{+}(A,R)$
$A \sqsubseteq \bot$	$\mapsto$	$Bot(A)$	$\exists R.Self \sqsubseteq C$	$\mapsto$	$Self^{-}(R,C)$
$\exists R.A \sqsubseteq C$	$\mapsto$	$Ex^{-}(R,A,C)$	$\{a\} \sqsubseteq C$	$\mapsto$	$SubClass(\alpha,C)$
$A \sqsubseteq \exists R.B$	$\mapsto$	$Ex^{+}(A,R,B,e^{A \sqsubseteq \exists R.B})$	$A \sqsubseteq \{c\}$	$\mapsto$	$SubClass(A,C)$
$R \sqsubseteq T$	$\mapsto$	$SubRole(R,T)$	$R \sqsubseteq C \times D$	$\mapsto$	$RProd^{+}(R,C,D)$
$R \circ S \sqsubseteq T$	$\mapsto$	$RChain^{-}(R,S,T)$	$A \times B \sqsubseteq R$	$\mapsto$	$RProd^{-}(A,B,R)$

$\alpha \in N_1 \mapsto Nom(a)$ $\quad$ $A \in N_C \mapsto Cls(A)$ $\quad$ $R \in N_R \mapsto Rol(R)$

$A,B,C,D \in N_C$,$R,S,T \in N_R$, $\alpha,b,c \in N_1$

Σχήμα 3.14. Η μετάφραση της εισόδου I_{inst}

(1)	$\mathrm{Nom}(x)$	$\rightarrow$	$\mathrm{isa}(x,x)$
(2)	$\mathrm{Nom}(x) \wedge \mathrm{spo}(x,v,x)$	$\rightarrow$	$\mathrm{isa}(x,v)$
(3)	$\mathrm{Top}(z) \wedge \mathrm{isa}(x,z^{\cdot})$	$\rightarrow$	$\mathrm{isa}(x,z)$
(4)	$\mathrm{Bot}(z) \wedge \mathrm{isa}(u,z) \wedge \mathrm{isa}(x,z^{\cdot}) \wedge \mathrm{Cls}(y)$	$\rightarrow$	$\mathrm{isa}(x,y)$
(5)	$\mathrm{SubClass}(y,z) \wedge \mathrm{isa}(x,y)$	$\rightarrow$	$\mathrm{isa}(x,z)$
(6)	$\mathrm{Conj}^{-}(y_1,y_2,z) \wedge \mathrm{isa}(x,y_1) \wedge \mathrm{isa}(x,y_2)$	$\rightarrow$	$\mathrm{isa}(x,z)$
(7)	$\mathrm{Ex}^{-}(v,y,z) \wedge \mathrm{spo}(x,v,x^{\cdot}) \wedge \mathrm{isa}(x^{\cdot},y)$	$\rightarrow$	$\mathrm{isa}(x,z)$
(8)	$\mathrm{Ex}^{-}(v,y,z) \wedge \mathrm{self}(x,v) \wedge \mathrm{isa}(x,y)$	$\rightarrow$	$\mathrm{isa}(x,z)$
(9)	$\mathrm{Ex}^{+}(y,v,z,x^{\cdot}) \wedge \mathrm{isa}(x,y)$	$\rightarrow$	$\mathrm{spo}(x,v,x^{\cdot})$
(10)	$\mathrm{Ex}^{+}(y,v,z,x^{\cdot}) \wedge \mathrm{isa}(x,y)$	$\rightarrow$	$\mathrm{isa}(x^{\cdot},z)$
(11)	$\mathrm{Self}^{-}(v,z) \wedge \mathrm{self}(x,v)$	$\rightarrow$	$\mathrm{isa}(x,z)$
(12)	$\mathrm{Self}^{+}(y,v) \wedge \mathrm{self}(x,y)$	$\rightarrow$	$\mathrm{self}(x,v)$
(13)	$\mathrm{SubRole}(v,w) \wedge \mathrm{spo}(x,v,x^{\cdot})$	$\rightarrow$	$\mathrm{spo}(x,w,x^{\cdot})$
(14)	$\mathrm{SubRole}(v,w) \wedge \mathrm{self}(x,v)$	$\rightarrow$	$\mathrm{self}(x,w)$
(15)	$\mathrm{RChain}^{-}(u,v,w) \wedge \mathrm{spo}(x,u,y) \wedge \mathrm{spo}(y,v,z)$	$\rightarrow$	$\mathrm{spo}(x,w,z)$
(16)	$\mathrm{RChain}^{-}(u,v,w) \wedge \mathrm{self}(x,u) \wedge \mathrm{spo}(x,v,x^{\cdot})$	$\rightarrow$	$\mathrm{spo}(x,w,x^{\cdot})$
(17)	$\mathrm{RChain}^{-}(u,v,w) \wedge \mathrm{spo}(x,u,x^{\cdot}) \wedge \mathrm{self}(x^{\cdot},v)$	$\rightarrow$	$\mathrm{spo}(x,w,x^{\cdot})$
(18)	$\mathrm{RChain}^{-}(u,v,w) \wedge \mathrm{self}(x,u) \wedge \mathrm{self}(x,v)$	$\rightarrow$	$\mathrm{spo}(x,w,x)$
(19)	$\mathrm{RProd}^{-}(y_1,y_2,w) \wedge \mathrm{isa}(x,y_1) \wedge \mathrm{isa}(x^{\cdot},y_2)$	$\rightarrow$	$\mathrm{spo}(x,w,x^{\cdot})$
(20)	$\mathrm{RProd}^{-}(y_1,y_2,w) \wedge \mathrm{isa}(x,y_1) \wedge \mathrm{isa}(x,y_2)$	$\rightarrow$	$\mathrm{self}(x,w)$
(21)	$\mathrm{RProd}^{+}(v,z_1,z_2) \wedge \mathrm{spo}(x,v,x^{\cdot})$	$\rightarrow$	$\mathrm{isa}(x,z_1)$
(22)	$\mathrm{RProd}^{+}(v,z_1,z_2) \wedge \mathrm{self}(x,v)$	$\rightarrow$	$\mathrm{isa}(x,z_1)$
(23)	$\mathrm{RProd}^{+}(v,z_1,z_2) \wedge \mathrm{spo}(x,v,x^{\cdot})$	$\rightarrow$	$\mathrm{isa}(x^{\cdot},z_2)$
(24)	$\mathrm{RProd}^{+}(v,z_1,z_2) \wedge \mathrm{self}(x,v)$	$\rightarrow$	$\mathrm{isa}(x,z_2)$
(25)	$\mathrm{isa}(x,y) \wedge \mathrm{Nom}(y) \wedge \mathrm{isa}(x,z)$	$\rightarrow$	$\mathrm{isa}(y,z)$
(26)	$\mathrm{isa}(x,y) \wedge \mathrm{Nom}(y) \wedge \mathrm{isa}(y,z)$	$\rightarrow$	$\mathrm{isa}(x,z)$
(27)	$\mathrm{isa}(x,y) \wedge \mathrm{Nom}(y) \wedge \mathrm{spo}(z,u,x)$	$\rightarrow$	$\mathrm{spo}(z,u,y)$

Σχήμα 3.15. Οι παραγωγικοί κανόνες P_{inst}

Η δεύτερη λειτουργία που μπορεί να επιτευχθεί μέσω της μεθόδου Krötzsch είναι η ταξινόμηση των κλάσεων (class subsumption), ο υπολογισμός δηλαδή όλων των υπονοούμενων από την βάση γνώσης σχέσεων υπαγωγής μεταξύ των ατομικών κλάσεων. Η ταξινόμηση μπορεί να απλοποιηθεί σε έλεγχο στιγμιότυπου: το $KB \models A \sqsubseteq B$ ισχύει αν $KB\{A(c)\} \models B(c)$ για ένα νέο c. Η διαδικασία αυτή απαιτεί αλλαγές

πάνω στην βάση γνώσης, οδηγώντας σε καινούργιες συνεπαγωγές, ακόμα και σε ασυνέπειες (iconsistencies). Έτσι η K_{inst} δεν μπορεί να χρησιμοποιηθεί απευθείας για ταξινόμηση και ουσιαστικά χρειάζεται ένα ξεχωριστό τρέξιμο της για κάθε υπόθεση (assumption) A(c), για να υπολογιστούν όλες οι συνεπαγωγές της μορφής $A \sqsubseteq B$.

Μια λύση είναι η «εσωτερικοποίηση» των τρεξιμάτων της K_{inst} , προεκτείνοντας όλα τα IDB κατηγορήματα με μια ακόμα παράμετρο που θα κωδικοποιεί την υπόθεση για την οποία ισχύει η συνεπαγωγή. Η υλοποίηση αυτή όμως δεν είναι πολύ αποδοτική γιατί παράγονται αληθή αποτελέσματα για κάθε υπόθεση A(c), άρα ο αριθμός των γεγονότων που παράγονται πολλαπλασιάζεται όσο πιο πολλές είναι οι κλάσεις.

Το πρόβλημα αυτό έχει ως αποτέλεσμα την αύξηση των ορισμάτων των παραγόμενων κατηγορημάτων από 3 σε 4, που οδηγεί και σε αυξημένες απαιτήσεις χώρου αποθήκευσης. Άρα ο αριθμός των ορισμάτων των παραγόμενων κατηγορημάτων είναι ένα καλό μέτρο σύγκρισης αποδοτικότητας, και μια υλοποίηση που χρησιμοποιεί 2 ή 3 τέτοια ορίσματα και ήταν ορθή και πλήρης για την ταξινόμηση της *SROEL(X)* θα ήταν πολύ αποδοτική. Με την βοήθεια των αποδεικτικών δέντρων (proof trees) αποδεικνύεται ότι τέτοια υλοποίηση δεν υπάρχει:

- Έστω L μια περιγραφική λογική με υπαγωγές γενικών εννοιών, υπαρξιακές ποσοτικοποιήσεις και αλυσίδες ρόλων. Κάθε υλοποίηση ταξινόμησης ή ελέγχου στιγμιότυπου για να είναι ορθή και πλήρης χρειάζεται ο αριθμός των ορισμάτων των παραγόμενων κατηγορημάτων να είναι τουλάχιστον 3.
- Έστω L μια περιγραφική λογική με υπαγωγές γενικών εννοιών, υπαρξιακές ποσοτικοποιήσεις και ονοματικές κλάσεις. Κάθε υλοποίηση ταξινόμησης για να είναι ορθή και πλήρης χρειάζεται ο αριθμός των ορισμάτων των παραγόμενων κατηγορημάτων να είναι τουλάχιστον 3.
- Έστω L μια περιγραφική λογική με υπαγωγές γενικών εννοιών, υπαρξιακές ποσοτικοποιήσεις, αλυσίδες ρόλων και ονοματικές κλάσεις. Κάθε υλοποίηση ταξινόμησης για να είναι ορθή και πλήρης χρειάζεται ο αριθμός των ορισμάτων των παραγόμενων κατηγορημάτων να είναι τουλάχιστον 4.

Εδώ καλύπτεται ουσιαστικά η ταξινόμηση ενός μικρότερου μέρους της *SROEL(X)* (συγκεκριμένα καλύπτονται όλες οι *OWL EL* οντολογίες που δεν περιέχουν ιδιότητες τύπων δεδομένων και κανένα από τα owl:Thing, owl:topObjectProperty, owl:bottomObjectProperty, owl:ObjectHasValue, ObjectOneOf, και HasKey) :

Έστω η υλοποίηση K_{scc} με το I_{scc} ορισμένο όπως το I_{inst} του Σχήματος 3.14, αλλά όχι για αξιώματα που περιέχουν ονοματικές έννοιες, $\top$, $\bot$, ή παράγωγα εννοιών (concept products) στην αριστερή πλευρά, και το πρόγραμμα Pscc που αποτελείται από τους κανόνες 1,2,5-18,21-24 του Σχήματος 3.15 μαζί με τον καινούργιο κανόνα $Cls(z) \rightarrow isa(z,z)$.

Για μία βάση γνώσης KB τέτοια ώστε το $I_{scc}(\alpha)$ ορίζεται για όλα τα $\alpha \in KB$, ορίζεται $P(KB)Q = P_{scc} \sqcup \{I_{scc}(\alpha) | \alpha \in KB\} \sqcup \{I_{scc}(s) | s \in \mathbf{N_I} \sqcup \mathbf{N_C} \sqcup \mathbf{N_R}\}$. Τότε για όλα τα $A,B \in \mathbf{N_C}$, η KB συνεπάγει $A \sqsubseteq B$ εάν και μόνο εάν η P(KB) συνεπάγει isa(A,B), όποτε η P(KB) ορίζεται. Άρα η K_{scc} υλοποιεί την ταξινόμηση για όλες τις SROEL(X) βάσεις

γνώσης που περιέχουν μόνο ⊓ (για έννοιες και ρόλους),∃ , Self, ο, και παράγωγα εννοιών στην δεξιά πλευρά.

3.2.3 Το σύστημα Orel

Το σύστημα Orel είναι σχεδιασμένο για την ταξινόμηση οντολογιών εκφραστικότητας $\mathcal{EL}$ και $\mathcal{RL}$ μέσω της χρήσης σχεσιακών συστημάτων διαχείρισης βάσεων (relational database management systems –RDBMS), και λειτουργεί στον σκληρό δίσκο. Έχει υλοποιηθεί στην γλώσσα προγραμματισμού Java και χρησιμοποιεί το σχεσιακό σύστημα διαχείρισης βάσεων MySql και το *OWL* API για την επεξεργασία *OWL* αρχείων.

Στο Σχήμα 3.16 παρουσιάζεται η κλασσική προσέγγιση μετάφρασης μιας γνώσης βάσης σε Datalog, όπου οι κλάσεις *OWL* μεταφράζονται σε μοναδιαία κατηγορήματα και οι ιδιότητες σε δυαδικά, ενώ επειδή η Datalog δεν επιτρέπει υπαρξιακές εξαρτήσεις, για την αναπαράσταση της κάθε σχέσης ObjectSomeValuesFrom εισάγεται μια σταθερά.

PhD ⊑ AcademicDegree PostDoc ⊑∃has.PhD Graduate ≡ ∃has.AcademicDegree

PhD(x)	→	AcademicDegree(x)		Graduate(x)	→	has(x,$d_{\exists has.AD}$)
PostDoc(x)	→	has(x,$d_{\exists has.PhD}$)		Graduate(x)	→	AcademicDegree($d_{\exists has.AD}$)
PostDoc(x)	→	PhD($d_{\exists has.PhD}$)	has(x,y) ∧ AcademicDegree(y)	→	Graduate(x)	

Σχήμα 3.16. Παράδειγμα μετάφρασης σε Datalog

Ακολουθώντας την μετάφραση αυτή το σύνολο των κανόνων που δημιουργείται μπορεί να γίνει πολύ μεγάλο, αφού αυξάνει γραμμικά με το μέγεθος της βάσης γνώσης. Επειδή το σύστημα Orel όμως χρησιμοποιεί RDBMS, όπου συμφέρει μεγάλος αριθμός δεδομένων και μικρότερος αριθμός κανόνων, ακολουθείται μια διαφορετική προσέγγιση. Η οντολογική πληροφορία (ontological information), όπως οι σχέσεις υπαγωγής, αποθηκεύονται ως γεγονότα, ενώ οι λογικές διακλαδώσεις (logical ramifications) αντιπροσωπεύονται από κανόνες που αντιστοιχούν στους κανόνες μιας παραγωγικής διαδικασίας (deduction calculus). Έτσι το παραπάνω παράδειγμα κωδικοποιείται ως εξής:

subclass(PhD, AcademicDegree)
someValues(PostDoc, Has, PhD, $d_{\exists has.AD}$)
someValues(Graduate, Has, AcademicDegree, $d_{\exists has.AD}$)
subSomeValues(Has, Academic, Degree, Graduate)

Στη συνέχεια η σημασιολογία μπορεί να κωδικοποιηθεί σε παραγωγικούς κανόνες όπως:

subClass(a,b∧inst(x,a)→ inst(x,b)
someValues(a,r,b,d)∧ inst(x,a) →triple(x,r,d)
someValues(a,r,b,d)∧ inst(x,a) →inst(d,b)
subSomeValues(r,a,b)∧ triple(x,r,y)∧ inst(y,a) →inst(x,b)

όπου το κατηγόρημα inst χρησιμοποιείται για στιγμιότυπα κλάσεων (class instances), το κατηγόρημα triple για ισχυρισμούς ρόλων (role assertions), a,b,c,d είναι κλάσεις, r,s,t ρόλοι, x,y,z ατομικές έννοιες και d βοηθητικές σταθερές.

Στο Σχήμα 3.17 παρουσιάζεται η μετάφραση των αξιωμάτων σε γεγονότα για τις $\mathcal{EL}$ και $\mathcal{RL}$, ενώ στο Σχήμα 3.18 φαίνονται οι αντίστοιχοι κανόνες. Στα δύο αυτά σχήματα χρησιμοποιούνται τα ονόματα των κλάσεων, των ρόλων, των ατομικών εννοιών, της καθολικής έννοιας T και της κενής έννοιας ⊥ ως σταθερές, ενώ στην υλοποίηση στην βάση τους έχουν δοθεί αριθμοί-ταυτότητες (numerical identifiers) για την αναπαράστασή τους. Θεωρείται ότι πριν την μετάφραση τα αξιώματα έχουν πρώτα κανονικοποιηθεί.

A(n)	↦	inst(n,A)	$A \sqsubseteq C$	↦	subClass(A,C)
R(n,m)	↦	triple(n,R,m)	$A \sqcap B \sqsubseteq C$	↦	subIntersect($\hat{A}$,$\hat{B}$,$\hat{C}$)
$\exists R.Self(n)$	↦	self(n,R)	$\exists R.Self \sqsubseteq C$	↦	selfImplies(R,$\hat{C}$)
$\exists R.A \sqsubseteq C$	↦	subSomeValues(R,$\hat{A}$,$\hat{C}$)	$A \sqsubseteq \exists R.Self$	↦	impliesSelf($\hat{A}$,R)
$A \sqsubseteq \exists R.B$	↦	someValues($\hat{A}$,R,$\hat{B}$, $d_{\exists R.B}$)	$A \sqsubseteq \leq 1 R.B$	↦	atMostOne($\hat{A}$,R,$\hat{B}$)
$A \sqsubseteq \forall R.B$	↦	allValuesFrom($\hat{A}$,R,$\hat{B}$)	Disj(R,S)	↦	disjoint(R,S)
$R \sqsubseteq T$	↦	subProperty(R,T)	$R^- \sqsubseteq S$	↦	subInverseOf(R,S)
$R \circ S \sqsubseteq T$	↦	subPropertyChain(R,S,T)			
Για κάθε άτομο n της οντολογίας, προστίθεται το γεγονός nom(n) στην μετάφραση.					
Για κάθε κλάση ή ονοματική κλάση A της οντολογίας, προστίθεται το γεγονός subclass(a,⊤).					

Σχήμα 3.17. Δημιουργία μιας βάσης γεγονότων από αξιώματα περιγραφικής λογικής στην Orel; για μια κλάση C ορίζεται $\hat{C}$:= n αν C={n} είναι μία ονοματική κλάση, και C :=$\hat{C}$ αν το C είναι κλάση, η καθολική ή η κενή έννοια

(1)	$nom(x)$	$\mapsto$	$inst(x,x)$
(2)	$nom(x) \wedge triple(x,r,x)$	$\mapsto$	$self(x,r)$
(3)	$subClass(\alpha,b) \wedge inst(x,\alpha)$	$\mapsto$	$inst(x,b)$
(4)	$subIntersect(\alpha,b,c) \wedge inst(x,\alpha) \wedge inst(x,b)$	$\mapsto$	$inst(x,c)$
(5)	$subSomeValues(r,a,c) \wedge triple(x,r,y) \wedge inst(y,a)$	$\mapsto$	$inst(x,c)$
(6)	$someValues(\alpha,p,b,d) \wedge inst(x,\alpha)$	$\mapsto$	$triple(x,p,d)$
(7)	$someValues(\alpha,p,b,d) \wedge inst(x,\alpha)$	$\mapsto$	$inst(d,b)$
(8)	$selfImplies(r,c) \wedge self(x,r)$	$\mapsto$	$inst(x,c)$
(9)	$impliesSelf(\alpha,r) \wedge inst(x,\alpha)$	$\mapsto$	$self(x,r)$
(10)	$impliesSelf(\alpha,r) \wedge inst(x,\alpha)$	$\mapsto$	$triple(x,r,x)$
(11)	$subProperty(r,t) \wedge triple(x,r,y)$	$\mapsto$	$triple(x,t,y)$
(12)	$subProperty(r,t) \wedge self(x,r)$	$\mapsto$	$self(x,t)$
(13)	$subPropertyChain(r,s,t) \wedge triple(x,r,y) \wedge triple(y,s,z)$	$\mapsto$	$triple(x,t,z)$
(14)	$disjoint(r,s) \wedge triple(x,r,y) \wedge triple(x,s,y)$	$\mapsto$	$inst(x,\bot)$
(15)	$inst(x,y) \wedge nom(y)$	$\mapsto$	$inst(y,x)$
(16)	$inst(x,y) \wedge nom(y)$	$\mapsto$	$nom(x)$
(17)	$triple(x_1,r,y) \wedge inst(x_2,y) \wedge nom(y)$	$\mapsto$	$triple(x_1,r,x_2)$
(18)	$allValuesFrom(\alpha,r,b) \wedge nom(x) \wedge nom(y) \wedge triple(x,r,y) \wedge inst(x,\alpha)$	$\mapsto$	$inst(y,b)$
(19)	$atMostOne(\alpha,r,b) \wedge nom(x) \wedge nom(y_1) \wedge nom(y_2) \wedge inst(x,\alpha) \wedge triple(x,r,y_1) \wedge inst(y_1,b) \wedge triple(x,r,y_2) \wedge inst(y_2,b)$	$\mapsto$	$inst(y_1,y_2)$
(20)	$subInverseOf(r,s) \wedge nom(x) \wedge nom(y) \wedge triple(x,r,y)$	$\mapsto$	$triple(y,s,x)$

Σχήμα 3.18. Κανόνες για την παραγωγή συμπερασμάτων στην Orel

Τα μόνο χαρακτηριστικά που λείπουν από τα σχήματα 3.17 και 3.18 είναι τα εύρη ιδιοτήτων (property ranges), ο καθολικός ρόλος (universal role) και τα πεδία ορισμού (concrete domains). Το σύστημα Orel μετατρέπει τα εύρη ιδιοτήτων σε αξιώματα της μορφής $T \sqsubseteq \forall R.C$, ενώ δεν υποστηρίζει τον καθολικό ρόλο. Τα πεδία ορισμών υποστηρίζονται με ένα επιπλέον σύνολο κανόνων, ενώ άλλα χαρακτηριστικά που έχουν παραληφθεί, όπως για παράδειγμα η ασυμμετρία ρόλων (asymmetry of roles) μπορούν να εκφραστούν μέσω των χαρακτηριστικών που υποστηρίζονται.

Το σύστημα διασφαλίζει ότι οι τυχόν ασυνέπειες (inconsistencies) οδηγούν στην παραγωγή κατηγορημάτων της μορφής $inst(n,\bot)$ για μια σταθερά n, και ελέγχει για την συνθήκη αυτή, ώστε να μπορεί να επιστρέψει σωστές απαντήσεις χωρίς να υλοποιεί όλα τα πιθανά συμπεράσματα στην βάση δεδομένων.

Με την παραπάνω μετατροπή των αξιωμάτων, το σύστημα μπορεί εύκολα να απαντήσει σε ελέγχους στιγμιότυπων ατομικών εννοιών. Για τις σύνθετες έννοιες χρειάζεται όμως η επέκταση της βάσης γνώσης με βοηθητικά αξιώματα και έπειτα η επανεκκίνηση της διαδικασίας. Επειδή όμως τα βοηθητικά αξιώματα δεν επηρεάζουν την σημασιολογία της βάσης γνώσης, μπορεί να γίνουν πολλοί τέτοιοι έλεγχοι πριν χρειαστεί η επανεκκίνηση.

Όσον αφορά τον έλεγχο υπαγωγής εννοιών, το σύστημα δεν ακολουθεί την κλασική προσέγγιση όπου το πρόβλημα μετατρέπεται σε έλεγχο στιγμιότυπου (για τον έλεγχο $A \sqsubseteq B$ εισάγεται στην βάση το άτομο c και ο ισχυρισμός A(c), οπότε αν ισχύει B(c), ισχύει και το $A \sqsubseteq B$). Η διαδικασία αυτή αποφεύγεται γιατί δεν διατηρεί την σημασιολογία της βάσης, αφού η εισαγωγή του A(c) σε αυτήν μπορεί να οδηγήσει σε ασυνέπεια. Έτσι απαγορεύεται η ταυτόχρονη εκτέλεση πολλών ερωτημάτων (ενώ στις βάσεις δεδομένων η ταυτόχρονη εκτέλεση ερωτημάτων είναι σύνηθες), ενώ χρειάζονται διαγραφές δεδομένων μετά τα ερωτήματα. Ταυτόχρονα δεν είναι δυνατή η εκτέλεση του κάθε ερωτήματος σε ξεχωριστό αντίγραφο της βάσης γιατί οι απαιτήσεις αποθήκευσης δεδομένων για την περίπτωση αυτή είναι πολύ μεγάλες.

Λόγω των προβλημάτων αυτών, το σύστημα ακολουθεί μια συνδυαστική προσέγγιση για τον έλεγχο υπαγωγής εννοιών. Χρησιμοποιούνται οι απλοί κανόνες όταν είναι σίγουρο ότι θα εκφέρουν αποτέλεσμα, αλλά δημιουργούνται επιπλέον αντίγραφα των αξιωμάτων όταν παράγονται αποτελέσματα που δεν μπορούν να χειριστούν με αυτόν τον τρόπο. Μέχρι τώρα ο Orel είναι πιο αποδοτικός όταν οι οντολογίες δεν περιέχουν ονοματικές κλάσεις, ενώ η απόδοσή του μειώνεται όταν υπάρχουν συνδυασμοί ονοματικών εννοιών, υπαρξιακός ποσοτικοποιητής (existencial quantifier) και χαρακτηριστικών της $\mathcal{RL}$.

Ο Orel χρησιμοποιεί την Mysql αποθηκεύοντας την επέκταση κάθε datalog κατηγορήματος σε ένα πίνακα στην βάση, οι στήλες του οποίου αντιστοιχούν στα ορίσματα του κατηγορήματος. Για παράδειγμα, με αυτόν τον τρόπο, ο κανόνας (3) του Σχήματος 3.18 μπορεί να υλοποιηθεί με το ακόλουθο SQL ερώτημα:

```
INSERT INTO inst(x,y) SELECT t1.x AS x, t2.y AS y
FROM subclass AS t1 INNER JOIN inst AS t2 ON t1.x=t2.y
```

Η εκτέλεση του παραπάνω ερωτήματος επεκτείνει τον πίνακα inst με όλα τα γεγονότα που μπορούν να παραχθούν σε μία εφαρμογή του κανόνα (3). Η επαναληπτική εφαρμογή του οδηγεί σε μεγάλο αριθμό περιττών υπολογισμών, αφού σε κάθε επανάληψη παράγει τα ίδια αποτελέσματα.

Για την επίλυση αυτού του προβλήματος χρησιμοποιείται η ημι-αφελής από κάτω προς τα πάνω αξιολόγηση (semi-naïve bottom-up evaluation), όπου κρατείται το βήμα στο οποίο παράχθηκε το αξίωμα και η εφαρμογή των κανόνων απαιτεί καινούργια γεγονότα. Γράφοντας $inst_i$ για τον συμβολισμό του κατηγορήματος που αντιστοιχεί στην επέκταση του inst στο βήμα i, η στρατηγική αυτή συνοψίζεται στην αποτίμηση του κανόνα:

$$subClass(x,y) \wedge inst^{i}(y,z) \rightarrow inst^{i+1}(x,z)$$

Παρόλα αυτά εξακολουθεί να παράγεται ένας μεγάλος αριθμός περιττών γεγονότων.

Για την αξιολόγηση του συστήματος Orel (το οποίο δεν είναι πλήρως βελτιστοποιημένο) χρησιμοποιήθηκε η οντολογία SNOMED-CT που περιέχει 425.000 αξιώματα. Για την φόρτωση της οντολογίας και την αποθήκευσή της στην MySQL ο Orel χρειάζεται (ανάλογα με τα διαφορετικά υπολογιστικά συστήματα) περίπου 9 με 20 λεπτά, με κύρια αιτία του μεγάλου χρόνου φόρτωσης την διαδικασία εγγραφής των δεδομένων στην βάση, ενώ αντίστοιχα για την ταξινόμηση χρειάζεται περίπου 2 ώρες.

3.3 Συνδυαστικές Μέθοδοι

Στην ενότητα αυτή παρουσιάζεται μια συνδυαστική μέθοδος που προτάθηκε από τους Haarslev, Möller και Wandelt και η οποία συνδυάζει την μέθοδο της Δομικής Υπαγωγής με τα tableau συστήματα συλλογιστικής ανάλυσης. Επίσης εξαιρετικό ενδιαφέρον για όλα τα συστήματα που παρουσιάζονται το κεφάλαιο 3 έχει η δουλειά που παρουσιάζεται στο [AH08], όπου προτείνεται ένας αλγόριθμος παράλληλης ταξινόμησης. Μέσω του αλγορίθμου αυτού ερευνάται ο αντίκτυπος παραμέτρων όπως ο αριθμός των νημάτων (threads), ο αριθμός των εννοιών που θα επεξεργάζονται σε κάθε νήμα (καλείται επίσης μέγεθος διαμερισμού – partition size), και η στρατηγική που θα ακολουθηθεί για τον διαμερισμό του δοσμένου συνόλου εννοιών, στην πληρότητα του αλγορίθμου, αφού θεωρείται ότι η παραλληλοποίηση θυσιάζει σκόπιμα την πληρότητα.

3.3.1 Μέθοδος Haarslev, Möller και Wandelt

Η μέθοδος αυτή χρησιμοποιείται για την ταξινόμηση οντολογιών με TBoxes των οποίων το μεγαλύτερο μέρος των αξιωμάτων χρησιμοποιούν μια λιγότερο εκφραστική γλώσσα, όπως η $\mathcal{ELH}$, ενώ ένα μικρό τους μέρος χρησιμοποιεί μια γλώσσα εκφραστική όσο η $\mathcal{SHIQ}$. Εφόσον η μέθοδος αφορά tableau συστήματα συλλογιστικής ανάλυσης, η όλη διαδικασία γίνεται στην μνήμη.

Το πρόβλημα που αντιμετωπίζουν οι περισσότερες $\mathcal{SHIQ}$ τεχνικές ταξινόμησης είναι ότι, στην χειρότερη περίπτωση, ένα πρόβλημα υπαγωγής $C \sqsubseteq D$ έχοντας ένα TBox T, μετατρέπεται στο πρόβλημα $\neg SAT_T(C \sqcap \neg D)$, δηλαδή η υπαγωγή μετατρέπεται σε μη-ικανοποιησιμότητα. Η εισαγωγή της άρνησης μπορεί να εισάγει διαζεύξεις, και εφόσον οι περιγραφές εννοιών συνήθως εισάγουν πολλές τομές, λόγω της άρνησης, το αποτέλεσμα είναι πολλές διαζεύξεις. Στις βασισμένες σε tableau μηχανές συλλογιστικής ανάλυσης χρησιμοποιείται μια τεχνική που καλείται οκνηρή αξιολόγηση (lazy evaluation) με αποτέλεσμα την εισαγωγή ακόμα περισσότερων διαζεύξεων όταν οι έννοιες αντικαθιστούνται με τους ορισμούς τους.

Στις περισσότερες ταξινομήσεις Tbox πρέπει να γίνουν πολλοί έλεγχοι υπαγωγής και οι περισσότεροι επιστρέφουν λάθος, δηλαδή το $C \sqcap \neg D$ είναι ικανοποιήσιμο, ενώ το πρόβλημα υπαγωγής λύνεται με έναν εκθετικό αλγόριθμο,

λόγω της άρνησης. Γενικά ο μεγάλος αριθμός των ελέγχων υπαγωγής που απαιτούνται κάνει τους χρόνους ταξινόμησης μη αποδεκτούς.

Η παραπάνω προσέγγιση όταν δοκιμάστηκε με τον RacerPro χρειάστηκε μέρες για να ταξινομήσει σωστά μία παραλλαγή της SNOMED-CT που χρησιμοποιεί μόνο την γλώσσα $\mathcal{ELH}$ και καλείται SNOMED_ELH. Στη συνέχεια εφαρμόστηκε η τεχνική που είναι γνωστή ως «αναγνώριση των ολοκληρωτικά ορισμένων εννοιών» (identification of completely defined concepts – CD optimization). Η τεχνική αυτή βασίζεται στην παρατήρηση ότι στα περισσότερα Tboxes του είδους, για έναν αρκετά μεγάλο αριθμό εννοιών, τα αξιώματα ορίζουν απευθείας την σωστή θέση στην ταξονομία, δηλαδή τα παιδιά και οι γονείς μπορούν να αναγνωριστούν μέσω στατικής ανάλυσης). Ο χρόνος εξακολούθησε να είναι μη αποδεκτός, αφού απλά μειώθηκε από μέρες σε ώρες.

Μια άλλη τεχνική είναι η συγχώνευση ψευδομοντέλων (pseudo model merging), όπου κάθε έννοια συνδέεται με το λεγόμενο ψευδομοντέλο δομής δεδομένων (pseudo model data structure) $M=(L, \neg L, S^{\exists}, S^{\forall})$, όπου το L είναι ένα σύνολο θετικών εννοιών, το $\neg L$ αρνητικών, το $S^{\exists}$ είναι ένα σύνολο περιγραφών εννοιών της μορφής $\exists R.C$, και το $S^{\forall}$ ένα σύνολο περιγραφών εννοιών της μορφής $\forall R.C$. Η ιδέα του αλγορίθμου είναι το να δείξει ότι το C και το $\neg D$ δεν αλληλεπιδρούν ούτε μέσω θετικών και αρνητικών εννοιών, ούτε μέσω υπαρξιακών περιορισμών (existential restrictions) ή περιορισμών τιμών (value restrictions). Αν δεν υπάρχει αλληλεπίδραση τότε το $C \sqcap \neg D$ είναι ικανοποιήσιμο και το C δεν υπάγεται στο D. Ενώ η τεχνική αυτή είναι αρκετά αποδοτική για διάφορες περιγραφικές λογικές, τα αποτελέσματα είναι αποθαρρυντικά για την SNOMED-ELH. Εφόσον οι έλεγχοι υπαγωγής εισάγουν διαζεύξεις, πολλές φορές δοκιμάζονται τα «λάθος» ψευδομοντέλα και δεν παράγονται σωστά αποτελέσματα.

Η μέθοδος που παρουσιάζεται εδώ, βασιζόμενη στην παρατήρηση ότι οι περισσότερες βάσεις γνώσεις χρησιμοποιούν μη εκφραστικές γλώσσες στο μεγαλύτερο μέρος των Tboxes τους, αντί για συγχώνευση των ψευδομοντέλων που υπολογίζονται για το C και το $\neg D$, προσπαθεί να ενσωματώσει το D στο ψευδομοντέλο του C. Η τεχνική αυτή είναι η Δομική Υπαγωγή (structural subsumption). Τα ψευδομοντέλα που δημιουργούνται μπορούν να θεωρηθούν ως περιγραφικοί γράφοι (description graphs), οπότε ένας αλγόριθμος που ελέγχει αν οι περιγραφικοί γράφοι μπορούν να ενσωματωθούν, μπορεί να χρησιμοποιηθεί για έλεγχο.

Στη συνέχεια ένας απλός αλγόριθμος ενσωμάτωσης γράφων (graph-embedding algorithm) δουλεύει από κάτω προς τα πάνω (top-down), δέχεται δύο ψευδομοντέλα ως είσοδο και ελέγχει για ενσωματώσεις γράφων λαμβάνοντας υπόψη τους υπαρξιακούς περιορισμούς. Ο αλγόριθμος αυτός είναι εκθετικός στην χειρότερη περίπτωση.

Ουσιαστικά, εφόσον τα ψευδομοντέλα παράγονται έτσι και αλλιώς κατά την ταξινόμηση σε ένα βασισμένο σε tableau αλγόριθμο, η μέθοδος προσπαθεί να εκμεταλλευτεί αυτές τις δομές δεδομένων για τον έλεγχο Δομικής Υπαγωγής στην $\mathcal{ELH}$.

Κάθε $\mathcal{ELH}$ έννοια συνδέεται με ένα ψευδομοντέλο δομής δεδομένων M=(L,{}, $S^{\exists}$,{}), δηλαδή δεν υπάρχουν αρνητικές έννοιες και περιορισμοί τιμών. Θεωρείται ότι κάθε δομή δεδομένων pmodel (C) που συνδέεται με την έννοια C υπολογίζεται κατά απαίτηση.

Αν κατά τη διάρκεια της ταξινόμησης πρέπει να ελεγχθεί το αν το A_1 υπάγεται στο A_2, καλείται ο αλγόριθμος με είσοδο τα δύο ψευδομοντέλα pmodel(A_1) και pmodel(A_2) και ελέγχεται (τόσο μέσω των εννοιών στα L1 και L2, όσο και μέσω των υπαρξιακών στα $S^{\exists}_1$ και $S^{\exists}_2$) το αν μπορεί να ενσωματωθεί το ψευδομοντέλο του A_2 στο ψευδομοντέλο του A_1 . Παρατηρείται το εξής πρόβλημα:

Έστω τα παρακάτω αξιώματα

$$\{A \equiv \exists R.C\ ,\ A_1 \sqsubseteq B \sqcap \exists R.C\ ,\ A_2 \equiv nB \sqcap A\}$$

Είναι φανερό ότι το A_1 υπάγεται στο A_2, και τα ψευδομοντέλα τους είναι

$$\text{pmodel}(A_1) = (\{A_1,B\}, \{\}, \{\exists R.C\ \},\{\})$$

$$\text{pmodel}(A_2) = (\{A_2, B, A\}, \{\}, \{\exists R.C\ \}, \{\})$$

Παρατηρείται εύκολα όμως ότι το ψευδομοντέλο του A_2 δεν μπορεί να ενσωματωθεί στο ψευδομοντέλο του A_1 , λόγω των εννοιών A_2 και A στην λίστα. Για την επίλυση του προβλήματος αυτού χρησιμοποιείται μια διαδικασία (εκθετική στην χειρότερη περίπτωση) η οποία διαγράφει από το L όλες τις έννοιες για τις οποίες έχει δοθεί ορισμός (concept definition), όπως στο παράδειγμα τα A_2 και A, και κρατάει μόνο αυτές για τις οποίες δεν υπάρχει ορισμός ή υπάρχει μόνο μια γενική υπαγωγή έννοιας (general concept inclusion – GCI) με την έννοια στην αριστερή πλευρά. Η διαδικασία αυτή εφαρμόζεται πάντα στην έννοια που ελέγχεται για το αν υπάγεται σε μια άλλη. Δηλαδή αν ελέγχεται το $C \sqsubseteq D$, η διαδικασία θα εφαρμοστεί στο C.

Για να εφαρμοστεί ο αλγόριθμος πρέπει να πληρούνται κάποιες προϋποθέσεις. Οι προϋποθέσεις αυτές είναι:

- Να έχουν απορροφηθεί όλες οι γενικές υπαγωγές εννοιών (GCIs), δηλαδή να μην έχει μείνει κανένα GCI ανεπεξέργαστο.
- Και στις δύο έννοιες που ελέγχονται στην σχέση υπαγωγής πρέπει να εφαρμοστεί μια διαδικασία που ελέγχει (μέσω στατικής ανάλυσης των αξιωμάτων του TBox) το αν όλες οι υπο-έννοιες της κάθε έννοιας, στις οποίες μπορεί κάποιος να φτάσει μέσω ορισμών εννοιών ή GCIs, είναι $\mathcal{ELH}$ έννοιες.
- Και οι δύο έννοιες που ελέγχονται στην σχέση υπαγωγής πρέπει να είναι ακυκλικές.

Ο αλγόριθμος ελέγχει πρώτα τις παραπάνω προϋποθέσεις και αν αυτές ισχύουν τότε εφαρμόζεται κανονικά. Αλλιώς εφαρμόζεται ο κλασικός έλεγχος υπαγωγής με τις τεχνικές που περιγράφηκαν πιο πάνω.

Στα πειράματα που έγιναν σε αυτή την εργασία η δομική υπαγωγή, υλοποιημένη μέσω ενός αλγορίθμου εκθετικού στην χειρότερη περίπτωση, επιτάχυνε σε μεγάλο βαθμό την ταξινόμηση της SNOMED-ELH, διατηρώντας την ορθότητα και την πληρότητα.

Στο Σχήμα 3.19 παρουσιάζονται τα αποτελέσματα των πειραμάτων που διεξήχθησαν για την ταξινόμηση μέσω του RacerPro της SNOMED-ELH, η οποία

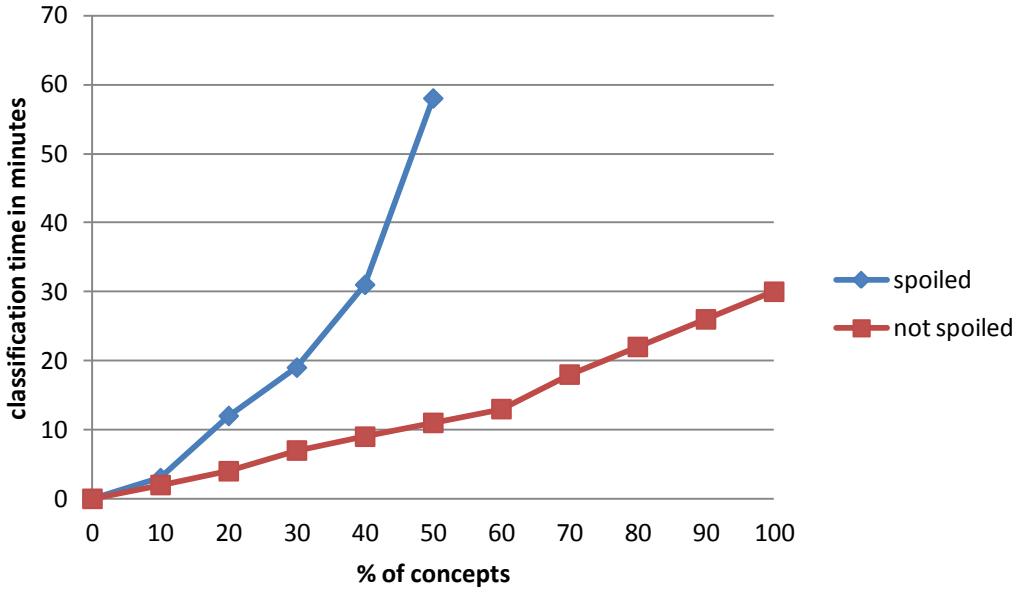
περιέχει 380.000 έννοιες. Παρατηρείται ότι η τεχνική αναγνώρισης των ολοκληρωτικά ορισμένων εννοιών είχε δραματική επίδραση στα αποτελέσματα, αλλά η ενσωμάτωση μοντέλων είχε ακόμα περισσότερη. Ο συνδυασμός και των δύο βελτίωσε περαιτέρω (σε πολύ μικρό βαθμό όμως) την επίδοση. Οι χρόνοι ήταν περίπου ίδιοι όταν αντί του RacerPro χρησιμοποιήθηκε ο CEL.

CD optimization	pseudo model embedding	ELH?	runtime
disabled	disabled	yes	several days
enabled	disabled	yes	several hours
disabled	disabled	yes	minutes (15 min)
enabled	disabled	yes	minutes(12 min)

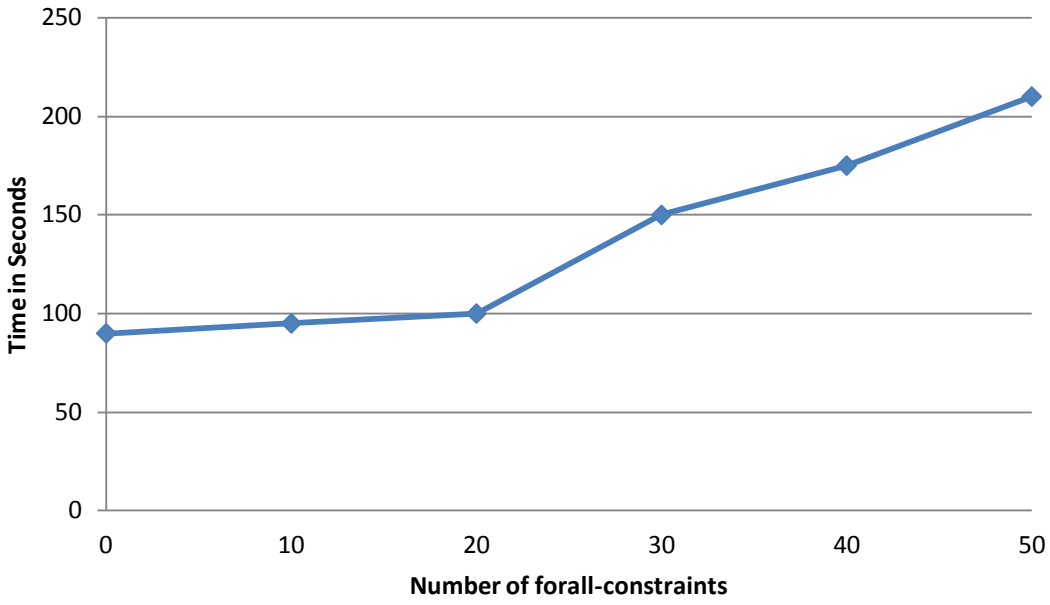
Σχήμα 3.19. Επιδράσεις τεχνικών βελτιστοποίησης

Ο RacerPro μπορεί ακόμα να χρησιμοποιηθεί αν προστεθεί στο TBox ένα $\mathcal{ALH}$ αξίωμα της μορφής $A_1 \sqsubseteq \forall R.A_2$ για αυθαίρετες SNOMED-ELH έννοιες A_1 και A_2 και ρόλο R.

Διεξήχθησαν περαιτέρω πειράματα για να ερευνηθεί σε ποιες περιπτώσεις η απόδοση θα μειωθεί. Στο Σχήμα 3.20 παρατηρείται ότι η επίδραση είναι άμεση, αφού η προσθήκη λίγων μόνων περιορισμών τιμών οδηγούν σε μεγάλους χρόνους ταξινόμησης. Η επίδραση είναι μικρότερη αν ο αριθμός των εννοιών είναι λιγότερος, όπως φαίνεται στο Σχήμα 3.21.



Σχήμα 3.20. Ταξινόμηση της SNOMED έχοντας προσθέσει περιορισμούς τιμών σε 50 ορισμούς τυχαίων γονέων της κενής έννοιας $\bot$. Χρησιμοποιήθηκε ένα αυξανόμενο μέρος του TBox (10%, 20%, ..., 100%). Η μπλε γραμμή είναι οι χρόνοι (σε λεπτά) χωρίς περιορισμούς τιμών, ενώ η κόκκινη είναι οι χρόνοι με περιορισμούς τιμών



Σχήμα 3.21. Ταξινόμηση της SNOMED με την πρόσθεση αυξανόμενου αριθμού περιορισμών τιμών στους ορισμούς τυχαίων γονέων της κενής έννοιας ⊥. Χρησιμοποιήθηκε μόνο το 10% της SNOMED (37900 έννοιες)

3.4 Συγκεντρωτικός Πίνακας

	CEL	DB	CB	Μέθοδος των Cali, Gottlob & Lukasiewiez	Μέθοδος Krötzsch	Orel	Μέθοδος Haarslev, Möller & Wandelt
Γλώσσα Προγραμματισμού	Lisp	ML	Java	Datalog	Datalog	Java	-
Εκφραστικότητα	$\mathcal{EL}^+$	$\mathcal{ELH}$	$\mathcal{H}orn\ \mathcal{SHIQ}$	$\mathcal{DL\text{-}Lite}_\mathcal{F}$, $\mathcal{DL\text{-}Lite}_\mathcal{R}$, και $\mathcal{DL\text{-}Lite}_\mathcal{A}$	$\mathcal{SROEL}(\mathcal{X})$	$\mathcal{EL}$,$\mathcal{RL}$	$\mathcal{ELH}$
Λειτουργία σε	μνήμη	σκληρό δίσκο	μνήμη	σκληρό δίσκο	σκληρό δίσκο	σκληρό δίσκο	μνήμη
Άδεια	Allegro CL License Agreement	GNU Lesser General Public License	GNU Lesser General Public License	-	-	GNU Lesser General Public License	-

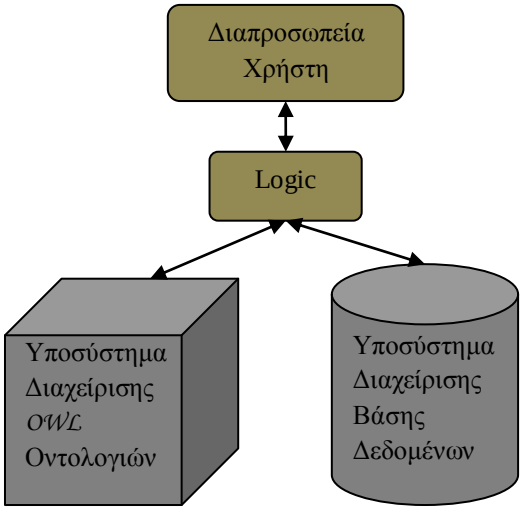
4

Ανάλυση Συστήματος

Στο κεφάλαιο αυτό αναλύουμε τις απαιτήσεις που ικανοποιεί το DBRS. Η μεθοδολογία παρουσίασης έχει ως βασικό άξονα το διαχωρισμό του συστήματος σε διακριτά υποσυστήματα (Ενότητα 4.1), την συνοπτική περιγραφή των λειτουργιών που αυτά επιτελούν (Ενότητα 4.2) και την ανάδειξη του τρόπου επικοινωνίας μεταξύ τους. Ως υποσύστημα ορίζουμε μια «οικογένεια» από εφαρμογές που είτε επιτελούν παρόμοιες λειτουργίες (και επομένως μπορούν να ομαδοποιηθούν) είτε συμμετέχουν στην εκτέλεση ενός αυτοτελούς καθήκοντος. Ένα υποσύστημα μπορεί να διαιρείται σε επιμέρους υποσυστήματα (υποσύνολα εφαρμογών) τα οποία, με τη σειρά τους, μπορεί επίσης να διαιρούνται σε άλλα κοκ.

4.1 Αρχιτεκτονική – Διαχωρισμός Υποσυστημάτων

Το DBRS αποτελείται από τρία βασικά υποσυστήματα. Το υποσύστημα του χρήστη, το υποσύστημα διαχείρισης της βάσης δεδομένων και το υποσύστημα διαχείρισης της μηχανής συλλογιστικής ανάλυσης. Το πρώτο από αυτά διαιρείται σε τέσσερα υποσυστήματα, ένα εκ των οποίων (το υποσύστημα φόρτωσης της οντολογίας) διαιρείται σε δύο επιπλέον. Αφαιρετικά, θα μπορούσαμε να πούμε ότι το DBRS διαθέτει μια τριεπίπεδη (three-layered) αρχιτεκτονική η οποία φαίνεται στο Σχήμα 4.1. Το πρώτο επίπεδο αφορά στη γραφική διαπροσωπία με την οποία ο χρήστης έρχεται σε επαφή με το σύστημα, ενώ το δεύτερο περιλαμβάνει όλα εκείνα τα υποσυστήματα του DBRS που αποτελούν το επίπεδο Λογικής (Logic) και μεσολαβούν μεταξύ του πρώτου και του τρίτου επιπέδου, δηλαδή μεταξύ της διαπροσωπίας του χρήστη και των υποσυστημάτων διαχείρισης της βάσης δεδομένων και διαχείρισης OWL οντολογιών. Το επίπεδο Λογικής του DBRS αποτελεί στην ουσία και τον πυρήνα (core) του συστήματος.



Σχήμα 4.1 Τριεπίπεδη Αρχιτεκτονική του συστήματος DBRS

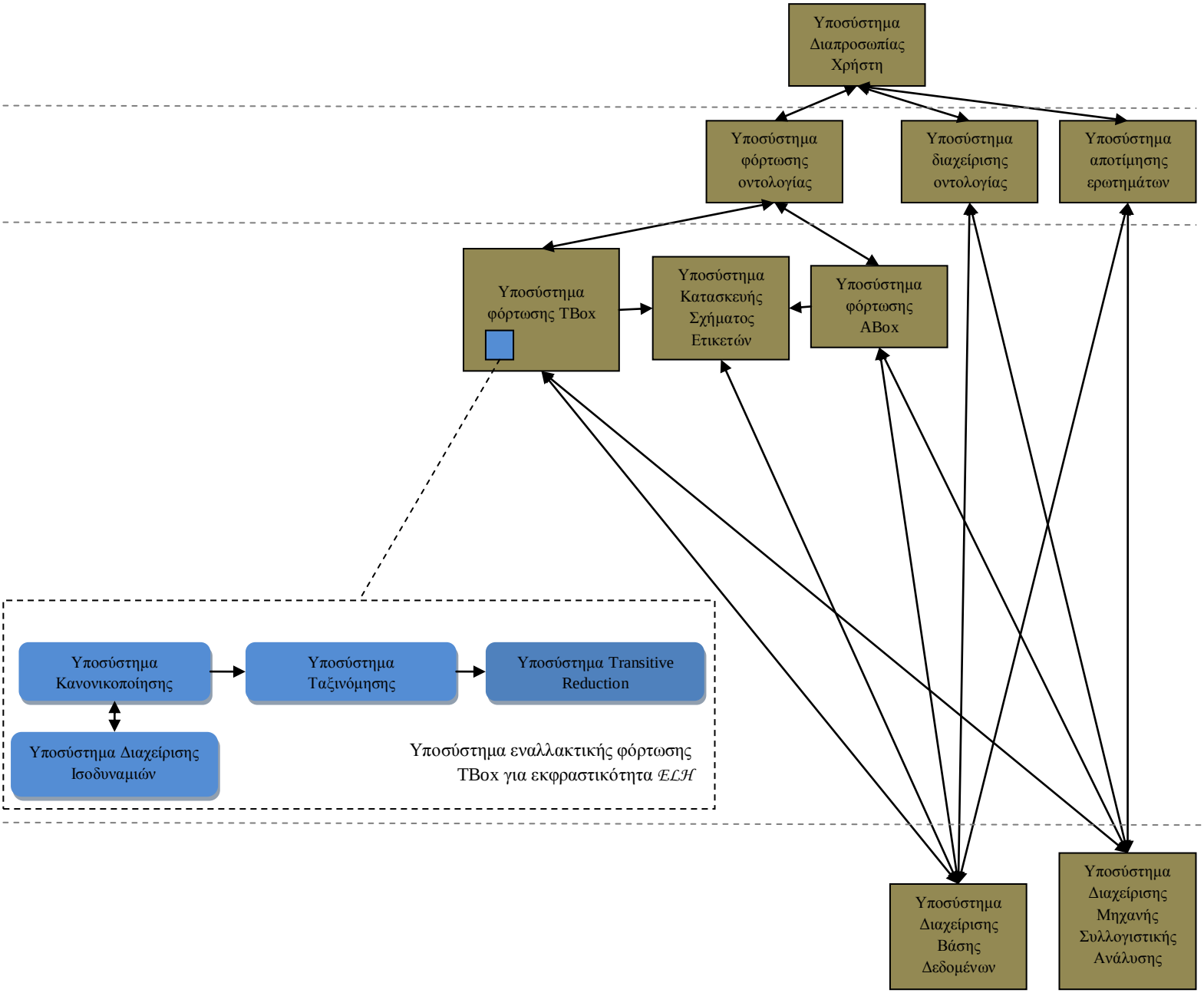
Πιο συγκεκριμένα και όπως φαίνεται στο Σχήμα 4.2, η δομή του DBRS είναι:

- Υποσύστημα γραφικής διαπροσωπίας χρήστη
- Υποσύστημα φόρτωσης οντολογίας
    - Υποσύστημα φόρτωσης TBox
        - Υποσύστημα κανονικοποίησης
        - Υποσύστημα διαχείρισης ισοδυναμιών
        - Υποσύστημα ταξινόμησης
        - Υποσύστημα transitive reduction
    - Υποσύστημα φόρτωσης ABox
- Υποσύστημα κατασκευής σχήματος ετικετών
- Υποσύστημα διαχείρισης οντολογίας
- Υποσύστημα αποτίμησης ερωτημάτων

- Υποσύστημα διαχείρισης Βάσης Δεδομένων
- Υποσύστημα διαχείρισης Μηχανής Συλλογιστικής Ανάλυσης

Η αντιστοιχία μεταξύ των δύο σχημάτων έχει ως εξής:

Το Υποσύστημα γραφικής διαπροσωπίας χρήστη του Σχήματος 4.2 αντιστοιχεί στη Διαπροσωπία Χρήστη, δηλαδή στο πρώτο επίπεδο του Σχήματος 4.1. Τα υπόλοιπα Υποσυστήματα του Σχήματος 4.2 αντιστοιχούν στο δεύτερο επίπεδο (Logic) του Σχήματος 4.1, με εξαίρεση τα Υποσυστήματα διαχείρισης Βάσης Δεδομένων και διαχείρισης OWL οντολογιών που αντιστοιχούν στο τρίτο και κατώτερο επίπεδο.



Σχήμα 4.2 Αρχιτεκτονική του συστήματος DBRS

4.2 Περιγραφή υποσυστημάτων

Στην ενότητα αυτή περιγράφεται συνοπτικά η λειτουργία κάθε υποσυστήματος που αναφέρθηκε προηγουμένως. Οι εφαρμογές που υλοποιούν αυτές τις λειτουργίες θα παρουσιαστούν λεπτομερώς στο επόμενο κεφάλαιο.

4.2.1 Υποσύστημα Γραφικής Διαπροσωπίας Χρήστη

Το Υποσύστημα γραφικής διαπροσωπίας χρήστη αποτελεί το περιβάλλον με το οποίο ο χρήστης έρχεται σε επαφή με το σύστημα. Μέσω αυτού, του δίνεται η πρόσβαση στις λειτουργίες που προσφέρει το DBRS σχετικά με τη φόρτωση και διαχείριση οντολογιών, αλλά και με τη διεξαγωγή ερωτημάτων.

4.2.2 Υποσύστημα Φόρτωσης Οντολογίας

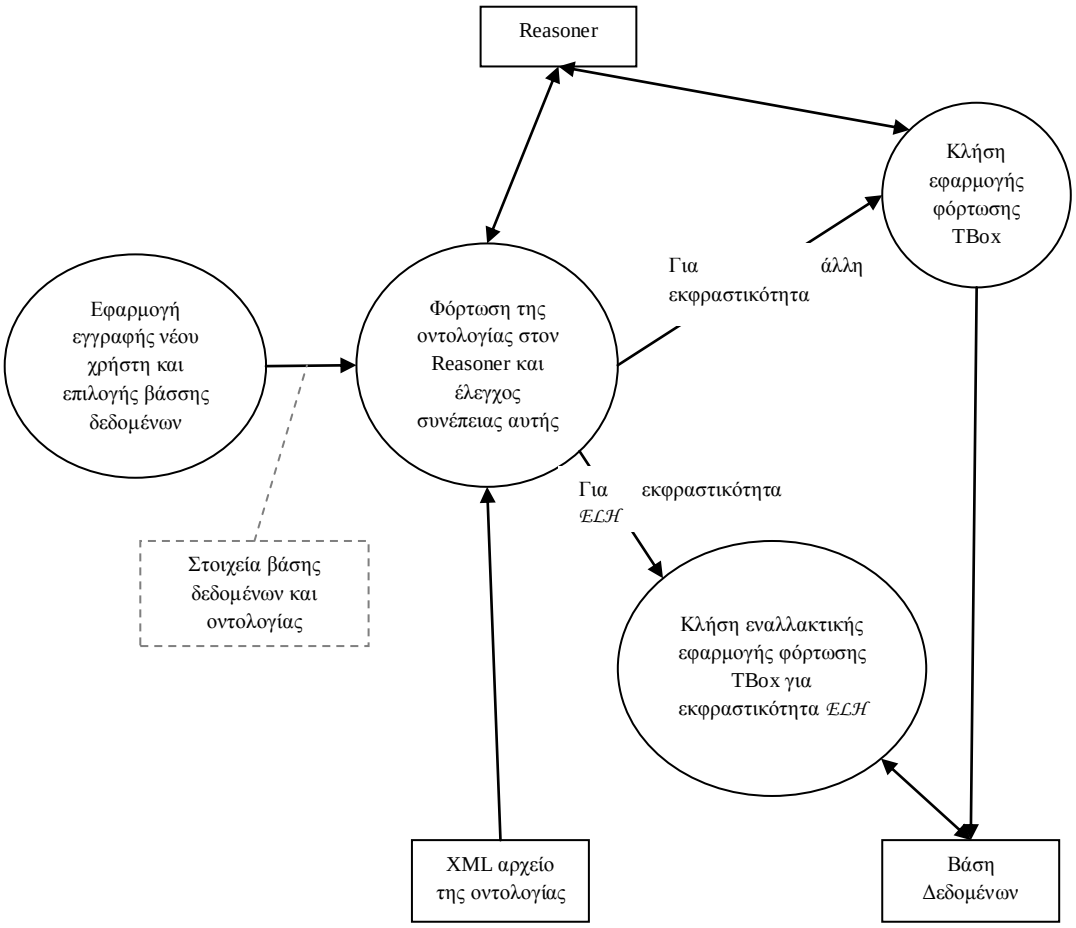
Το Υποσύστημα φόρτωσης οντολογίας περιλαμβάνει τις λειτουργίες που επιτελούνται κατά τη φόρτωση της οντολογίας στη βάση δεδομένων του DBRS, παραμετροποιημένες με βάση τις επιλογές του χρήστη. Διαιρείται στο Υποσύστημα φόρτωσης TBox και στο Υποσύστημα φόρτωσης ABox που σχετίζονται με τις λειτουργίες φόρτωσης του TBox και του ABox αντίστοιχα. Ειδική μνεία θα γίνει στα υποσυστήματα που σχετίζονται με την φόρτωση του TBox στην περίπτωση που η εκφραστικότητα της οντολογίας είναι $\mathcal{ELH}$.

4.2.2.1 Υποσύστημα Φόρτωσης TBox

Το Υποσύστημα φόρτωσης TBox περιλαμβάνει τις λειτουργίες που σχετίζονται με τη φόρτωση του TBox της οντολογίας. Αυτές είναι:

- Αποθήκευση Namespaces οντολογίας
- Αποθήκευση κλάσεων και κωδικοποίηση της ιεραρχίας τους μέσω κλήσης του Υποσυστήματος κατασκευής σχήματος ετικετών
- Αποθήκευση πληροφορίας περί ασυμβατότητας κλάσεων
- Αποθήκευση ρόλων και κωδικοποίηση της ιεραρχίας τους μέσω κλήσης του Υποσυστήματος κατασκευής σχήματος ετικετών
- Αποθήκευση πληροφορίας περί αντίστροφων ρόλων
- Αποθήκευση πληροφορίας περί χαρακτηριστικών ρόλων
- Αποθήκευση πληροφορίας σχετικά με το πεδίο (domain) και το εύρος (range) των ρόλων
- Αποθήκευση πληροφορίας περί ασυμβατότητας ρόλων
- Αποθήκευση ορισμών κλάσεων
- Κατασκευή και αποθήκευση του διευρυμένου TBox
- Αποθήκευση ψευδομοντέλων κλάσεων

Στο υποσύστημα φόρτωσης TBox μελετάμε ένα συγκεκριμένο εναλλακτικό υποσύστημα φόρτωσης για την περίπτωση που η εκφραστικότητα της υπό εξέταση οντολογίας διαπιστωθεί ότι είναι *ELH*. Το διάγραμμα ροής για την επιλογή του εναλλακτικού σεναρίου φαίνεται στο Σχήμα 4.3 και τα επιμέρους υποσυστήματα αναλύονται στη συνέχεια.



Σχήμα 4.3 Διάγραμμα ροής δεδομένων για τη διαδικασία φόρτωσης των αξιωμάτων ανάλογα με την εκφραστικότητα της οντολογίας

4.2.2.1.1 Υποσύστημα Κανονικοποίησης

Το Υποσύστημα κανονικοποίησης περιλαμβάνει τις λειτουργίες που επιτελούνται κατά την διαδικασία της κανονικοποίησης της οντολογίας και της εισαγωγής των εξαγόμενων δεδομένων στη βάση δεδομένων του DBRS. Πιο συγκεκριμένα επιτελεί τις παρακάτω λειτουργίες.

- Φόρτωση του σώματος της οντολογίας στο DBRS για την περαιτέρω επεξεργασία αυτής.
- Διάκριση του τύπου του κάθε αξιώματος που περιέχεται στην οντολογία.

- Επεξεργασία του κάθε τύπου αξιώματος με την επαναληπτική εφαρμογή τυπικών καθορισμένων κανόνων κανονικοποίησης ανάλογα με τον τύπο του αξιώματος.
- Ειδική μεταχείριση των αξιωμάτων ισοδυναμίας μέσω κλήσης του υποσυστήματος διαχείρισης ισοδυναμιών.
- Απόδοση ενός μοναδικού αναγνωριστικού σε όλες τις 'σημασιολογικές ψηφίδες' που συμμετέχουν στη σύνταξη των αξιωμάτων της οντολογίας μέσω τη χρήσης μιας συνάρτησης κατακερματισμού (hash) η οποία για την ίδια είσοδο επιστρέφει πάντα την ίδια έξοδο. Ο όρος 'σημασιολογικές ψηφίδες' μπορεί να αναφέρεται σε ατομικές κλάσεις, σύνθετες κλάσεις[35] και ρόλους.
- Αποθήκευση των πληροφοριών που προέκυψαν από την εφαρμογή των κανόνων κανονικοποίησης σε συγκεκριμένη μορφή στη βάση δεδομένων κατάλληλη για την επεξεργασία αυτών στο επόμενο στάδιο.

4.2.2.1.2 Υποσύστημα Διαχείρισης Ισοδυναμιών

Το Υποσύστημα διαχείρισης ισοδυναμιών περιλαμβάνει όλες τις λειτουργίες που σχετίζονται με τη διαχείριση των αξιωμάτων ισοδυναμίας κλάσεων τα οποία εμφανίζονται στο σώμα της οντολογίας. Το συγκεκριμένο υποσύστημα καλείται να επιτελέσει έναν τριπλό ρόλο:

- Να καταχωρεί σε μία συγκεκριμένη δομή όλες τις κλάσεις που είναι ισοδύναμες μεταξύ τους όπως αυτό καθορίζεται στα αξιώματα ισοδυναμίας.
- Μόλις ολοκληρώσει την εισαγωγή στην δομή όλων των ισοδύναμων κλάσεων να δημιουργήσει το βέλτιστο mapping ώστε όλες οι ισοδύναμες κλάσεις να αντιστοιχήσουν σε ένα μοναδικό hashcode (ένα από τα hashcode που έχει αποδοθεί σε κάποια από τις ισοδύναμες κλάσεις) και να αποθηκεύσει αυτή τη δομή στη βάση δεδομένων.
- Να ενημερώσει όλους τους πίνακες της βάσης δεδομένων που προκύπτουν από την διαδικασία της κανονικοποίησης με τα αναθεωρημένα hashcode.

4.2.2.1.3 Υποσύστημα Ταξινόμησης

Το υποσύστημα ταξινόμησης περιλαμβάνει όλες τις λειτουργίες που σχετίζονται με τον υπολογισμό όλων των υπονοούμενων σχέσεων ιεραρχίας μεταξύ των εννοιών της οντολογία μέσω της εξαντλητικής εφαρμογής ενός συνόλου από επαγωγικούς κανόνες και μπορεί να θεωρηθεί ως τον υπολογισμό ενός εκτεταμένου transitive closure. Επίσης περιλαμβάνει όλες τις λειτουργίες που επιτελούνται κατά την επιλογή του σχεδίου με το οποίο θα εφαρμοστούν οι επαγωγικοί κανόνες καθώς και τις λειτουργίες που σχετίζονται με την εφαρμογή των επαγωγικών κανόνων. Πιο συγκεκριμένα επιτελεί τις παρακάτω λειτουργίες.

[35] Επειδή έχουμε εκφραστικότητα $\mathcal{ELH}$ οι σύνθετες κλάσεις μπορεί να είναι μία τομή ατομικών κλάσεων, μία υπαρξιακή ποσοτικοποίηση, τομή άλλων σύνθετων κλάσεων και συνδυασμός αυτών.

- Δημιουργία διακριτών σταδίων εκτέλεσης του αλγορίθμου.
- Καθορισμός των συμπερασματικών κανόνων που θα εφαρμοστούν και της σειράς αυτών ανάλογα με το στάδιο που βρίσκεται η διαδικασία
- Προετοιμασία της βάσης δεδομένων με τον καθορισμό συγκεκριμένων παραμέτρων με σκοπό την βελτιστοποίηση της διαδικασίας ανάλογα με το στάδιο που θα ακολουθήσει (π.χ. δημιουργία ευρετηρίων, κατάργηση ευρετηρίων, χρήση προσωρινών πινάκων, χρήση εναλλακτικών πινάκων κτλ.).
- Μετασχηματισμό του κάθε συμπερασματικού κανόνα στην αντίστοιχη αλληλουχία sql statements που απευθύνεται στη βάση δεδομένων και την δημιουργία των ερωτημάτων για την αποθήκευση των εξαγόμενων από τη διαδικασία δεδομένων σε κατάλληλη μορφή στη βάση δεδομένων και την ενημέρωση των ήδη υπαρχόντων.

4.2.2.1.4 Υποσύστημα transitive reduction

Το υποσύστημα transitive reduction περιλαμβάνει όλες τις λειτουργίες που σχετίζονται με την επεξεργασία των δεδομένων που προέκυψαν από την διαδικασία της ταξινόμησης με σκοπό την δημιουργία μιας ταξονομίας στην οποία θα περιλαμβάνονται μόνο άμεσες σχέσεις υπαγωγής μεταξύ των ατομικών εννοιών.

4.2.2.2 Υποσύστημα Φόρτωσης Abox

Το Υποσύστημα φόρτωσης ABox περιλαμβάνει τις λειτουργίες που σχετίζονται με τη φόρτωση του ABox της οντολογίας. Αυτές είναι:

- Αποθήκευση ατόμων
- Αποθήκευση πληροφορίας περί ταυτότητας ατόμων
- Αποθήκευση πληροφορίας περί διαφορετικών ατόμων
- Αποθήκευση δηλώσεων κλάσεων
- Αποθήκευση δηλώσεων ρόλων αντικειμένου και ρόλων τύπου δεδομένων
- Αποθήκευση δηλώσεων μεταβατικών ρόλων και κωδικοποίησή τους μέσω κλήσης του Υποσυστήματος κατασκευής σχήματος ετικετών
- Αποθήκευση ψευδομοντέλων ατόμων

4.2.3 Υποσύστημα Κατασκευής Σχήματος Ετικετών

Το Υποσύστημα κατασκευής Σχήματος Ετικετών αναλαμβάνει την κωδικοποίηση (μέσω ετικετών) των σχέσεων ιεραρχίας της οντολογίας και την αποθήκευσή τους στη βάση δεδομένων. Οι σχέσεις ιεραρχίας μπορεί να αναφέρονται σε κλάσεις, ρόλους, αλλά και δηλώσεις μεταβατικών ρόλων.

4.2.4 Υποσύστημα Διαχείρισης Οντολογίας

Το Υποσύστημα διαχείρισης οντολογίας περιλαμβάνει όλες τις λειτουργίες που σχετίζονται με τη διαχείριση των οντολογιών οι οποίες είναι αποθηκευμένες σε βάση δεδομένων.

4.2.5 Υποσύστημα Αποτίμησης Ερωτημάτων

Το Υποσύστημα αποτίμησης ερωτημάτων αναλαμβάνει την αναγνώριση των ερωτημάτων που υποβάλλονται από τον χρήστη μέσω του GUI, τον έλεγχο των παραμέτρων τους και την επιλογή της στρατηγικής αποτίμησής τους.

4.2.6 Υποσύστημα διαχείρισης Βάσης Δεδομένων

Το Υποσύστημα διαχείρισης Βάσης Δεδομένων είναι υπεύθυνο για την επικοινωνία των υποσυστημάτων του DBRS με τη βάση δεδομένων και ως εκ τούτου αναλαμβάνει την εκτέλεση όλων εκείνων των λειτουργιών που σχετίζονται με εισαγωγή, ενημέρωση και εξαγωγή πληροφορίας από αυτήν. Επίσης, διεκπεραιώνει όλες τις αιτήσεις που αφορούν είτε σε δημιουργία νέας βάσης είτε σε διαχείριση υπάρχουσας, αλλά και τις αιτήσεις για εγγραφή νέου χρήστη στο DBMS.

4.2.7 Υποσύστημα διαχείρισης Μηχανής Συλλογιστικής Ανάλυσης

Το Υποσύστημα διαχείρισης Μηχανής Συλλογιστικής Ανάλυσης είναι υπεύθυνο για τη επικοινωνία των υποσυστημάτων του DBRS με τον Reasoner. Στην ουσία, αναλαμβάνει την εκτέλεση όλων εκείνων των λειτουργιών που σχετίζονται με εξαγωγή πληροφορίας από τη βάση γνώσης (στην κύρια μνήμη) του Reasoner.

5

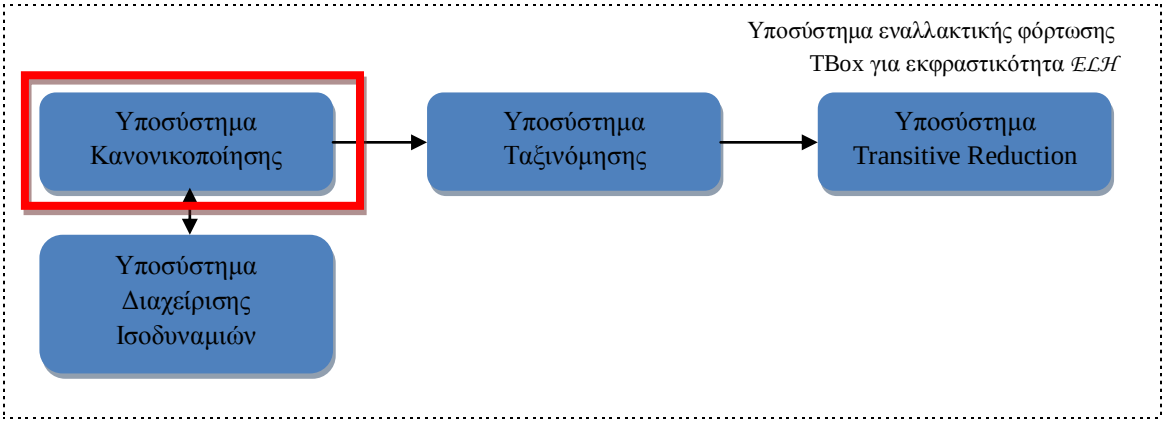
Σχεδίαση Συστήματος

Στο κεφάλαιο αυτό περιγράφονται οι εφαρμογές κάθε υποσυστήματος του συστήματός μας και, όπου απαιτείται, παρατίθενται διαγράμματα ροής για την καλύτερη κατανόηση των επιμέρους διαδικασιών που εμπλέκονται σε αυτές. Δεν περιγράφεται εδώ όλη η αρχιτεκτονική του DBRS (στο οποίο ενσωματώνεται το σύστημά μας), αλλά μόνο το κομμάτι που υλοποιήσαμε εμείς[36], η διαχείριση δηλαδή της οντολογίας από την στιγμή που θα ταυτοποιηθεί ως μία *ELH* οντολογία και μετά. Η δομή της παρουσίασης είναι παρόμοια με αυτή του Κεφαλαίου 4, μόνο που εδώ αναφερόμαστε στο επίπεδο των εφαρμογών. Στο τέλος του Κεφαλαίου δίνεται ένα παράδειγμα εκτέλεσης.

[36] Η αρχιτεκτονική του υπόλοιπου DBRS παρουσιάζεται αναλυτικά εδώ http://www.dbnet.ece.ntua.gr/pubs/details.php?id=1523&clang=0

5.1 Υποσύστημα κανονικοποίησης

Στο υποσύστημα κανονικοποίησης ανήκουν οι εφαρμογές που αναλαμβάνουν την εφαρμογή των κανόνων κανονικοποίησης στα αξιώματα της οντολογίας και την αποθήκευση της κανονικοποιημένης οντολογίας στο σχεσιακό DBMS. Αρχικά μέσω της εφαρμογής διαχείρισης αξιωμάτων επιτυγχάνεται η μετατροπή των αξιωμάτων της οντολογίας στα κανονικοποιημένα αξιώματα ($A \sqsubseteq B$, $A_1 \sqcap A_2 \sqsubseteq B$, $A \sqsubseteq \exists r.B$, $\exists r.A \sqsubseteq B$, $r \sqsubseteq s$) και η αποθήκευση τους στις αντίστοιχες προσωρινές δομές σε αρχεία. Στη συνέχεια γίνεται η μεταφορά της οντολογίας από τις προσωρινές δομές στο σχεσιακό DBMS μέσω κλήσης της αντίστοιχης εφαρμογής.



Σχήμα 5.1 Η θέση του υποσυστήματος κανονικοποίησης στο ολικό σύστημα

5.1.1 Εφαρμογή διαχείρισης αξιωμάτων

Η εφαρμογή αυτή αποτελεί τον πυρήνα της εφαρμογής των κανόνων κανονικοποίησης. Σκοπός της διαδικασίας είναι η μετατροπή των αξιωμάτων της οντολογίας στα κανονικοποιημένα αξιώματα και η αποθήκευση αυτών στις αντίστοιχες προσωρινές δομές. Τα βήματα της διαδικασίας παρατίθενται παρακάτω:

- Από το μοντέλο της οντολογίας που έχει κατασκευαστεί προηγουμένως εξαιρούμε τα αξιώματα δήλωσης (declaration). Για τα υπόλοιπα αξιώματα ξεκινάμε την επεξεργασία τους σειριακά.
- Διακρίνεται ο τύπος του κάθε αξιώματος μεταξύ των επιτρεπόμενων:
  - αξιώματα υπαγωγής εννοιών (subclass_of)
  - αξιώματα υπαγωγής ρόλων (sub_object_property)
  - αξιώματα ισοδυναμίας εννοιών (equivalent_classes).
- Γίνεται κλήση της εφαρμογής απόδοσης μοναδικού αναγνωριστικού για όλες τις *OWL* εκφράσεις που συμμετέχουν σε κάθε αξίωμα, δηλαδή τα subClass, superClass στα αξιώματα υπαγωγής εννοιών, τα subProperty, superProperty στα αξιώματα υπαγωγής ρόλων και όλες τις ισοδύναμες OWLClassExpression

στα αξιώματα ισοδυναμίας εννοιών με σκοπό την απόδοση των αντίστοιχων id.

- Το κάθε ζευγάρι αναγνωριστικών (id1-id2) αποθηκεύεται στις αντίστοιχες προσωρινές δομές κανονικοποιημένων αξιωμάτων (A ⊑ B ή r ⊑ s) μέσω κλήσης της εφαρμογής προσωρινής αποθήκευσης κανονικοποιημένων σχέσεων.
- Στην περίπτωση αξιωμάτων ισοδυναμίας τα αναγνωριστικά αποθηκεύονται ανά ζεύγη σε μια προσωρινή δομή αναπαράστασης ισοδυναμιών.
- Για κάθε *OWL* έκφραση εννοιών (OWLClassExpression) που έχουμε συναντήσει στα αξιώματα υπαγωγής εννοιών ή στα αξιώματα ισοδυναμίας εννοιών καλούμε την εφαρμογή χειρισμού *OWL* εκφράσεων εννοιών (OWLClassExpression).

5.1.2 Εφαρμογή προσωρινής αποθήκευσης κανονικοποιημένων αξιωμάτων

Η εφαρμογή προσωρινής αποθήκευσης κανονικοποιημένων αξιωμάτων σχετίζεται με την χρήση προσωρινών αρχείων για την αποθήκευση των κανονικοποιημένων αξιωμάτων που προκύπτουν σταδιακά από τη διαδικασία της κανονικοποίησης. Για κάθε έναν τύπο κανονικοποιημένου αξιώματος υπάρχει και ένα αντίστοιχο προσωρινό αρχείο. Επιπλέον υπάρχει ένα ακόμα αρχείο στο οποίο αποθηκεύονται τα ζευγάρια που προκύπτουν από τα αξιώματα ισοδυναμίας και ένα αρχείο που αντιστοιχεί όλες τις *OWL* εκφράσεις της οντολογίας με το μοναδικό id που τους έχει αντιστοιχηθεί. Βασική λειτουργικότητα αυτή της εφαρμογής είναι η απουσία διπλοεγγραφών μέσα στα προσωρινά αρχεία.

5.1.3 Εφαρμογή απόδοσης μοναδικού αναγνωριστικού

Η εφαρμογή απόδοσης μοναδικού αναγνωριστικού σχετίζεται με την διαδικασία απόδοσης ενός μοναδικού id σε κάθε *OWL* έκφραση που περιέχεται στην οντολογία. Βασική επιδίωξη μας είναι να μην χρειάζεται να ανατρέχουμε σε όλα τα id που έχουμε ήδη αποδώσει πριν από την απόδοση id σε μία νέα έκφραση και να είμαστε σίγουροι ότι κάθε φορά που η διαδικασία κανονικοποίησης θα συναντά την ίδια *OWL* έκφραση θα αποδίδει το ίδιο id. Για να γίνει αυτό εφικτό χρειαζόμαστε μία δομή στη μνήμη με τα χαρακτηριστικά ενός hash map που θα αντιστοιχεί το URI κάθε *OWL* έκφρασης με το αντίστοιχο id. Τα βήματα αυτής της διαδικασίας περιγράφονται παρακάτω και φαίνονται στο διάγραμμα ροής του σχήματος :

- Απόδοση προσωρινού id με τη χρήση μιας συνάρτησης κατακερματισμού (hashcode).
- Έλεγχος αν υπάρχει ήδη αποθηκευμένο μέσα στη δομή hash map αυτό το id. Αν αυτό δεν υπάρχει τότε το προσωρινό id γίνεται οριστικό και αποθηκεύεται

το αντίστοιχο ζευγάρι URI – id στην δομή hash map. Αν υπάρχει το id ήδη αποθηκευμένο προχωράμε στο επόμενο βήμα.

- Ελέγχουμε αν το URI που βρίσκεται αποθηκευμένο στην δομή hash map στο αντίστοιχο id είναι το ίδιο με αυτό που έχει η υπό μελέτη *OWL* έκφραση. Αν αυτό συμβαίνει σταματάμε την διαδικασία σε αυτό το σημείο και χρησιμοποιούμε αυτό το id για την υπόλοιπη διαδικασία της κανονικοποίησης. Αν όχι συνεχίζουμε στο επόμενο βήμα.
- Αυξάνουμε το προσωρινό id κατά 1 και επιστρέφουμε πάλι στο βήμα ii.

5.1.4 Εφαρμογή χειρισμού OWL εκφράσεων εννοιών (OWLClassExpression)

Η εφαρμογή χειρισμού εκφράσεων *OWL* εννοιών (OWLClassExpression) σχετίζεται με τον τρόπο που γίνεται το unfolding των σύνθετων εννοιών έως ότου να έχουν δημιουργηθεί όλα τα κανονικοποιημένα αξιώματα και περιέχει 2 εφαρμογές για τον χειρισμό των κατασκευαστών που υπάρχουν στην εκφραστικότητα *ELH*. Αν η υπό εξέταση έκφραση *OWL* κλάσης είναι ήδη μια ατομική ονοματισμένη κλάση (named classes) τότε η διαδικασία δεν συνεχίζει αν όμως η έκφραση *OWL* κλάσης ταυτοποιηθεί ότι είναι μία έκφραση τομής ή μια έκφραση υπαρξιακού ποσοδείκτη τότε γίνεται κλήση των αντίστοιχων εφαρμογών που περιγράφονται στη συνέχεια.

5.1.4.1. Εφαρμογή χειρισμού του κατασκευαστή της τομής

Η εφαρμογή χειρισμού του κατασκευαστή της τομής σχετίζεται με τον χειρισμό των εκφράσεων *OWL* κλάσεων όταν έχει ταυτοποιηθεί ότι είναι μία έκφρασης τομής. Τα βήματα της εφαρμογής περιγράφονται παρακάτω:

- Αρχικά μέσω της εφαρμογής απόδοσης μοναδικού χαρακτηριστικού αποδίδεται ένα id στο σύνολο της *OWL* έκφρασης.
- Απομονώνονται όλες οι N *OWL* εκφράσεις που συμμετέχουν στην αρχική τομή.
- Δημιουργούνται 2 νέες εκφράσεις *OWL* κλάσεων. Η πρώτη περιέχει την πρώτη έκφραση *OWL* κλάσης της αρχικής τομής και η δεύτερη τις υπόλοιπες N-1 εκφράσεις.
- Με κλήση της εφαρμογής απόδοσης μοναδικού χαρακτηριστικού αποδίδονται id στις νέες εκφράσεις (id1 , id2)
- Μέσω της εφαρμογής προσωρινής αποθήκευσης κανονικοποιημένων αξιωμάτων αποθηκεύονται τα ζεύγη αναγνωριστικών (id-id1 και id-id2) που αντιστοιχούν στις 2 νέες κανονικοποιημένες σχέσεις υπαγωγής ($A \sqsubseteq B$) που υπονοούνται από τον κατασκευαστή της τομής και η τριπλέτα αναγνωριστικών (id1-id2-id) που αντιστοιχεί στην κανονικοποιημένη αναπαράσταση της τομής ($A_1 \sqcap A_2 \sqsubseteq B$).

- Για κάθε μία από τις δύο νέες εκφράσεις *OWL* κλάσεων καλείται πάλι η εφαρμογή χειρισμού εκφράσεων *OWL* κλάσεων για την συνέχιση της διαδικασίας "ξεδιπλώματος".

5.1.4.2 Εφαρμογή χειρισμού του υπαρξιακού κατασκευαστή

Η εφαρμογή χειρισμού του υπαρξιακού κατασκευαστή σχετίζεται με τον χειρισμό των εκφράσεων *OWL* κλάσεων όταν έχει ταυτοποιηθεί ότι είναι μία έκφρασης πλήρους υπαρξιακής ποσοτικοποίησης. Τα βήματα της εφαρμογής περιγράφονται παρακάτω:

- Αρχικά μέσω της εφαρμογής απόδοσης μοναδικού χαρακτηριστικού αποδίδεται ένα id στο σύνολο της *OWL* έκφρασης.
- Απομονώνεται η έκφραση *OWL* ιδιότητας αντικειμένων και η έκφραση *OWL* κλάσης που συμμετέχουν στην αρχική έκφραση.
- Με κλήση της εφαρμογής απόδοσης μοναδικού χαρακτηριστικού αποδίδονται id στις νέες εκφράσεις (r_id , c_id)
- Μέσω της εφαρμογής προσωρινής αποθήκευσης κανονικοποιημένων αξιωμάτων αποθηκεύεται η τριπλέτα αναγνωριστικών (id-r_id-c_id) που αντιστοιχεί στην κανονικοποιημένη αναπαράσταση του υπαρξιακού κατασκευαστή ($A \sqsubseteq \exists r.B$). Αν ο υπαρξιακός κατασκευαστής συμμετέχει αρνητικά[37] στο αξίωμα τότε εισάγεται και η αντίστοιχη τριπλέτα αναγνωριστικών (r_id-c_id-id) στην κανονικοποιημένη αναπαράσταση του αρνητικού υπαρξιακού κατασκευαστή ($\exists r.A \sqsubseteq B$).
- Για την νέα έκφραση *OWL* κλάσης καλείται πάλι η εφαρμογή χειρισμού εκφράσεων *OWL* κλάσεων για την συνέχιση της διαδικασίας "ξεδιπλώματος".

5.1.5 Εφαρμογή αποθήκευσης της οντολογίας στο σχεσιακό DBMS

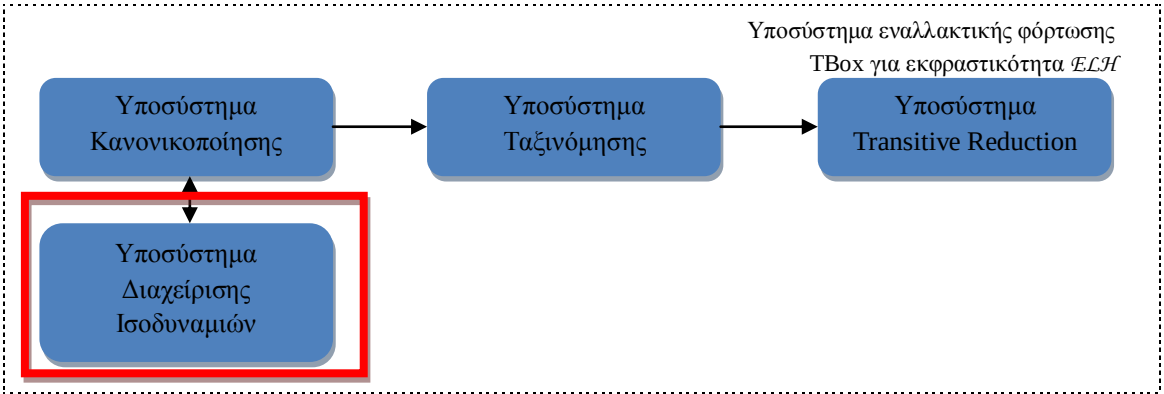
Η εφαρμογή αποθήκευσης της οντολογίας στο σχεσιακό DBMS σχετίζεται με τις διαδικασίες για την μεταφορά των προσωρινών δομών με τα κανονικοποιημένα αξιώματα στο αντίστοιχο σχήμα βάσης που θα χρησιμοποιηθεί και στα επόμενα στάδια.

5.2 Υποσύστημα διαχείρισης ισοδυναμιών

Το υποσύστημα διαχείρισης ισοδυναμιών επιτελεί όλες εκείνες τις λειτουργίες που είναι απαραίτητες για την διαχείριση των ισοδυναμιών. Για την ορθότητα του αλγορίθμου είναι αναγκαίο ισοδύναμες εκφράσεις *OWL* κλάσεων να σχετίζονται με

[37] Στην συγκεκριμένη εκφραστικότητα αυτό συμβαίνει μόνο όταν συμμετέχει σε σχέσεις ισοδυναμίας

το ίδιο αναγνωριστικό id σε παρέκκλιση όσων ισχύουν για την απόδοση αναγνωριστικών γενικά. Αυτή η ανάγκη επιτελείται μέσω των δύο εφαρμογών που αποτελούν αυτό το υποσύστημα και αναλύονται στη συνέχεια.



Σχήμα 5.2 Η θέση του υποσυστήματος διαχείρισης ισοδυναμιών στο ολικό σύστημα

5.2.1 Εφαρμογή ομαδοποίησης ισοδυναμιών

Η εφαρμογή ομαδοποίησης ισοδυναμιών σχετίζεται με την χρήση ενός αλγορίθμου ο οποίος ομαδοποιεί όλες τις ισοδύναμες εκφράσεις *OWL* κλάσεων που μπορεί να εμφανίστηκαν από διαφορετικά αξιώματα ισοδυναμιών στο στάδιο της κανονικοποίησης (π.χ. A≡B και B≡C). Σε κάθε ομάδα ισοδύναμων εκφράσεων αποφασίζεται πιο αναγνωριστικό θα είναι το χαρακτηριστικό αυτής της ομάδας και δημιουργεί ένα mapping.

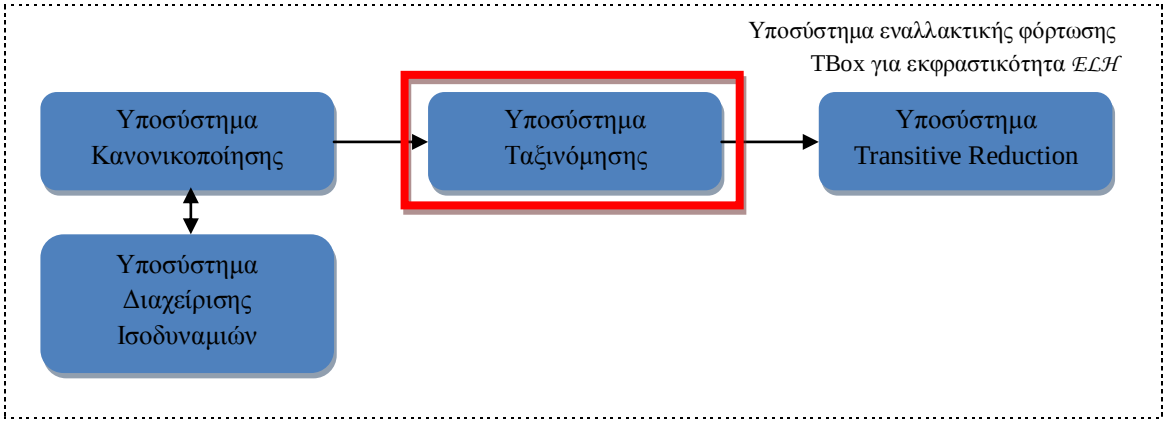
5.2.2 Εφαρμογή ενημέρωσης σχήματος βάσης

Η εφαρμογή ενημέρωσης σχήματος βάσης σχετίζεται με την ενημέρωση του σχήματος βάσης της κανονικοποιημένης οντολογίας σχετικά με την πληροφορία που εξάγεται από τα αξιώματα ισοδυναμίας. Αυτό επιτυγχάνεται μέσω της εφαρμογής μιας σειράς από sql statements στο σχήμα της βάσης.

5.3 Υποσύστημα ταξινόμησης

Στο υποσύστημα ταξινόμησης ανήκουν οι εφαρμογές που αναλαμβάνουν την εξαντλητική εφαρμογή των επαγωγικών κανόνων (fix-point) στο σχήμα βάσης που έχει αποθηκευθεί η ταξινομημένη οντολογία. Ουσιαστικά, το υποσύστημα ταξινόμησης υλοποιεί έναν αλγόριθμο ταξινόμησης που έχει σκοπό την εξασφάλιση της παραγωγής όλων των σχέσεων ιεραρχίας μεταξύ των ατομικών ονοματισμένων κλάσεων (named classes). Για να είναι αυτό εφικτό απαιτείται ο υπολογισμός όλων των "ενδιάμεσων" σχέσεων ιεραρχίας μεταξύ των εννοιών που προκύπτουν κυρίως

από τους κατασκευαστές της εκφραστικότητας. Οι εφαρμογές που αποτελούν το υποσύστημα ταξινόμησης παρουσιάζονται αναλυτικά παρακάτω.

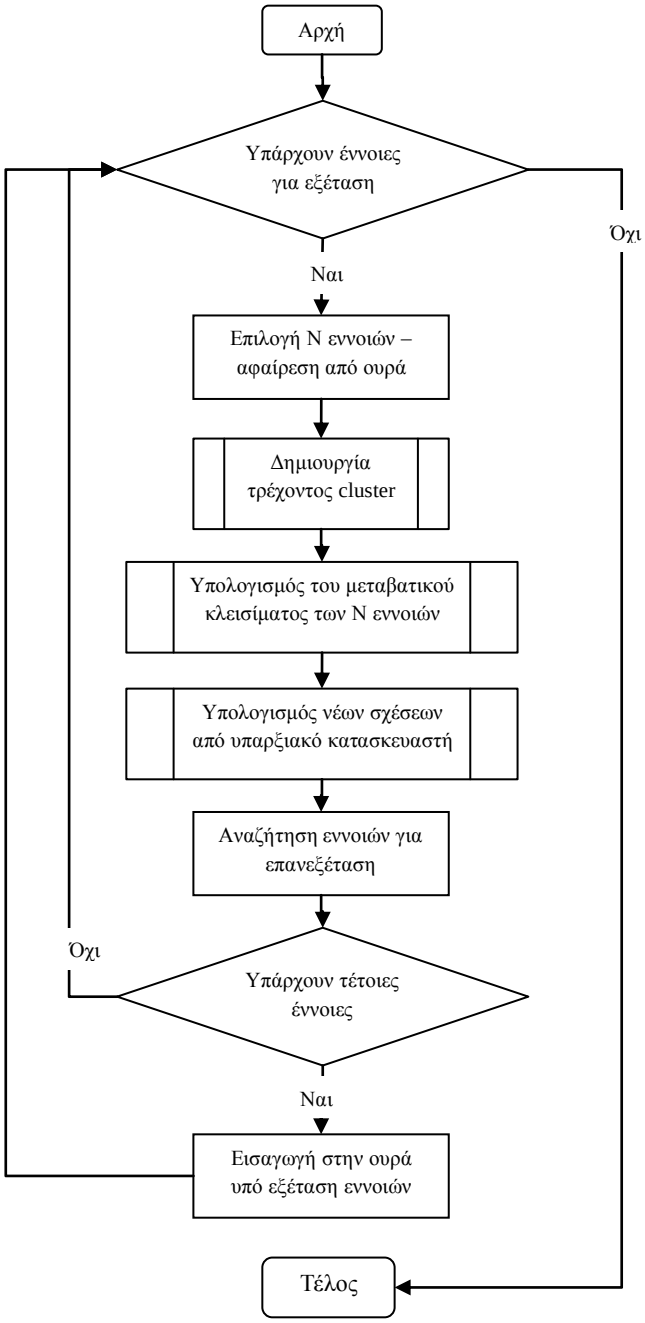


Σχήμα 5.3 Η θέση του υποσυστήματος ταξινόμησης στο ολικό σύστημα

Σε αυτό το σημείο ας αναφέρουμε για μια ακόμα φορά τους επαγωγικούς κανόνες για την εκφραστικότητα $\mathcal{ELH}$, οι οποίοι εφαρμόζονται μετά την μετάφρασή τους σε ένα αντίστοιχο σύνολο sql statement.

1. Αν $A \in T$ τότε $T = T \cup \{A \sqsubseteq A , A \sqsubseteq \top\}$
2. Αν $A \sqsubseteq B \in T$ και $B \sqsubseteq C \in T$, τότε $T = T \cup \{A \sqsubseteq C\}$
3. Αν $A \sqsubseteq B \in T$ και $A \sqsubseteq C \in T$ και $B \sqcap C \sqsubseteq D \in T$, τότε $T = T \cup \{A \sqsubseteq D\}$
4. Αν $A \sqsubseteq B \in T$ και $B \sqsubseteq \exists r.C \in T$, τότε $T = T \cup \{A \sqsubseteq \exists r.C\}$
5. Αν $A \sqsubseteq \exists r.B \in T$ και $r \sqsubseteq s \in T$, τότε $T = T \cup \{A \sqsubseteq \exists s.B\}$
6. Αν $A \sqsubseteq \exists r.B \in T$ και $B \sqsubseteq C \in T$ και $\exists r.C \sqsubseteq D \in T$, τότε $T = T \cup \{A \sqsubseteq D\}$

Επιπλέον, πριν από την αναλυτική παρουσίαση των εφαρμογών του υποσυστήματος της κανονικοποίησης παρατίθεται διάγραμμα ροής για όλη τη διαδικασία της κανονικοποίησης στο παρακάτω σχήμα.



Σχήμα 5.4 Διάγραμμα ροής υποσυστήματος ταξινόμησης

5.3.1 Εφαρμογή υπολογισμού μεταβατικού κλεισίματος ιδιοτήτων

Η εφαρμογή υπολογισμού μεταβατικού κλεισίματος ιδιοτήτων (transitive closure) ιδιοτήτων σχετίζεται με τις λειτουργίες που επιτελούνται για τον υπολογισμό νέων σχέσεων υπαγωγής ιδιοτήτων μεταξύ εκφράσεων *OWL* ιδιοτήτων αντικειμένων από τις κανονικοποιημένες σχέσεις υπαγωγής ιδιοτήτων που έχουν αποθηκευτεί στο σχήμα της βάσης μετά την διαδικασία της κανονικοποίησης. Αυτή η λειτουργία

επιτυγχάνεται με την επαναληπτική εφαρμογή ενός sql statement στον αντίστοιχο πίνακα του σχήματος βάσης μέχρι αν μην βγάζει νέες σχέσεις και την αποθήκευση του αποτελέσματος σε έναν νέο πίνακα του σχήματος βάσης.

5.3.2 Εφαρμογή απαλοιφής επαγωγικού κανόνα 5

Η εφαρμογή απαλοιφής του επαγωγικού κανόνα 5 (Αν $A \sqsubseteq \exists r.B \in T$ και $r \sqsubseteq s \in T$, τότε $T = T \cup \{A \sqsubseteq \exists s.B\}$) σχετίζεται με τις λειτουργίες που επιτελούνται ώστε να είναι εφικτό ο επαγωγικός κανόνας 5 να εξαιρεθεί από την επαναληπτική εφαρμογή. Δεδομένου ότι έχει προηγηθεί ο υπολογισμός του μεταβατικού κλεισίματος των ιδιοτήτων μπορούμε να εφαρμόσουμε τον επαγωγικό κανόνα 5 μία φορά και να είμαστε σίγουροι ότι δεν χρειάζεται να επαναληφθεί στο μέλλον καθώς θα έχουν εξαχθεί όλες οι σχέσεις ιεραρχίας που θα μπορούσαν να δημιουργηθούν από την εφαρμογή αυτού του κανόνα.

5.3.3 Εφαρμογή δημιουργίας τρέχοντος cluster

Η εφαρμογή δημιουργίας ενός τρέχοντος cluster σχετίζεται με τις απαραίτητες λειτουργίες για την προετοιμασία του σχήματος βάσης πριν από κάθε επανάληψη του αλγορίθμου για την εφαρμογή των επαγωγικών κανόνων σε ένα μόνο τμήμα της οντολογίας. Βασική αρχή του υποσυστήματος της ταξινόμησης είναι ότι οι επαγωγικοί κανόνες δεν εφαρμόζονται στο σύνολο της οντολογίας αλλά σε ένα τμήμα αυτής, το μέγεθος του οποίου καθορίζεται από μία παράμετρο του χρήστη. Ουσιαστικά η παράμετρος αυτή καθορίζει πόσες από τις υπο εξέταση κλάσεις θα εξεταστούν σε κάθε επανάληψη του αλγορίθμου. Επομένως, θα πρέπει να δημιουργηθεί ένα cluster του ολικού σχήματος βάσης που να περιέχει μόνο όσα δεδομένα (σχέσεις υπαγωγής και κανονικοποιημένες μορφές αναπαράστασης των κατασκευαστών) έχουν κάποια σχέση με αυτές τις επιλεγμένες υπό εξέταση κλάσεις. Αυτή ακριβώς τη λειτουργία επιτελεί η εφαρμογή δημιουργίας τρέχοντος cluster. Επιπλέον σε αυτή την εφαρμογή υλοποιείται και ο επαγωγικός κανόνας 1 με την αρχικοποίηση των αντίστοιχων πινάκων.

1. Αν $A \in T$ τότε $T = T \cup \{A \sqsubseteq A, A \sqsubseteq \top\}$

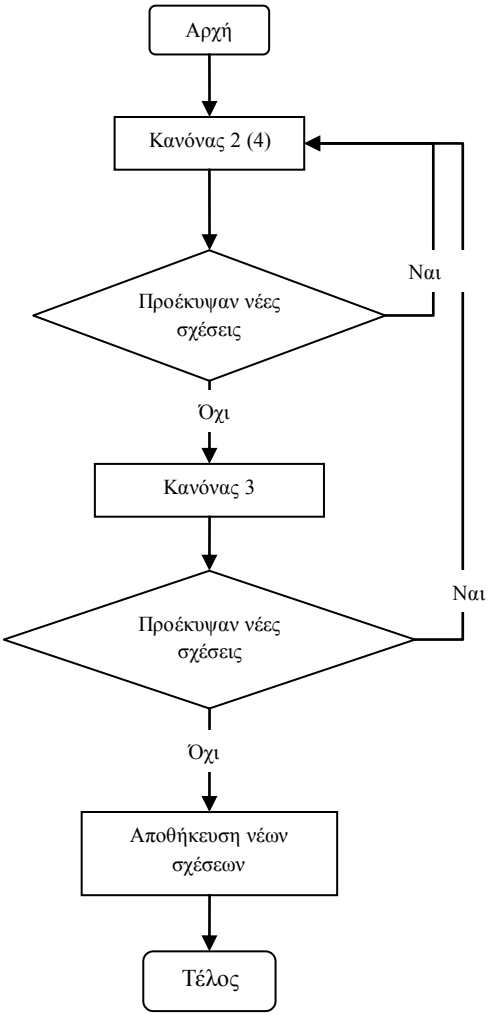
5.3.4 Εφαρμογή υπολογισμού τρέχοντος μεταβατικού κλεισίματος κλάσεων

Η εφαρμογή υπολογισμού τρέχοντος μεταβατικού κλεισίματος κλάσεων σχετίζεται με τις απαραίτητες λειτουργίες για τον υπολογισμό του μεταβατικού κλεισίματος (transitive closure) για τις υπό εξέταση κλάσεις. Ουσιαστικά πρόκειται για την εφαρμογή που υλοποιεί τους επαγωγικούς κανόνες 2, 3 και 4 με την εφαρμογή επαναληπτικών sql statements για την δημιουργία του transitive closure.

2. Αν $A \sqsubseteq B \in T$ και $B \sqsubseteq C \in T$, τότε $T = T \cup \{A \sqsubseteq C\}$
3. Αν $A \sqsubseteq B \in T$ και $A \sqsubseteq C \in T$ και $B \sqcap C \sqsubseteq D \in T$, τότε $T = T \cup \{A \sqsubseteq D\}$
4. Αν $A \sqsubseteq B \in T$ και $B \sqsubseteq \exists r.C \in T$, τότε $T = T \cup \{A \sqsubseteq \exists r.C\}$

Με τον τρόπο που έχει δημιουργηθεί η κανονικοποιημένη οντολογία ο κανόνας 4 και ο κανόνας 2 εφαρμόζονται ταυτόχρονα από το ίδιο σύνολο sql statements καθώς στο πίνακα με τις σχέσεις υπαγωγής δεν γίνεται διάκριση των id ανάλογα με το τι αντιπροσωπεύουν. Επίσης πριν από το τέλος αυτής της εφαρμογής οι νέες υπονοούμενες σχέσεις υπαγωγής εισάγονται στον πίνακα των τελικών αποτελεσμάτων. Το διάγραμμα ροής των βημάτων αυτής της διαδικασίας φαίνεται στη συνέχεια ενώ τα βήματα είναι:

- Εξαντλητική επαναληπτική εφαρμογή του κανόνα 2 με τρέχων σύνολο των πίνακα που αποθηκεύονται οι σχέσεις ιεραρχίας για τις υπό εξέταση έννοιες και σύνολο βάσης τον πίνακα μερικών αποτελεσμάτων.
- Εφαρμογή του κανόνα 3 από τον οποίο προκύπτουν σχέσεις ιεραρχίας εξαιτίας του κατασκευαστή της τομής.
- Αν μετά την εφαρμογή του κανόνα 3 έχουμε νέα αποτελέσματα τότε επαναλαμβάνουμε τη διαδικασία από την αρχή αν όχι συνεχίζουμε στο επόμενο βήμα.
- Αποθήκευση των νέων σχέσεων ιεραρχίας που προέκυψαν στον πίνακα των τελικών αποτελεσμάτων.



Σχήμα 5.5 Διάγραμμα ροής εφαρμογής υπολογισμού τρέχοντος μεταβατικού κλεισίματος

5.3.5 Εφαρμογή υπολογισμού σχέσεων ιεραρχίας λόγω του υπαρξιακού κατασκευαστή

Η εφαρμογή αυτή σχετίζεται με τις λειτουργίες που είναι απαραίτητες για τον υπολογισμό νέων σχέσεων ιεραρχίας που προκύπτουν από την σημασιολογία του πλήρους υπαρξιακού ποσοδείκτη. Επομένως, αναφερόμαστε στην περίπτωση εφαρμογή του επαγωγικού κανόνα 6.

6. Αν $A \sqsubseteq \exists r.B \in T$ και $B \sqsubseteq C \in T$ και $\exists r.C \sqsubseteq D \in T$, τότε $T = T \cup \{A \sqsubseteq D\}$

Και σε αυτή την περίπτωση ο επαγωγικός κανόνας, δηλαδή τα αντίστοιχα sql statements, δεν εκτελείται σε όλη την οντολογία αλλά σε ένα τμήμα αυτής. Συγκεκριμένα εφαρμόζουμε τον κανόνα 6 μόνο για το σύνολο σχέσεων ιεραρχίας που προέκυψαν από την εφαρμογή υπολογισμού τρέχοντος μεταβατικού κλεισίματος. Τα

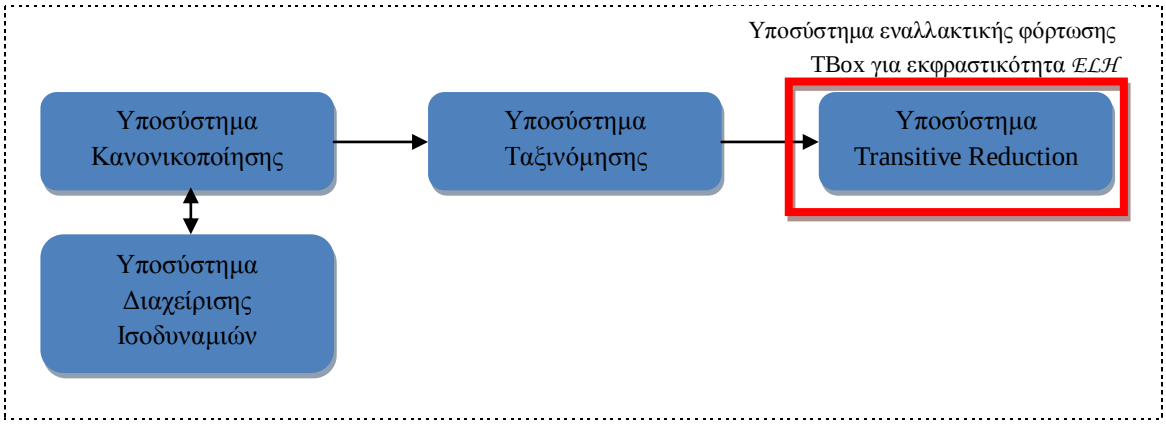
αποτελέσματα αυτής της εφαρμογής αποθηκεύονται ταυτόχρονα σε έναν προσωρινό πίνακα και προστίθενται στον πίνακα των μερικών αποτελεσμάτων.

5.3.6 Εφαρμογή εύρεσης εννοιών που πρέπει να επανεξεταστούν

Η εφαρμογή εύρεσης εννοιών που πρέπει να επανεξεταστούν σχετίζεται με τις λειτουργίες που είναι απαραίτητες για τον εντοπισμό εννοιών που έχουν εξεταστεί σε προηγούμενες επαναλήψεις του αλγορίθμου και πρέπει να εξεταστούν ξανά για να είναι πλήρης ο αλγόριθμος ταξινόμησης. Για τον εντοπισμό αυτών των εννοιών επεξεργαζόμαστε τις σχέσεις ιεραρχίας που προέκυψαν από τον επαγωγικό κανόνα 6 σε συνδυασμό με τα τελικά αποτελέσματα μέχρι αυτή τη στιγμή. Για όσες έννοιες υπάρχουν ενδείξεις ότι μπορούν να εμπλακούν κατά τη διάρκεια υπολογισμού του μεταβατικού κλεισίματος με κάποια από τις νέες υπονοούμενες σχέσεις του επαγωγικού κανόνα 6, τις θέτουμε υπό εξέταση. Οι έννοιες που θα βρεθούν από αυτή τη διαδικασία τοποθετούνται στο τέλος της ουράς των προς εξέταση στοιχείων.

5.4 Υποσύστημα Transitive Reduction

Το υποσύστημα Transitive Reduction είναι υπεύθυνο για την δημιουργία μιας ταξονομίας, η οποία περιέχει μόνο τις άμεσες σχέσεις υπαγωγής μεταξύ των ατομικών ονοματισμένων κλάσεων της οντολογίας. Η λειτουργία του στην ουσία είναι πολύ απλή. Χρησιμοποιώντας τα δεδομένα που έχουν παραχθεί από το υποσύστημα ταξινόμησης, από όλες τις σχέσεις υπαγωγής μεταξύ εννοιών (αυτές που υπήρχαν στο αρχικό TBox συν αυτές που παράχθηκαν από την ταξινόμηση) κρατάει μόνο αυτές που τόσο η υποκλάση, όσο και η υπερκλάση είναι ατομικές ονοματισμένες κλάσεις.



Σχήμα 5.6 Η θέση του υποσυστήματος Transitive Reduction στο ολικό σύστημα

5.5 Παράδειγμα εκτέλεσης

Έστω ότι έχουμε το ακόλουθο TBox :

$$\text{MuscularOrgan} \equiv \text{Organ} \sqcap \exists \text{isPartOf.MuscularSystem}$$
$$\text{Heart} \sqsubseteq \text{Organ} \sqcap \exists \text{belongsTo.}(\text{MuscularSystem} \sqcap \text{CirculatorySystem})$$
$$\text{belongsTo} \sqsubseteq \text{isPartOf}$$

Στο Σχήμα 5.7 φαίνεται η αναπαράσταση του παραπάνω TBox στην βάση, μετά την ενεργοποίηση της εφαρμογής διαχείρισης αξιωμάτων (5.1.1) και πριν ενεργοποιηθεί η εφαρμογή χειρισμού *OWL* εκφράσεων εννοιών (5.1.4). Η ισοδυναμία των αναγνωριστικών 1345 και 6758 έχει κρατηθεί σε μία προσωρινή δομή αναπαράστασης ισοδυναμιών.

Concepts

concept_name	concept_id
MuscularOrgan	1345
Organ ⊓ ∃isPartOf.MuscularSystem	6758
Heart	7689
Organ ⊓ ∃belongsTo.(MuscularSystem ⊓ CirculatorySystem)	4523

Atomic Concepts

concept_name	concept_id
MuscularOrgan	1345
Heart	7689

Roles

role_name	role_id
isPartOf	1457
belongsTo	6657

ax_class

sub	super
7689	4523

tmp_inter

sub1	sub2	super

tmp_exis_pos

sub	super_r	super_c

ax_object

sub	super
6657	1457

tmp_exis_neg

sub_r	sub_c	super

Σχήμα 5.7 Κανονικοποίηση του TBox σε αρχικό στάδιο

Στο Σχήμα 5.8 παρατηρείται η διαμόρφωση του αρχικού TBox μετά την ενεργοποίηση της εφαρμογής χειρισμού του κατασκευαστή της τομής (5.1.4.1) για τις δύο αρχικές τομές (με αναγνωριστικά 6758 και 4523) που βρίσκονται εκτός των υπαρξιακών ποσοτικοποιήσεων.

Στο Σχήμα 5.9 παρατίθεται το TBox μετά την ενεργοποίηση της εφαρμογής χειρισμού του υπαρξιακού κατασκευαστή (5.1.4.2), ενώ στο Σχήμα 5.10 παρατηρείται το Tbox μετά την επανάκληση της εφαρμογής χειρισμού του κατασκευαστή της τομής για την τομή που συναντάμε μέσα στην μία σχέση υπαρξιακής ποσοτικοποίησης (αναγνωριστικό 7413).

Concepts	
concept_name	concept_id
MuscularOrgan	1345
Organ ⊓ ∃isPartOf.MuscularSystem	6758
Heart	7689
Organ ⊓ ∃belongsTo.(MuscularSystem ⊓ CirculatorySystem)	4523
Organ	***5476***
∃isPartOf.MuscularSystem	***9876***
∃belongsTo.(MuscularSystem ⊓ CirculatorySystem)	***9987***

Atomic Concepts	
concept_name	concept_id
MuscularOrgan	1345
Heart	7689
Organ	***5476***

Roles	
role_name	role_id
isPartOf	1457
belongsTo	6657

ax_class	
sub	super
7689	4523
6758	***5476***
6758	***9876***
4523	***5476***
4523	***9987***

tmp_inter		
sub1	sub2	super
5476	***9876***	***6758***
5476	***9987***	***4523***

tmp_exis_pos		
sub	super_r	super_c

tmp_exis_neg		
sub_r	sub_c	super

ax_object	
sub	super
6657	1457

Σχήμα 5.8 Κανονικοποίηση του TBox μετά την επεξεργασία των αρχικών τομών

Concepts	
concept_name	concept_id
MuscularOrgan	1345
Organ ⊓ ∃isPartOf.MuscularSystem	6758
Heart	7689
Organ ⊓ ∃belongsTo.(MuscularSystem ⊓ CirculatorySystem)	4523
Organ	5476
∃isPartOf.MuscularSystem	9876
∃belongsTo.(MuscularSystem ⊓ CirculatorySystem)	9987
MuscularSystem ⊓ CirculatorySystem	***7413***
MuscularSystem	***2345***

Atomic Concepts	
concept_name	concept_id
MuscularOrgan	1345
Heart	7689
Organ	5476
MuscularSystem	***2345***

Roles	
role_name	role_id
isPartOf	1457
belongsTo	6657

ax_class	
sub	super
7689	4523
6758	5476
6758	9876
4523	5476
4523	9987

tmp_inter		
sub1	sub2	super
5476	9876	6758
5476	9987	4523

tmp_exis_pos		
sub	super_r	super_c
9876	***1457***	***2345***
9987	***6657***	***7413***

ax_object	
sub	super
6657	1457

tmp_exis_neg		
sub_r	sub_c	super
1457	***2345***	***9876***

Σχήμα 5.9 Κανονικοποίηση του TBox μετά την επεξεργασία των υπαρξιακών ποσοτικοποιήσεων

Concepts	
concept_name	concept_id
MuscularOrgan	1345
Organ ⊓ ∃isPartOf.MuscularSystem	6758
Heart	7689
Organ ⊓ ∃belongsTo.(MuscularSystem ⊓ CirculatorySystem)	4523
Organ	5476
∃isPartOf.MuscularSystem	9876
∃belongsTo.(MuscularSystem ⊓ CirculatorySystem)	9987
MuscularSystem ⊓ CirculatorySystem	7413
MuscularSystem	2345
CirculatorySystem	***1113***

Atomic Concepts	
concept_name	concept_id
MuscularOrgan	1345
Heart	7689
Organ	5476
MuscularSystem	2345
CirculatorySystem	***1113***

Roles	
role_name	role_id
isPartOf	1457
belongsTo	6657

ax_class	
sub	super
7689	4523
6758	5476
6758	9876
4523	5476
4523	9987
7413	***2345***
7413	***1113***

tmp_inter		
sub1	sub2	super
5476	9876	6758
5476	9987	4523
2345	***1113***	***7413***

tmp_exis_pos		
sub	super_r	super_c
9876	1457	2345
9987	6657	7413

ax_object	
sub	super
6657	1457

tmp_exis_neg		
sub_r	sub_c	super
1457	2345	9876

Σχήμα 5.10 Κανονικοποίηση του TBox μετά την επεξεργασία όλων των κατασκευαστών

Τέλος στο Σχήμα 5.11 παρουσιάζεται το Tbox μετά το τέλος της κανονικοποίησης, δηλαδή μετά και την ενεργοποίηση του υποσυστήματος διαχείρισης ισοδυναμιών (5.2). Να σημειωθεί ότι σε οποιοδήποτε στοιχείο αποδίδουμε ένα αναγνωριστικό, αυτό γίνεται πάντα μέσω της εφαρμογής απόδοσης μοναδικού αναγνωριστικού (5.1.3).

Concepts	
concept_name	concept_id
MuscularOrgan	1345
Organ ⊓ ∃isPartOf.MuscularSystem	~~6758~~ ~~1345~~
Heart	7689
Organ ⊓ ∃belongsTo.(MuscularSystem ⊓ CirculatorySystem)	4523
Organ	5476
∃isPartOf.MuscularSystem	9876
∃belongsTo.(MuscularSystem ⊓ CirculatorySystem)	9987
MuscularSystem ⊓ CirculatorySystem	7413
MuscularSystem	2345
CirculatorySystem	1113

Atomic Concepts	
concept_name	concept_id
MuscularOrgan	1345
Heart	7689
Organ	5476
MuscularSystem	2345
CirculatorySystem	1113

Roles	
role_name	role_id
isPartOf	1457
belongsTo	6657

ax_class	
sub	super
7689	4523
~~6758~~ ~~1345~~	5476
~~6758~~ ~~1345~~	9876
4523	5476
4523	9987
7413	2345
7413	1113

tmp_inter		
sub1	sub2	super
5476	9876	6758
5476	9987	4523
2345	1113	7413

tmp_exis_pos		
sub	super_r	super_c
9876	1457	2345
9987	6657	7413

ax_object	
sub	super
6657	1457

tmp_exis_neg		
sub_r	sub_c	super
1457	2345	9876

Σχήμα 5.10 Τελική κανονικοποίηση του TBox

Στη συνέχεια ξεκινά την λειτουργία του το υποσύστημα ταξινόμησης και συγκεκριμένα η εφαρμογή υπολογισμού μεταβατικού κλεισίματος ιδιοτήτων (5.3.1), η οποία στο συγκεκριμένο παράδειγμα δεν παράγει επιπλέον αποτελέσματα. Έπειτα ενεργοποιείται η εφαρμογή απαλοιφής του επαγωγικού κανόνα 5 (5.3.2), τα αποτελέσματα της κλήσης της οποίας παρουσιάζονται στο Σχήμα 5.11.

Concepts	
concept_name	concept_id
MuscularOrgan	1345
Organ ⊓ ∃isPartOf.MuscularSystem	~~6758~~ 1345
Heart	7689
Organ ⊓ ∃belongsTo.(MuscularSystem ⊓ CirculatorySystem)	4523
Organ	5476
∃isPartOf.MuscularSystem	9876
∃belongsTo.(MuscularSystem ⊓ CirculatorySystem)	9987
MuscularSystem ⊓ CirculatorySystem	7413
MuscularSystem	2345
CirculatorySystem	1113

Atomic Concepts	
concept_name	concept_id
MuscularOrgan	1345
Heart	7689
Organ	5476
MuscularSystem	2345
CirculatorySystem	1113

Roles	
role_name	role_id
isPartOf	1457
belongsTo	6657

ax_class	
sub	super
7689	4523
~~6758~~ 1345	5476
~~6758~~ 1345	9876
4523	5476
4523	9987
7413	2345
7413	1113

tmp_inter		
sub1	sub2	super
5476	9876	6758
5476	9987	4523
2345	1113	7413

tmp_exis_pos		
sub	super_r	super_c
9876	1457	2345
9987	6657	7413

ax_object	
sub	super
6657	1457

tmp_exis_neg		
sub_r	sub_c	super
1457	2345	9876
6657	***2345***	***9876***

Σχήμα 5.11 Διαμόρφωση του TBox μετά την κλήση της εφαρμογής υπολογισμού μεταβατικού κλεισίματος ιδιοτήτων

Από το παραπάνω σχήμα μπορούμε να δούμε πολύ εύκολα τις έννοιες οι οποίες θα πρέπει να ελεγχθούν κατά την διαδικασία της ταξινόμησης. Είναι όλες οι έννοιες του πίνακα Atomic Concepts συν όλες τις έννοιες της στήλης super_c του πίνακα tmp_exis_pos. Στο συγκεκριμένο παράδειγμα ο αριθμός N που έχουμε επιλέξει (και που αντιστοιχεί στον αριθμό των εννοιών που θα εξετάζονται σε κάθε επανάληψη) είναι N = 3. Στο Σχήμα 5.12 μπορούμε να παρατηρήσουμε την βάση μας μετά το πέρας της πρώτης εκτέλεσης της διαδικασίας ταξινόμησης για τις τρεις πρώτες έννοιες με αναγνωριστικά 1345, 5476 και 2345. Στον πίνακα tmp_cl_c μπορούμε να διαπιστώσουμε εύκολα τις αρχικές εγγραφές που έχουν προστεθεί από την εφαρμογή δημιουργίας τρέχοντος cluster (5.3.3), και τις δύο νέες που προστίθενται από την εφαρμογή υπολογισμού τρέχοντος μεταβατικού κλεισίματος κλάσεων (5.3.4). Ο πίνακας cl_class περιέχει αθροιστικά όλους τους πίνακες tmp_cl_c που παράγονται σε κάθε επανάληψη της διαδικασίας για N έννοιες. Στον πίνακα tmp_axioms

αποθηκεύονται οι αρχικές εγγραφές του πίνακα ax_class (Σχήμα 5.11) και αθροιστικά όλους τους πίνακες tmp_ext_cl για κάθε επανάληψη της διαδικασίας για N έννοιες. Ο πίνακας αυτός περιέχει σε κάθε επανάληψη τις εγγραφές που παράγονται από την εφαρμογή υπολογισμού σχέσεων ιεραρχίας λόγω υπαρξιακού κατασκευαστή (5.3.5)

tmp_cl_c		
sub	super	step
1345	1345	0
5476	5476	0
2345	2345	0
1345	***5476***	***1***
1345	***9876***	***1***

cl_class	
sub	super
1345	***1345***
5476	***5476***
2345	***2345***
1345	***5476***
1345	***9876***

tmp_axioms	
sub	super
1345	5476
1345	9876
7689	4523
4523	5476
4523	9987
7413	2345
7413	1113

tmp_ext_cl	
sub	super

Τίποτα για επανεξέταση

Σχήμα 5.12 Διαμόρφωση της βάσης μετά την εκτέλεση της διαδικασίας ταξινόμησης για τις τρεις πρώτες έννοιες

Στο Σχήμα 5.13 παρατίθεται η βάση μετά την επανεκτέλεση της διαδικασίας ταξινόμησης για τις τρεις επόμενες έννοιες, δηλαδή για τις έννοιες με αναγνωριστικά 7689, 1113 και 7413.

tmp_cl_c		
sub	super	step
7689	7689	0
1113	1113	0
7413	7413	0
7689	***4523***	***1***
7413	***2345***	***1***
7413	***1113***	***1***
7689	***5476***	***2***
7689	***9987***	***2***
7689	***4523***	***3***

tmp_axioms	
sub	super
1345	5476
1345	9876
7689	4523
4523	5476
4523	9987
7413	2345
7413	1113
9987	***9876***

cl_class	
sub	super
1345	1345
5476	5476
2345	2345
1345	5476
1345	9876
7689	***7689***
1113	***1113***
7413	***7413***
7689	***4523***
7413	***2345***
7413	***1113***
7689	***5476***
7689	***9987***
7689	***4523***

tmp_ext_cl	
sub	super
9987	***9876***

Επανεξέταση το 7689

Σχήμα 5.13 Διαμόρφωση της βάσης μετά την εκτέλεση της διαδικασίας ταξινόμησης για τις τρεις επόμενες έννοιες

Στο Σχήμα 5.14 παρουσιάζεται η διαμόρφωση της βάσης μετά το πέρας της διαδικασίας ταξινόμησης, δηλαδή μετά και την επανεξέταση του στοιχείου με αναγνωριστικό 7689. Με μπλε χρώμα οι νέες σχέσεις ιεραρχίας που προέκυψαν μετά το τέλος της ταξινόμησης και αντιστοιχούν στα αξιώματα:

$$\text{Heart} \sqsubseteq \exists \text{belongsTo.MuscularSystem}$$
$$\text{Heart} \sqsubseteq \exists \text{isPartOf.MuscularSystem}$$
$$\text{Heart} \sqsubseteq \text{MuscularOrgan}$$

tmp_cl_c		
sub	super	step
7689	7689	0
7689	4523	0
7689	5476	0
7689	9987	0
7689	4523	0
7689	***9876***	***1***
7689	***1345***	***2***

tmp_axioms	
sub	super
1345	5476
1345	9876
7689	4523
4523	5476
4523	9987
7413	2345
7413	1113
9987	9876

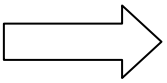
cl_class	
sub	super
1345	1345
5476	5476
2345	2345
1345	5476
1345	9876
7689	7689
1113	1113
7413	7413
7689	4523
7413	2345
7413	1113
7689	5476
7689	9987
7689	4523
7689	***9876***
7689	***1345***

tmp_ext_cl	
sub	super

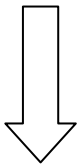
Σχήμα 5.14 Διαμόρφωση της βάσης μετά το πέρας της διαδικασίας ταξινόμησης

Στη συνέχεια αναλαμβάνει το υποσύστημα Transitive Reduction (5.4) το οποίο χρησιμοποιώντας τον πίνακα cl_class που έχει παραχθεί από το υποσύστημα ταξινόμησης, δημιουργεί μία ταξονομία που περιλαμβάνει μόνο τις άμεσες σχέσεις υπαγωγής μεταξύ των ατομικών ονοματισμένων κλάσεων. Στο Σχήμα 5.15 παρατηρούμε την τελική μορφή της βάσης μας για το συγκεκριμένο παράδειγμα, μετά και την ενεργοποίηση του συστήματος Transitive Reduction.

cl_class	
sub	super
1345	1345
5476	5476
2345	2345
1345	5476
1345	9876
7689	7689
1113	1113
7413	7413
7689	4523
7413	2345
7413	1113
7689	5476
7689	9987
7689	4523
7689	***9876***
7689	***1345***



cl_class_reduct	
sub	super
1345	***5476***
7689	***5476***
7689	***1345***



MuscularOrgan ⊑ Organ

Heart ⊑ Organ

Heart ⊑ MuscularOrgan

Σχήμα 5.15 Διαμόρφωση της βάσης μετά το πέρας της διαδικασίας υπολογισμού του Transitive Reduction

6

Υλοποίηση

Στο κεφάλαιο αυτό παρουσιάζεται η υλοποίηση της επέκτασης του υπάρχοντος συστήματος DBRS που δημιουργήσαμε, για τον υπολογισμό όλων των υπονοούμενων σχέσεων ιεραρχίας μεταξύ των εννοιών μιας οντολογίας με εκφραστικότητα περιορισμένη στο τμήμα *EL* της γλώσσας αναπαράστασης *OWL*. Στην πρώτη ενότητα παρουσιάζονται διάφορες διαδικασίες που επιτελούνται από το σύστημα και οι οποίες παρουσιάζουν ιδιαίτερο ενδιαφέρον. Στην ενότητα δύο αναφέρονται οι πλατφόρμες στις οποίες ελέγχθηκε η λειτουργία του συστήματός μας και τα προγραμματιστικά εργαλεία με τα οποία δημιουργήθηκε. Στην ενότητα τρία δίνονται οι οδηγίες για την εγκατάσταση του συστήματος και στην ενότητα τέσσερα παρουσιάζονται οι Java κλάσεις του και τα ευρετήρια που χρησιμοποιήθηκαν.

6.1 Αλγόριθμοι

Στην ενότητα αυτή παρουσιάζονται οι πιο σημαντικοί αλγόριθμοι του συστήματός μας. Αρχικά επεξηγείται η λειτουργία του κάθε αλγορίθμου και στη συνέχεια παρατίθεται ο ίδιος ο αλγόριθμος σε ψευδοκώδικα.

6.1.1 Απόδοση μοναδικού αναγνωριστικού

Στο σύστημά μας κάθε *OWL* έκφραση αναπαριστάται από ένα μοναδικό αναγνωριστικό. Αρχικά χρησιμοποιήθηκε η μέθοδος της java hashcode(), η οποία δέχεται ένα αντικείμενο και του προσδίδει κανονικά ένα μοναδικό αναγνωριστικό (έναν ακέραιο αριθμό). Όμως παρατηρήθηκε ότι η μέθοδος αυτή κάποιες φορές τείνει να δίνει το ίδιο αναγνωριστικό σε δύο αντικείμενα τελείως διαφορετικά μεταξύ τους, με αποτέλεσμα τη μη μοναδικότητα. Για τον λόγο αυτό δημιουργήθηκε ο αλγόριθμος απόδοσης μοναδικού αναγνωριστικού.

Ο αλγόριθμος αυτός δέχεται ως είσοδο το αναγνωριστικό του αντικειμένου που του επιδόθηκε από την μέθοδο hashcode() και το ίδιο το αντικείμενο ως String, τα αποθηκεύει σε ένα HashMap και επιστρέφει ένα μοναδικό αναγνωριστικό. Αυτό γίνεται ως εξής: Όταν δοθεί ένα ζευγάρι αναγνωριστικού-String προς αποθήκευση, τότε η μέθοδος πριν το προσθέσει στο HashMap ελέγχει πρώτα αν υπάρχει ήδη αποθηκευμένο το αναγνωριστικό. Αν όχι, τότε απλά προσθέτει το ζευγάρι και το μοναδικό αναγνωριστικό που επιστρέφει είναι αυτό που του δόθηκε ως είσοδος (αυτό δηλαδή που επιδόθηκε από την hashcode()). Αν όμως το αναγνωριστικό υπάρχει τότε υπάρχουν δύο περιπτώσεις:

- Το String που αντιστοιχεί στο αποθηκευμένο αναγνωριστικό είναι ίδιο με αυτό που δόθηκε ως είσοδος. Τότε απλά το ζευγάρι αυτό έχει ήδη προστεθεί και επιστρέφεται το αναγνωριστικό.
- Το String είναι διαφορετικό. Τότε το αναγνωριστικό αυτό χρησιμοποιείται ήδη για άλλο αντικείμενο. Η μέθοδος προσθέτει στο αναγνωριστικό τον αριθμό 1 και επαναλαμβάνει την διαδικασία. Έτσι είτε θα καταλήξει να βρει το ζευγάρι αναγνωριστικό-String ήδη αποθηκευμένο στο HashMap, είτε θα καταλήξει σε ένα ελεύθερο αναγνωριστικό για να αντιστοιχήσει στο String.

Γενικά οι συγκρούσεις αυτές (conflicts) που δημιουργούνται λόγω της μεθόδου hashcode() είναι λίγες, οπότε συνήθως αντιμετωπίζουμε την πρώτη περίπτωση.

Ακολουθεί η περιγραφή της μεθόδου του αλγορίθμου για την εισαγωγή μιας εγγραφής στο HashMap (μέθοδος addEntry της κλάσης CheckSet.java).

Αλγόριθμος

addEntry(int code, String name):
Αν το HashMap δεν περιέχει το code **τότε**
{
    Προσθήκη στο HashMap του ζευγαριού (code,name);
    Επέστρεψε το code;

```
}
Διαφορετικά
{
        Αν το String που αντιστοιχεί ήδη το HashMap στο code είναι ίδιο με το name
τότε
        {
                Επέστρεψε το code;
        }
        Επανέλαβε
        {
                Προσθήκη στο code του αριθμού 1;
                Αν το HashMap δεν περιέχει το code τότε
                {
                        Προσθήκη στο HashMap του ζευγαριού (code,name);
                        Επέστρεψε το code;
                }
                Αν το String που αντιστοιχεί ήδη το HashMap στο code είναι ίδιο με
το                      name τότε
                {
                        Επέστρεψε το code;
                }
        }
}
```

6.1.2 Διαχείριση ισοδύναμων κλάσεων

Κατά την κανονικοποίηση της οντολογίας όταν συναντάται ένα αξίωμα ισοδυναμίας δίνονται στον αλγόριθμο διαχείρισης ισοδυναμιών τα δύο αναγνωριστικά των δύο κλάσεων, έστω id1 και id2. Τα αναγνωριστικά αυτά είναι διαφορετικά, αλλά στην τελικά μορφή της βάσης μετά την κανονικοποίηση θέλουμε οι δύο αυτές κλάσεις και όσες άλλες είναι ισοδύναμες με αυτές να αναπαριστούνται από το ίδιο αναγνωριστικό.

Για να το επιτύχει αυτό ο αλγόριθμος χρησιμοποιεί την δομή HashMap. Η λογική είναι ότι από όλες τις ισοδύναμες μεταξύ τους κλάσεις N, οι N-1 κλάσεις δείχνουν την κλάση N. Έτσι μπορεί να χρησιμοποιηθεί το αναγνωριστικό της N για την αναπαράσταση όλων των ισοδύναμων κλάσεων. Ο αλγόριθμος αποτελείται ουσιαστικά από δύο μεθόδους.

Η πρώτη μέθοδος καλείται όπως αναφέρθηκε μετά από κάθε αξίωμα ισοδυναμίας και δέχεται το ζευγάρι (id1,id2) ως είσοδο. Υπάρχουν τέσσερις περιπτώσεις:

- Τα id1 και id2 δεν περιέχονται ως κλειδιά στο HashMap. Τότε προστίθενται οι εγγραφές (id1,id1) και (id2,id1). Ουσιαστικά δηλαδή η id1 δείχνει τον εαυτό της και η id2 την id1.

- Περιέχεται μόνο το id1 ως κλειδί. Προφανώς η id1 δείχνει ή τον εαυτό της ή κάποια άλλη κλάση. Αν δείχνει τον εαυτό της προστίθεται η εγγραφή (id2,id1). Αν όχι, βρίσκεται το αναγνωριστικό της κλάσης που δείχνει η «αλυσίδα» του id1, έστω rec1 [πχ (id5,id4) , (id4,id1) , (id1,id2) , (id2,rec1) , (rec1,rec1)]. Το rec1 είναι πάντα εύκολο να βρεθεί γιατί είναι η μόνη εγγραφή στην αλυσίδα που δείχνει τον εαυτό της.
- Περιέχεται μόνο το id2 ως κλειδί. Αντίστοιχα με την προηγούμενη περίπτωση προστίθεται η εγγραφή (id1,id2) αν το id2 δείχνει τον εαυτό του ή η εγγραφή (id1,rec2).
- Το HashMap περιέχει ως κλειδιά και το id1 και το id2. Αφαιρείται η εγγραφή με κλειδί το rec2 από το HashMap και προστίθεται η εγγραφή (rec2,rec1).

Η δεύτερη μέθοδος καλείται όταν έχουν διαβαστεί όλα τα αξιώματα της οντολογίας. Για κάθε κλειδί (έστω id) του HashMap που δεν δείχνει στον εαυτό του βρίσκει το αναγνωριστικό της κλάσης που δείχνει η αλυσίδα του (έστω rec) και προσθέτει σε ένα καινούργιο HashMap την εγγραφή (id,rec) [πχ η αλυσίδα που δόθηκε πιο πάνω θα γινότανε (id5,rec1) , (id4,rec1) , (id1,rec1) , (id2,rec1)].

Οι εγγραφές του τελικού αυτού HashMap τυπώνονται σε ένα προσωρινό αρχείο .txt και περνάνε σαν πίνακας στην βάση. Μέσω του πίνακα αυτού και SQL ερωτημάτων ενημερώνονται όλοι οι υπόλοιποι πίνακες της βάσης πριν το τέλος της διαδικασίας κανονικοποίησης και έτσι όλες οι ισοδύναμες κλάσεις αποκτούν το ίδιο αναγνωριστικό σε όλους τους πίνακες.

Ακολουθεί η περιγραφή των δύο μεθόδων του αλγορίθμου (addEquiv και finalHash της κλάσης MapHash.java).

Αλγόριθμος

addEquiv(int ιd1, int id2):
Νέα δυαδική μεταβλητή exists1 := false;
Νέα μεταβλητή rec1 := id1;
Αν το id1 περιέχεται στο HashMap **τότε**
{
    rec1 := το αναγνωριστικό που αντιστοιχεί το HashMap στο id1;
    Όσο id1 <> rec1
    {
        id1 := rec1;
        rec1 := το αναγνωριστικό που αντιστοιχεί το HashMap στο id1;
    }
    exists1 := true;
}
Νέα δυαδική μεταβλητή exists2 := false;
Νέα μεταβλητή rec2 := id2;
Αν το id2 περιέχεται στο HashMap **τότε**
{

rec2 := το αναγνωριστικό που αντιστοιχεί το HashMap στο id2;
Όσο id2 <> rec2
{
id2 := rec2;
rec2 := το αναγνωριστικό που αντιστοιχεί το HashMap στο id2;
}
exists2 := true;
}
Αν exists1 := false **και** exists2 := false **τότε**
{
Προσθήκη στο HashMap της εγγραφής (rec1 , rec1);
Προσθήκη στο HashMap της εγγραφής (rec2 , rec1);
}
Διαφορετικά αν exists1 := true **και** exists2 := false **τότε**
{
Προσθήκη στο HashMap της εγγραφής (rec2 , rec1);
}
Διαφορετικά αν exists1 := false **και** exists2 := true **τότε**
{
Προσθήκη στο HashMap της εγγραφής (rec1 , rec2);
}
Διαφορετικά exists1 := true **και** exists2 := true **τότε**
{
Αφαίρεση από το HashMap της εγγραφής με κλειδί το rec2;
Προσθήκη στο HashMap της εγγραφής (rec2 , rec1);
}

finalHash():
Πάρε τα κλειδιά του αρχικού HashMap;
Όσο υπάρχουν κλειδιά
{
Μεταβλητή key := το επόμενο κλειδί;
Μεταβλητή value := το αναγνωριστικό που αντιστοιχεί το αρχικό HashMap στο κλειδί;
Προσωρινή μεταβλητή temp := key;
Αν key <> value **τότε**
{
Όσο temp <> value
{
temp := value;
value := το αναγνωριστικό που αντιστοιχεί το αρχικό HashMap στο temp;
}

```
                Προσθήκη στο τελικό HashMap της εγγραφής (key, value);
        }
}
```

6.1.3 Κανονικοποίηση της οντολογίας

Όπως έχει προαναφερθεί, προκειμένου το σύστημα να προχωρήσει στον υπολογισμό των υπονοούμενων σχέσεων ιεραρχίας μεταξύ των εννοιών μιας οντολογίας εκφραστικότητας $\mathcal{EL}$, η οντολογία αυτή πρέπει να επέλθει πρώτα σε μια κανονικοποιημένη μορφή (αξιώματα της μορφής $A \sqsubseteq B$, $A_1 \sqcap A_2 \sqsubseteq B$, $A \sqsubseteq \exists r.B$, $\exists r.A \sqsubseteq B$, $r \sqsubseteq s$).

Αρχικά ανεβάζεται ολόκληρη η οντολογία στην μνήμη και στην συνέχεια προσπελαύνονται το ένα μετά το άλλο τα αξιώματά της (εξαιρουμένων των αξιωμάτων δήλωσης – declaration axioms). Οι περιπτώσεις που μπορούν να προκύψουν είναι τρεις:

- Αξίωμα υπαγωγής εννοιών (subclass_of) : Δίνεται στην υποκλάση και την υπερκλάση του αξιώματος ένα μοναδικό αναγνωριστικό και τα δύο αυτά αναγνωριστικά αποθηκεύονται μαζί σε μια προσωρινή δομή αρχείου (αντιστοιχούν στην μορφή $A \sqsubseteq B$). Στη συνέχεια ελέγχεται αν η υπερκλάση ή η υποκλάση του αξιώματος είναι μια τομή από έννοιες (απλές ή σύνθετες) ή ένας υπαρξιακός περιορισμός και αν ναι τότε προωθούνται στον αλγόριθμο για την διαχείρισή τους (περιγραφή στην ενότητα 6.1.4).
- Αξίωμα υπαγωγής ρόλων (sub_object_property) : Δίνεται και πάλι στην υποκλάση και την υπερκλάση του αξιώματος ένα μοναδικό αναγνωριστικό και τα δύο αυτά αναγνωριστικά αποθηκεύονται μαζί σε μια δεύτερη προσωρινή δομή αρχείου (αντιστοιχούν στην μορφή $r \sqsubseteq s$).
- Αξίωμα ισοδυναμίας εννοιών (equivalent classes) : Δίνεται ένα μοναδικό αναγνωριστικό σε κάθε μία από τις κλάσεις και καλείται ο αλγόριθμος διαχείρισης ισοδύναμων κλάσεων (περιγράφεται στην ενότητα 6.1.2). Στη συνέχεια ελέγχεται αν κάποια από τις δύο είναι μια τομή από έννοιες (απλές ή σύνθετες) ή ένας υπαρξιακός περιορισμός και αν ναι τότε προωθούνται στον αλγόριθμο για την διαχείρισή τους.

Από τα αξιώματα υπαγωγής εννοιών και από τα αξιώματα υπαγωγής ρόλων που διαβάστηκαν έχουν δημιουργηθεί δύο προσωρινά αρχεία .txt που περιέχουν σχέσεις $A \sqsubseteq B$ (όπου A,B απλές ή σύνθετες έννοιες) και $r \sqsubseteq s$ (όπου r,s ρόλοι) αντίστοιχα. Ο αλγόριθμος διαχείρισης τομών και υπαρξιακών περιορισμών προσθέτει σχέσεις στο πρώτο από αυτά και δημιουργεί άλλα τρία όπου περιέχουν αντίστοιχα αξιώματα της μορφής $A_1 \sqcap A_2 \sqsubseteq B$, $A \sqsubseteq \exists r.B$, $\exists r.A \sqsubseteq B$. Δύο επιπλέον αρχεία δημιουργούνται από τον αλγόριθμο απόδοσης μοναδικού αναγνωριστικού και από τον αλγόριθμο διαχείρισης ισοδύναμων κλάσεων. Από όλα αυτά τα παραπάνω αρχεία ο αλγόριθμος κανονικοποίησης

της οντολογίας δημιουργεί τους πίνακες της βάσης και χρησιμοποιώντας τον πίνακα που περιέχει τις ισοδύναμες κλάσεις ενημερώνει όλους τους υπόλοιπους πίνακες ώστε οι ισοδύναμες κλάσεις να έχουν όλες το ίδιο μοναδικό αναγνωριστικό.

Αλγόριθμος

Normalization():
Δημιουργία και σύνδεση με τη βάση, φόρτωση της οντολογίας στην μνήμη και δημιουργία των προσωρινών αρχείων μέσω της κλάσης SystemInitializer ;
OWLLogicalAxiom := τα λογικά αξιώματα της οντολογίας;
Νέα δομή Checkset για την απόδοση μοναδικού αναγνωριστικού;
Νέα δομή MapHash για την διαχείριση των ισοδύναμων κλάσεων;
Όσο υπάρχουν ακόμα λογικά αξιώματα
{
Πάρε το επόμενο αξίωμα;

Αν το αξίωμα είναι υπαγωγής εννοιών **τότε**
{
Απόδοση στις δύο κλάσεις μοναδικών αναγνωριστικών;
Πρόσθεση του ζευγαριού αναγνωριστικών στο προσωρινό αρχείο που περιέχει τις σχέσεις A ⊑ B;
Αν η υπερκλάση είναι τομή εννοιών **τότε**
{
Δίνεται ως είσοδος στον αλγόριθμο διαχείρισης τομών;
}
Διαφορετικά αν η υπερκλάση είναι υπαρξιακός περιορισμός **τότε**
{
Δίνεται ως είσοδος στον αλγόριθμο διαχείρισης υπαρξιακών περιορισμών;
}
Αν η υποκλάση είναι τομή εννοιών **τότε**
{
Δίνεται ως είσοδος στον αλγόριθμο διαχείρισης τομών;
}
Διαφορετικά αν η υποκλάση είναι υπαρξιακός περιορισμός **τότε**
{
Δίνεται ως είσοδος στον αλγόριθμο διαχείρισης υπαρξιακών περιορισμών;
}
}
Διαφορετικά αν το αξίωμα είναι υπαγωγής ρόλων **τότε**
{
Απόδοση στις δύο κλάσεις μοναδικών αναγνωριστικών;

Πρόσθεση του ζευγαριού αναγνωριστικών στο προσωρινό αρχείο που περιέχει τις σχέσεις r ⊑ s;

}

Διαφορετικά αν το αξίωμα ισοδυναμίας εννοιών **τότε**

{

Απόδοση στις δύο κλάσεις μοναδικών αναγνωριστικών;

Το ζευγάρι αναγνωριστικών δίνεται ως είσοδος στον αλγόριθμο διαχείρισης ισοδύναμων κλάσεων;

Για κάθε μία από τις δύο κλάσεις

{

Αν η κλάση είναι τομή εννοιών **τότε**

{

Δίνεται ως είσοδος στον αλγόριθμο διαχείρισης τομών;

}

Διαφορετικά αν η κλάση είναι υπαρξιακός περιορισμός **τότε**

{

Δίνεται ως είσοδος στον αλγόριθμο διαχείρισης υπαρξιακών περιορισμών;

}

}

}

}

Κλήση της μεθόδου για να λάβει η δομή MapHash την τελική της μορφή;

Κλήση της μεθόδου για την αντιγραφή των περιεχομένων του κάθε προσωρινού αρχείου στον αντίστοιχο πίνακα της βάσης;

Αποστολή και εκτέλεση ενός SQL ερωτήματος για κάθε πίνακα της βάσης, όπου ο κάθε πίνακας ενημερώνεται με βάση την δομή MapHash και αλλάζουν τα αναγνωριστικά των ισοδύναμων κλάσεων ώστε να έχουν όλες το ίδιο αναγνωριστικό;

6.1.4 Διαχείριση τομών και υπαρξιακών περιορισμών

Ο αλγόριθμος αυτός αποτελείται ουσιαστικά από δύο ξεχωριστούς αλγορίθμους, έναν για την διαχείριση τομών και έναν για την διαχείριση υπαρξιακών περιορισμών.

6.1.4.1 Διαχείριση τομών

Ο αλγόριθμος αυτός καλείται όταν κατά την κανονικοποίηση της οντολογίας αναγνωρίζεται μια *OWL* έκφραση ως μία τομή κλάσεων. Δέχεται ως είσοδο την τομή των κλάσεων αυτών, μια λογική (boolean) μεταβλητή, και δύο δομές SystemInitializer και CheckSet. Η λογική μεταβλητή δηλώνει το αν η τομή ήταν στο αριστερό ή στο δεξί μέλος του αξιώματος (true για negative ή false για positive), ενώ

οι δύο δομές χρησιμοποιούνται για το πέρασμα δεδομένων στα προσωρινά αρχεία και για την απόδοση μοναδικού αναγνωριστικού αντίστοιχα.

Αρχικά δίνεται ένα μοναδικό αναγνωριστικό σε όλη τη τομή και λαμβάνονται όλες οι *OWL* εκφράσεις που περιέχονται στην τομή αυτή σε μία λίστα. Απομονώνεται από την λίστα αυτή η πρώτη έκφραση και της δίνεται ένα αναγνωριστικό. Στη συνέχεια ελέγχεται αν η έκφραση αυτή είναι ένας υπαρξιακός περιορισμός και αν ναι, τότε προωθείται στην μέθοδο διαχείρισης υπαρξιακών περιορισμών. Τέλος αφαιρείται η έκφραση από τη λίστα.

Αν η λίστα περιέχει πλέον μία μόνο έκφραση τότε αυτή είναι είτε μια ατομική έννοια, είτε ένας υπαρξιακός περιορισμός. Σε κάθε περίπτωση αποδίδεται ένα αναγνωριστικό και αν είναι υπαρξιακός περιορισμός προωθείται και πάλι για επεξεργασία. Αντίθετα αν η λίστα περιέχει ακόμα πάνω από μία εκφράσεις τότε αυτές αποτελούν μια ακόμα τομή, οπότε συντίθενται πάλι σε μια ενιαία τομή, της δίνεται ένα αναγνωριστικό και προωθείται εκ νέου στον αλγόριθμο.

Τέλος ο αλγόριθμος ενημερώνει τα προσωρινά αρχεία. Συγκεκριμένα, έστω id το αναγνωριστικό της αρχικής τομής, id1 το αναγνωριστικό της πρώτης έκφρασης και id2 το αναγνωριστικό της εναπομείναντας, προσθέτει στο προσωρινό αρχείο που περιέχει τις σχέσεις $A \sqsubseteq B$ τις εγγραφές id-id1 και id-id2 και στο προσωρινό αρχείο που περιέχει τις σχέσεις $A_1 \sqcap A_2 \sqsubseteq B$ την εγγραφή id1-id2-id.

Ακολουθεί η περιγραφή του αλγορίθμου (HashInter της κλάσης HashNormal.java).

Αλγόριθμος

HashInter(OWLObjectIntersectionOf inter, boolean neg, SystemInitializer system, CheckSet checkset):
Μεταβλητή id := το αναγνωριστικό που δίνεται στην τομή;
Τοποθετούνται όλες οι *OWL* εκφράσεις που συμμετέχουν στην τομή σε μία λίστα;
Μεταβλητή id1 := το αναγνωριστικό που δίνεται στην πρώτη έκφραση της λίστας;
Αν η έκφραση αυτή είναι μία έκφραση υπαρξιακού περιορισμού **τότε**
{
    Δίνεται ως είσοδος στον αλγόριθμο διαχείρισης υπαρξιακών περιορισμών;
}
Αφαιρείται η πρώτη έκφραση από την λίστα;
Αν το μέγεθος της λίστας := 1 **τότε**
{
    Μεταβλητή id2 := το αναγνωριστικό που δίνεται στην έκφραση που έχει απομείνει στην λίστα;
    Αν η έκφραση αυτή είναι μία έκφραση υπαρξιακού περιορισμού **τότε**
    {
        Δίνεται ως είσοδος στον αλγόριθμο διαχείρισης υπαρξιακών περιορισμών;
    }
}

Διαφορετικά
{
Οι εκφράσεις που έχουν απομείνει στην λίστα συντίθενται σε μία τομή;
Μεταβλητή id2 := το αναγνωριστικό που δίνεται στην νέα αυτή τομή;
Καλείται εκ νέου ο αλγόριθμος, με είσοδο την τομή αυτή;
}
Προσθήκη στο προσωρινό αρχείο που περιέχει τις σχέσεις $A \sqsubseteq B$ των id-id1 και id-id2;
Προσθήκη στο προσωρινό αρχείο που περιέχει τις σχέσεις $A_1 \sqcap A_2 \sqsubseteq B$ των id1-id2-id;

6.1.4.2 Διαχείριση Υπαρξιακών Περιορισμών

Ο αλγόριθμος αυτός καλείται όταν κατά την κανονικοποίηση της οντολογίας αναγνωρίζεται μια *OWL* έκφραση ως ένας υπαρξιακός περιορισμός. Όπως και ο αντίστοιχος αλγόριθμος των τομών δέχεται ως είσοδο την τομή των κλάσεων αυτών, μια λογική (boolean) μεταβλητή, και δύο δομές SystemInitializer και CheckSet. Η λογική μεταβλητή δηλώνει το αν η τομή ήταν στο αριστερό ή στο δεξί μέλος του αξιώματος (true για negative ή false για positive), ενώ οι δύο δομές χρησιμοποιούνται για το πέρασμα δεδομένων στα προσωρινά αρχεία και για την απόδοση μοναδικού αναγνωριστικού αντίστοιχα.

Αποδίδεται ένα αναγνωριστικό σε όλη την έκφραση, απομονώνονται ο ρόλος και η *OWL* έκφραση που περιέχονται σε αυτήν και τους δίνονται αναγνωριστικά. Επιπλέον αν η *OWL* έκφραση που περιέχεται στην αρχική είναι μία τομή ή ένας υπαρξιακός περιορισμός, τότε καλούνται εκ νέου οι αντίστοιχοι αλγόριθμοι χειρισμού τους.

Ενημερώνονται τέλος τα προσωρινά αρχεία. Αν θέσουμε id το αναγνωριστικό της αρχικής έκφρασης, idr το αναγνωριστικό του ρόλου και id1 το αναγνωριστικό της έκφρασης που περιέχεται στην αρχική, τότε ο αλγόριθμος προσθέτει στο προσωρινό αρχείο που περιέχει τις σχέσεις $A \sqsubseteq \exists r.B$ την εγγραφή id-idr-id1, ενώ αν ο αρχικός υπαρξιακός περιορισμός συμμετέχει αρνητικά στο αξίωμα προσθέτει και στο προσωρινό αρχείο που περιέχει τις σχέσεις $\exists r.A \sqsubseteq B$ την εγγραφή idr-id1-id.

Ακολουθεί η περιγραφή του αλγορίθμου (HashSome της κλάσης HashNormal.java).

Αλγόριθμος

HashSome(OWLObjectSomeValueFrom some, boolean neg, SystemInitializer system, CheckSet checkset):
Μεταβλητή id := το αναγνωριστικό που δίνεται στον υπαρξιακό περιορισμό;
Μεταβλητή idr := το αναγνωριστικό που δίνεται στον ρόλο που συμμετέχει στον υπαρξιακό περιορισμό;

Μεταβλητή id1 := το αναγνωριστικό που δίνεται στην *OWL* έκφραση που περιέχεται στον υπαρξιακό περιορισμό;
Προσθήκη στο προσωρινό αρχείο που περιέχει τις σχέσεις A ⊑ ∃r.B των id-idr-id1;
Αν η neg είναι true **τότε**
{
Προσθήκη στο προσωρινό αρχείο που περιέχει τις σχέσεις ∃r.A ⊑ B των idr-id1- id;
}
Αν η *OWL* έκφραση που περιέχεται στον υπαρξιακό περιορισμό είναι μια τομή εκφράσεων **τότε**
{
Δίνεται ως είσοδος στον αλγόριθμο διαχείρισης τομών;
}
Διαφορετικά αν η *OWL* έκφραση που περιέχεται στον υπαρξιακό περιορισμό είναι ένας υπαρξιακός περιορισμός **τότε**
{
Καλείται εκ νέου ο αλγόριθμος, με είσοδο τον υπαρξιακό αυτόν περιορισμό;
}

6.1.5 Ταξινόμηση της οντολογίας

Ο αλγόριθμος αυτός είναι υπεύθυνος για τον υπολογισμό όλων των υπονοούμενων σχέσεων ιεραρχίας μεταξύ των ονοματισμένων κλάσεων της οντολογίας μέσω της εξαντλητικής εφαρμογής των παρακάτω επαγωγικών κανόνων:

1. Αν $A \in T$ τότε $T = T \cup \{A \sqsubseteq A, A \sqsubseteq \top\}$
2. Αν $A \sqsubseteq B \in T$ και $B \sqsubseteq C \in T$, τότε $T = T \cup \{A \sqsubseteq C\}$
3. Αν $A \sqsubseteq B \in T$ και $A \sqsubseteq C \in T$ και $B \sqcap C \sqsubseteq D \in T$, τότε $T = T \cup \{A \sqsubseteq D\}$
4. Αν $A \sqsubseteq B \in T$ και $B \sqsubseteq \exists r.C \in T$, τότε $T = T \cup \{A \sqsubseteq \exists r.C\}$
5. Αν $A \sqsubseteq \exists r.B \in T$ και $r \sqsubseteq s \in T$, τότε $T = T \cup \{A \sqsubseteq \exists s.B\}$
6. Αν $A \sqsubseteq \exists r.B \in T$ και $B \sqsubseteq C \in T$ και $\exists r.C \sqsubseteq D \in T$, τότε $T = T \cup \{A \sqsubseteq D\}$

Αρχικά υπολογίζεται το μεταβατικό κλείσιμο των ρόλων ώστε να μπορέσουν να παραχθούν όλες οι υπονοούμενες σχέσεις μεταξύ τους και στη συνέχεια εφαρμόζεται ο επαγωγικός ρόλος 5. Εφόσον έχει προηγηθεί το μεταβατικό κλείσιμο των ρόλων με τη μία αυτή εφαρμογή παράγονται όλες οι σχέσεις ιεραρχίας που είναι δυνατόν να παραχθούν από αυτόν.

Στη συνέχεια δημιουργείται ένας πίνακας που θα περιέχει όλες τις έννοιες που θέλουμε να εξετάσουμε (πίνακας tmp_queue). Στον πίνακα αυτόν εισάγονται όλες οι ατομικές έννοιες και όλες οι έννοιες που συμμετέχουν σε θετικούς υπαρξιακούς περιορισμούς. Η εξέταση των εννοιών αυτών γίνεται ανά N έννοιες, όπου το N ορίζεται από τον χρήστη. Αποθηκεύονται όλες οι αρχικές σχέσεις A ⊑ B σε έναν πίνακα (tmp_axioms) που θα χρησιμοποιηθεί κατά την εφαρμογή των κανόνων και η διαδικασία από εκεί και πέρα χωρίζεται σε τρία στάδια. Το πρώτο

στάδιο είναι αυτό στο οποίο εξετάζονται οι ατομικές έννοιες, το δεύτερο στάδιο είναι αυτό στο οποίο εξετάζονται οι έννοιες που συμμετέχουν σε θετικούς υπαρξιακούς περιορισμούς και το τρίτο και τελευταίο είναι αυτό στο οποίο εξετάζονται όλες οι έννοιες που εισάχθηκαν για επανεξέταση.

Η διαδικασία που ακολουθείται για το πρώτο στάδιο είναι η εξής:

1. Δημιουργούνται δύο πίνακες που περιέχουν αρχικά ο πρώτος (πίνακας tmp_cl_c) τα αποτελέσματα της εφαρμογής του επαγωγικού κανόνα 1 στις N πρώτες από τις υπό εξέταση έννοιες και ο δεύτερος (πίνακας tmp_ext_pos_part) όλους τους θετικούς υπαρξιακούς περιορισμούς που σχετίζονται με τις N έννοιες και μπορούν να παράξουν νέες σχέσεις μέσω του επαγωγικού κανόνα 6 . Οι πίνακες αυτοί σε συνδυασμό με τον tmp_axioms περιέχουν τώρα πλέον όλες τις σχέσεις υπαγωγής που χρειάζονται για την εφαρμογή των επαγωγικών κανόνων.
2. Εφαρμόζονται εξαντλητικά οι κανόνες 2 και 4 (οι δύο αυτοί κανόνες εφαρμόζονται ουσιαστικά ταυτόχρονα αφού οι σχέσεις υπαγωγής είναι αποθηκευμένες μέσω των αναγνωριστικών της υποκλάσης και της υπερκλάσης και δεν μας ενδιαφέρει αν η υπερκλάση είναι υπαρξιακός περιορισμός ή όχι) και στη συνέχεια μία φορά ο κανόνας 3. Τα αποτελέσματα αποθηκεύονται (πίνακας tmp_cl_c) και αν ο κανόνας 3 παρήγαγε νέα αποτελέσματα τότε η διαδικασία επαναλαμβάνει το βήμα 2. Όταν οι επαναλήψεις σταματήσουν οι νέες εγγραφές που παράχθηκαν αποθηκεύονται στον πίνακα τελικών αποτελεσμάτων (cl_class).
3. Τέλος εφαρμόζεται ο επαγωγικός κανόνας 6 (τα αποτελέσματα αποθηκεύονται στον tmp_axioms) και η διαδικασία ξεκινά από την αρχή έως ότου επεξεργαστούν όλες οι ατομικές έννοιες από τον πίνακα των υπό εξέταση εννοιών.

Στα στάδιο 2 και 3 στο αντίστοιχο βήμα 1, στην αρχικοποίηση του πίνακα tmp_cl_c προστίθενται επιπλέον και όλες οι σχέσεις υπαγωγής που παράχθηκαν σε προηγούμενα στάδια και συνδέονται με κάποια από τις N υπό εξέταση έννοιες. Επιπλέον στα στάδια αυτά στο βήμα 3 ελέγχεται αν από τις σχέσεις που παράχθηκαν μετά την εφαρμογή του επαγωγικού κανόνα 6 προκύπτει ότι κάποια έννοια χρειάζεται επανεξέταση. Οι έννοιες αυτές εισάγονται στο τέλος του πίνακα tmp_queue. Στο στάδιο 1 επειδή οι έννοιες που επεξεργάζονται είναι ατομικές δεν προκύπτουν έννοιες για επανεξέταση.

Μετά το τέλος της διαδικασίας ταξινόμησης εφαρμόζεται ένα SQL ερώτημα με το οποίο δημιουργείται η ταξονομία ονοματισμένων OWL κλάσεων εννοιών από τον πίνακα cl_class και αποθηκεύεται στον πίνακα cl_class_reduct.

Ακολουθεί η περιγραφή του αλγορίθμου σε ψευδοκώδικα (ClassificationFinal.java).

Αλγόριθμος

Classification():

Μεταβλητή s := 5000 που καθορίζει πόσες κλάσεις θα εξετάζονται από τον πίνακα tmp_queue σε κάθε επανάληψη;
Μεταβλητή num_Process := 0 που χρησιμοποιείται για να σταματήσει την επαναληπτική εκτέλεση των διαδικασιών;
Αποστολή SQL ερωτήματος με το οποίο εισάγονται στον πίνακα tmp_axioms όλες οι σχέσεις A ⊑ B που έχουν αποθηκευτεί από την κανονικοποίηση της οντολογίας;
Κλήση της μεθόδου Role_CR1():
{
Λογική μεταβλητή flag := true;
Επανέλαβε
{
flag := false;
Αποστολή SQL ερωτήματος με το οποίο εφαρμόζεται ο επαγωγικός κανόνας 2 για τους ρόλους (μεταβατικό κλείσιμο των ρόλων);
Αν το query επέστρεψε αποτέλεσμα **τότε**
{
flag := true;
}
}**όσο ισχύει** flag := true;
Αποστολή SQL ερωτήματος με το οποίο εισάγονται στον πίνακα που περιέχει τις υπαγωγές ρόλων όλοι οι ρόλοι ως υποσύνολα του εαυτού τους;
Αποστολή SQL ερωτήματος με το οποίο εισάγονται στον πίνακα που περιέχει τους αρνητικούς υπαρξιακούς περιορισμούς ∃r.A ⊑ B, οι σχέσεις ∃s.A ⊑ B για κάθε s ⊑ r (επαγωγικός κανόνας 5);
}
Αποστολή SQL ερωτήματος με το οποίο εισάγονται στον πίνακα tmp_queue όλες οι ατομικές έννοιες;
Μεταβλητή count1 := ο αριθμός των ατομικών αυτών εννοιών;
Μεταβλητή num1 := (count1 / s) +1 πόσες φορές θα εκτελεστεί η μέθοδος Process_Stage1;
Αποστολή SQL ερωτήματος με το οποίο εισάγονται στον πίνακα tmp_queue όλες οι έννοιες που συμμετέχουν ως υπερκλάσεις στον πίνακα θετικών υπαρξιακών περιορισμών;
Μεταβλητή count2 := ο αριθμός των εννοιών αυτών;
Μεταβλητή num2 := (count2 / s)+count1 πόσες φορές θα εκτελεστεί η μέθοδος Process_Stage2;
Κλήση της μεθόδου Process_Stage1(num1):
{
Λογική μεταβλητή ok := true;
num_Process := 0;
Μεταβλητή num_l := 0;

Όσο ok := true
{
num_Process++;
Μεταβλητή count := ο αριθμός των εγγραφών στον tmp_queue;
Αν count := 0 ***τότε***
{
ok := false;
}
Διαφορετικά
{
Κλήση της reload_part_Stage1(s):
{
Αποστολή SQL ερωτήματος για το άδειασμα των πινάκων tmp_part, tmp_cl_c, tmp_ext_pos_part;
Αποστολή SQL ερωτήματος με το οποίο εισάγονται στον πίνακα tmp_part οι s πρώτες εγγραφές από τον tmp_queue;
Αποστολή SQL ερωτήματος με το οποίο διαγράφονται οι s αυτές εγγραφές από τον tmp_queue;
Αποστολή SQL ερωτήματος με το οποίο για κάθε εγγραφή X του tmp_part εισάγεται στον πίνακα tmp_cl_c η εγγραφή X,X,1 (επαγωγικός κανόνας 1);
Αποστολή SQL ερωτήματος για την εισαγωγή στον πίνακα tmp_ext_pos_part των εγγραφών από τον πίνακα των θετικών υπαρξιακών περιορισμών που σχετίζονται με τις εγγραφές του tmp_part;
}
Κλήση της compute_classify_tmp_Stage1():
{
Λογική μεταβλητή ok := true;
Λογική μεταβλητή rule1 := true που χρησιμοποιείται για να δείχνει αν παράχθηκαν νέες εγγραφές από τους επαγωγικούς κανόνες 2 και 4;
Λογική μεταβλητή rule2 := true που χρησιμοποιείται για να δείχνει αν παράχθηκαν νέες εγγραφές από τον επαγωγικό κανόνα 3;
Μεταβλητή len := 1 το βήμα που βρίσκεται η μέθοδος;
Μεταβλητή last_inter := 0 το τελευταίο βήμα που είχαμε εφαρμογή των επαγωγικών κανόνων 2 και 4;

Όσο ok := true
{
Όσο ok := true
{
rule1 := **Κλήση της compute_imply(“tmp_axioms”, “tmp_cl_c”, len):**
{
Αποστολή SQL ερωτήματος για την εκτέλεση των επαγωγικών κανόνων 2,4 και εισαγωγή των αποτελεσμάτων στον tmp_cl_c με βήμα len+1;
Αν το query είχε αποτελέσματα **τότε**
{
Επέστρεψε true;
}
Επέστρεψε false;
}
Αν rule1 := true **τότε**
{
len++;
}
Διαφορετικά
{
ok := false;
}
}
rule2 := **Κλήση της μεθόδου compute_inter(len, last_inter):**
{
Αποστολή SQL ερωτήματος για την εκτέλεση του επαγωγικού κανόνα 3 και εισαγωγή των αποτελεσμάτων στον tmp_cl_c με βήμα len+1;
Αν δεν προστέθηκαν εγγραφές **τότε**
{
Επέστρεψε false;
}
Επέστρεψε true;
}
last_inter := len;

```
                                    Αν rule2 := true τότε
                                    {
                                        ok := true;
                                        len++;
                                    }
                                }
                                Αποστολή SQL ερωτήματος για την εισαγωγή στον
                                πίνακα cl_class όλων των νέων εγγραφών που
                                παράχθηκαν από την εφαρμογή των επαγωγικών
                                κανόνων 2,3,4;
                            }
                            num_l ++;
                            Αν num_l > 4 τότε
                            {
                                Αποστολή SQL ερωτήματος για την ανάλυση του
                                πίνακα tmp_axioms;
                                num_l := 0;
                            }
                            Κλήση της compute_exist_Stage1():
                            {
                                Αποστολή SQL ερωτήματος για το άδειασμα του
                                πίνακα tmp_ext_cl;
                                Αποστολή SQL ερωτήματος για την εφαρμογή του
                                επαγωγικού κανόνα 6 και πρόσθεση των
                                αποτελεσμάτων στον tmp_ext_cl;
                                Αποστολή SQL ερωτήματος για την πρόσθεση των
                                εγγραφών του tmp_ext_cl στον tmp_axioms;
                            }
                        }
                        Αν num_Process := num1 τότε
                        {
                            ok := false;
                        }
                    }
                }
Κλήση της μεθόδου Process_Stage2(num2):
{
    Αποστολή SQL ερωτημάτων για την δημιουργία indexes στον πίνακα
    cl_class και την ανάλυσή του;
    Λογική μεταβλητή ok := true;
    Μεταβλητή num_l := 0;
    Όσο ok := true
    {
        num_Process++;
```

Αν ο πίνακας tmp_queue είναι κενός **τότε**
{
ok := false;
}
Διαφορετικά
{
Κλήση της reload_part_Stage23(s):
{
Αποστολή SQL ερωτήματος για το άδειασμα των πινάκων tmp_part, tmp_cl_c, tmp_ext_pos_part;
Αποστολή SQL ερωτήματος με το οποίο εισάγονται στον πίνακα tmp_part οι s πρώτες εγγραφές από τον tmp_queue;
Αποστολή SQL ερωτήματος με το οποίο διαγράφονται οι s αυτές εγγραφές από τον tmp_queue;
Αποστολή SQL ερωτήματος με το οποίο εισάγεται στον tmp_cl_c η εγγραφή X,Y,1 για κάθε εγγραφή X του tmp_part και X,Y του cl_class ή του cl_class2;
Αποστολή SQL ερωτήματος με το οποίο για κάθε εγγραφή X του tmp_part εισάγεται στον πίνακα tmp_cl_c η εγγραφή X,X,1 (επαγωγικός κανόνας 1);;
Αποστολή SQL ερωτήματος για την εισαγωγή στον πίνακα tmp_ext_pos_part των εγγραφών από τον πίνακα των θετικών υπαρξιακών περιορισμών που σχετίζονται με τις εγγραφές του tmp_part;
}
Κλήση της compute_classify_tmp_Stage2():
{
Λογική μεταβλητή ok := true;
Λογική μεταβλητή rule1 := true που χρησιμοποιείται για να δείχνει αν παράχθηκαν νέες εγγραφές από τους επαγωγικούς κανόνες 2 και4;
Λογική μεταβλητή rule2 := true που χρησιμοποιείται για να δείχνει αν παράχθηκαν νέες εγγραφές από τον επαγωγικό κανόνα 3;
Μεταβλητή len := 1 το βήμα που βρίσκεται η μέθοδος;
Μεταβλητή last_inter := 0 το τελευταίο βήμα που είχαμε εφαρμογή των επαγωγικών κανόνων 2 και 4;
Όσο ok := true

```
{
    Όσο ok := true
    {
        rule1 := Κλήση της compute_imply("tmp_axioms", "tmp_cl_c", len):
        {
            Αποστολή SQL ερωτήματος για την εκτέλεση των επαγωγικών κανόνων 2,4 και εισαγωγή των αποτελεσμάτων στον tmp_cl_c με βήμα len+1;
            Αν το query είχε αποτελέσματα τότε
            {
                Επέστρεψε true;
            }
            Επέστρεψε false;
        }
        Αν rule1 := true τότε
        {
            len++;
        }
        Διαφορετικά
        {
            ok := false;
        }
    }
    rule2 := Κλήση της μεθόδου compute_inter(len, last_inter):
    {
        Αποστολή SQL ερωτήματος για την εκτέλεση του επαγωγικού κανόνα 3 και εισαγωγή των αποτελεσμάτων στον tmp_cl_c με βήμα len+1;
        Αν δεν προστέθηκαν εγγραφές τότε
        {
            Επέστρεψε false;
        }
        Επέστρεψε true;
    }
    last_inter := len;
    Αν rule2 := true τότε
```

```
                                {
                                        ok := true;
                                        len++;
                                }
                        }
                        Αποστολή SQL ερωτήματος για την εισαγωγή στον
                        πίνακα cl_class2 όλων των νέων εγγραφών που
                        παράχθηκαν από την εφαρμογή των επαγωγικών
                        κανόνων 2,3,4;
                }
                num_l ++;
                Αν num_l > 2 τότε
                {
                        Αποστολή SQL ερωτήματος για την ανάλυση των
                        πινάκων tmp_axioms και cl_class2;
                        num_l := 0;
                }
                Κλήση της compute_exist_Stage23():
                {
                        Αποστολή SQL ερωτήματος για το άδειασμα του
                        πίνακα tmp_ext_cl;
                        Αποστολή SQL ερωτήματος για την εφαρμογή του
                        επαγωγικού κανόνα 6 και πρόσθεση των
                        αποτελεσμάτων στον tmp_ext_cl;
                        Αποστολή SQL ερωτήματος για την διαγραφή
                        όσων από αυτές τις εγγραφές υπάρχουν ήδη σε
                        κάποιον από τους πίνακες tmp_axioms, cl_class ή
                        cl_class2;
                        Αποστολή SQL ερωτήματος για την εγγραφή στον
                        tmp_queue των εννοιών που χρειάζονται
                        επανεξέταση;
                        Αποστολή SQL ερωτήματος για την πρόσθεση των
                        εγγραφών του tmp_ext_cl στον tmp_axioms;
                }
        }
        Αν num_Process := num2 τότε
        {
                ok := false;
        }
    }
}
Κλήση της μεθόδου Process_Stage3():
{
```

Αποστολή SQL ερωτημάτων για την εισαγωγή όλων των εγγραφών του cl_class2 στον cl_class, άδειασμα του cl_class2 και ανάλυση και των δύο πινάκων;
Λογική μεταβλητή ok := true;
Όσο ok := true
{
Αν ο πίνακας tmp_queue είναι κενός **τότε**
{
ok := false;
}
Διαφορετικά
{
Κλήση της reload_part_Stage23(s):
{
Αποστολή SQL ερωτήματος για το άδειασμα των πινάκων tmp_part, tmp_cl_c, tmp_ext_pos_part;
Αποστολή SQL ερωτήματος με το οποίο εισάγονται στον πίνακα tmp_part οι s πρώτες εγγραφές από τον tmp_queue;
Αποστολή SQL ερωτήματος με το οποίο διαγράφονται οι s αυτές εγγραφές από τον tmp_queue;
Αποστολή SQL ερωτήματος με το οποίο εισάγεται στον tmp_cl_c η εγγραφή X,Y,1 για κάθε εγγραφή X του tmp_part και X,Y του cl_class ή του cl_class2;
Αποστολή SQL ερωτήματος με το οποίο για κάθε εγγραφή X του tmp_part εισάγεται στον πίνακα tmp_cl_c η εγγραφή X,X,1 (επαγωγικός κανόνας 1);;
Αποστολή SQL ερωτήματος για την εισαγωγή στον πίνακα tmp_ext_pos_part των εγγραφών από τον πίνακα των θετικών υπαρξιακών περιορισμών που σχετίζονται με τις εγγραφές του tmp_part;
}
Κλήση της compute_classify_tmp_Stage3():
{
Λογική μεταβλητή ok := true;
Λογική μεταβλητή rule1 := true που χρησιμοποιείται για να δείχνει αν παράχθηκαν νέες εγγραφές από τους επαγωγικούς κανόνες 2 και 4;
Λογική μεταβλητή rule2 := true που χρησιμοποιείται για να δείχνει αν παράχθηκαν νέες εγγραφές από τον επαγωγικό κανόνα 3;

Μεταβλητή len := 1 το βήμα που βρίσκεται η μέθοδος;
Μεταβλητή last_inter := 0 το τελευταίο βήμα που είχαμε εφαρμογή των επαγωγικών κανόνων 2 και 4;
Όσο ok := true
{
Όσο ok := true
{
rule1 := **Κλήση της compute_imply(“tmp_axioms”, “tmp_cl_c”, len):**
{
Αποστολή SQL ερωτήματος για την εκτέλεση των επαγωγικών κανόνων 2,4 και εισαγωγή των αποτελεσμάτων στον tmp_cl_c με βήμα len+1;
Αν το query είχε αποτελέσματα **τότε**
{
Επέστρεψε true;
}
Επέστρεψε false;
}
Αν rule1 := true **τότε**
{
len++;
}
Διαφορετικά
{
ok := false;
}
}
rule2 := **Κλήση της μεθόδου compute_inter(len, last_inter):**
{
Αποστολή SQL ερωτήματος για την εκτέλεση του επαγωγικού κανόνα 3 και εισαγωγή των αποτελεσμάτων στον tmp_cl_c με βήμα len+1;
Αν δεν προστέθηκαν εγγραφές **τότε**
{

Επέστρεψε false;
}
Επέστρεψε true;
}
last_inter := len;
Αν rule2 := true **τότε**
{
ok := true;
len++;
}
}
Αποστολή SQL ερωτήματος για την εισαγωγή στον πίνακα cl_class2 όλων των νέων εγγραφών που παράχθηκαν από την εφαρμογή των επαγωγικών κανόνων 2,3,4;
}
Αποστολή SQL ερωτήματος για την ανάλυση του πίνακα cl_class2;
Κλήση της compute_exist_Stage23():
{
Αποστολή SQL ερωτήματος για το άδειασμα του πίνακα tmp_ext_cl;
Αποστολή SQL ερωτήματος για την εφαρμογή του επαγωγικού κανόνα 6 και πρόσθεση των αποτελεσμάτων στον tmp_ext_cl;
Αποστολή SQL ερωτήματος για την διαγραφή όσων από αυτές τις εγγραφές υπάρχουν ήδη σε κάποιον από τους πίνακες tmp_axioms, cl_class ή cl_class2;
Αποστολή SQL ερωτήματος για την εγγραφή στον tmp_queue των εννοιών που χρειάζονται επανεξέταση;
Αποστολή SQL ερωτήματος για την πρόσθεση των εγγραφών του tmp_ext_cl στον tmp_axioms;
}
}
}
}

6.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Το σύστημά μας έχει δοκιμαστεί και τρέχει κανονικά σε Windows 7 και Windows Vista, μπορεί όμως να τρέξει σε οποιοδήποτε εμπορικό λειτουργικό σύστημα αφού είναι γραμμένο σε Java και χρησιμοποιεί το DBMS PostgreSQL, διανομές των οποίων κυκλοφορούν για τα περισσότερα εμπορικά συστήματα.

Συγκεκριμένα το σύστημά μας υλοποιήθηκε με το Java SDK 1.6 και η έκδοση της PostgreSQL που χρησιμοποιήθηκε ήταν η PostgreSQL Server 8.4, ενώ έχει δοκιμαστεί και η έκδοση 9.0.

Για την ανάπτυξη και διαχείριση της βάσης δεδομένων χρησιμοποιήθηκε ο pgAdmin III Version 1.10.5, ενώ για την σύνδεση με τον PostgreSQL Server χρησιμοποιήθηκε ο οδηγός JDBC 8.4-701.jdbc3 (για την έκδοση 8.4 του server) και ο οδηγός JDBC 8.4-701.jdbc4 (για την έκδοση 9.0 του server).

Η συγγραφή του κώδικα έγινε με την χρήση του αναπτυξιακού περιβάλλοντος Eclipse Version 3.3.1.1 και η επεξεργασία των οντολογιών μέσω του Protégé Ontology Editor v.4 Alpha.

Τα παραπάνω προγράμματα είναι όλα ανοικτού κώδικα και υπόκεινται στο GPL.

6.3 Εγκατάσταση συστήματος

Αρχικά χρειάζεται η εγκατάσταση των παρακάτω προγραμμάτων:

- PostgreSQL Server 8.4 ή ανώτερος
- Java SDK 1.6 ή ανώτερο
- Eclipse 3.3.x

Είναι απαραίτητο επίσης να κατεβάσετε τον οδηγό JDBC που αντιστοιχεί στην έκδοση του PostgreSQL Server που εγκαταστήσατε, καθώς και το owlapi-distribution-3.4.5.jar[38]. Επίσης οι οντολογίες που φορτώνονται στο σύστημα μας χρειάζεται να μετατραπούν σε OWL_functional_syntax. Για την διαδικασία αυτή εμείς χρησιμοποιήσαμε και προτείνουμε την χρήση του Protégé Ontology Editor.

Το σύστημά μας δίνεται μαζί με το σύστημα DBRS το οποίο επεκτείνει. Ακολουθούν οι οδηγίες για την από κοινού εγκατάσταση και των δύο συστημάτων.

Διαδικασία εγκατάστασης:

i. Ανοίξτε το Eclipse και πηγαίνετε File → New → Other → Java Project from Existing Ant Buildfile. Επιλέξτε Browse και πηγαίνετε στον φάκελο που έχετε αποθηκεύσει το DBRS_ext. Επιλέξτε DBRS_ext → DBRS → build.xml. Ονομάστε το project σας και πατήστε OK.

[38] http://sourceforge.net/projects/owlapi/files/OWL%20API%20%28for%20OWL%202.0%29/3.4.5/

ii. Πηγαίνετε “Project name” → Properties → Java Build Path → Libraries και πατήστε Add External JARs. Επιλέξτε τα JAR αρχεία του JDBC Driver και του owlapi.v.3 που έχετε κατεβάσει και κάντε τα add.

iii. Πηγαίνετε στην κλάση Classify_ELH.java στο package ntua.DBRS_ELH και ορίστε τις τιμές των παρακάτω μεταβλητών:

- **static** String *main_database* = το όνομα που θέλουμε να δώσουμε στην βάση.
- **static** String *main_username* = το όνομα του χρήστη στον οποίο ανήκει η βάση.
- **static** String *main_password* = ο κωδικός του χρήστη.
- **static final** String *main_adminUserName* = το όνομα του admin της database. Συνήθως είναι “postgres”.
- **static final** String *main_adminPassword* = ο κωδικός του admin της database.
- **static** String *main_Ontology_File_Input* = το path του αρχείου της οντολογίας που θέλετε να φορτώσετε.
- **static** String *main_Path_Dir_Temp* = το path στο οποίο θα αποθηκεύονται τα προσωρινά αρχεία (πχ. `="C:/Program Files/PostgreSQL/8.4/bin/"`). Τα αρχεία αυτά πρέπει να σβήνονται πριν κάθε φόρτωμα νέας οντολογίας.
- **static final** String *main_dbTablesScriptPath* = το path του CreateDBTables.sql

iv. Κάντε δεξί κλικ πάνω στην κλάση Classify_ELH.java και πατήστε Run As-> Open Run Dialog… επιλέξετε την καρτέλα Arguments και προσθέστε στα Program Arguments: -Xmx*NUMBER*m , όπου στη θέση του *NUMBER* γράφετε το μέγιστο μέγεθος μνήμης σε MB που θα επιτραπεί να δεσμεύσει ο Java Heap (Προτεινόμενη τιμή 1500).

v. Κάντε δεξί κλικ πάνω στην κλάση Classify_ELH.java και πατήστε Run As → Java Application.

Από το iii μέχρι το v καλύπτονται τα βήματα για την λειτουργία του συστήματος που δημιουργήσαμε εμείς ως επέκταση του DBRS. Για να τρέξετε το DBRS πρέπει να εκτελέσετε και τα ακόλουθα βήματα.

vi. Πηγαίνετε στην κλάση ntua.DBRS.SystemInitializer.java και ορίστε για τις πρώτες τρεις final String μεταβλητές τα paths των αντίστοιχων αρχείων και για τις επόμενες δύο το όνομα και τον κωδικό του διαχειριστή του DBMS αντίστοιχα.

vii. Πηγαίνετε στην κλάση ntua.DBRS.SystemUtils.java και ορίστε την final String μεταβλητή *filesPath* ομοίως με παραπάνω.

viii. Κάντε δεξί κλικ πάνω στην κλάση ntua.DBRS.SystemGUI και πατήστε Run As → Open Run Dialog… επιλέξετε την καρτέλα Arguments και προσθέστε στα Program Arguments: -Xmx*NUMBER*m , όπου στη θέση του *NUMBER* γράφετε το μέγιστο μέγεθος μνήμης σε MB που θα επιτραπεί να δεσμεύσει ο Java Heap (Προτεινόμενη τιμή >1500).

ix. Κάντε δεξί κλικ πάνω στην κλάση ntua.DBRS.SystemGUI και πατήστε Run As → Java Application.

6.4 Λεπτομέρειες Υλοποίησης

Στην ενότητα αυτήν περιγράφονται τα ευρετήρια που χρησιμοποιήθηκαν στην βάση μας και οι java κλάσεις της.

6.4.1 Ευρετήρια της βάσης δεδομένων

Παρακάτω αναφέρονται τα ευρετήρια (indexes) που επιλέχθηκαν για κάθε πίνακα της βάσης δεδομένων. Οι επιλογές των ευρετηρίων έτσι ώστε να εκτελούνται όσο πιο γρήγορα γίνεται τα SQL ερωτήματα που γίνονται στη βάση δεδομένων κατά την κανονικοποίηση και την ταξινόμηση της οντολογίας, εκμεταλλευόμενοι την ποικιλία των ευρετηρίων που προσφέρει η PostgreSQL.

- Πίνακας tmp_axioms
  Τα ερωτήματα που γίνονται εδώ είναι ταυτοποίησης είτε βάσει του πεδίου sub, είτε βάση του πεδίου sub και του πεδίου sup μαζί. Ορίστηκε επομένως ένα ευρετήριο btree και για τα δύο πεδία της σχέσης μαζί.

- Πίνακας tmp_clc_c
  Εδώ συναντάμε ερωτήματα ταυτοποίησης για τα πεδία sub και sup μαζί, ερωτήματα ταυτοποίησης για τα πεδία sup και length μαζί και ερωτήματα σύγκρισης για το πεδίο length. Χρησιμοποιούνται τρία btree ευρετήρια, ένα για το πεδίο length, ένα για το πεδίο sub και ένα για τα πεδία sub και sup μαζί.

- Πίνακας tmp_ext_neg
  Εδώ γίνονται ερωτήματα ταυτοποίησης και σύγκρισης στο πεδίο sub_o, ερωτήματα ταυτοποίησης στα πεδία sub_o και sub_c και ερωτήματα σύγκρισης στο πεδίο sup. Έχει οριστεί ένα ευρετήριο btree και για τα τρία πεδία μαζί.

- Πίνακας tmp_ext_pos
  Συναντάμε ερωτήματα ταυτοποίησης για το πεδίο sup_c. Έχουν οριστεί δύο ευρετήρια btree, ένα για το πεδίο sub και ένα για τα πεδία sup_c και sup_o μαζί.

- Πίνακας tmp_ext_pos_part
  Συναντάμε ερωτήματα σύγκρισης για το πεδίο sub και ερωτήματα ταυτοποίησης για τα πεδία sup_o και sup_c μαζί, οπότε χρησιμοποιούμε ένα ευρετήριο btree για το πεδίο sub και ένα ευρετήριο btree για τα πεδία sup_c και sup_o μαζί.

- Πίνακας tmp_int
  Τα ερωτήματα εδώ είναι ταυτοποίησης για τα πεδία id1 και id2 μαζί. Χρησιμοποιούνται δύο ευρετήρια btree, ένα για τα πεδία id1 και id2 μαζί και ένα για το πεδίο id2.

- Πίνακας tmp_queue
  Εδώ συναντάμε ερωτήματα ταυτοποίησης για το πεδίο ord. Χρησιμοποιείται ένα ευρετήριο btree για το πεδίο id.

6.4.2 Περιγραφή Java κλάσεων

Στην ενότητα αυτή περιγράφονται συνοπτικά οι Java κλάσεις του κώδικα (source)του συστήματός μας. Για κάθε κλάση παρουσιάζονται τα σημαντικότερα μέλη της και οι μέθοδοί της.

6.4.2.1 Η κλάση DBWrapper

Μέλη της κλάσης

- `Connection db`
  Η σύνδεση με τη βάση δεδομένων

- `Statement sql`
  Μια δήλωση SQL για την υποβολή ερωτημάτων

- `DatabaseMetaData dbmd`
  Τα στοιχεία της βάσης δεδομένων με την οποία έχει πραγματοποιηθεί σύνδεση

- `boolean dbExistence`
  Παίρνει την τιμή true μόνο αν πραγματοποιηθεί επιτυχής σύνδεση με τη βάση

- `boolean dbUserExistence`
  Παίρνει την τιμή true μόνο αν πραγματοποιηθεί επιτυχής σύνδεση με τη βάση για τον δοσμένο όνομα χρήστη

Μέθοδοι της κλάσης

- `public DBWrapper(String database, String username, String password)`
  Ο κατασκευαστής της κλάσης. Πραγματοποιεί τη σύνδεση με τη βάση δεδομένων βάσει των στοιχείων που του δίνονται ως ορίσματα και αρχικοποιεί τα μέλη της κλάσης.

- `public DBWrapper(String userName, String userPassword)`
  Κατασκευαστής της κλάσης. Χρησιμοποιείται μόνο για τον έλεγχο ύπαρξης βάσης δεδομένων και χρήστη.
- `public void loadScript(String scriptFilePath, String userName)`
  Δέχεται ως όρισμα την τοποθεσία ενός script αρχείου κατασκευής σχήματος βάσης δεδομένων (ή ευρετηρίων) και το όνομα του χρήστη στον οποίο θα ανήκει αυτή και το εκτελεί.

- `public int RunQuery(String sqlText)`
  Εκτελεί το query που δέχεται ως όρισμα.

- `public int RunQueryDB(String sqlText)`
  Εκτελεί το query που δέχεται ως όρισμα και επιστρέφει τον αριθμό των αποτελεσμάτων.

- **public void** closeConnection()
  Κλείνει την σύνδεση με τη βάση.

- **public boolean** dbExistenceChecker()
  Επιστρέφει την τιμή του μέλους της κλάσης dbExistence.

- **public boolean** isDBEmpty()
  Επιστρέφει true αν η βάση δεδομένων με την οποία έχει πραγματοποιηθεί σύνδεση περιέχει δεδομένα.

- **public void** createDatabase(String databaseName, String userName)
  Δέχεται ως ορίσματα το όνομα μιας νέας βάσης δεδομένων και το όνομα του ιδιοκτήτη της και την κατασκευάζει.

- **public void** createDBUser(String userName, String userPassword
  Δέχεται ως ορίσματα το όνομα ενός νέου χρήστη και τον κωδικό του και τον εγγράφει στο DBMS.

- **public boolean** dbUserExistenceChecker()
  Επιστρέφει την τιμή του μέλους της κλάσης dbUserExistence.

- **public void** copy_into_table (String table,String file)
  Δέχεται ως είσοδο το όνομα ενός πίνακα και το path ενός αρχείου και αντιγράφει τα περιεχόμενα του αρχείου στον πίνακα.

6.4.2.2 Η κλάση SystemInitializer

Μέλη της κλάσης

- **static final** String dbTablesScriptPath
  Εδώ αποθηκεύεται η διαδρομή (path) του script αρχείου κατασκευής του σχήματος της βάσης δεδομένων του DBRS.

- **static final** String adminUsername
  Εδώ αποθηκεύεται το όνομα του διαχειριστή του DBMS.

- **static final** String adminPassword
  Εδώ αποθηκεύεται ο κωδικός του διαχειριστή του DBMS.

- **public static** OWLOntologyManager *manager*
  Αντικείμενο της κλάσης OWLOntologyManager, απαραίτητο για την κατασκευή του μοντέλου της οντολογίας στην κύρια μνήμη.

- **public static** OWLDataFactory *factory*
  Αντικείμενο της κλάσης OWLDataFactory. Χρησιμοποιείται για την σύνθεση *OWL* εκφράσεων σε μια τομή.

- `FillTables fillax_c_incl`
  Αντικείμενο της κλάσης FillTables για την δημιουργία του προσωρινού αρχείου που γεμίζει τον πίνακα ax_class.

- `FillTables fillax_r_incl`
  Αντικείμενο της κλάσης FillTables για την δημιουργία του προσωρινού αρχείου που γεμίζει τον πίνακα ax_object.

- `FillTables fillax_conj`
  Αντικείμενο της κλάσης FillTables για την δημιουργία του προσωρινού αρχείου που γεμίζει τον πίνακα tmp_int.

- `FillTables fillax_exis_neg`
  Αντικείμενο της κλάσης FillTables για την δημιουργία του προσωρινού αρχείου που γεμίζει τον πίνακα tmp_ext_neg.

- `FillTables fillax_exis_pos`
  Αντικείμενο της κλάσης FillTables για την δημιουργία του προσωρινού αρχείου που γεμίζει τον πίνακα tmp_ext_pos.

- `DBWrapper dbHandler`
  Αντικείμενο της κλάσης DBWrapper για τη σύνδεση με τη βάση δεδομένων.

- **`private`** `String userName`
  Το όνομα του χρήστη.

- **`private`** `String userPassword`
  Ο κωδικός του χρήστη.

- **`private`** `String dbName`
  Το όνομα της βάσης δεδομένων.

Μέθοδοι της κλάσης

- **`public`** `SystemInitializer(String user_Name,String user_Password, String db_Name)`
  Δημιουργία νέου αντικειμένου κλάσης SystemInitializer για την αρχικοποίηση του συστήματος και τη δημιουργία (όταν απαιτείται) και φόρτωση της βάσης δεδομένων. Δέχεται ως ορίσματα το όνομα και τον κωδικό του χρήστη, καθώς και το όνομα της βάσης.

- **`private void`** `createDataBase(DBWrapper checkDBHandler)`
  Δημιουργεί, αν δεν υπάρχει ήδη, ένα νέο ρόλο στο DBMS με τα στοιχεία του χρήστη. Στη συνέχεια, κατασκευάζει τη βάση δεδομένων εκτελώντας το αντίστοιχο script αρχείο.

- **`public`** `DBWrapper getDBWrapper()`
  Επιστρέφει το αντικείμενο της κλάσης `dbHandler` DBWrapper.

6.4.2.3 Η κλάση CheckSet

Μέλη της κλάσης

- `static String Path_Dir_Temp`
  Το path που αποθηκεύονται τα προσωρινά αρχεία.

- `public HashMap<Integer,String> data`
  Χρησιμοποιείται για την αποθήκευση των μοναδικών αναγνωριστικών.

- `private BufferedWriter out`
  Χρησιμοποιείται για την εκτύπωση των δεδομένων του HashMap στο αρχείο.

Μέθοδοι της κλάσης

- `public CheckSet()`
  Ο κατασκευαστής της κλάσης.

- `public int addEntry(int code,String name)`
  Δέχεται ως ορίσματα ένα αναγνωριστικό και ένα String. Ελέγχει αν το αναγνωριστικό είναι μοναδικό, αν όχι αποδίδει ένα καινούργιο, αποθηκεύει το ζευγάρι μοναδικό αναγνωριστικό-String και επιστρέφει το μοναδικό αναγνωριστικό.

- `public void createFile()`
  Εκτυπώνει τα δεδομένα του HashMap σε ένα αρχείο.

6.4.2.4 Η κλάση FillTables

Μέλη της κλάσης

- `static String Path_Dir_Temp`
  Το path που αποθηκεύονται τα προσωρινά αρχεία.

- `public Set<String> data`
  Χρησιμοποιείται για την αποθήκευση των δεδομένων.

- `private BufferedWriter out`
  Χρησιμοποιείται για την εκτύπωση των δεδομένων του Set στο αρχείο.

Μέθοδοι της κλάσης

- `public FillTables(String table)`
  Ο κατασκευαστής της κλάσης. Ανάλογα με το όνομα του πίνακα που δέχεται σαν όρισμα δημιουργεί και διαφορετικό Filewriter που θα χρησιμοποιηθεί για την εκτύπωση των δεδομένων του Set σε ένα αρχείο.

- **public void** addSet(String new_data)
  Προσθέτει μια νέα εγγραφή στο Set.

- **public void** createFile()
  Εκτυπώνει τα δεδομένα του Set σε ένα αρχείο.

6.4.2.5 Η κλάση MapHash

Μέλη της κλάσης

- **static** String Path_Dir_Temp
  Το path που αποθηκεύονται τα προσωρινά αρχεία.

- HashMap<Integer,Integer> map
  Χρησιμοποιείται για την αποθήκευση των ισοδύναμων εννοιών. Στη δομή αυτή κάθε έννοια δείχνει σε μία ισοδύναμή της.

- HashMap<Integer,Integer> final_map
  Χρησιμοποιείται για την αποθήκευση των εννοιών που χρειάζονται να αντικατασταθούν στην βάση με μία ισοδύναμή τους.

- **private** BufferedWriter out
  Χρησιμοποιείται για την εκτύπωση των δεδομένων του final_map σε αρχείο.

Μέθοδοι της κλάσης

- **public** MapHash(String table)
  Ο κατασκευαστής της κλάσης.

- **public void** addEquiv(**int** hash1,**int** hash2)
  Δέχεται ως είσοδο δύο αναγνωριστικά ισοδύναμων εννοιών και τα προσθέτει στο map. Εξασφαλίζει ότι όλες οι ισοδύναμες μεταξύ τους έννοιες συνδέονται μέσω μιας «αλυσίδας» εγγραφών.

- **public void** finalHash()
  Επεξεργάζεται τα δεδομένα του map και αποθηκεύει στο final_map όλες τις έννοιες εκείνες που χρειάζονται να αντικατασταθούν στην βάση με μία ισοδύναμή τους. Εξασφαλίζει ότι όλες οι μεταξύ τους ισοδύναμες έννοιες θα αντικατασταθούν από την ίδια έννοια.

- **public void** createFile()
  Εκτυπώνει τα δεδομένα του final_map σε ένα αρχείο.

6.4.2.6 Η κλάση HashNormal

Μέθοδοι της κλάσης

- **public static void** HashInter(OWLObjectIntersectionOf inter, **boolean** neg, SystemInitializer system,CheckSet checkset)
  Διαχειρίζεται τις τομές από *OWL* εκφράσεις κατά την κανονικοποίηση της οντολογίας. Δέχεται ως εισόδους:
  - `OWLObjectIntersectionOf inter` : η τομή από *OWL* εκφράσεις.
  - **boolean** `neg` : Δηλώνει το αν η τομή συμμετείχε αρνητικά ή θετικά στο αξίωμα.
  - `SystemInitializer system` : Χρησιμοποιείται για την αποθήκευση των νέων σχέσεων που εξάγονται στα αντίστοιχα Set (από τα Set αυτά σχηματίζονται τα προσωρινά αρχεία και από εκεί οι πίνακες της βάσης).
  - `CheckSet checkset` : Αντικείμενο της κλάσης CheckSet για την απόδοση μοναδικού αναγνωριστικού.

  Αναλύει την τομή που δέχεται ως είσοδο και καλεί τις απαραίτητες μεθόδους για την συνέχιση της επεξεργασίας της. Προσθέτει σχέσεις στα δύο προσωρινά αρχεία που περιέχουν σχέσεις μορφής $A \sqsubseteq B$ και $A_1 \sqcap A_2 \sqsubseteq B$ αντίστοιχα.

- **public static void** HashSome (OWLObjectSomeValuesFrom some, **boolean** neg, SystemInitializer system,CheckSet checkset)
  Διαχειρίζεται τους υπαρξιακούς περιορισμούς κατά την κανονικοποίηση της οντολογίας. Δέχεται ως εισόδους:
  - `OWLObjectSomeValuesFrom some` : η τομή από *OWL* εκφράσεις.
  - **boolean** `neg` : Δηλώνει το ο υπαρξιακός περιορισμός συμμετείχε αρνητικά ή θετικά στο αξίωμα.
  - `SystemInitializer system` : Χρησιμοποιείται για την αποθήκευση των νέων σχέσεων που εξάγονται στα αντίστοιχα Set (από τα Set αυτά σχηματίζονται τα προσωρινά αρχεία και από εκεί οι πίνακες της βάσης).
  - `CheckSet checkset` : Αντικείμενο της κλάσης CheckSet για την απόδοση μοναδικού αναγνωριστικού.

  Αναλύει τον υπαρξιακό περιορισμό που δέχεται ως είσοδο και καλεί τις απαραίτητες μεθόδους για την συνέχιση της επεξεργασίας του. Προσθέτει σχέσεις στα δύο προσωρινά αρχεία που περιέχουν σχέσεις μορφής $A \sqsubseteq \exists r.B$ και $\exists r.A \sqsubseteq B$ αντίστοιχα.

6.4.2.7 Η κλάση Normalization

Μέλη της κλάσης

- **private** String database
  Το όνομα της βάσης δεδομένων.

- **private** String username
  Το όνομα του χρήστη.

- **private** String userpassword
  Ο κωδικός του χρήστη.

- **static** String Ontology_File_Input
  Το path του αρχείου της οντολογίας.

- **static** String Path_Dir_Temp
  Το path που αποθηκεύονται τα προσωρινά αρχεία.

Μέθοδοι της κλάσης

- **public** Normalization()
  Η μέθοδος αυτή είναι υπεύθυνη για την κανονικοποίηση της οντολογίας. Διαβάζει ένα προς ένα τα αξιώματά της και καλεί τις αντίστοιχες μεθόδους για την διαχείρισή τους. Προσθέτει εγγραφές στα προσωρινά αρχεία που περιέχουν σχέσεις μορφής $A \sqsubseteq B$ και $r \sqsubseteq s$ και καλεί τις μεθόδους για την κατασκευή της βάσης και το πέρασμα των δεδομένων όλων των προσωρινών αρχείων στους πίνακές της.

- **public static void** equiv_update()
  Ανανεώνει όλους τους πίνακες ώστε οι ισοδύναμες κλάσεις που περιέχονται σε αυτούς να έχουν όλες το ίδιο μοναδικό χαρακτηριστικό.

6.4.2.8 Η κλάση ClassificatioFinal

Μέλη της κλάσης

- **public static int** *s*
  Πόσες από τις υπό εξέταση έννοιες θα ελέγχονται κάθε φορά. Η τιμή καθορίζεται από τον χρήστη.

- **public static int** *num_Process*
  Χρησιμοποιείται για τον τερματισμό των τριών διαφορετικών σταδίων της ταξινόμησης.

- **public static** String sqlText_m
  Χρησιμοποιείται για την αποστολή SQL queries στην βάση.

- **public static** DBWrapper dbwrap_m
  Αντικείμενο της κλάσης DBWrapper, απαραίτητο για την σύνδεση με την βάση.

- **static** String database
  Το όνομα της βάσης δεδομένων.

- **static** String username
  Το όνομα του χρήστη.

- **static** String userpassword
  Ο κωδικός του χρήστη.

Μέθοδοι της κλάσης

- **public** Classify()
  Ξεκινά την διαδικασία ταξινόμησης και υπολογίζει την διάρκειά της.

- **public static void** Open_Data_Base()
  Ανοίγει μία σύνδεση με την βάση.

- **public static void** Close_Data_Base()
  Κλείνει την σύνδεση με τη βάση.

- **public static void** Process_All()
  Αρχικοποιεί τον πίνακα με τις έννοιες υπό εξέταση και καλεί τα διάφορα στάδια της διαδικασίας.

- **public static void** Role_CR1()
  Εκτελεί το μεταβατικό κλείσιμο των ρόλων και εφαρμόζει τον επαγωγικό κανόνα 5.

- **public static void** Process_Stage1(**int** number_Stop_Process)
  Αποτελεί το πρώτο στάδιο της διαδικασίας. Είναι η υπεύθυνη μέθοδος για την επαναληπτική εκτέλεση των μεθόδων που εφαρμόζουν τους επαγωγικούς κανόνες στη βάση για τις ατομικές έννοιες από τις υπό εξέταση έννοιες. Ως είσοδο δέχεται έναν αριθμό που δείχνει μετά από πόσες επαναλήψεις πρέπει να τερματίσει .

- **public static void** Process_Stage2(**int** number_Stop_Process)
  Αποτελεί το δεύτερο στάδιο της διαδικασίας. Είναι η υπεύθυνη μέθοδος για την επαναληπτική εκτέλεση των μεθόδων που εφαρμόζουν τους επαγωγικούς κανόνες στη βάση για τις έννοιες που προήλθαν από υπαρξιακούς περιορισμούς από τις υπό εξέταση έννοιες. Ως είσοδο δέχεται έναν αριθμό που δείχνει μετά από πόσες επαναλήψεις πρέπει να τερματίσει .

- **public static void** Process_Stage3()
  Αποτελεί το τρίτο στάδιο της διαδικασίας. Είναι η υπεύθυνη μέθοδος για την επαναληπτική εκτέλεση των μεθόδων που εφαρμόζουν τους επαγωγικούς κανόνες στη βάση για τις έννοιες που χρειάζονται επανεξέταση.

- **public static boolean** compute_imply(String tbl_base, String tbl_tmp,int l)

Η μέθοδος αυτή εφαρμόζει τους επαγωγικούς κανόνες 2 και 4. Αν η εφαρμογή του κανόνα παρήγαγε αποτελέσματα επιστρέφει true. Δέχεται ως είσοδο δύο πίνακες και το βήμα στο οποίο βρίσκεται η διαδικασία. Ο πρώτος πίνακας περιέχει τα αρχικά αξιώματα, ενώ ο δεύτερος περιέχει τα αξιώματα που συνδέονται με τις υπό εξέταση έννοιες και αυτά που έχουν παραχθεί από τους επαγωγικούς κανόνες κατά το τρέχον στάδιο.

- `public static boolean compute_inter(int len, int last_inter)`
  Η μέθοδος αυτή εφαρμόζει τον επαγωγικό κανόνα 3. Αν η εφαρμογή του κανόνα παρήγαγε αποτελέσματα επιστρέφει true. Δέχεται το βήμα στο οποίο βρίσκεται η διαδικασία και το τελευταίο βήμα της διαδικασίας στο οποίο παράχθηκε αποτέλεσμα από την εκτέλεση του κανόνα3.

- `public static void compute_classify_tmp_Stage1()`
  Καλεί τις μεθόδους για την εφαρμογή των επαγωγικών κανόνων 2,3,4 κατά το πρώτο στάδιο της διαδικασίας και προσθέτει τα αποτελέσματά τους στον πίνακα των τελικών αποτελεσμάτων.

- `public static void compute_classify_tmp_Stage2()`
  Αντίστοιχη λειτουργικότητα με την προηγούμενη, αλλά για το δεύτερο στάδιο της διαδικασίας.

- `public static void compute_classify_tmp_Stage3()`
  Αντίστοιχη λειτουργικότητα με την προηγούμενη, αλλά για το τρίτο και τελευταίο στάδιο της διαδικασίας.

- `public static void reload_part_Stage1(int s)`
  Δημιουργεί το απαραίτητο cluster της βάσης για τις *s* πρώτες από τις υπό εξέταση έννοιες και εφαρμόζει τον επαγωγικό κανόνα 1 κατά το πρώτο στάδιο της διαδικασίας.

- `public static void reload_part_Stage23(int s)`
  Δημιουργεί το απαραίτητο cluster της βάσης για τις *s* πρώτες από τις υπό εξέταση έννοιες και εφαρμόζει τον επαγωγικό κανόνα 1 κατά το δεύτερο και τρίτο στάδιο της διαδικασίας.

- `public static void compute_exist_Stage1()`
  Εφαρμόζει τον επαγωγικό κανόνα 6 κατά το πρώτο στάδιο της διαδικασίας.

- `public static void compute_exist_Stage23()`

Εφαρμόζει τον επαγωγικό κανόνα 6 κατά το δεύτερο και τρίτο στάδιο της διαδικασίας. Επιπλέον βρίσκει τις έννοιες που χρειάζονται επανεξέταση και τις προσθέτει εκ νέου στον αντίστοιχο πίνακα.

- `public static int Table_Count(String table_name)`
  Δέχεται σαν όρισμα το όνομα ενός πίνακα. Μετράει πόσες εγγραφές έχει και επιστρέφει τον αριθμό αυτόν.

6.4.2.9 Η κλάση Classify_ELH

Μέλη της κλάσης

- **static** String *main_database*
  Το όνομα της βάσης δεδομένων.

- **static** String *main_username*
  Το όνομα του χρήστη

- **static** String *main_password*
  Ο κωδικός του χρήστη

- **static final** String *main_adminUserName*
  Το όνομα του admin της βάσης

- **static final** String *main_adminPassword*
  Ο κωδικός του admin της βάσης

- **static** String *main_Ontology_File_Input*
  Το path στο οποίο βρίσκεται η οντολογία προς φόρτωμα

- **static** String *main_Path_Dir_Temp*
  Το path που αποθηκεύονται τα προσωρινά αρχεία.

- **static final** String *main_dbTablesScriptPath*
  Εδώ αποθηκεύεται η διαδρομή (path) του script αρχείου κατασκευής του σχήματος της βάσης δεδομένων του DBRS

Μέλη της κλάσης

- `public static void main`
  Καλεί όλες τις υπόλοιπες μεθόδους. Ξεκινά και τερματίζει την λειτουργία του συστήματός μας.

7

Έλεγχος και αξιολόγηση

Στο κεφάλαιο αυτό χρησιμοποιώντας έναν αριθμό πραγματικών οντολογιών αξιολογούμε την απόδοση του συστήματός μας. Αρχικά εξετάζουμε την επίδραση του μεγέθους της οντολογίας στον χρόνο εκτέλεσης του συστήματός μας (τον χρόνο δηλαδή που καταναλώνεται για την φόρτωση, αποθήκευση, κανονικοποίηση και ταξινόμηση της οντολογίας) και στη συνέχεια, παρατηρούμε επίσης τις αλλαγές που επιφέρει η αύξηση της μνήμης RAM που χρησιμοποιείται από το σύστημα στην απόδοση του συστήματός μας. Στην δεύτερη περίπτωση για να αναγκάσουμε την εφαρμογή στην χρήση παραπάνω RAM μεταβάλλουμε και τον αριθμό N των υπό εξέταση εννοιών ανά κύκλο.

7.1 Μεθοδολογία ελέγχου

Ο έλεγχος της λειτουργίας του συστήματός μας μπορεί να χωριστεί σε δύο σκέλη. Το πρώτο σκέλος αφορά τον έλεγχο της ορθής και πλήρης (sound and complete) λειτουργίας του συστήματος, δηλαδή στο κατά πόσο είναι ορθή και πλήρης τόσο η κανονικοποίηση, όσο και η ταξινόμηση της οντολογίας που φορτώνει το σύστημα, ενώ το δεύτερο σκέλος αφορά την αξιολόγηση της απόδοσής του.

Η πληρότητα του συστήματος μπορεί να ελεγχθεί και με οντολογίες μικρού μεγέθους, αλλά υψηλής εκφραστικότητας. Αντίθετα για τον έλεγχο της απόδοσης του συστήματος επιβάλλεται η χρήση οντολογιών μεγάλου όγκου όπως αυτές που χρησιμοποιούνται σε πραγματικές εφαρμογές (άλλωστε εκεί παρατηρούνται τα πλεονεκτήματα της χρήσης του DBMS).

7.2 Αναλυτική παρουσίαση ελέγχου

Επικεντρωνόμαστε μόνο στον έλεγχο της απόδοσης του συστήματός μας και συγκεκριμένα στον χρόνο εκτέλεσής του. Η πληρότητα (soundness και completeness) των αλγορίθμων που χρησιμοποιούνται μπορεί να αποδειχθεί θεωρητικά και δεν θα μας απασχολήσει εδώ.

Στην παράγραφο 7.2.1 παρουσιάζονται τα χαρακτηριστικά των οντολογιών που χρησιμοποιήθηκαν για τον έλεγχο της απόδοσης του συστήματος, στην παράγραφο 7.2.2 ελέγχεται η επίδραση του μεγέθους των οντολογιών στον χρόνο εκτέλεσης του προγράμματός μας και στην παράγραφο 7.2.3 η απόδοση του συστήματος όσο αυξάνεται η χρήση της μνήμης RAM.

Οι μετρήσεις που παρατίθενται παρακάτω πραγματοποιήθηκαν σε υπολογιστή με επεξεργαστή Intel(R) Core(TM) i5 CPU στα 2.67GHz, με μνήμη RAM 4GB και λειτουργικό Windows 7 Home Premium.

Επιπλέον η λειτουργία του συστήματός μας έγινε μέσω του περιβάλλοντος Eclipse Version 3.3.1.1, η έκδοση της Java που χρησιμοποιήθηκε ήταν η Java SDK 1.6, ενώ το DBMS που χρησιμοποιήθηκε ήταν ο PostgreSQL Server 8.4.

7.2.1 Οντολογίες που χρησιμοποιήθηκαν κατά τον έλεγχο

Για τον έλεγχο της απόδοσης του συστήματός μας χρησιμοποιήθηκαν 6 οντολογίες τα χαρακτηριστικά των οποίων παρατίθενται στον πίνακα 7.1.

Οι οντολογίες αυτές χρησιμοποιούνται για την κωδικοποιημένη περιγραφή ιατρικών δεδομένων. Διακρίνονται για τα μεγάλα Tbox τους και τον μεγάλο αριθμό κλάσεων που περιέχουν. Η τελευταία από αυτές, η SNOMED CT x2 δεν υφίσταται στην πραγματικότητα και δημιουργήθηκε από εμάς με μοναδικό σκοπό την χρησιμοποίησή της για τον έλεγχο του συστήματος μας.

Συγκεκριμένα, η οντολογία EL GALEN[39] αποτελεί μία $\mathcal{EL}$ εκδοχή της GALEN οντολογίας, η οποία χρησιμοποιείται για την περιγραφή ιατρικών όρων, ανατομίας και φαρμάκων. Αντίστοιχα η οντολογία Gene Ontology[40] (GO) χρησιμοποιείται για την περιγραφή γονιδίων και των χαρακτηριστικών τους και η οντολογία Thesaurus[41], που έχει αναπτυχθεί στα πλαίσια ενός προγράμματος του Αμερικανικού Ινστιτούτου Καρκίνου (US National Cancer Institute), για την περιγραφή δεδομένων που αφορούν είδη καρκίνων, ασθένειες που συνδέονται με αυτούς, θεραπείες και άλλα λοιπά δεδομένα. Τέλος η μεγαλύτερη οντολογία από αυτές που χρησιμοποιήσαμε, η SNOMED CT[42] δημιουργήθηκε από το Κολλέγιο Αμερικανών Παθολόγων (College of American Pathologists) και χρησιμοποιείται για την ιεραρχική περιγραφή ιατρικών δεδομένων.

	EL GALEN	GO	NCI Thesaurus 11.04d	NCI Thesaurus 11.05e	SNOMED CT	SNOMED CT x2
Κλάσεις	23136	34427	84786	85319	379692	759384
Ρόλοι	950	6	103	103	61	121
Αξιώματα	36489	60975	325329	331439	585291	1170582
Αξιώματα Ισοδυναμίας	9968	0	11392	11536	38719	77438
Αξιώματα Υπαγωγής Ρόλων	958	2	17	17	11	22
Αρχικές Σχέσεις	53883	60975	399771	407647	781265	1562472
Συνολικός Αριθμός Σχέσεων	596921	515403	1164085	1188695	11724467	23576073
Νέες Σχέσεις Ιεραρχίας	543038	454428	764314	781048	10943202	22013601
Νέες Σχέσεις Ιεραρχίας ως ποσοστό επί του Συνολικού Αριθμού Σχέσεων	90%	88%	65%	65%	93%	93%
Νέες Σχέσεις Ιεραρχίας που αφορούν Named Κλάσεις	349399	339317	553799	559755	5708527	11465980
Νέες Σχέσεις Ιεραρχίας που αφορούν Named Κλάσεις ως	58%	65%	47%	47%	48%	48%

[39] http://code.google.com/p/condor-reasoner/downloads/detail?name=EL-GALEN.owl&can=2&q=
[40] http://www.geneontology.org/
[41] http://ncit.nci.nih.gov/ncitbrowser/
[42] http://www.ihtsdo.org/snomed-ct/

ποσοστό επί του Συνολικού Αριθμού Σχέσεων						
Αρχικές Σχέσεις ως ποσοστό επί των Νέων Σχέσεων Ιεραρχίας	9%	11%	34%	34%	6%	6%

Πίνακας 7.1: Ποσοτικά χαρακτηριστικά των οντολογιών που χρησιμοποιήθηκαν για τον έλεγχο της απόδοσης του συστήματος

Στον παραπάνω πίνακα ο συνολικός αριθμός σχέσεων ιεραρχίας είναι το άθροισμα των αρχικών σχέσεων και των νέων σχέσεων ιεραρχίας που παράγονται μέσω της διαδικασίας ταξινόμησης.

7.2.2 Η επίδραση του μεγέθους της οντολογίας στον χρόνο εκτέλεσης

Για την μελέτη της επίδρασης του μεγέθους της οντολογίας στον χρόνο εκτέλεσης του συστήματός μας χρησιμοποιήσαμε ως μέτρο σύγκρισης την μηχανή συλλογιστικής ανάλυσης DB reasoner (παρουσιάστηκε στο Κεφάλαιο 3). Στην εκλογή αυτή μας οδήγησε το γεγονός ότι όπως και το σύστημά μας ο DB reasoner βασίζεται στην μέθοδο της δομικής υπαγωγής και στην χρησιμοποίηση του σκληρού δίσκου και όχι της μνήμης RAM για την επεξεργασία της οντολογίας.

Στον παρακάτω πίνακα παρουσιάζονται τα αποτελέσματα των μετρήσεων για τις οντολογίες της παραγράφου 7.2.1:

	DB reasoner	Το σύστημά μας
EL GALEN	56	52
GO	30	18
NCI Thesaurus 11.04d	114	144
NCI Thesaurus 11.05e	117	158
SNOMED CT	1136	1512
SNOMED CT x2	3934	8083

Πίνακας 7.2 : Σύγκριση των χρόνων εκτέλεσης του συστήματος μας με τον DB reasoner για οντολογίες διαφορετικών μεγεθών (seconds)

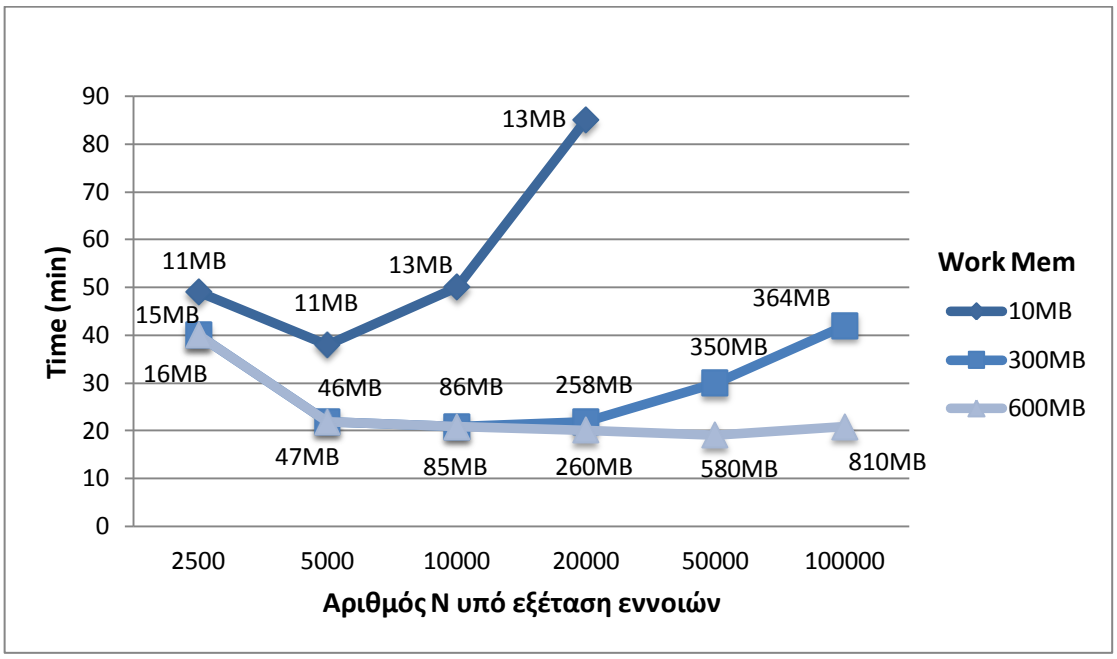
Παρατηρείται εύκολα από τους παραπάνω χρόνους εκτέλεσης ότι η αύξηση του μεγέθους της οντολογίας οδηγεί αυτόματα στην αύξηση του χρόνου εκτέλεσης. Κάτι τέτοιο ήταν αναμενόμενο αφού μεγαλύτερες οντολογίες περιέχουν περισσότερα

δεδομένα και άρα το σύστημα χρειάζεται περισσότερο χρόνο για την αποθήκευση και επεξεργασία τους. Συγκεκριμένα η μεγαλύτερη επιβάρυνση του συστήματος έγκειται στην ανάγκη για πολλαπλάσιες εφαρμογές των επαγωγικών κανόνων της ταξινόμησης (που παρουσιάστηκαν στα προηγούμενα κεφάλαια) σε σχέση με τις μικρότερες οντολογίες.

Όπως φαίνεται από τον Πίνακα 7.2 η απόδοση του συστήματος μας σε σχέση με τον DB reasoner μειώνεται σταδιακά όσο αυξάνεται το μέγεθος της οντολογίας. Συγκεκριμένα, στις μικρότερες οντολογίες όπως η EL GALEN και η GO επιτυγχάνουμε καλύτερους χρόνους, σε μεγαλύτερες όμως οντολογίες όπως οι Thesaurus και η SNOMED CT ο DB μας ξεπερνά, ενώ στην SNOMED CT x2 η διαφορά είναι πολύ μεγάλη. Παρόλα αυτά, και αν λάβουμε υπόψη ότι αυτή τη στιγμή στην κατηγορία των οντολογιών εκφραστικότητας $\mathcal{ELH}$ η SNOMED CT είναι η μεγαλύτερη οντολογία που θα συναντήσουμε, οι χρόνοι εκτέλεσης που επιτυγχάνουμε είναι ικανοποιητικοί.

7.2.3 Η επίδραση της αύξησης της μνήμης RAM στον χρόνο εκτέλεσης

Για την μελέτη της επίδρασης της αύξησης της χρήσης μνήμης RAM στον χρόνο εκτέλεσης του συστήματός μας χρησιμοποιήσαμε αποκλειστικά την οντολογία SNOMED CT. Η κύρια παράμετρος της PostgreSQL που επηρεάζει την απόδοση του συστήματός μας είναι η τιμή της παραμέτρου work_mem και αντιστοιχεί στην μέγιστη μνήμη που μπορεί να διαθέσει η PostgreSQL για κάθε λειτουργία (operation). Επιπλέον για να εξαναγκάσουμε το σύστημα στην χρήση παραπάνω μνήμης RAM αυξάνουμε ταυτόχρονα και τον αριθμό N των υπό εξέταση εννοιών ανά κύκλο. Τα αποτελέσματα των πειραμάτων παρουσιάζονται στο Διάγραμμα 7.1:



Διάγραμμα 7.1 : Επίδραση της μνήμης RAM στην απόδοση του συστήματος

Στον παραπάνω πίνακα οι αριθμοί στα σημεία καμπής των διαγραμμάτων αντιστοιχούν στην μνήμη RAM που καταναλώνεται στην πραγματικότητα από το σύστημά μας. Δηλαδή η τιμή της παραμέτρου work_mem που εμείς θέτουμε, αποτελεί την μέγιστη τιμή της RAM που μπορεί να διατεθεί, χωρίς αυτό να σημαίνει ότι χρησιμοποιείται πάντα ολόκληρη.

Όπως διαπιστώνεται εύκολα για χαμηλές τιμές της RAM της τάξης των 10MB το σύστημά μας δεν έχει καλή απόδοση. Οι χρόνοι εκτέλεσης είναι πολύ μεγάλοι και ειδικά με την αύξηση του αριθμού N των υπό εξέταση εννοιών εκτοξεύονται. Αντίθετα, με την αύξηση της μνήμης RAM στα 300MB παρατηρούμε ότι οι χρόνοι εκτέλεσης βελτιώνονται αισθητά. Επιπλέον παρατηρούμε ότι η αύξηση των υπό εξέταση εννοιών ναι μεν οδηγεί στην χρήση περισσότερης μνήμης, αλλά μετά τον αριθμό των 5000 εννοιών ανά κύκλο οι χρόνοι εκτέλεσης είτε παραμένουν ίδιοι, είτε αυξάνονται, δηλαδή δεν υπάρχει ουσιαστική βελτίωση, αλλά άσκοπη κατανάλωση μνήμης. Τέλος όταν αυξήσουμε και άλλο την μνήμη RAM που θα χρησιμοποιήσει το σύστημα σε τιμές της τάξης των 600MB, παρατηρούμε ότι οι χρόνοι εκτέλεσης κυμαίνονται σταθεροί γύρω στα 20 λεπτά, ενώ η επιπλέον RAM χρησιμοποιείται μόνο σε πολύ μεγάλα μεγέθη του αριθμού N των υπό εξέταση εννοιών.

Καταλήγουμε λοιπόν με τη βοήθεια του παραπάνω διαγράμματος και με γνώμονα το ότι μας ενδιαφέρει η ελάχιστη κατανάλωση της μνήμης RAM, στο συμπέρασμα ότι το σύστημά μας λειτουργεί βέλτιστα χρησιμοποιώντας περίπου 46MB μνήμης RAM και εξετάζοντας 5000 έννοιες ανά κύκλο. Για να τονιστεί η μεγάλη διαφορά στην κατανάλωση μνήμης με τις σύγχρονες μηχανές συλλογιστικής ανάλυσης που βασίζονται σε tableau αλγόριθμο, παρατίθενται στον Πίνακα 7.4 τέσσερα από τα πιο γνωστά τέτοια συστήματα, με την αντίστοιχη μνήμη RAM που καταναλώνουν για την ταξινόμηση τριών από τις οντολογίες που παρουσιάστηκαν στην παράγραφο 7.2.1.

	EL GALEN	NCI Thesaurus	SNOMED CT
Hermit 1.3.4 [HERM]	150	600	-
Pellet 2.2.2 [PEL]	600	1300	-
FACT v1.5.2 [FACT]	50	150	350
CEL 1.1.2 [CEL]	70	250	700

Πίνακας 7.4 : Κατανάλωση μνήμης RAM από μηχανές συλλογιστικής ανάλυσης βασισμένες σε tableau (οι μετρήσεις είναι σε MB)

8

Επίλογος

Στο κεφάλαιο αυτό παρουσιάζεται μια σύντομη ανασκόπηση της διπλωματικής μας και συνοψίζονται τα αποτελέσματά της και η συνεισφορά της. Επίσης προτείνονται βάση της εμπειρίας μας μελλοντικές επεκτάσεις και γενικές βελτιώσεις του συστήματός μας (Ενότητα 8.3).

8.1 Σύνοψη

Η συνεχής ανάπτυξη του Σημασιολογικού Ιστού (Semantic Web) καθιστά απαραίτητη την δυνατότητα ολοκληρωμένης διαχείρισης μεταδεδομένων, οργανωμένων με την μορφή οντολογιών (ontologies). Οι οντολογίες αυτές χρησιμοποιούνται για την αναπαράσταση γνώσης σε πολλούς διαφορετικούς επιστημονικούς κλάδους (βιοϊατρική, βιοχημεία, τεχνητή νοημοσύνη, κτλ) και χαρακτηρίζονται από τον μεγάλο όγκο τους και την πλούσια εκφραστικότητά τους.

Η πλειονότητα των οντολογιών έχει μεγάλο σε όγκο ABox (δεδομένα) και μικρό σε όγκο TBox (μεταδεδομένα). Όμως υπάρχουν οντολογίες (κυρίως αυτές που χρησιμοποιούνται για την μοντελοποίηση δεδομένων) που έχουν πολύ μεγάλο TBox και μικρό, ή σε μερικές περιπτώσεις και καθόλου, ABox. Τα TBoxes των οντολογιών αυτών, τα οποίο ανήκουν εξολοκλήρου ή κατά ένα μεγάλο μέρος τους στο profile $OWL\ 2\ EL$, χαρακτηρίζονται από μεγάλο αριθμό αξιωμάτων ιεραρχίας και πολύπλοκους ορισμούς κλάσεων, καθιστώντας έτσι την διαδικασία ανάλυσής τους ιδιαίτερα δύσκολη, ειδικά αν αναλογιστούμε ότι ο όγκος των προαναφερθεισών οντολογιών τείνει να αυξάνει συνεχώς, αφού εμπλουτίζονται ανά τακτά χρονικά διαστήματα με την προσθήκη επιπλέον γνώσης (νέα αξιώματα). Η δημιουργία συστημάτων ικανών για την αποδοτική συλλογιστική ανάλυση των οντολογιών αυτών συνιστά υπαρκτό πρόβλημα, διότι οι οντολογίες αυτές χρησιμοποιούνται ευρέως σε πεδία όπως η ιατρική και η βιολογία και ανανεώνονται συνεχώς.

Τα περισσότερα από τα συστήματα διαχείρισης οντολογιών που υπάρχουν σήμερα χρησιμοποιούν κάποια Μηχανή Συλλογιστικής Ανάλυσης (Reasoner). Το πρόβλημα που αντιμετωπίζουν τα συστήματα αυτά είναι ότι οι Μηχανές Συλλογιστικής Ανάλυσης λειτουργούν αποκλειστικά στην κύρια μνήμη. Επειδή η λειτουργία τους βασίζεται στην υλοποίηση του αλγόριθμου Tableau, δηλαδή κατασκευάζουν μοντέλα, καταναλώνουν ένα μεγάλο μέρος της κύριας μνήμης και αυτό έχει ως αποτέλεσμα, σε ορισμένες περιπτώσεις όπου το μέγεθος της οντολογίας είναι πολύ μεγάλο, ακόμα και την αδυναμία ανάλυσης της οντολογίας από ορισμένα από αυτά. Επιδεικνύουν δηλαδή κακή κλιμάκωση αναλογικά με το μέγεθος του TBox των οντολογιών, καθιστώντας αναγκαία την ανάπτυξη συστημάτων και τεχνικών για την αντιμετώπιση του προβλήματος.

Στην κατεύθυνση αυτή, η παρούσα διπλωματική εργασία αποσκοπούσε στον σχεδιασμό και την ανάπτυξη ενός συστήματος υπολογισμού όλων των σχέσεων ιεραρχίας που υπονοούνται μεταξύ των εννοιών μιας οντολογίας με εκφραστικότητα περιορισμένη στο τμήμα $\mathcal{EL}$ της γλώσσας αναπαράστασης OWL, χωρίς την χρησιμοποίηση όμως κάποιας Μηχανής Συλλογιστικής Ανάλυσης. Το τμήμα $\mathcal{EL}$ επιλέχθηκε πρώτον γιατί, όπως προαναφέρθηκε, θεωρείται επαρκές για την περιγραφή των οντολογιών που μας ενδιαφέρουν και δεύτερον διότι στο συγκεκριμένο τμήμα εκφραστικότητας οι διαδικασίες συλλογιστικής ανάλυσης απαιτούν πολυωνυμικό χρόνο σε σχέση με τον αριθμό των αρχικών (explicit) αξιωμάτων της οντολογίας, εξασφαλίζοντας μας έτσι την ευκολία (tractability) που απαιτείται για τη διαχείριση δεδομένων μεγάλου όγκου.

Κεντρική ιδέα της υλοποίησής μας ήταν η αντικατάσταση της χρήσης Μηχανής Συλλογιστικής Ανάλυσης που υλοποιεί τον Tableau αλγόριθμο με την τεχνική της Δομικής Υπαγωγής (Structural Subsumption) υλοποιημένη σε ένα Σχεσιακό Σύστημα Βάσεων Δεδομένων (RDBMS). Η τεχνική της Δομικής Υπαγωγής έχει αποδειχθεί ότι αποτελεί έναν ορθό και πλήρη αλγόριθμο ταξινόμησης μιας

οντολογίας για το τμήμα *EL* της *OWL*, εξασφαλίζοντας μας έτσι την πληρότητα του συστήματος.

Με την εξάλειψη του Reasoner στόχος μας ήταν η αποδοτικότερη χρησιμοποίηση της κύριας μνήμης και η εκμετάλλευση της δευτερεύουσας μνήμης για την ανάλυση της οντολογίας. Για τον σκοπό αυτό αναπτύξαμε αλγορίθμους που υλοποιούν την τεχνική της Δομικής Υπαγωγής βασιζόμενοι αποκλειστικά στο RDBMS, αποφεύγοντας κατά αυτό τον τρόπο την επιβάρυνση της κύριας μνήμης και τα προβλήματα που αυτή συνεπάγεται.

8.2 Αποτελέσματα – Συνεισφορά

Για τον έλεγχο του συστήματος μας χρησιμοποιήθηκαν, όπως είδαμε στο κεφάλαιο 7, πραγματικές οντολογίες διαφορετικού όγκου δεδομένων η καθεμία. Σαν βάση σύγκρισης χρησιμοποιήθηκε το σύστημα συλλογιστικής ανάλυσης DB reasoner , το οποίο χρησιμοποιεί και αυτό την μέθοδο της Δομικής Υπαγωγής.

Διαπιστώσαμε ότι το σύστημα που δημιουργήσαμε αντιμετωπίζει το πρόβλημα της ταξινόμησης οντολογιών με μεγάλου όγκου TBox αποδοτικά, με καλύτερη χρήση μνήμης και καλύτερη κλιμάκωση (scalability) σε σύγκριση με τα συστήματα συλλογιστικής που χρησιμοποιούν Tableau. Τα αποτελέσματα στο πεδίο του χρόνου εκτέλεσης δεν είναι το ίδιο ενθαρρυντικά αλλά σε αρκετές περιπτώσεις είναι συγκρίσιμα με τα υπόλοιπα συστήματα.

Γενικά τα αποτελέσματα που πήραμε είναι ενθαρρυντικά και γι' αυτό θεωρούμε, δεδομένου ότι υπάρχουν ακόμα πολλά περιθώρια βελτίωσης και επέκτασης του συστήματός μας, ότι ο συνδυασμός της τεχνικής της Δομικής Υπαγωγής με ένα Σχεσιακό Σύστημα Βάσεων Δεδομένων για την εκμετάλλευση της δευτερεύουσας μνήμης για την συλλογιστική ανάλυση οντολογιών, αποτελεί ένα πολύ ενδιαφέρον τομέα, άξιου περαιτέρω έρευνας.

Η συνεισφορά της διπλωματικής μας εργασίας μπορεί να συνοψιστεί στα παρακάτω:

- Υλοποιήσαμε την τεχνική της Δομικής Υπαγωγής σε ένα Σχεσιακό Σύστημα Βάσεων Δεδομένων, καθιστώντας έτσι δυνατή την ταξινόμηση του TBox οντολογιών χωρίς την χρήση εξωτερικής Μηχανής Συλλογιστικής Ανάλυσης. Συγκεκριμένα αναπτύξαμε διακριτούς αλγόριθμους, οι οποίοι υλοποιούν τα δύο διαφορετικά στάδια της Δομικής Υπαγωγής, την κανονικοποίηση και την ταξινόμηση της οντολογίας. Παράγουμε επιπλέον και το transitive reduction της οντολογίας, δηλαδή μια ταξονομία, η οποία περιέχει μόνο τις άμεσες σχέσεις υπαγωγής μεταξύ των ατομικών ονοματισμένων κλάσεων της οντολογίας
- Αναπτύξαμε αλγόριθμο απόδοσης μοναδικού αναγνωριστικού που αντιμετωπίζει το πρόβλημα διπλοτύπων που παρουσιάστηκε από την χρήση της μεθόδου hashcode() της γλώσσας προγραμματισμού Java.

- Όσον αφορά το βιβλιογραφικό κομμάτι, μελετήσαμε και παρουσιάσαμε 7 διαφορετικά συστήματα συλλογιστικής ανάλυσης που βασίζονται κατά κύριο λόγο στην μέθοδο της Δομικής Υπαγωγής.

8.3 Μελλοντικές επεκτάσεις

Παρακάτω παρατίθενται μερικές πιθανές βελτιώσεις και επεκτάσεις του συστήματός μας:

- Επέκταση του συστήματός μας ώστε να μπορεί να διαχειρίζεται οντολογίες μεγαλύτερης εκφραστικότητας, όπως για παράδειγμα η *Horn SHIQ*. Μια τέτοια οντολογία θα περιέχει περισσότερους κατασκευαστές και καινούργια είδη αξιωμάτων, κάτι που καθιστά απαραίτητη την επέκταση κάθε μέρους του συστήματός μας, από την κατασκευή της βάσης, μέχρι και την ταξινόμηση. Παρόλα αυτά, μεγαλύτερη εκφραστικότητα σημαίνει δυνατότητα για αναπαράσταση περισσότερης γνώσης, κάτι που καθιστά την συγκεκριμένη επέκταση από τις πιο σημαντικές.
- Ανάπτυξη αλγορίθμων που θα διαχειρίζονται την περίπτωση προσθήκης αξιωμάτων (incremental classification). Οι οντολογίες όπως αναφέραμε εμπλουτίζονται ανά τακτά χρονικά διαστήματα με νέα αξιώματα. Βέλτιστη λειτουργία του συστήματος θα ήταν αντί να χρειάζεται να ξαναφορτώσουμε όλη την οντολογία από την αρχή (όπως χρειάζεται να γίνει για το σύστημά μας στην τωρινή μορφή του), να υπάρχουν αλγόριθμοι που θα διαχειρίζονται μόνο τα νέα αξιώματα και θα ενημερώνουν το RDBMS αντίστοιχα.
- Ανάπτυξη τεχνικών για την εκμετάλλευση του παράλληλου προγραμματισμού. Η παράλληλη ταξινόμηση της οντολογίας, μέσω διαφορετικών νημάτων, θα επέφερε μεγάλη βελτίωση στους χρόνους λειτουργίας του συστήματός μας, επειδή όμως η παραλληλοποίηση θυσιάζει την πληρότητα (που σε αρκετές οντολογίες, όπως για παράδειγμα οι βιοϊατρικές, θεωρείται απολύτως απαραίτητη) χρειάζονται αρκετές ακόμα έρευνες για την εκμετάλευση της συγκεκριμένης τεχνικής. Μια πολύ ενδιαφέρουσα έρευνα πάνω στο συγκεκριμένο θέμα παρουσιάζεται στο

9

Βιβλιογραφία

[DBRS] DBRS
http://www.dbnet.ece.ntua.gr/pubs/details.php?id=1523&clang=0

[LF09] John Liagouris, Tryfonas Farmakakis : DataBase supported Reasoning System (DBRS). Master Thesis, School of Electrical and Computer Engineering, Division of Computer Science, National Technical University of Athens,2009

[Pellet] Pellet : open source OWL 2 reasoner
http://clarkparsia.com/pellet/

[Postgres] PostgreSQL : open source database
http://www.postgresql.org/

[Eclipse] The Eclipse Platform
http://www.eclipse.org/

[OWL 2] OWL 2 Specification
http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/

[AH04] Grigoris Antoniou, Frank van Harmelen : Web Ontology Language: OWL. In Staab, S., Studer, R., eds.: Handbook on Ontologies, Springer (2004)

[XML] Extensible Markup Language (XML):
http://www.w3.org/XML/

[RDF] RDF Tutorial
http://www.w3.org/RDF

[OWLAPI] OWLAPI v.3
http://owlapi.sourceforge.net/index.html

[PROT] Protégé Ontology Editor
http://protege.stanford.edu/

[OWL EL] OWL 2 EL profile
http://www.w3.org/TR/owl2-profiles/#OWL_2_EL

[OWL QL] OWL 2 QL profile
http://www.w3.org/TR/owl2-profiles/#OWL_2_QL

[OWL OWL 2 RL profile
http://www.w3.org/TR/owl2-profiles/#OWL_2_RL

[Ontology] Ontologies : What is an Ontology?
http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

[SNOMED] Snomed CT ontology
http://www.ihtsdo.org/index.php?id=545

[GALEN] Galen ontology
http://bioportal.bioontology.org/ontologies/1055

[GO] Go ontology
http://www.geneontology.org/

[NCI] NCI Thesaurus ontology
http://evs.nci.nih.gov/ftp1/NCI_Thesaurus

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, Peter F. Patel-Schneider: The Description Logic Handbook: Theory, Implementation, Applications. Cambridge University Press,Cambridge, UK, 2003.

[DK09] Vincent Delaitre and Yevgeny Kazakov : Classifying ELH Ontologies in SQL Databases. Proceedings of OWL: Experiences and Directions, Chantilly, Virginia, USA, 2009

[DB] The DB Reasoner
https://code.google.com/p/db-reasoner/

[BLS05] Franz Baader, Carsten Lutz, Boontawee Suntisrivaraporn : Is Tractable Reasoning in Extensions of the Description Logic EL Useful in Practice?* . In Proceedings of the Methods for Modalities Workshop (M4M-05), Berlin, 2005

[CEL] The CEL Reasoner
http://code.google.com/p/cel/

[K09] Yevgeny Kazakov : Consequence-Driven Reasoning for Horn SHIQ Ontologies. International Joint Conference On Artificial Intelligence, IJCAI-09, Pasadena, California, USA, 2009

[CB] The CB Reasoner
https://code.google.com/p/cb-reasoner/

[HMW08] V.Haarslev, R.Moller, S.Wandelt : The revival of structural subsumption in tableau-based description logic reasoners. Proceedings of International Workshop on Description Logics, Dresden, Germany, 2008

[RacerPro] The RacerPro Reasoner
http://www.franz.com/agraph/racer/

[AH08] Mina Aslani, Volker Haarslev : Towards Parallel Classification of TBoxes. Proceedings of the International Workshop on Description Logics, Dresden, Germany, 2008

[KMR10 Markus Krötzsch, Anees ul Mehdi, Sebastian Rudolph : Orel: Database-Driven Reasoning for OWL 2 Profiles. In Proceedings:Description Logics, DL2010,Waterloo, Canada, 2010

[ORL] The Orel Reasoner
https://code.google.com/p/orel/

[K11] Markus Krötzsch : Efficient Rule-Based Inferencing for OWL EL, In Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI-11, Barcelona, Catalonia, Spain, 2011

[ELK] The ELK Reasoner
https://code.google.com/p/elk-reasoner/

[CGL09] Andrea Cali, Georg Gottlob, Thomas Lukasiewiez : A General Datalog-Based Framework for Tractable-Query Answering over Ontologies. Proceedings of Principles of Database Systems,PODS-09, New York, USA, 2009

[HER] The Hermit Reasoner
http://hermit-reasoner.com/

[FACT] The Fact Reasoner
http://owl.man.ac.uk/factplusplus/

[JDBC] The JDBC driver
http://jdbc.postgresql.org/

[BBL05] Franz Baader, Sebastian Brandt and Carsten Lutz : Pushing the EL envelope. Nineteenth International Joint Conference on Artificial Intelligence, IJCAI-05, Edinburgh, Scotland, 2005

[BBL08] Franz Baader, Sebastian Brandt and Carsten Lutz : Pushing the EL envelope further. In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions, Washington,USA, 2008

[BHNP94] Franz Baader, Bernhard Hollunder, Bernhard Nebel and Hans-Jürgen Profitlich : An Empirical Analysis of Optimization Techniques for Terminological Representation Systems - or: Making KRIS get a move on. Applied Artificial Intelligence. Special Issue on Knowledge Base Management, 1994

[ABJ89] Rakesh Agrawal, .Alexander Borgida, H.V.Jagadish : Efficient Management of Transitive Relationships In Large Data And Knowledge Bases. SIGMOD '89: Proceedings of the 1989 ACM SIGMOD international conference on Management of data, New York, USA, 1989

[LTW09] Carsten Lutz, David Toman, Frank Wolter : Conjunctive query answering in the description logic EL using a relational database system. IJCAI'09 Proceedings of the 21st International Joint Conference on Artifical intelligence, Pasadena, California, Usa, 2009

[KM05] Ralf Kusters, Ralf Molitor : Structural Subsumption and Least Common Subsumers in a Description Logic with Existential and Number Restrictions, Studia Logica, 81:227–259, 2005

[DMO93] Robert Dionne, Eric Mays and Frank Oles : The Equivalence of Model-Theoretic and Structural Subsumption in Description Logics. Proceedings 13th International Joint Conference on Artificial Intelligence (IJCAI-93), Chamberry, France, 1993

[B04] Sebastian Brandt : Polynomial Time Reasoning in a Description Logic with Existential Restrictions, GCI Axioms, and - What Else? Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, Valencia, Spain, 2004

[M98] Ralf Molitor : Structural Subsumption for ALN. Technical Report LTCS-98-03, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1998

[BKM98] Franz Baader, Ralf Küsters and Ralf Molitor : Structural Subsumption considered from a automata-theoretic point of view. Proceedings of the 1998 International Workshop on Description Logics (DL 98), IRST, Povo - Trento, Italy, 1998

[S09] Boontawee Suntisrivaraporn : Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies. PhD thesis, TU Dresden, Germany, 2009

[MS09] Julian Mendez and Boontawee Suntisrivaraporn : Reintroducing CEL as an OWL 2 EL Reasoner. Proceedings of the 2009 International Workshop on Description Logics (DL2009), Oxford, United Kingdom, 2009

[NB10] Daniele Nardi, Ronald J. Brachman : An Introduction to Description Logics. Cambridge Univ. Press, 2010