ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Μετασχηματισμός Οντολογικής Γνώσης Από Εκφραστικές Σε Βατές Περιγραφικές Λογικές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΝΑΣΤΑΣΙΟΥ-ΑΝΔΡΕΑ ΓΕΩΡΓΟΥΛΑ

Επιβλέπων : Γιώργος Στάμου
Λέκτορας Ε.Μ.Π.

Αθήνα, Ιούλιος 2010

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Μετασχηματισμός Οντολογικής Γνώσης Από Εκφραστικές Σε Βατές Περιγραφικές Λογικές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΝΑΣΤΑΣΙΟΥ-ΑΝΔΡΕΑ ΓΕΩΡΓΟΥΛΑ

Επιβλέπων : Γιώργος Στάμου
Λέκτορας Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 20η Ιουλίου 2010

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γιώργος Στάμου	Στέφανος Κόλλιας	Ανδρέας-Γεώργιος Σταφυλοπάτης
Λέκτορας Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2010

(Υπογραφή)

.....

ΑΝΑΣΤΑΣΙΟΣ-ΑΝΔΡΕΑΣ ΓΕΩΡΓΟΥΛΑΣ

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

Περίληψη

Ένα από τα θεμελιώδη προβλήματα της αναπαράστασης γνώσης είναι η εύρεση μίας ικανοποιητικής ισορροπίας μεταξύ εκφραστικής ισχύος και πολυπλοκότητας. Το πρόβλημα αυτό αποκτά νέα σημασία με την ανάπτυξη του Σημασιολογικού Ιστού και τις ανάγκες για ταχεία συλλογιστική που αυτός επιβάλλει. Η συλλογιστική στις Περιγραφικές Λογικές έχει μελετηθεί σε βάθος και είναι γνωστοί και εφαρμοσμένοι αλγόριθμοι για την υλοποίησή της, ωστόσο υποφέρουν από υψηλή πολυπλοκότητα όταν αυξάνεται αρκετά η εκφραστικότητα της γλώσσας. Ένας τρόπος να αντιμετωπιστεί αυτό το πρόβλημα είναι ο μετασχηματισμός γνώσης.

Η παρούσα διπλωματική εργασία αναφέρεται στη μετατροπή οντολογιών από εκφραστικές γλώσσες αναπαράστασης σε λιγότερο εκφραστικές, αλλά βατές, γλώσσες. Υλοποιείται ένας αλγόριθμος που να επιλύει το παραπάνω πρόβλημα, χρησιμοποιώντας τεχνικές για τη βελτίωση της απόδοσής. Παρουσιάζεται επίσης ένα ολοκληρωμένο σύστημα συλλογιστικής για την απάντηση συζευκτικών ερωτημάτων, η λειτουργία του οποίου βασίζεται στην υλοποίηση του αλγορίθμου μετασχηματισμού. Τέλος, αξιολογείται η συμπεριφορά και η επίδοση του συστήματος, χρησιμοποιώντας μία σειρά οντολογιών διαθέσιμων στον Ιστό, και γίνεται απόπειρα εξαγωγής συμπερασμάτων από τα παρατηρούμενα αποτελέσματα.

Λέξεις Κλειδιά: Αναπαράσταση Γνώσης, Περιγραφικές Λογικές, Οντολογία, Απάντηση Ερωτημάτων, DL-Lite, Μετασχηματισμός Γνώσης

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

One of the fundamental problems in knowledge representation is finding an acceptable equilibrium between expressiveness and complexity. This problem has acquired a new significance due to development of the Semantic Web, and the requirements for speedy reasoning which it imposes. Reasoning in Description Logics has been well-studied, and a number of algorithms are known and used to implement reasoning services, however they suffer from high complexity when the expressiveness of the language increases to a sufficient degree. One way of dealing with the problem is knowledge compilation.

This thesis deals with transformation of ontologies from expressive representation languages to less expressive, but tractable, languages. An algorithm solving the above problem is implemented, using techniques to improve its performance. Furthermore, a complete reasoning system for the answering of conjunctive queries is presented, whose function is based on this implementation of the knowledge compilation algorithm. Finally, its behavior and performance and are evaluated using a series of ontologies available on the Web, and we attempt to draw conclusions from the observed results.

Keywords: Knowledge Representation, Description Logics, Ontology, Query Answering, DL-Lite, Knowledge Compilation

Η σελίδα αυτή είναι σκόπιμα λευκή.

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο του Εργαστηρίου Ευφυών Συστημάτων, Περιεχομένου και Αλληλεπίδρασης της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών. Με την αφορμή της ολοκλήρωσής της, θα ήθελα να ευχαριστήσω ιδιαίτερα το Λέκτορα ΕΜΠ Γιώργο Στάμου για τη συνολική του βοήθεια, την επίβλεψη και την καθοδήγησή του καθ' όλη τη διάρκειά της, καθώς και για τη συνεχώς υψηλή ποιότητα της διδασκαλίας του, έναν παράγοντα που αδιαμφισβήτητα συνετέλεσε στο ενδιαφέρον μου για τον τομέα της αναπαράστασης γνώσης και αποτέλεσε κίνητρο για την ανάληψη της συγκεκριμένης εργασίας. Οφείλω επίσης να δώσω θερμές ευχαριστίες στον Υποψήφιο Διδάκτορα του Εργαστηρίου Τάσο Βενέτη, για τη βοήθεια, τις συμβουλές και την υποστήριξη που μου προσέφερε, και τη συνεισφορά του στην ολοκλήρωση τόσο του προγραμματιστικού μέρους, όσο και της συγγραφής αυτής της εργασίας.

Θα ήθελα ακόμα να ευχαριστήσω την οικογένεια και τους φίλους μου για τη συμπαράστασή τους στις δυσκολίες του τελευταίου έτους. Τέλος, την οικογένεια Νάζου, που (έστω και εν αγνοία τους) μου δάνεισαν τη δομή τους.

Αναστάσης Γεωργούλας

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Αναπαράσταση Γνώσης και ο Σημασιολογικός Ιστός

Η αναπαράσταση γνώσης (knowledge representation) είναι ένας κλάδος της Τεχνητής Νοημοσύνης που ασχολείται με την κωδικοποίηση δεδομένων με τρόπους που να ευνοούν την εξαγωγή συμπερασμάτων από αυτά [1]. Αντικείμενό της είναι συμβολισμοί και γλώσσες, όπως η λογική πρώτης τάξης, που μπορούν να χρησιμοποιηθούν για να αναπαραστήσουμε τη γνώση γύρω από ένα οποιοδήποτε αντικείμενο. Η διαδικασία της αναπαράστασης συνδέεται άμεσα με τη συλλογιστική, δηλαδή τον τρόπο από τον οποίο μπορεί κάποιος να αξιοποιήσει τη δεδομένη γνώση για να εξάγει συμπεράσματα και να παράγει νέα γνώση, ιδιαίτερα με τρόπο μηχανικό ή αυτόματο, ώστε να μπορεί να πραγματοποιηθεί από έναν υπολογιστή. Το ενιαίο αυτό αντικείμενο αποτελεί μία προσπάθεια προσέγγισης της διαδικασίας χειρισμού της γνώσης από τα ανθρώπινα όντα, που άπτεται όχι μόνο υπολογιστικών-τεχνολογικών, αλλά και φιλοσοφικών ζητημάτων.

Για την αναπαράσταση γνώσης έχουν προταθεί και χρησιμοποιηθεί διάφοροι φορμαλισμοί, από τη λογική πρώτης τάξης μέχρι τα πλαίσια [2], τα σενάρια [3] και τα σημασιολογικά δίκτυα [4], κάθε μια από τις οποίες είναι πιο κατάλληλη για την αναπαράσταση συγκεκριμένου τύπου γνώσης, ή προσεγγίζει το πρόβλημα από διαφορετική σκοπιά και, όπως είναι φυσικό, διαθέτει τα δικά της πλεονεκτήματα και μειονεκτήματα. Με την πάροδο του χρόνου και την εξέλιξη των διαθέσιμων φορμαλισμών, προέκυψε η επιθυμία για μία γλώσσα αναπαράστασης με δύο χαρακτηριστικά: αφενός, να έχει μία αυστηρά ορισμένη

σημασιολογία (κάτι που έλειπε από, π.χ., τα πλαίσια)· αφετέρου, να είναι σχεδιασμένη έχοντας κατά νου τη διαδικασία της συλλογιστικής. Από αυτές τις απαιτήσεις προέκυψε, τη δεκαετία του 1980, το πρώτο σύστημα συλλογιστικής βασισμένο σε ένα νέο φορμαλισμό, τις Περιγραφικές Λογικές [5]. Από τότε, οι Περιγραφικές Λογικές έχουν χρησιμοποιηθεί για την περιγραφή της γνώσης που σχετίζεται με πληθώρα χώρων ενδιαφέροντος. Μεταξύ των τομέων στους οποίους έχουν βρει ιδιαίτερη σημαντική απήχηση βρίσκονται η βιολογία και η ιατρική, αν και, όπως και με άλλους φορμαλισμούς, ο συγκεκριμένος τομέας γνώσης που αναπαρίσταται δεν έχει άμεση επίδραση στη χρήση τους.

Καθ' όλη την εξέλιξη των γλωσσών αναπαράστασης γνώσης, ένα από τα πρωτεύοντα ζητήματα είναι αυτό της ισορροπίας μεταξύ εκφραστικής δύναμης και πολυπλοκότητας εξαγωγής συμπερασμάτων. Έχει παρατηρηθεί ότι, καθώς περνάμε σε πιο εκφραστικές γλώσσες, γίνεται πιο δύσκολη, περίπλοκη και χρονοβόρα η διαδικασία της συλλογιστικής. Δημιουργείται, δηλαδή, ένα πρόβλημα εξισορρόπησης, από τη μία μεριά της εκφραστικότητας (ώστε να μπορούμε να περιγράψουμε όλα αυτά που θέλουμε) και από την άλλη της πολυπλοκότητας (ώστε να μπορούμε να λαμβάνουμε συμπεράσματα μέσα σε αποδεκτά χρονικά περιθώρια). Η σημασία του προβλήματος αυτού γίνεται σαφής όταν αναλογιστεί κάποιος ότι ο σκοπός της χρήσης μίας τυπικής αναπαράστασης είναι να μπορεί να πραγματοποιείται η διαδικασία της συλλογιστικής από μηχανήματα, τα οποία δεν μπορούν να χρησιμοποιήσουν την ανθρώπινη διαίσθηση ή να «κόψουν δρόμο», αλλά υφίστανται έντονα τις συνέπειες της υψηλής πολυπλοκότητας.

Μία λύση είναι να περιοριζόμαστε σε υποσύνολα της γλώσσας τα οποία, αν και δε μας επιτρέπουν να εκφραστούμε με όλη την ελευθερία που ίσως θέλαμε, προσφέρονται για την εξεύρεση ειδικών διαδικασιών συλλογιστικής που δεν υποφέρουν από απαγορευτική πολυπλοκότητα. Ένα παράδειγμα ενός τέτοιου «συμβιβασμού» είναι οι προτάσεις Horn: είναι περιορισμένες στο τι μπορούν να εκφράσουν σε σχέση με τη λογική πρώτης τάξης, όμως μπορεί κάποιος να εξάγει συμπεράσματα με την τεχνική της ανάλυσης SLD [6], η οποία έχει καλύτερα υπολογιστικά χαρακτηριστικά από τη γενική διαδικασία ανάλυσης για τη λογική πρώτης τάξης. Αντίστοιχες ιδέες μπορούν να εφαρμοστούν και στο φορμαλισμό των Περιγραφικών Λογικών, όπως και έχει γίνει.

Τα τελευταία χρόνια, ένας τομέας που έχει προσδώσει αυξημένη σημασία στην αναπαράσταση γνώσης είναι η ανάπτυξη του Σημασιολογικού Ιστού (Semantic Web) [7]. Ο όρος αναφέρεται σε μία προέκταση του Παγκόσμιου Ιστού στην οποία θα είναι δυνατή η άντληση πληροφοριών με αυτόματο τρόπο, από μηχανές αναζήτησης και πράκτορες. Η τεράστια ποσότητα πληροφορίας που είναι διαθέσιμη σήμερα μπορεί να χρησιμεύσει σε σημαντικότατο βαθμό σε κάποιον ανθρώπινο χρήστη του Ιστού, όμως από μόνη της δεν προσφέρεται για εύκολη αναζήτηση και άντληση στοιχείων από κάποια μηχανή. Για να γίνει

αυτό δυνατό, θα πρέπει η πληροφορία που βρίσκεται στον Παγκόσμιο Ιστό να είναι κατανοητή όχι μόνο από τον άνθρωπο αλλά και από κατάλληλα σχεδιασμένες μηχανές, πράγμα που με τη σειρά του απαιτεί η αποθηκευμένη γνώση να είναι δομημένη με κάποιον κατάλληλο, τυπικό τρόπο. Κάτι τέτοιο θα επιτρέψει την ευκολότερη και αυτοματοποιημένη διαχείριση της πληροφορίας, και επομένως θα συνεισφέρει στην αυτόματη διεκπεραίωση λειτουργιών του Ιστού.

Η διαδικασία της υλοποίησης του Σημασιολογικού Ιστού δεν έχει ακόμα ολοκληρωθεί, αλλά ήδη έχουν υιοθετηθεί πρότυπα για την αναπαράσταση πληροφοριών σε διάφορα επίπεδα. Μερικές από αυτές τις γλώσσες-πρότυπα είναι, σε σειρά από χαμηλότερο επίπεδο (πιο κοντά στη μηχανή) προς υψηλότερο (πιο κοντά στην ανθρώπινη σκέψη), οι RDF [8], RDF-S [9] και OWL [10]. Η τελευταία χρησιμοποιείται για την αναπαράσταση γνώσης και την πραγματοποίηση συλλογιστικής, στο επίπεδο μίας πολύ εκφραστικής Περιγραφικής Λογικής. Η αυξημένη αυτή εκφραστικότητα, εκτός των διευκολύνσεων και δυνατοτήτων που προσφέρει, επιφέρει, σύμφωνα με τα παραπάνω, και αυξημένη πολυπλοκότητα. Παρουσιάζει ενδιαφέρον, επομένως, η έρευνα πάνω σε τρόπους μείωσης αυτής της πολυπλοκότητας, κάνοντας χρήση μίας λιγότερο εκφραστικής γλώσσας.

1.2 Αντικείμενο εργασίας

Για να τύχουν οι Περιγραφικές Λογικές ευρείας αποδοχής και χρήσης, ιδιαίτερα στα πλαίσια του Σημασιολογικού Ιστού, πρέπει να διαθέτουν ορισμένα επιθυμητά χαρακτηριστικά, μεταξύ των οποίων είναι και η διεξαγωγή συλλογιστικής σε αποδεκτούς χρόνους, ακόμα και όταν πρόκειται για μεγάλες βάσεις γνώσης. Αυτή η απαίτηση, όμως, έρχεται σε σύγκρουση με ένα από τα βασικά χαρακτηριστικά τους, δηλαδή την εγγενή υψηλή πολυπλοκότητα των εκφραστικών Περιγραφικών Λογικών. Ο συμβιβασμός των δύο αντικρουόμενων αυτών καταστάσεων έχει μεγάλο ενδιαφέρον και σημασία, καθώς αποτελεί σημαντικό βήμα για τη διεύρυνση της χρήσης τόσο των Περιγραφικών Λογικών όσο και του ίδιου του Σημασιολογικού Ιστού.

Μία λύση για την επιτάχυνση της συλλογιστικής είναι, όπως αναφέρθηκε και παραπάνω, η χρήση λιγότερο εκφραστικών Περιγραφικών Λογικών. Αυτό θα είχε το επιθυμητό αποτέλεσμα όσον αφορά τη μείωση του χρόνου, αλλά ταυτόχρονα θα περιόριζε την ποικιλία και την πολυπλοκότητα της γνώσης που θα μπορούσε να αναπαρασταθεί. Το γεγονός ότι η OWL, η πρότυπη γλώσσα αναπαράστασης οντολογιών στον Ιστό, υποστηρίζει πολύ υψηλή εκφραστικότητα είναι μία ένδειξη ότι η δυνατότητα έκφρασης αρκετά σύνθετης γνώσης αποτελεί επίσης επιθυμητό στόχο, και άρα η μείωση της εκφραστικότητας δεν είναι αποδεκτή πορεία δράσης.

Μία άλλη λύση που έχει προταθεί είναι η υιοθέτηση μίας προσεγγιστικής συλλογιστικής, που πραγματοποιείται χαλαρώνοντας την έννοια της λογικής συνεπαγωγής [11]. Αυτό επίσης θα βελτίωνε την πολυπλοκότητα, ωστόσο η χρήση τέτοιων προσεγγίσεων δε θα μπορούσε να εγγυηθεί την ορθότητα και την πληρότητα της διαδικασίας, άλλες ιδιότητες που θα θέλαμε να έχει η συλλογιστική.

Τέλος, μία τρίτη μέθοδος, η οποία ακολουθείται εδώ και θα περιγραφεί αναλυτικότερα στη συνέχεια, είναι αυτή του μετασχηματισμού γνώσης. Αυτός συνίσταται στην κατάλληλη επεξεργασία μίας αρχικής γνώσης, ώστε να δημιουργηθεί μία νέα γνώση εκφρασμένη σε άλλη γλώσσα από την αρχική, με σκοπό η συλλογιστική να είναι ταχύτερη στη δεύτερη γνώση. Η διαδικασία αυτή προϋποθέτει το επιπλέον βήμα της επεξεργασίας της αρχικής γνώσης, αλλά το αντισταθμίζει με την ταχύτερη συλλογιστική στη συνέχεια, ενώ ταυτόχρονα δεν περιορίζει το χρήστη όσον αφορά στην εκφραστικότητα που του παρέχεται και, υπό κάποιες προϋποθέσεις, μπορεί να προσφέρει ορθή και πλήρη συλλογιστική.

Το πρωτεύον αντικείμενο με το οποίο καταπιάνεται η παρούσα διπλωματική εργασία είναι η υλοποίηση μεθόδων μετασχηματισμού οντολογικής γνώσης από μία εκφραστική γλώσσα (όπως η OWL) σε μία λιγότερη εκφραστική, αλλά βατή Περιγραφική Λογική (συγκεκριμένα, την DL-Lite). Ένας αλγόριθμος για αυτή τη διαδικασία έχει προταθεί στη βιβλιογραφία, επομένως το κύριο ζήτημα είναι η μελέτη του, η διερεύνηση πιθανών αλλαγών και βελτιστοποιήσεων και η υλοποίησή του. Επιπρόσθετα, μετά την υλοποίηση του αλγορίθμου, και για την παρουσίασή του σε ένα ενιαίο πλαίσιο, η εργασία συνεχίστηκε με το σχεδιασμό μίας εφαρμογής που θα χρησιμοποιεί την παραπάνω υλοποίηση, σε συνδυασμό με άλλα διαθέσιμα εργαλεία, για να διεκπεραιώνει μία πλήρη διαδικασία συλλογιστικής.

1.2.1 Συνεισφορά

Στα πλαίσια της διπλωματικής εργασίας:

1. Πραγματοποιήθηκε μία υλοποίηση του αλγορίθμου που έχει προταθεί για τη μετατροπή οντολογιών από OWL σε DL-Lite, ενσωματώνοντας σε αυτόν κάποιες βελτιστοποιήσεις
2. Υλοποιήθηκε ένα ολοκληρωμένο σύστημα που να αναλαμβάνει την απάντηση ενός ερωτήματος πάνω σε μια OWL οντολογία, μετασχηματίζοντάς την και διεξάγοντας συλλογιστική πάνω στην προκύπτουσα οντολογία
3. Μελετήθηκαν και αξιολογήθηκαν τα αποτελέσματα του μετασχηματισμού, για να εξαχθούν συμπεράσματα σχετικά με την απόδοσή του και τους παράγοντες που την επηρεάζουν

1.3 Οργάνωση κειμένου

Στο Κεφάλαιο 2 παρουσιάζονται θεωρητικά θέματα σχετικά με τις Περιγραφικές Λογικές και τη συλλογιστική. Το Κεφάλαιο 3 περιγράφει τη διαδικασία μετασχηματισμού οντολογιών από εκφραστικές σε λιγότερο εκφραστικές Περιγραφικές Λογικές. Λεπτομέρειες για την υλοποίηση του συστήματος μετατροπής περιέχονται στο Κεφάλαιο 4, ενώ το Κεφάλαιο 5 περιέχει παρατηρήσεις και αξιολόγηση της διαδικασίας μετατροπής που υλοποιήθηκε. Το Κεφάλαιο 6 αναφέρεται στα εξωτερικά εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση της εργασίας. Τέλος, στον επίλογο συνοψίζονται τα πεπραγμένα και τα συμπεράσματα της εργασίας, πριν γίνει αναφορά σε πιθανές προεκτάσεις της.

2

Περιγραφικές Λογικές

Οι Περιγραφικές Λογικές είναι ένας φορμαλισμός αναπαράστασης γνώσης που αναπτύχθηκε ως εξέλιξη προηγούμενων τεχνικών, όπως τα Πλαίσια και τα Σημασιολογικά Δίκτυα. Επιτρέπουν την καταγραφή, με τυπικό τρόπο, της γνώσης που αντιστοιχεί σε κάποιο συγκεκριμένο πεδίο ορισμού (world), χρησιμοποιώντας ένα προκαθορισμένο σύνολο συμβολισμών. Επιπλέον, οι Περιγραφικές Λογικές προσφέρουν υπηρεσίες συλλογιστικής, δηλαδή επιτρέπουν την εξαγωγή νέας γνώσης και συμπερασμάτων αξιοποιώντας την αρχική γνώση. Στη συνέχεια θα παρουσιαστούν τα βασικά τους χαρακτηριστικά, η χρήση τους για την αναπαράσταση γνώσης καθώς και η διαδικασία της συλλογιστικής. Για περαιτέρω πληροφορίες και ανάλυση, ο αναγνώστης παραπέμπεται στο [12].

Τα πρωταρχικά στοιχεία που χρησιμοποιούνται στις Περιγραφικές Λογικές είναι οι έννοιες (concepts), οι ρόλοι (roles) και τα άτομα (individuals). Η σημασιολογία αυτών θα οριστεί αυστηρά στη συνέχεια, αλλά διαισθητικά θα μπορούσαμε να πούμε ότι τα άτομα αναπαριστούν συγκεκριμένα στοιχεία του πεδίου αναφοράς, οι έννοιες αναπαριστούν σύνολα στοιχείων με κάποιο κοινό χαρακτηριστικό, ενώ οι ρόλοι σχέσεις μεταξύ δύο στοιχείων. Παραδείγματα εννοιών θα μπορούσαν να είναι οι έννοιες Άνθρωπος, Φοιτητής, ΚαλόςΚαθηγητής, Πανεπιστήμιο, Χώρα, ενώ ως ρόλους μπορούμε να σκεφτούμε τους φοιτείΣε, έχειΚαθηγητήΤον, ΕδρεύειΣε.

Ξεκινώντας από ένα αρχικό αλφάβητο εννοιών και ρόλων, που ονομάζονται ατομικές έννοιες (atomic concepts) και ατομικοί ρόλοι (atomic roles), μπορούμε να δομήσουμε πιο πολύπλοκες έννοιες χρησιμοποιώντας κατασκευαστές εννοιών. Για παράδειγμα, μπορούμε να

θεωρήσουμε τις σύνθετες έννοιες (ΦοιτητήςΚαθηγητής) και (ΦοιτητήςΠΈχειΚαθηγητήΤον.ΚαλόςΚαθηγητής). Στα παραδείγματα αυτά, τα ⊔, ⊓ και ∀ είναι κατασκευαστές εννοιών, η σημασία των οποίων θα παρουσιαστεί στη συνέχεια, και που χρησιμοποιούνται για να ορίσουμε τις έννοιες που αναφέρονται, αντίστοιχα, σε όσους είναι φοιτητές ή καθηγητές, και σε εκείνους του φοιτητές που έχουν μόνο καλούς καθηγητές.

Το σύνολο των κατασκευαστών εννοιών που μπορούν να χρησιμοποιηθούν καθορίζει την εκφραστικότητα της εκάστοτε Περιγραφικής Λογικής. Έτσι, υπάρχουν Περιγραφικές Λογικές περιορισμένης εκφραστικότητας (που επιτρέπουν τη χρήση λίγων μόνο κατασκευαστών) ή πολύ εκφραστικές (που προσφέρουν μεγάλη ποικιλία κατασκευαστών, και άρα μπορούν να περιγράψουν πιο περίπλοκες έννοιες). Σε αυτό το σημείο πρέπει να τονιστεί ότι η εκφραστικότητα μίας Περιγραφικής Λογικής συνδέεται και με την πολυπλοκότητα της συλλογιστικής της, όπως θα γίνει φανερό παρακάτω. Εν συνεχεία θα παρουσιαστούν ορισμένοι κατασκευαστές και οι Περιγραφικές Λογικές που τους χρησιμοποιούν, ενώ η σημασιολογία τους θα εξηγηθεί αναλυτικά στην επόμενη ενότητα.

2.1 Εκφραστικότητα

Μία ολόκληρη κατηγορία Περιγραφικών Λογικών είναι η οικογένεια που βασίζεται στη γλώσσα $\mathcal{AL}$ (attributive language), που προτάθηκε το 1991 [13]. Στην $\mathcal{AL}$, οι έννοιες μπορούν να παραχθούν από τις ατομικές έννοιες και τους ατομικούς ρόλους, με βάση την παρακάτω σύνταξη:

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

όπου C, D είναι περιγραφές εννοιών, A μία ατομική έννοια και R ένας ατομικός ρόλος. Τα σύμβολα ⊤ και ⊥ αναπαριστούν δύο ειδικές έννοιες που ονομάζονται καθολική (universal concept) και κενή έννοια (bottom concept), αντίστοιχα. Διαισθητικά, η πρώτη περιγράφει όλα τα αντικείμενα του πεδίου που περιγράφουμε, ενώ η δεύτερη κανένα (αναπαριστά το κενό σύνολο). Ο κατασκευαστής ∀ χρησιμοποιείται για την κατασκευή εννοιών της μορφής ∀R.C, που ονομάζονται καθολικοί περιορισμοί ή περιορισμοί τιμής (universal restriction, value restriction). Αντίστοιχα, μία έννοια της μορφής ∃R.⊤ ονομάζεται περιορισμένος υπαρξιακός περιορισμός (limited existential restriction). Τέλος, η τομή εννοιών συμβολίζεται με τον κατασκευαστή ⊓, και η άρνηση με τον ¬, ο οποίος (στη $\mathcal{AL}$) μπορεί να εφαρμοστεί μόνο σε ατομικές έννοιες, όπως φαίνεται και στον παραπάνω συντακτικό κανόνα.

Το σύνολο των κατασκευαστών αυτό μπορεί να επεκταθεί, αυξάνοντας την εκφραστικότητα της γλώσσας. Προκύπτουν έτσι διαφορετικές Περιγραφικές Λογικές, το όνομα των οποίων

αντικατοπτρίζει τις επιπλέον δυνατότητες που προσφέρουν- συγκεκριμένα, είθισται το όνομα αυτών των γλωσσών να προκύπτει προσθέτοντας κάποια γράμματα που συμβολίζουν τους επιπλέον κατασκευαστές. Για παράδειγμα, η γλώσσα $\mathcal{ALU}$ υποστηρίζει, εκτός των παραπάνω, τον κατασκευαστή της ένωσης (union) εννοιών, με τον οποίο θα μπορούσαμε, για παράδειγμα, να χρησιμοποιήσουμε την έννοια Λέκτορας ⊔ Επίκουρος ⊔ Αναπληρωτής, για να αναπαραστήσουμε το διδακτικό προσωπικό που ανήκει σε τουλάχιστον μία από τις αντίστοιχες βαθμίδες.

Ανάμεσα στις άλλες επεκτάσεις που μπορούν να ενσωματωθούν περιλαμβάνονται:

- Ο πλήρης υπαρξιακός περιορισμός (full existential qualification) ∃R.C, που συμβολίζεται με $\mathcal{E}$ και περιγράφει τα άτομα που σχετίζονται μέσω του ρόλου R με (τουλάχιστον) ένα άτομο που ανήκει στην έννοια C. Να σημειωθεί ότι στην AL επιτρέπονται μόνο έννοιες της μορφής ∃R.⊤ (π.χ. η έννοια ∃έχειΚαθηγητήΤον.⊤, που μπορεί να σημαίνει "έχει κάποιον καθηγητή"), ενώ ο κατασκευαστής E επιτρέπει να προσδιορίζεται οποιαδήποτε έννοια (επομένως μπορούμε να αποδώσουμε την περιγραφή "έχει κάποιον καθηγητή που είναι καλός" με την έννοια ∃έχειΚαθηγητήΤον.ΚαλόςΚαθηγητής).
- Ο περιορισμός πληθυκότητας (number restriction) $\mathcal{N}$[1], που επιτρέπει να προσδιορίζουμε ότι κάποιο άτομο συνδέεται με τουλάχιστον (ή το πολύ) συγκεκριμένο πλήθος ατόμων μέσω κάποιου ρόλου. Έτσι, μπορούμε να ορίσουμε την έννοια ≥3έχειΚαθηγητήΤον (≤3έχειΚαθηγητήΤον), για να περιγράψουμε όσους έχουν τουλάχιστον (το πολύ) 3 καθηγητές. Ωστόσο, ο κατασκευαστής $\mathcal{N}$ δεν επαρκεί για να εκφράσουμε, για παράδειγμα, την έννοια "έχει τουλάχιστον 3 καθηγητές που είναι καλοί). Για αυτόν το λόγο, μπορούμε να εισάγουμε τον κατασκευαστή $\mathcal{Q}$, που καλείται προσοντούχος περιορισμός πληθυκότητας (qualified number restriction), και μας δίνει τη δυνατότητα να γράψουμε την έννοια ≥3έχειΚαθηγητή.ΚαλόςΚαθηγητής. Η επέκταση αυτή είναι αντίστοιχη με την εισαγωγή του πλήρους υπαρξιακού περιορισμού ($\mathcal{E}$) για να καλύψουμε τη μειωμένη εκφραστικότητα του περιορισμένου υπαρξιακού περιορισμού της γλώσσας $\mathcal{AL}$.
- Ως ειδική περίπτωση του κατασκευαστή $\mathcal{N}$ μπορεί να αναφερθεί ο συναρτησιακός περιορισμός πληθυκότητας (functional number restriction) $\mathcal{F}$, ο οποίος προκύπτει αν περιοριστούμε σε εκφράσεις της μορφής ≤1R, όπου R κάποιος ρόλος.
- Η εφαρμογή της άρνησης σε οποιαδήποτε έννοια, και όχι μόνο σε ατομικές. Η επέκταση αυτή συμβολίζεται με το γράμμα $\mathcal{C}$. Μπορεί να φαίνεται απλή αλλαγή,

[1] Στην πραγματικότητα, καθένας από τους $\mathcal{N}$, $\mathcal{Q}$ αποτελείται από δύο κατασκευαστές, ένα για τη μορφή «το-πολύ» και ένα για τη μορφή «τουλάχιστον»

αλλά στην πραγματικότητα η επιπλέον εκφραστικότητα που προσφέρει είναι σημαντική, και μάλιστα αρκεί για να εκφράσουμε την ένωση εννοιών και τον πλήρη υπαρξιακό περιορισμό, ακόμα και χωρίς τη χρήση των αντίστοιχων κατασκευαστών $\mathcal{U}$ και $\mathcal{E}$. Για αυτό το λόγο, η γλώσσα $\mathcal{ALC}$ ουσιαστικά έχει την ίδια εκφραστικότητα με τη γλώσσα $\mathcal{ALUEC}$ και μπορεί να χρησιμοποιηθεί το όνομα της πρώτης χωρίς να υπονοείται μειωμένη εκφραστική δύναμη σε σχέση με την πρώτη.

- Ο ορισμός αντίστροφων ρόλων (συμβολίζεται με $\mathcal{I}$). Ο αντίστροφος ρόλος (inverse role) ενός ατομικού ρόλου R συμβολίζεται με R^- και ορίζεται ως εξής: Το άτομο a συνδέεται με το άτομο b μέσω του R^- αν και μόνο αν το b συνδέεται με το a μέσω του ρόλου R. Για παράδειγμα, ο ρόλος έχειΚαθηγητήΤον (που συνδέει φοιτητές με καθηγητές) μπορεί να θεωρηθεί αντίστροφος του ρόλου έχειΦοιτητήΤον (που συνδέει καθηγητές με φοιτητές), όπως επίσης και ο ρόλος είναιΠαιδίΤου με το ρόλο είναιΓονιόςΤου.

Η παραπάνω λίστα δεν είναι εξαντλητική, παρά μόνο ενδεικτική. Μία πληρέστερη παρουσίαση μπορεί να βρεθεί στα [12] και [14]. Στην πράξη, χρησιμοποιούνται και Περιγραφικές Λογικές εκτός αυτών της οικογένειας της $\mathcal{AL}$, ωστόσο η εκφραστικότητα που υποστηρίζουν μπορεί συχνά να εκφραστεί παρόμοια με τρόπο παρόμοιο με τους παραπάνω όρους.

2.2 Σημασιολογία

Ένα από τα βασικά χαρακτηριστικά των Περιγραφικών Λογικών, που τις ξεχωρίζουν από άλλες μεθόδους αναπαράστασης γνώσης, είναι ότι είναι εφοδιασμένες με μία αυστηρή σημασιολογία, που όμως ταυτόχρονα είναι αρκετά διαισθητική και ευνόητη.

Ο προσδιορισμός της σημασιολογίας μίας Περιγραφικής Λογικής συνίσταται στο να ερμηνευτούν όλες οι έννοιες και οι ρόλοι σε σχέση με τον χώρο που επιδιώκουμε να περιγράψουμε. Για παράδειγμα, αν θεωρήσουμε ότι θέλουμε να περιγράψουμε τα μέλη μίας συγκεκριμένης πανεπιστημιακής κοινότητας, θα πρέπει η έννοια ∃έχειΚαθηγητήΤον.ΚαλόςΚαθηγητής της γλώσσας $\mathcal{ALE}$ να απεικονιστεί με τέτοιο τρόπο ώστε να αντιστοιχεί ακριβώς σε όλα τα άτομα που έχουν (τουλάχιστον) έναν καλό καθηγητή. Για το σκοπό αυτό, αρχικά προσδιορίζουμε το χώρο ερμηνείας, δηλαδή το σύνολο των αντικειμένων από τα οποία αποτελείται ο χώρος που περιγράφουμε. Για παράδειγμα, αν θέλουμε να περιγράψουμε ένα πανεπιστήμιο, θα θέλαμε ίσως να αναφερθούμε στους διάφορους μαθητές, στους καθηγητές, στις σχολές και στο ίδιο το πανεπιστήμιο, επομένως ο χώρος ερμηνείας θα αποτελείται από όλα αυτά τα στοιχεία.

Μαθηματικά, μία ερμηνεία $\mathcal{I}$ είναι ένα ζεύγος $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, όπου $\Delta^{\mathcal{I}}$ ο χώρος ερμηνείας, που είναι ένα μη κενό σύνολο, και $\cdot^{\mathcal{I}}$ μία συνάρτηση ερμηνείας. Η τελευταία, εφαρμοζόμενη σε έννοιες και ρόλους της περιγραφής, αποδίδει τη σημασία τους. Αρχικά, κάθε ατομική έννοια A ερμηνεύεται ως ένα υποσύνολο $A^{\mathcal{I}}$ του $\Delta^{\mathcal{I}}$, ενώ κάθε ατομικός ρόλος R ως ένα υποσύνολο $R^{\mathcal{I}}$ του $\Delta^{\mathcal{I}}$ x $\Delta^{\mathcal{I}}$. Στη συνέχεια, η συνάρτηση ερμηνείας μπορεί να επεκταθεί και στις πιο σύνθετες έννοιες που παράγονται με τη χρήση κατασκευαστών. Έτσι, για τις έννοιες της γλώσσας AL ισχύει η ακόλουθη σημασιολογία:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$$

$$\bot^{\mathcal{I}} = \varnothing$$

$$(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} / A^{\mathcal{I}}$$

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$

$$(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \forall b \in \Delta^{\mathcal{I}}.\ (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$$

$$(\exists R.\top)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b \in \Delta^{\mathcal{I}}.\ (a,b) \in R^{\mathcal{I}}\}$$

Με αντίστοιχο τρόπο ορίζεται η σημασιολογία για τις επεκτάσεις της $\mathcal{AL}$. Συγκεκριμένα, για καθέναν από τους κατασκευαστές που αναφέρθηκαν παραπάνω, η ερμηνεία του ορίζεται ως εξής:

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$

$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b \in \Delta^{\mathcal{I}}.\ (a,b) \in R^{\mathcal{I}} \text{ και } b \in C^{\mathcal{I}}\}$$

$$(\leq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \#\{b \mid (a,b) \in R^{\mathcal{I}}\} \leq n\}$$

$$(\geq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \#\{b \mid (a,b) \in R^{\mathcal{I}}\} \geq n\}$$

$$(\leq nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \#\{b \mid (a,b) \in R^{\mathcal{I}} \text{ και } b \in C^{\mathcal{I}}\} \leq n\}$$

$$(\geq nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \#\{b \mid (a,b) \in R^{\mathcal{I}} \text{ και } b \in C^{\mathcal{I}}\} \geq n\}$$

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} / C^{\mathcal{I}}$$

$$(R^{-})^{\mathcal{I}} = \{(a,b) \in \Delta^{\mathcal{I}} \text{ x } \Delta^{\mathcal{I}} \mid (b,a) \in R^{\mathcal{I}}\}$$

2.3 Χρήση Περιγραφικών Λογικών

Για την αναπαράσταση της γνώσης ενός πεδίου, ενδέχεται να χρειάζεται να κωδικοποιήσουμε σχέσεις μεταξύ των διαφόρων εννοιών. Για παράδειγμα, θα ήταν χρήσιμο να δηλώναμε ότι όποιος είναι καθηγητής είναι και προσωπικό του πανεπιστημίου- με άλλα λόγια, η έννοια Καθηγητής είναι υπο-έννοια της έννοιας Προσωπικό. Η σύνταξη των Περιγραφικών Λογικών

δεν εξαντλείται στον ορισμό σύνθετων εννοιών, αλλά περιλαμβάνει και αξιώματα, με τα οποία δηλώνονται σχέσεις μεταξύ των διαφορετικών εννοιών.

Η πρώτη μορφή αξιωμάτων είναι τα αξιώματα υπαγωγής (subsumption ή inclusion axioms), που έχουν τη μορφή $C \sqsubseteq D$, όπου C και D έννοιες. Το παραπάνω αξίωμα δηλώνει ότι η έννοια C υπάγεται στην έννοια D, ή αλλιώς ότι η D είναι πιο γενική από τη C. Η παραπάνω περιγραφή, για παράδειγμα, θα μπορούσε να γραφεί ως Καθηγητής $\sqsubseteq$ Προσωπικό.

Το δεύτερο είδος αξιωμάτων μεταξύ εννοιών είναι τα αξιώματα ισοδυναμίας (equivalence ή equality axioms). Ένα τέτοιο αξίωμα γράφεται $C \equiv D$ και η σημασία του είναι ότι δύο έννοιες είναι ταυτόσημες. Τα αξιώματα ισοδυναμίας μπορούν να χρησιμοποιηθούν και για να ορίσουν έννοιες, όπως για παράδειγμα

ΆτυχοςΦοιτητής $\equiv$ Φοιτητής $\sqcap$ ∀έχειΚαθηγητήΤον.($\neg$ΚαλόςΚαθηγητής)

Ένα σύνολο από αξιώματα υπαγωγής και ισοδυναμίας ονομάζεται σώμα ορολογίας ή TBox (Terminological Box). Όπως είναι αναμενόμενο, η σημασιολογία των Περιγραφικών Λογικών επεκτείνεται και στα αξιώματα που μπορεί να περιέχει ένα TBox. Λέμε ότι μία ερμηνεία I ικανοποιεί ένα αξίωμα υπαγωγής $C \sqsubseteq D$ (αντίστοιχα, ένα αξίωμα ισοδυναμίας $C \equiv D$)αν και μόνο αν $C^I \subseteq D^I$ (αντίστοιχα, $C^I = D^I$). Επίσης, μία ερμηνεία I ικανοποιεί ένα TBox $\mathcal{T}$ αν και μόνο αν ικανοποιεί όλα τα αξιώματα που περιέχει το $\mathcal{T}$. Σε αυτή την περίπτωση, η I λέγεται μοντέλο (model) του $\mathcal{T}$.

Ανάλογα με τη μοφή των αξιωμάτων ενός TBox, αυτό χαρακτηρίζεται ως απλό (simple), γενικευμένο (general) ή κυκλικό (cyclic). Ένα TBox καλείται απλό όταν στο αριστερό μέλος των αξιωμάτων υπαγωγής[2] του εμφανίζονται μόνο ατομικές έννοιες. Αντίθετα, ένα γενικό TBox μπορεί να περιέχει υπαγωγές γενικών εννοιών (general concept inclusions ή GCIs), δηλαδή αξιώματα υπαγωγής των οποίων το αριστερό μέλος να είναι μία οποιαδήποτε (εν γένει μη ατομική) έννοια. Τέλος, ένα TBox λέγεται κυκλικό όταν στα αξιώματά του εμφανίζονται κύκλοι. Ως παράδειγμα των παραπάνω, μπορούμε να θεωρήσουμε τα εξής TBoxes: $\mathcal{T}_1 = \{A \sqsubseteq B \sqcap C\}$, $\mathcal{T}_2 = \{\exists R.A \sqsubseteq B\}$, $\mathcal{T}_3 = \{A \sqsubseteq \exists R.A\}$. Θεωρώντας ότι τα A, B, C είναι ατομικές έννοιες και το R ρόλος, και με βάση τους προηγούμενους ορισμούς, το $\mathcal{T}_1$ είναι ένα απλό TBox, το $\mathcal{T}_2$ γενικευμένο και το $\mathcal{T}_3$ κυκλικό. Αυτή η κατηγοριοποίηση επηρεάζει τον τρόπο που η γνώση του TBox λαμβάνεται υπ' όψιν κατά τη διαδικασία της συλλογιστικής, όπως θα αναλυθεί αργότερα.

Εκτός από τον ορισμό των σχέσεων μεταξύ των εννοιών, που πραγματοποιείται με τις ορολογίες, χρειάζεται ένας τρόπος για να περιγράψουμε τον κόσμο της εκάστοτε εφαρμογής,

[2] Γίνεται λόγος μόνο για τα αξιώματα υπαγωγής, καθώς ένα αξίωμα ισοδυναμίας $A \equiv B$ μπορεί πάντα να αντικατασταθεί με το ζεύγος $\{A \sqsubseteq B, B \sqsubseteq A\}$.

κάνοντας αναφορές σε συγκεκριμένα άτομα ή αντικείμενα του χώρου ερμηνείας. Συγκεκριμένα, επιθυμούμε να ορίσουμε σχέσεις στιγμιότυπου μεταξύ ενός ατόμου και μίας έννοιας (για παράδειγμα, για να δηλώσουμε ότι ο Βασίλης είναι Φοιτητής), ή μεταξύ ενός ζεύγους ατόμων και ενός ρόλου (ο Βασίλης έχει καθηγητή τον Κωνσταντίνο).

Στις Περιγραφικές Λογικές, αυτό γίνεται μέσω ισχυρισμών εννοιών (concept assertions) και ισχυρισμών ρόλων (role assertions), αντίστοιχα. Ένας ισχυρισμός έννοιας γράφεται ως a:C ή C(a), και δηλώνει ότι το άτομο a είναι στιγμιότυπο της έννοιας C. Ένας ισχυρισμός ρόλου γράφεται ως (a,b):R ή R(a,b), και δηλώνει ότι το ζεύγος (a,b) είναι στιγμιότυπο του ρόλου R, δηλαδή ότι τα άτομα a και b συνδέονται μέσω του ρόλου R. Ένα σύνολο από ισχυρισμούς ονομάζεται σώμα ισχυρισμών ή ABox (Assertional Box).

Για να ορίσουμε τη σημασιολογία ενός ABox, πρέπει πρώτα να αποδώσουμε μία σημασία στα άτομα που εμφανίζονται στους ισχυρισμούς. Αυτό γίνεται μέσω της συνάρτησης ερμηνείας $\cdot^I$, η οποία αντιστοιχίζει άτομα σε αντικείμενα (στοιχεία) του χώρου ερμηνείας. Έτσι, αν a είναι το όνομα ενός ατόμου (π.χ. σε έναν ισχυρισμό έννοιας), το a^I υποδηλώνει το αντικείμενο του Δ^I που αντιπροσωπεύει το άτομο a. Μία σύμβαση που συχνά υιοθετείται είναι η λεγόμενη υπόθεση μοναδικότητας ονομάτων(unique names assumption). Σύμφωνα με αυτή, σε κάθε αντικείμενο αντιστοιχεί μόνο ένα άτομο. Αυτό σημαίνει ότι, αν στο ABox εμφανίζονται δύο διαφορετικά ονόματα ατόμων, π.χ. a και b, τότε αυτά αντιστοιχούν σίγουρα σε διαφορετικά αντικείμενα του χώρου ερμηνείας (τυπικά: αν a, b είναι διαφορετικά ονόματα, τότε $a^I \neq b^I$).

Δεδομένης της παραπάνω επέκτασης της συνάρτησης ερμηνείας, μπορούμε να ερμηνεύσουμε τους ισχυρισμούς ενός ABox. Λέμε ότι μία ερμηνεία I ικανοποιεί έναν ισχυρισμό έννοιας a:C (αντίστοιχα, έναν ισχυρισμό ρόλου (a,b):R) αν και μόνο αν $a^I \in C^I$ (αντίστοιχα, $(a^I,b^I) \in R^I$). Όπως και στην περίπτωση του TBox, μία ερμηνεία I ικανοποιεί ένα ABox $\mathcal{A}$ αν και μόνο αν ικανοποιεί όλους τους ισχυρισμούς του A, και τότε η I καλείται μοντέλο του $\mathcal{A}$. Τέλος, λέμε ότι μία ερμηνεία ικανοποιεί ένα ABox $\mathcal{A}$ με βάση το (μβτ) TBox $\mathcal{T}$ αν είναι ταυτόχρονα μοντέλο και του $\mathcal{A}$ και του $\mathcal{T}$.

Κάποιες Περιγραφικές Λογικές επιτρέπουν να ορίζονται αντίστοιχες σχέσεις (υπαγωγής και ισοδυναμίας) όχι μόνο για τις έννοιες, αλλά και για τους ρόλους [15]. Έτσι, για παράδειγμα, μπορούμε να πούμε ότι η σχέση είναιΠρόγονοςΤου είναι πιο γενική από τη σχέση είναιΠατέραςΤου, και αυτό μπορεί να εκφραστεί με το αξίωμα είναιΠατέραςΤου$\sqsubseteq$είναιΠρόγονοςΤου. Αξιώματα της μορφής $R \sqsubseteq S$, όπου R, S ρόλοι, ονομάζονται αξιώματα υπαγωγής ρόλων (role inclusion axioms). Η επιπλέον αυτή εκφραστικότητα (δημιουργία ιεραρχιών ρόλων) δηλώνεται στο όνομα της αντίστοιχης γλώσσας με το γράμμα $\mathcal{H}$.

Επίσης, κάποιες γλώσσες επιτρέπουν δηλώσεις σχετικά με ιδιότητες ρόλων, όπως π.χ. μεταβατικότητα. Ένα παράδειγμα μεταβατικού ρόλου είναι η σχέση έχειΑπόγονοΤον: Αν ο a έχει απόγονο τον b και επίσης ο b έχει απόγονο τον c, τότε και ο a έχει απόγονο τον c. Το αξίωμα που δηλώνει ότι ο ρόλος R είναι μεταβατικός γράφεται **Trans**(R) ή **Tr**(R). Η επέκταση της $\mathcal{ALC}$ που υποστηρίζει τέτοιες δηλώσεις ονομάζεται $\mathcal{ALC}_{R^+}$ ή, για απλούστευση, $\mathcal{S}$.

Κατ' αναλογία με το TBox, ένα σύνολο αξιωμάτων που αναφέρονται σε ρόλους, όπως τα παραπάνω, ονομάζεται σώμα ρόλων ή RBox (Role Box). Η σημασιολογία των αξιωμάτων ρόλων ορίζεται με τρόπο αντίστοιχο με αυτή για τα TBoxes ABoxes. Για παράδειγμα, μία ερμηνεία I ικανοποιεί ένα αξίωμα υπαγωγής ρόλων $R \sqsubseteq S$ αν και μόνο αν $R^I \subseteq S^I$ (μόνο που σε αυτή την περίπτωση, τα R^I, S^I είναι υποσύνολα όχι του Δ^I, αλλά του $\Delta^I \times \Delta^I$).

Μία βάση γνώσης (knowledge base) ορίζεται ως ένα ζεύγος $\mathcal{K} = (\mathcal{T},\mathcal{A})$, όπου $\mathcal{T}$ είναι ένα TBox και $\mathcal{A}$ ένα ABox. Αν η Περιγραφική Λογική είναι αρκετά εκφραστική ώστε να υποστηρίζει και αξιώματα ρόλων, τότε η βάση γνώσης ορίζεται ως μία τριάδα $(\mathcal{T},\mathcal{R},\mathcal{A})$ όπου $\mathcal{T}$, $\mathcal{A}$ όπως παραπάνω και $\mathcal{R}$ ένα RBox.

2.4 Συλλογιστική

Στη συνέχεια θα γίνει αναφορά σε ένα άλλο θεμελιώδες χαρακτηριστικό των Περιγραφικών Λογικών: την ικανότητα εξαγωγής συμπερασμάτων. Μέχρι τώρα, έχει αναλυθεί το πώς μπορούμε να κωδικοποιήσουμε τη γνώση που είναι διαθέσιμη για κάποιο τομέα ενδιαφέροντος. Από αυτή τη γνώση, όμως, ενδέχεται να μπορούν να αντληθούν και άλλα συμπεράσματα, που δεν περιλαμβάνονται εκπεφρασμένα στην περιγραφή που έχει γίνει. Ένα σημαντικό ερώτημα που προκύπτει, επομένως, είναι πώς μπορούμε να εξάγουμε νέα γνώση (συμπεράσματα) από την περιγραφή μίας γνώσης σε κάποια γλώσσα Περιγραφικής Λογικής- και, φυσικά, πώς αυτό μπορεί να γίνει με τυπικό τρόπο ώστε η διαδικασία να μπορεί να αυτοματοποιηθεί.

Η ικανότητα εξαγωγής συμπερασμάτων είναι συνυφασμένη με τις Περιγραφικές Λογικές και, μαζί με την ακριβή σημασιολογία, ήταν από τους αρχικούς στόχους που επιδιώχθησαν κατά τη σχεδίαση των πρώτων συστημάτων βασισμένων σε Περιγραφικές Λογικές. Η σημασία που έχει το ζήτημα αυτό φαίνεται και από το γεγονός ότι η αναζήτηση μεθόδων και αλγορίθμων για την αποδοτική υλοποίηση της συλλογιστικής σε Περιγραφικές Λογικές έχει αποτελέσει αντικείμενο εκτενούς μελέτης κατά τη διάρκεια ετών, και συνεχίζει να αποτελεί.

Οι Περιγραφικές Λογικές προσφέρουν συγκεκριμένες υπηρεσίες συλλογιστικής με βάση το TBox ή/και το ABox της βάσης γνώσης. Αρχικά θα παρουσιαστούν οι υπηρεσίες πάνω στο TBox, οι οποίες έχουν να κάνουν κυρίως με σχέσεις μεταξύ εννοιών. Οι υπηρεσίες αυτές

ορίζονται με βάση την έννοια του μοντέλου που παρουσιάστηκε παραπάνω, και είναι οι ακόλουθες:

- Ικανοποιησιμότητα: Μία έννοια C είναι ικανοποιήσιμη (satisfiable) με βάση το (μβτ) TBox $\mathcal{T}$ αν και μόνο αν υπάρχει μοντέλο $\mathcal{I}$ του $\mathcal{T}$, τέτοιο ώστε $C^{\mathcal{I}} \neq \emptyset$.
- Υπαγωγή: Μία έννοια C υπάγεται σε (is subsumed by) μία έννοια D μβτ $\mathcal{T}$, αν και μόνο αν, για κάθε μοντέλο $\mathcal{I}$ του $\mathcal{T}$, ισχύει $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Συμβολικά, γράφουμε $\mathcal{T} \models C \sqsubseteq D$.
- Ισοδυναμία: Μία έννοια C είναι ισοδύναμη με (is equivalent to) μία έννοια D μβτ $\mathcal{T}$ (συμβολικά: $\mathcal{T} \models C \equiv D$), αν και μόνο αν, για κάθε μοντέλο $\mathcal{I}$ του $\mathcal{T}$, ισχύει $C^{\mathcal{I}} = D^{\mathcal{I}}$.
- Ξένες έννοιες: Μία έννοια C είναι ξένη με (is disjoint with) μία έννοια D μβτ $\mathcal{T}$, αν και μόνο αν, για κάθε μοντέλο $\mathcal{I}$ του $\mathcal{T}$, ισχύει $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$.

Οι παραπάνω υπηρεσίες εξαγωγής συμπερασμάτων δεν είναι τελείως ανεξάρτητες μεταξύ τους. Μάλιστα, μπορούν όλες να αναχθούν στην υπαγωγή εννοιών, αρκεί η γλώσσα της Περιγραφικής Λογικής να υποστηρίζει τομή εννοιών και την κενή έννοια. Συγκεκριμένα,

- Μία έννοια C είναι μη ικανοποιήσιμη αν και μόνο αν υπάγεται στην κενή έννοια, $\bot$.
- Δύο έννοιες C και D είναι ισοδύναμες αν και μόνο αν η C υπάγεται στη D και η D υπάγεται στη C.
- Δύο έννοιες C και D είναι ξένες μεταξύ τους αν και μόνο αν η τομή τους $C \sqcap D$ υπάγεται στην κενή έννοια, $\bot$.

Παρομοίως, αν η γλώσσα υποστηρίζει τομή εννοιών και άρνηση σύνθετων (όχι μόνο ατομικών) εννοιών, ισχύουν οι παρακάτω αναγωγές στη μη-ικανοποιησιμότητα:

- Μία έννοια C υπάγεται σε μία έννοια D αν και μόνο αν η έννοια $C \sqcap \neg D$ είναι μη-ικανοποιήσιμη.
- Δύο έννοιες C και D είναι ισοδύναμες αν και μόνο αν οι έννοιες $C \sqcap \neg D$ και $C \sqcap \neg D$ είναι και οι δύο μη-ικανοποιήσιμες.
- Δύο έννοιες C και D είναι ξένες μεταξύ τους αν και μόνο αν η τομή τους $C \sqcap D$ είναι μη-ικανοποιήσιμη.

Όσον αφορά τις υπηρεσίες συλλογιστικές πάνω σε ABox, αρχικά πρέπει να παρατηρήσουμε ότι οι ισχυρισμοί που περιέχονται στο ABox ενδεχομένως να αναφέρονται σε έννοιες οι οποίες ορίζονται (ή, γενικότερα, για τις οποίες περιέχεται πληροφορία) σε κάποιο TBox. Αυτό σημαίνει ότι πρέπει να λαμβάνεται υπ' όψιν το περιεχόμενο του TBox, όπως φαίνεται και από τους παρακάτω ορισμούς και θα αναλυθεί στη συνέχεια. Οι υπηρεσίες συλλογιστικής που προσφέρονται πάνω σε ένα ABox $\mathcal{A}$ και ένα TBox $\mathcal{T}$ είναι οι ακόλουθες:

- Συνέπεια: Το $\mathcal{A}$ είναι συνεπές (consistent) μβτ $\mathcal{T}$ αν και μόνο αν υπάρχει μοντέλο του $\mathcal{T}$ το οποίο να είναι και μοντέλο του $\mathcal{A}$.
- Συνεπαγωγή: Το $\mathcal{A}$ συνεπάγεται (entails) έναν ισχυρισμό (έννοιας ή ρόλου) φ μβτ $\mathcal{T}$ αν και μόνο αν κάθε ερμηνεία που είναι ταυτόχρονα μοντέλο του $\mathcal{A}$ και του $\mathcal{T}$ ικανοποιεί τον ισχυρισμό. Συμβολικά, αυτό γράφεται $\mathcal{T},\mathcal{A} \models \varphi$.

Όπως και στην περίπτωση του TBox, και εδώ μπορούμε να αναγάγουμε τις παραπάνω υπηρεσίες στο πρόβλημα της συνέπειας. Συγκεκριμένα, ένα ABox $\mathcal{A}$ συνεπάγεται έναν ισχυρισμό φ μβτ TBox $\mathcal{T}$ αν και μόνο αν το $\mathcal{A} \cup \{\neg\varphi\}$ είναι ασυνεπές μβτ $\mathcal{T}$. Επίσης, μία έννοια C είναι ικανοποιήσιμη μβτ TBox $\mathcal{T}$ αν και μόνο αν το ABox {C(a)} είναι συνεπές μβτ $\mathcal{T}$ (όπου a ένα νέο άτομο, που δεν εμφανίζεται στη βάση γνώσης).

Το πρώτο βήμα για την πραγματοποίηση της συλλογιστικής, όπως αναφέρθηκε παραπάνω, είναι να ληφθεί υπ' όψιν η γνώση που περιέχει το TBox. Η βασική ιδέα είναι να εφαρμοστεί κάποια επεξεργασία πάνω στο TBox, μετά το πέρας της οποίας η διαδικασία της συλλογιστικής μπορεί να συνεχίσει αγνοώντας το, δηλαδή να πραγματοποιηθεί μβτ κενό TBox. Ο συγκεκριμένος τρόπος με τον οποίο θα γίνει αυτή η απαλοιφή εξαρτάται από τη μορφή των αξιωμάτων που περιέχονται στο TBox, δηλαδή το αν αυτό είναι απλό, γενικευμένο ή κυκλικό.

Για τα απλά TBoxes, είναι αρκετή η μέθοδος του ξεδιπλώματος ή της επέκτασης (unfolding ή expansion). Συνοπτικά, με βάση τη μέθοδο αυτή, όλα τα αξιώματα υπαγωγής αντικαθίστανται από αξιώματα ισοδυναμίας (ορισμούς) και στη συνέχεια οι σύνθετες έννοιες αντικαθίστανται από τον ισοδύναμο ορισμό τους. Λαμβάνεται έτσι ένα νέο TBox, που καλείται επέκταση του αρχικού. Το μέγεθος της επέκτασης μπορεί σε ορισμένες περιπτώσεις να είναι εκθετικά μεγαλύτερο εκείνου του αρχικού TBox, έχουν όμως προταθεί παραλλαγές της τεχνικής αυτής, οι οποίες προσφέρουν επέκταση με πολυωνυμικό μέγεθος. Η διαδικασία του ξεδιπλώματος δεν προσφέρεται για την απαλοιφή γενικευμένων ή/και κυκλικών TBoxes, γιατί στην μεν πρώτη περίπτωση δεν μπορεί να πραγματοποιηθεί, στη δε δεύτερη δεν τερματίζει ποτέ. Για να αντιμετωπιστεί αυτή η αδυναμία, χρησιμοποιείται η μέθοδος της εσωτερίκευσης (internalization). Με αυτή, καταλήγουμε σε ένα μοναδικό αξίωμα υπαγωγής το οποίο κωδικοποιεί την πληροφορία των αξιωμάτων του αρχικού TBox, και το οποίο χρησιμοποιείται για την πραγματοποίηση της συλλογιστικής.

Μετά την απαλοιφή του TBox, πρέπει να χρησιμοποιηθεί κάποιος αλγόριθμος συλλογιστικής (reasoning alogrithm) που να υλοποιεί την υπηρεσία συλλογιστικής. Μία κατηγορία τέτοιων αλγορίθμων είναι οι αλγόριθμοι tableau (tableau algorithms) [16], οι οποίοι εφαρμόζουν διαδοχικά ένα σύνολο κανόνων και, ανάλογα με το αν καταλήξουν σε κάποια αντίφαση ή όχι,

αποφαίνονται για την ικανοποιησιμότητα ή μη κάποιας έννοιας (υπενθυμίζεται ότι, υπό προϋποθέσεις, όλες οι υπηρεσίες συλλογιστικής σε TBox ανάγονται στη μη-ικανοποιησιμότητα εννοιών). Η εφαρμογή των κανόνων απαιτεί μία προ-επεξεργασία των εννοιών, ενώ η διαδικασία τερματίζει όταν δεν υπάρχει κανένας κανόνας που να μπορεί να εφαρμοστεί ή όταν εντοπιστεί μία σύγκρουση (clash), δηλαδή κάποιο σύνολο ισχυρισμών το οποίο είναι προφανώς αδύνατο να ισχύει. Οι συγκεκριμένοι κανόνες και συγκρούσεις εξαρτώνται από την εκφραστικότητα της Περιγραφικής Λογικής, δηλαδή τους κατασκευαστές που αυτή προσφέρει.

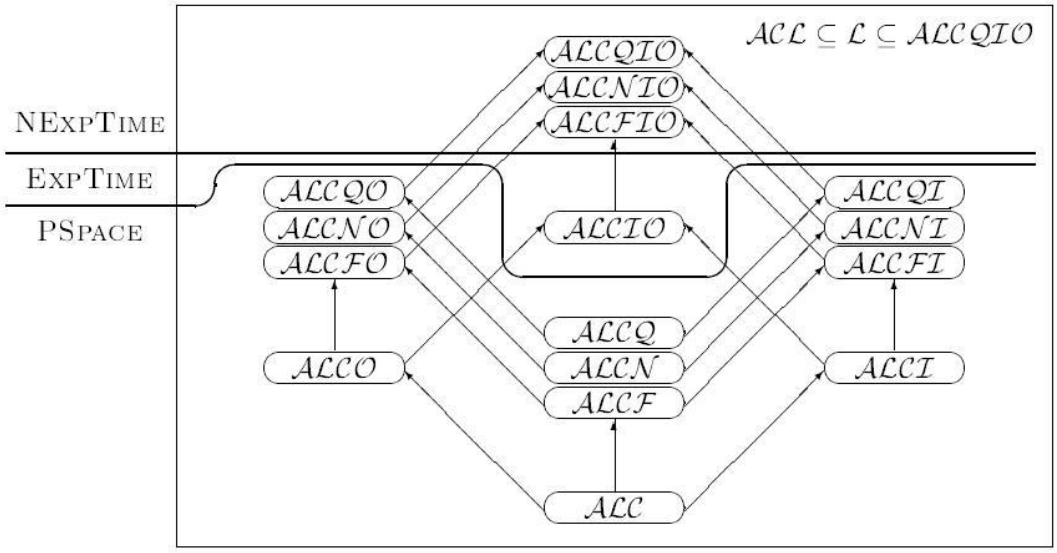
Οι αλγόριθμοι tableau έχουν χρησιμοποιηθεί σε διάφορα συστήματα συλλογιστικής (reasoners) για Περιγραφικές Λογικές, όπως τα Pellet [17], FaCT++ [18], HermiT [19] κ.ά. Εκτός από τη βασική λογική των αλγορίθμων tableau, αυτά ενσωματώνουν συνήθως και κάποιες βελτιστοποιήσεις για την αντιμετώπιση ορισμένων καταστάσεων και κατασκευαστών που αυξάνουν ιδιαίτερα την πολυπλοκότητα.

Ένα σημείο το οποίο χρήζει ιδιαίτερης σημασίας είναι η πολυπλοκότητά των αλγορίθμων πινάκων. Όπως είναι ίσως αναμενόμενο, αυτή εξαρτάται από την εκφραστικότητα της Περιγραφικής Λογικής. Αυτό συμβαίνει γιατί οι διάφορες μορφές σύνθετων εννοιών (π.χ. τομή, ένωση, περιορισμοί πληθυκότητας) επιβάλλουν διαφορετικούς κανόνες κατά την εκτέλεση του αλγορίθμου, μερικοί από τους οποίους επηρεάζουν έντονα την πολυπλοκότητα. Ακόμα, ορισμένοι από τους κανόνες (όπως αυτοί που σχετίζονται με την ένωση εννοιών ή τους περιορισμούς πληθυκότητας) είναι μη-ντετερμινιστικοί, πράγμα που εισάγει ακόμα μία πηγή πολυπλοκότητας, καθώς αυτοί οι κανόνες δεν προσδιορίζουν ένα μοναδικό μονοπάτι εκτέλεσης, αλλά επιτρέπουν μία σειρά από πιθανές εκτελέσεις οι οποίες να πρέπει ενδεχομένως να ελεγχθούν όλες.

Ως παράδειγμα, το πρόβλημα της συλλογιστικής[3] στη γλώσσα $\mathcal{ALC}$ ανήκει στην κλάση πολυπλοκότητας PSPACE. Ακόμα και με διάφορες επεκτάσεις, όπως την προσθήκη αντίστροφων ρόλων, ονοματικών εννοιών ή περιορισμών πληθυκότητας (αλλά όχι συνδυασμού αυτών) ή αξιωμάτων μεταβατικότητας ρόλων, το πρόβλημα δεν ξεφεύγει από αυτή την κλάση. Ωστόσο, ο συνδυασμός ονοματικών εννοιών και αντίστροφων ρόλων οδηγεί σε μεγάλη αύξηση της πολυπλοκότητας, με αποτέλεσμα το πρόβλημα να ανήκει πλέον στην κλάση EXPTIME, ενώ η ύπαρξη και περιορισμών πληθυκότητας (προσοντούχων ή μη) ταυτόχρονα κάνει το πρόβλημα μέλος της κλάσης NEXPTIME. Μία πληρέστερη παρουσίαση της πολυπλοκότητας των αλγορίθμων συλλογιστικής για διάφορες γλώσσες μπορεί να βρεθεί στο [20].

[3] Συγκεκριμένα, της μη-ικανοποιησιμότητας εννοιών μβτ κενό TBox

Σε γενικές γραμμές, υπάρχει μία «ισορροπία» (tradeoff) μεταξύ εκφραστικότητας και καλής πολυπλοκότητας: η αυξημένη πολυπλοκότητα της συλλογιστικής είναι το τίμημα που πληρώνει ο χρήστης για τη μεγαλύτερη εκφραστικότητα. Αντίστροφα, για την αντιμετώπιση του υψηλού κόστους της συλλογιστικής στις πολύ εκφραστικές γλώσσες, έχουν προταθεί διάφορες εναλλακτικές Περιγραφικές Λογικές, οι οποίες προσφέρουν χαμηλότερη πολυπλοκότητα θυσιάζοντας μέρος της εκφραστικής ισχύος. Οι γλώσσες αυτές απευθύνονται σε χρήστες που δε χρειάζονται το πλήρες εύρος μίας πολύ εκφραστικής γλώσσας αλλά μόνο μέρος της. Αν, επομένως, κάποιος μπορεί να αρκεστεί σε ένα υποσύνολο των κατασκευαστών και των δυνατοτήτων της γλώσσας, δεν υπάρχει λόγος να βαρύνεται από την υπολογιστικά ακριβή συλλογιστική της πλήρους γλώσσας. Μερικές από αυτές τις γλώσσες, οι οποίες μάλιστα δεν είναι τόσο περιοριστικές στις εκφράσεις που επιτρέπουν αλλά διατηρούν χαμηλή πολυπλοκότητα, είναι η οικογένεια της $\mathcal{EL}$ [21] καθώς και η DL-Lite, η οποία θα μελετηθεί αναλυτικότερα στη συνέχεια.



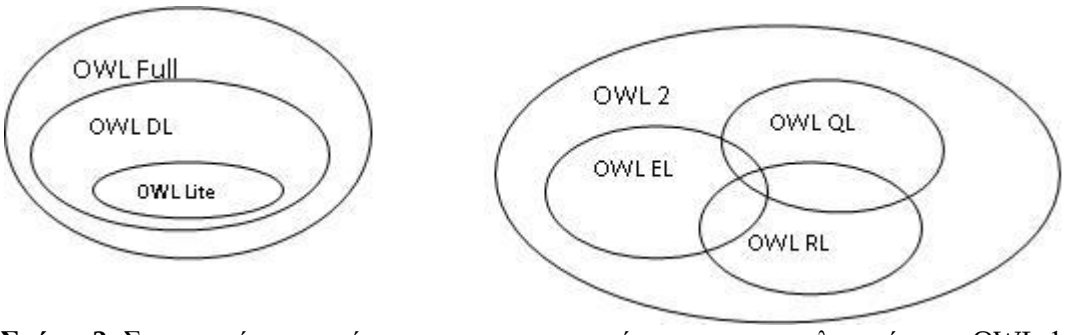
Σχήμα 1: Πολυπλοκότητα του προβλήματος της ικανοποιησιμότητας εννοιών μβτ κενό TBox για ορισμένες Περιγραφικές Λογικές διαφορετικής εκφραστικότητας (από το [22])

2.5 Η γλώσσα OWL

Η OWL (Web Ontology Language) είναι μία γλώσσα περιγραφής οντολογιών σχεδιασμένη για το Σημασιολογικό Ιστό και καθιερωμένη ως πρότυπο από την Κοινοπραξία του Παγκόσμιου Ιστού (World Wide Web Consortium, W3C) [10]. Η χρήση και η σημασιολογία της είναι παρόμοιες με αυτές των Περιγραφικών Λογικών και, όπως θα φανεί, ορισμένα τμήματα της γλώσσας είναι εκφραστικά ισοδύναμα με συγκεκριμένες Περιγραφικές Λογικές.

Η αρχική έκδοση της OWL προβλέπει τρεις υπογλώσσες, με διαφορετικό βαθμό εκφραστικής ισχύος. Αυτές είναι, σε σειρά αυξανόμενης εκφραστικότητας οι OWL Lite, OWL DL και OWL Full, κάθε μία εκ των οποίων είναι υπερσύνολο της προηγούμενης. Οι δύο πρώτες παρέχουν εκφραστικότητα αντίστοιχη αυτή των Περιγραφικών Λογικών $\mathcal{SHIF}(\mathbf{D})$ και $\mathcal{SHOIN}(\mathbf{D})$, αντίστοιχα, και η συλλογιστική σε αυτές είναι πάντα αποφασίσιμη, με την πρώτη να παρέχει καλύτερη πολυπλοκότητα από τη δεύτερη. Αντίθετα, η OWL Full επιτρέπει την ελεύθερη χρήση όλων των χαρακτηριστικών της RDF, με αποτέλεσμα να μην μπορεί να εκφραστεί στα πλαίσια κάποιας Περιγραφικής Λογικής και να είναι μη-αποφασίσιμη.

Η δεύτερη έκδοση του προτύπου [23], που υιοθετήθηκε από την W3C το 2009 με το όνομα OWL 2, προσφέρει ακόμα μεγαλύτερη εκφραστικότητα, αντίστοιχη της $\mathcal{SROIQ}(\mathbf{D})$. Επίσης, η νέα έκδοση προσδιορίζει τρία προφίλ (profiles) της γλώσσας, τα OWL EL, OWL QL και OWL RL, που είναι υπογλώσσες κατάλληλες για διαφορετικές χρήσεις και απαιτήσεις σε πολυπλοκότητα [24]. Σε αντίθεση με τις υπογλώσσες της OWL 1, κανένα από αυτά τα προφίλ δεν είναι υποσύνολο άλλου. Το προφίλ OWL EL, που βασίζεται στην οικογένεια Περιγραφικών Λογικών EL, είναι κυρίως κατάλληλο για οντολογίες με πολύ μεγάλο αριθμό κλάσεων ή/και ιδιοτήτων (για παράδειγμα, βιοϊατρικές οντολογίες) και προσφέρει έλεγχο συνέπειας, υπαγωγής και συνεπαγωγής ισχυρισμών σε πολυωνυμικό χρόνο. Το προφίλ OWL QL βασίζεται στις Περιγραφικές Λογικές της οικογένειας της DL-Lite και προσφέρει απάντηση ερωτημάτων σε λογαριθμικό χώρο. Είναι ιδιαίτερα κατάλληλο για εφαρμογές που ανακτούν τους ισχυρισμούς τους από ένα σύστημα διαχείρισης σχεσιακών Βάσεων Δεδομένων (RDBMS). Τέλος, το προφίλ OWL RL έχει επιλεγεί ώστε να είναι κατάλληλο για εφαρμογές που υλοποιούνται με συστήματα κανόνων και προσφέρεται ιδιαίτερα για τη διαχείριση και τον εμπλουτισμό RDF δεδομένων [25].



Σχήμα 2: Σχηματική αναπαράσταση της εκφραστικότητας των υπογλωσσών της OWL 1 και των προφίλ της OWL 2

Ως γλώσσα του Σημασιολογικού Ιστού, η OWL διαθέτει σύνταξη συμβατή με τις γλώσσες RDF(S) και XML. Ωστόσο, προβλέπεται επιπλέον μία μορφή σύνταξης πιο κατανοητή και

εύχρηστη για την περιγραφή οντολογιών, που ονομάζεται αφηρημένη σύνταξη (abstract syntax). Αν και η σημασιολογία είναι παρόμοια με αυτή των Περιγραφικών Λογικών, η σύνταξη και το λεξιλόγιο της OWL διαφέρουν σε κάποια σημεία. Έτσι, κατ' αναλογία με τις έννοιες και τους ρόλους των Περιγραφικών Λογικών, μία OWL οντολογία διαθέτει κλάσεις (classes) και ιδιότητες (properties), αντίστοιχα. Επίσης, παρέχει τη δυνατότητα ορισμού αξιωμάτων κλάσεων (class axioms), αξιωμάτων ιδιοτήτων (property axioms) και αξιωμάτων ατόμων (individual axioms), καθώς και γεγονότων (facts), δηλαδή σχέσεων στιγμιότυπου μεταξύ ενός ατόμου (δύο ατόμων) και μίας κλάσης (ενός ρόλου), κατ' αντιστοιχία με τους ισχυρισμούς στις Περιγραφικές Λογικές. Η αντιστοίχηση μεταξύ των δύο λεξιλογίων είναι απλή και κάποια παραδείγματα παρουσιάζονται, ενδεικτικά, στον παρακάτω πίνακα (περισσότερες αντιστοιχίες μπορούν να βρεθούν στο [26]):

Αφηρημένη σύνταξη OWL	**Σύνταξη Περιγραφικών Λογικών**
owl:Thing / owl:Nothing	$\top$ / $\bot$
intersectionOf(C_1,…,C_n)	$C_1 \sqcap \ldots \sqcap C_n$
complementOf(C)	$\neg C$
restriction(R allValuesFrom(C))	$\forall R.C$
restriction(R someValuesFrom(C))	$\exists R.C$
SubClassOf(C_1, C_2)	$C_1 \sqsubseteq C_2$
DisjointClasses(C_1, …, C_n)	$C_i \sqcap C_j \sqsubseteq \bot$, $1 \leq i < j \leq n$

Πίνακας 1: Ενδεικτικές δηλώσεις OWL και αντίστοιχες δηλώσεις Περιγραφικών Λογικών

Να σημειωθεί ότι υπάρχουν αξιώματα της OWL, όπως τα αξιώματα πεδίου ορισμού και συνόλου τιμών ρόλων, τα οποία δεν έχουν άμεσα αντίστοιχες δηλώσεις σε Περιγραφικές Λογικές, μπορούν όμως να εκφραστούν με αξιώματα υπαγωγής, αρκεί η Περιγραφική Λογική να διαθέτει την κατάλληλη εκφραστικότητα. Η παραπάνω αντιστοιχία σημαίνει ότι η συλλογιστική στην OWL μπορεί να αναχθεί σε προβλήματα συλλογιστικής Περιγραφικών Λογικών [27], πράγμα που κάνει δυνατή την αξιοποίηση εργαλείων και τεχνικών που έχουν αναπτυχθεί για την αποδοτική διεξαγωγή συλλογιστικής σε αυτές.

2.6 Η Περιγραφική Λογική DL-Lite

Η DL-Lite είναι μία Περιγραφική Λογική που προτάθηκε το 2004 [28], με σκοπό να συνδυάσει αρκετά υψηλή εκφραστικότητα με χαμηλή (πολυωνυμική) πολυπλοκότητα. Για ένα αλφάβητο ατομικών εννοιών και ρόλων, η σύνταξη της DL-Lite επιτρέπει τον ορισμό

βασικών εννοιών (basic concept) και γενικών εννοιών (general concept) σύμφωνα με τους κανόνες:

$$B \rightarrow \ A \mid \exists R \mid \exists R^{-}$$

$$C \rightarrow \ B \mid \neg B \mid C_1 \sqcap C_2$$

όπου A μία ατομική έννοια, B μία βασική έννοια, C, C_1, C_2 γενικές έννοιες, και R ένας (ατομικός) ρόλος.

Όπως και στις Περιγραφικές Λογικές που αναλύθηκαν προηγουμένως, ένα TBox της DL-Lite περιέχει αξιώματα σχετικά με τις έννοιες της γλώσσας. Συγκεκριμένα, η γλώσσα υποστηρίζει αξιώματα υπαγωγής της μορφής $B \sqsubseteq C$, όπου B μία βασική έννοια και C μία οποιαδήποτε (γενική) έννοια. Ένα ABox μπορεί να περιέχει ισχυρισμούς της μορφής B(a) και R(a,b), όπου B βασική έννοια, R ατομικός ρόλος και a, b άτομα.

Η σημασιολογία της γλώσσας ορίζεται με τρόπο παρόμοιο με αυτόν για τις Περιγραφικές Λογικές της οικογένειας $\mathcal{AL}$, όπως παρουσιάστηκε παραπάνω, με μικρές προσαρμογές για τις ιδιαιτερότητες της DL-Lite.

Εξαρχής είναι φανερό ότι η DL-Lite υπολείπεται σε εκφραστική δύναμη της OWL (ακόμα και της OWL Lite). Μεταξύ των δυνατοτήτων των οποίων στερείται είναι η άρνηση μη-βασικών εννοιών, η ένωση εννοιών και οι καθολικοί περιορισμοί, ενώ επιπλέον επιβάλλει περιορισμούς στη μορφή των αξιωμάτων υπαγωγής. Ωστόσο, ακόμα και με αυτές τις "ελλείψεις", η DL-Lite παραμένει αρκετά εκφραστική για να μπορεί να περιγράψει έννοιες που εμφανίζονται σε φορμαλισμούς της τεχνολογίας λογισμικού, όπως το μοντέλο οντοτήτων-συσχετίσεων και τα διαγράμματα κλάσεων της UML. Επιπλέον, εξαιτίας ακριβώς αυτής της μειωμένης εκφραστικότητας, η συλλογιστική στη γλώσσα παραμένει βατή, ακόμα και για υπηρεσίες πιο σύνθετες από αυτές που υποστηρίζουν οι συνήθεις Περιγραφικές Λογικές.

Η κυριότερη υπηρεσία συλλογιστικής που προσφέρεται στη DL-Lite είναι η απάντηση ερωτημάτων (query answering). Ένα συζευκτικό ερώτημα (conjuctive query) ορίζεται ως μία έκφραση της μορφής

$$q(\vec{x}) \leftarrow \exists \vec{y}.\, conj(\vec{x}, \vec{y})$$

όπου $\vec{x}$ ένα διάνυσμα μεταβλητών που καλούνται διακεκριμένες μεταβλητές, $\vec{y}$ ένα διάνυσμα μη-διακεκριμένων μεταβλητών, και $conj(\vec{x}, \vec{y})$ είναι σύζευξη εκφράσεων της μορφής B(z) ή $R(z_1,z_2)$ όπου B βασική έννοια και R ατομικός ρόλος της βάσης γνώσης, και z, z_1, z_2 άτομα της βάσης ή μεταβλητές των $\vec{x}$, $\vec{y}$. Η απάντηση ενός συζευκτικού ερωτήματος q της παραπάνω μορφής συνίσταται στην επιστροφή όλων εκείνων των πλειάδων t από άτομα οι

οποίες, όταν αντικατασταθούν στη θέση του x, ικανοποιούν $\mathcal{K} \models q(t)$. Για παράδειγμα, αν έχουμε τις ατομικές έννοιες Φοιτητής, Καθηγητής και το ρόλο έχειΚαθηγητήΤον, μπορούμε να πάρουμε όλα τα ζεύγη διδασκόντων-διδασκομένων με το ερώτημα $q(x, y) \leftarrow Καθηγητής(x),\ Φοιτητής(y),\ έχειΚαθηγητήΤον(x, y)$. Είναι σημαντικό να παρατηρήσουμε ότι η απάντηση συζευκτικών ερωτημάτων είναι πιο γενική από τις υπηρεσίες ελέγχου ικανοποιησιμότητας εννοιών και συνεπαγωγής ισχυρισμών, οι οποίες μάλιστα μπορούν εύκολα να αναχθούν σε απάντηση ενός ερωτήματος.

Η συλλογιστική στη DL-Lite δε βασίζεται σε αλγορίθμους πινάκων, αλλά ανάγει την απάντηση ενός συζευκτικού ερωτήματος σε αποτίμηση ενός ερωτήματος SQL πάνω σε μια βάση δεδομένων. Η αναγωγή αυτή είναι δυνατή χάρις στην περιορισμένη εκφραστικότητα της γλώσσας. Συνοπτικά, δεδομένης μίας βάσης γνώσης (δηλαδή, ενός TBox και ενός ABox) και ενός συζευκτικού ερωτήματος προς απάντηση, η διαδικασία είναι η εξής:

1. Κανονικοποίηση (normalization) της βάσης γνώσης: Με κατάλληλη προεπεξεργασία, κατασκευάζεται ένα νέο TBox και ένα νέο ABox, που είναι ισοδύναμα με τα αρχικά. Σκοπός είναι να φέρουμε τη βάση γνώσης σε μία μορφή που θα διευκολύνει τα επόμενα βήματα.
2. Αποθήκευση του ABox: Δημιουργείται μία βάση δεδομένων, της οποίας το σχήμα και τα περιεχόμενα προκύπτουν από και αναπαριστούν το (κανονικοποιημένο) ABox. Ορίζεται, δηλαδή, μία αντιστοίχηση από τις έννοιες και τους ρόλους της βάσης γνώσης στους πίνακες της ΒΔ.
3. Ικανοποιησιμότητα της βάσης γνώσης: Ελέγχεται, μέσω κατάλληλων ερωτημάτων SQL, το αν η βάση γνώσης είναι ικανοποιήσιμη- δηλαδή, αν μπορεί να υπάρξει κοινό μοντέλο του TBox και του ABox. Το βήμα αυτό είναι απαραίτητο, καθώς, αν η γνώση δεν είναι ικανοποιήσιμη, κάθε πλειάδα ατόμων είναι απάντηση σε οποιοδήποτε ερώτημα, και άρα η απάντηση ερωτημάτων δεν έχει νόημα.
4. Επαναδιατύπωση (reformulation) του ερωτήματος: Με βάση το δοσμένο ερώτημα, αξιοποιείται η γνώση που περιέχεται στο (κανονικοποιημένο) TBox, και κατασκευάζεται ένα σύνολο συζευκτικών ερωτημάτων.
5. Αποτίμηση των ερωτημάτων: Για καθένα από τα ερωτήματα που λαμβάνονται από το προηγούμενο βήμα, σχηματίζεται ένα ερώτημα SQL και υποβάλλεται στη βάση δεδομένων, επιστρέφοντας έτσι τις απαντήσεις του αρχικού συζευκτικού ερωτήματος.

Αποδεικνύεται ότι η παραπάνω διαδικασία είναι ορθή: μία πλειάδα ατόμων c είναι απάντηση ενός συζευκτικού ερωτήματος αν και μόνο αν ανήκει στο σύνολο των απαντήσεων που επιστρέφει ο παραπάνω αλγόριθμος. Σχετικά με την πολυπλοκότητα, μπορεί να δειχθεί ότι το

πρόβλημα της απάντησης συζευκτικών ερωτημάτων στη DL-Lite είναι NP-πλήρες όσον αφορά τη συνδυασμένη πολυπλοκότητα (combined complexity), δηλαδή, ως προς το μέγεθος της βάσης και του ερωτήματος. Ωστόσο, όσον αφορά την πολυπλοκότητα δεδομένων (data complexity), δηλαδή ως προς το μέγεθος του ABox, το πρόβλημα βρίσκεται στην κλάση πολυπλοκότητας PTIME [29].

Τέλος, αξίζει να σημειωθεί ότι η σύνταξη που αναφέρεται παραπάνω έχει αποτελέσει βάση για τη δημιουργία και άλλων Περιγραφικών Λογικών, οι οποίες επεκτείνουν τις εκφραστικές δυνατότητες της DL-Lite [30]. Τέτοιες γλώσσες είναι, μεταξύ άλλων, οι DL-Lite$_R$ (που επιτρέπει αξιώματα υπαγωγής ρόλων) και DL-Lite$_F$ (που περιλαμβάνει συναρτησιακούς ρόλους). Η οικογένεια DL-Lite έχει γνωρίσει σημαντική χρήση και αποδοχή, με αποτέλεσμα, όπως αναφέρθηκε προηγουμένως, να επιλεγεί ως βάση για το προφίλ OWL QL της OWL 2. Στη συνέχεια αυτού του εγγράφου, ως DL-Lite θα αναφέρεται η εκδοχή της γλώσσας που περιγράφηκε στην αρχή αυτής της ενότητας. Αυτή διαφέρει ελαφρώς από την περιγραφή που δόθηκε αρχικά από τους δημιουργούς της γλώσσας (στα [28], [29]), αλλά επιλέχθηκε ώστε να είναι πιο κοντά στο προφίλ OWL QL.

3

Μετασχηματισμός Οντολογιών

Στο κεφάλαιο αυτό παρουσιάζεται το πρόβλημα με το οποίο καταπιάνεται το κύριο μέρος της εργασίας, που είναι είναι το εξής: Δεδομένης μίας οντολογίας γραμμένης σε OWL DL, ζητείται να κατασκευαστεί μία σημασιολογική προσέγγισή της σε DL-Lite. Ένας αλγόριθμος για την επίλυση έχει προταθεί από τους Pan και Thomas στο [31] και θα παρουσιαστεί στη συνέχεια.

3.1 Μετασχηματισμός Γνώσης

Το πρόβλημα του μετασχηματισμού γνώσης (knowledge compilation) συνίσταται στο μετασχηματισμό δηλώσεων από μία γλώσσας αναπαράστασης σε μία άλλη, συνήθως από μία περισσότερο σε μία λιγότερο εκφραστική γλώσσα. Το όφελος της ιδέας αυτής είναι διπλό: αφενός, επιτρέπει στο χρήστη να απολαμβάνει χαμηλότερη πολυπλοκότητα συλλογιστικής, χωρίς να τον προβληματίζει η περιορισμένη εκφραστικότητα που θα είχε αν χρησιμοποιούσε κατευθείαν τη δεύτερη γλώσσα· αφετέρου, πληροφορία που υπονοούνταν στην αρχική μορφή μπορεί κατά τη διαδικασία αυτή να προστεθεί στη γνώση εκπεφρασμένα (explicitly), με αποτέλεσμα να είναι ακόμα γρηγορότερη η συλλογιστική στην προκύπτουσα γνώση [32]. Για την προσέγγιση μίας οντολογίας, υπάρχουν δύο τεχνικές: συντακτική (syntactic approximation) και σημασιολογική προσέγγιση (semantic approximation). Στην πρώτη από αυτές, ο σκοπός είναι να μετασχηματιστούν τα αξιώματα ώστε να είναι έγκυρα στη λιγότερο εκφραστική γλώσσα. Αυτό έχει τον εγγενή κίνδυνο κάποια αξιώματα να εξαφανιστούν

(καθώς δεν μπορούν να εκφραστούν στη δεύτερη γλώσσα) ενώ άλλα να αλλοιωθούν, ώστε να συμμορφωθούν με τους περιορισμούς της γλώσσας. Επομένως, η διαδικασία δεν είναι ορθή, γιατί η νέα γνώση μπορεί να συνεπάγεται προτάσεις που δε συνεπάγονται από την αρχική.

Αντίθετα, η σημασιολογική προσέγγιση εστιάζει στη δημιουργία μίας νέας οντολογίας η οποία θα περιέχει ακριβώς εκείνα τα αξιώματα και ισχυρισμούς που είναι συνέπειες της αρχικής και που μπορούν να εκφραστούν στη νέα γλώσσα. Εξ ορισμού, επομένως, η διαδικασία αυτή είναι ορθή, μπορεί όμως να μην είναι πλήρης.

Ως παράδειγμα για τη διαφορά των δύο προσεγγίσεων, έστω ότι θέλουμε να μετασχηματίσουμε τη βάση γνώσης $\mathcal{K} = (\mathcal{T},\mathcal{A})$, όπου $\mathcal{T}$ είναι το TBox {C⊑D, ∃R.C⊑B} και $\mathcal{A}$ το ABox {(a,b):R}, από την Περιγραφική Λογική ALE σε DL-Lite. Μία συντακτική προσέγγιση της K είναι η $\mathcal{K}_1$ = ({C⊑D, ∃R⊑B}, {(a,b):R}). Η $\mathcal{K}_1$ φαίνεται να μην υστερεί ιδιαίτερα της K στη γνώση που κωδικοποιεί, όμως συνεπάγεται τον ισχυρισμό a:B, που δεν αποτελεί συνεπαγωγή της $\mathcal{K}$. Είναι, επομένως, μία μη-ορθή προσέγγιση. Από την άλλη, μία σημασιολογική προσέγγιση της $\mathcal{K}$ είναι η $\mathcal{K}_2$ = ({C⊑D}, {(a,b):R}). Σε αυτή την περίπτωση, δεν υπάρχουν συνεπαγωγές της $\mathcal{K}_2$ που να μην προκύπτουν και από την $\mathcal{K}$ (δηλαδή η προσέγγιση είναι ορθή), αλλά βλέπει κανείς ότι έχει χαθεί πλήρως η πληροφορία του δεύτερου αξιώματος υπαγωγής, πράγμα που σημαίνει ότι ενδεχομένως να υπάρχουν συμπεράσματα της $\mathcal{K}$ που να μην μπορούν να εξαχθούν από την $\mathcal{K}_2$ (πρόκειται επομένως για μη-πλήρη προσέγγιση). Για να διορθωθεί το μειονέκτημα αυτό, θα μπορούσαμε να συμπεριλάβουμε στην $\mathcal{K}_2$ εκπεφρασμένα τις συνεπαγωγές της $\mathcal{K}$ που «αποκόπτονται» κατά τη διαδικασία της προσέγγισης.

Το ιδανικό θα ήταν να βρεθεί μία προσέγγιση της $\mathcal{K}$ που να είναι ταυτόχρονα ορθή και πλήρης, αλλά να μπορεί να εκφραστεί στη νέα γλώσσα αναπαράστασης. Καθώς αυτό μπορεί να μην είναι πάντα εφικτό, το πρόβλημα με το οποίο θα ασχοληθούμε στη συνέχεια είναι πώς μπορεί να κατασκευαστεί μία ορθή (σημασιολογική) προσέγγιση, η οποία να περιέχει όσο το δυνατόν περισσότερες από τις συνέπειες της αρχικής γνώσης γίνεται (εξασφαλίζοντας, δηλαδή, μία όσο το δυνατόν πληρέστερη προσέγγιση).

3.2 «Αφελής» προσέγγιση

Αρχικά, ακολουθούν κάποιοι ορισμοί για τη σαφέστερη διατύπωση του προβλήματος. Έστω μία αρχική οντολογία O_1, γραμμένη σε μία γλώσσα L_1. Συμβολίζουμε με N_C, N_P και N_I το σύνολο των ατομικών εννοιών, ρόλων και τον ατόμων, αντίστοιχα, της O_1. Εφόσον η O_1 είναι πεπερασμένη, και τα τρία σύνολα είναι επίσης προφανώς πεπερασμένα.. Έστω επίσης μία γλώσσα L_2 (γλώσσα-στόχος). Το σύνολο αξιωμάτων (axiom set) της L_2 ως προς τα N_C, N_P, N_I ορίζεται ως το σύνολο όλων των αξιωμάτων που μπορούν να εκφραστούν στη γλώσσα L_2

χρησιμοποιώντας τα στοιχεία των N_C, N_P, N_I, και συμβολίζεται με $AS(L_2,N_C,N_P,N_I)$. Το σύνολο συνεπαγωγών (entailment set) της O_1 ως προς την L_2 είναι το σύνολο όλων των αξιωμάτων που μπορούν να εκφραστούν στην L_2 (χρησιμοποιώντας τα στοιχεία των N_C, N_P, N_I) και τα οποία συνεπάγονται από την O_1, συμβολίζεται δε ως $ES(L_2,O_1)$. Είναι προφανές ότι $ES(L_2,O_1) \subseteq AS(L_2,N_C,N_P,N_I)$.

Η αρχική ιδέα του αλγορίθμου των Pan – Thomas αποτελείται από δύο βήματα: Πρώτον, αφού η μορφή των αξιωμάτων στη DL-Lite είναι δεδομένη και απλή, μπορούμε να κατασκευάσουμε όλο το $AS(DL\text{-}Lite,N_C,N_P,N_I)$. Στη συνέχεια, μπορούμε να ελέγξουμε ποια μέλη του συνόλου αυτού συνεπάγονται από την O_1, και αυτά θα αποτελέσουν το $ES(DL\text{-}Lite,O_1)$, που θα είναι και η σημασιολογική προσέγγιση της αρχικής οντολογίας. Ο αλγόριθμος περιγράφεται περιληπτικά στα επόμενα σχήματα. Να σημειωθεί ότι ο έλεγχος της συνεπαγωγής πρέπει να γίνει με κάποιον reasoner που να υποστηρίζει την εκφραστικότητα της αρχικής γλώσσας (στην προκειμένη περίπτωση, OWL DL) και όχι απλώς της DL-Lite.

Αλγόριθμος 1: Σύνολο αξιωμάτων
Είσοδοι: N_C, N_P, N_I
Έξοδος: $AS(DL\text{-}Lite,N_C,N_P,N_I)$

```
AS := ∅
//κατασκευή συνόλου βασικών εννοιών (BS)
BS := N_C
for R ∈ N_P:
        BS := BS ∪ {∃R, ∃R⁻}
//κατασκευή συνόλου γενικών εννοιών (CS)
CS := BS
for B ∈ BS:
        CS := CS ∪ {¬B}
//κατασκευή αξιωμάτων εννοιών
for B ∈ BS:
        for C ∈ CS:
                AS = AS ∪ {B⊑C}
//κατασκευή ισχυρισμών εννοιών
for B ∈ BS:
        for a ∈ N_I:
                AS = AS ∪ {C(a)}
//κατασκευή ισχυρισμών ρόλων
for R ∈ N_P:
        for a ∈ N_I:
                for b ∈ N_I:
                        AS = AS ∪ {R(a,b)}
return AS
```

Σχήμα 3: Αλγόριθμος για τον υπολογισμό του συνόλου αξιωμάτων

Αλγόριθμος 2: Σύνολο συνεπαγωγών
Είσοδοι: O_1, N_C, N_P, N_I
Έξοδος: ES(O_1,DL-Lite)

ES := ∅
AS := AS(DL-Lite,N_C,N_P,N_I)
//*«φιλτράρισμα» των αξιωμάτων του συνόλου αξιωμάτων (AS)*
for a ∈ AS:
    if (O_1 ⊨ a) **then**:
        ES := ES ∪ {a}
return ES

Σχήμα 4: Αφελής αλγόριθμος για τον υπολογισμό του συνόλου συνεπαγωγών

3.3 Βελτιστοποιήσεις

Το μειονέκτημα της παραπάνω προσέγγισης είναι η πολυπλοκότητά της, καθώς στο πρώτο βήμα πρέπει να υπολογιστεί όλο το AS(DL-Lite,N_C,N_P,N_I), επομένως μπορεί να παράγεται μεγάλος αριθμός αξιωμάτων προς έλεγχο, από τα οποία ενδέχεται μόνο ένα μικρό μέρος να γίνουν αποδεκτά στη συνέχεια ως μέλη του συνόλου συνεπαγωγών ES(DL-Lite,O_1). Για την ακρίβεια, αν $|N_C|$, $|N_P|$. $|N_I|$ το πλήθος των κλάσεων, ιδιοτήτων και ατόμων της O_1, αντίστοιχα, τότε το AS(DL-Lite,N_C,N_P,N_I) θα αποτελείται από $2(|N_C|+2|N_P|)^2$.αξιώματα εννοιών και $2(|N_C|+2|N_P|)|N_I| + |N_P||N_I|^2$ ισχυρισμούς. Το πλήθος αυτό είναι μεν πολυωνυμικό ως προς τα $|N_C|$, $|N_P|$. $|N_I|$, αλλά μπορεί να είναι σημαντικά μεγαλύτερο του πλήθους των αξιωμάτων που πραγματικά χρειάζεται να παραχθούν.

Για αυτό το λόγο, οι Pan - Thomas πρότειναν μία πιο αποδοτική λύση, η οποία αντιμετωπίζει με διαφορετικό τρόπο τα διάφορα είδη αξιωμάτων και αξιοποιεί τη δομή και τη σημασιολογία της αρχικής οντολογίας για να μειώσει το χρόνο εκτέλεσης. Συγκεκριμένα, αφού κατασκευαστεί το σύνολο των βασικών και των γενικών κλάσεων, ακολουθούνται διαφορετικές διαδικασίες για τα αξιώματα υπαγωγής εννοιών, τους ισχυρισμούς εννοιών και τους ισχυρισμούς ρόλων. Κάθε μία από αυτές τις διαδικασίες παράγει ένα σύνολο αξιωμάτων, η ένωση των οποίων θα αποτελέσει την τελική οντολογία.

3.3.1 Αξιώματα υπαγωγής εννοιών

Για να επιταχυνθεί η διαδικασία του ελέγχου της υπαγωγής εννοιών, χρησιμοποιείται η υπηρεσία ταξινόμησης (classification). Για να κάνουμε χρήση αυτής, επεκτείνεται η αρχική οντολογία με τον εξής τρόπο: Για κάθε γενική κλάση C, προσθέτουμε τον ορισμό $A_C \equiv C$. Αυτή η προσθήκη εισάγει απλώς νέα ονόματα εννοιών, χωρίς να επηρεάζει τη σημασιολογία

της οντολογίας. Μετά από αυτή τη διαδικασία, ένα αξίωμα υπαγωγής $C \sqsubseteq D$ ισχύει αν και μόνο αν $A_C \sqsubseteq A_D$. Τα αξιώματα της τελευταίας μορφής, όμως, μπορούν να ελεγχθούν εύκολα ταξινομώντας την αρχική οντολογία.

Η διαδικασία που ακολουθείται δεν είναι σημαντικά διαφορετική από αυτή της «αφελούς» έκδοσης. Το πλεονέκτημα αυτής της βελτιστοποίησης είναι ότι, μετά την ταξινόμηση της οντολογίας, η υπαγωγή εννοιών μπορεί να ελεγχθεί πιο γρήγορα και αποδοτικά απ' ό,τι προηγουμένως, με αποτέλεσμα να βελτιώνεται ο χρόνος εκτέλεσης του αλγορίθμου. Επίσης, χάρη στην προσθήκη των αξιωμάτων ισοδυναμίας, στην ταξινόμηση της οντολογίας εμφανίζονται ουσιαστικά όλες οι γενικές κλάσεις και όχι μόνο οι ατομικές, επιτρέποντας τη γρήγορη επαλήθευση αξιωμάτων υπαγωγής στα οποία συμμετέχουν ακόμα και ανώνυμες περιγραφές.

3.3.2 Ισχυρισμοί εννοιών

Ο έλεγχος των ισχυρισμών εννοιών μπορεί να γίνει πιο αποδοτικός αν ληφθεί υπόψιν η ιεραρχία των εννοιών της αρχικής οντολογίας. Η ιδέα της βελτιστοποίησης είναι η εξής: Αν ένα άτομο a είναι στιγμιότυπο μίας έννοιας A, τότε είναι στιγμιότυπο και όλων των εννοιών N στις οποίες υπάγεται (άμεσα ή έμμεσα) η A. Κατ' αυτόν τον τρόπο, ξεκινώντας από τα φύλλα της ιεραρχίας εννοιών (έννοιες που δεν έχουν υποέννοιες), μπορούμε να αποφύγουμε τον έλεγχο ισχυρισμών που ξέρουμε σίγουρα ότι θα ισχύουν και να ελέγξουμε μόνο εκείνους τους ισχυρισμούς που πραγματικά χρειάζονται έλεγχο. Η διαδικασία αυτή λαμβάνει χώρα μετά την ταξινόμηση της οντολογίας, καθώς χρησιμοποιεί πληροφορίες για την ιεραρχία των κλάσεων.

Ο αλγόριθμος χρησιμοποιεί μία ουρά, στην οποία περιέχονται οι κλάσεις που πρόκειται να ελεγχθούν, και ένα σύνολο για τις κλάσεις που έχουν ήδη ελεγχθεί. Επιστρέφει ένα σύνολο από ισχυρισμούς εννοιών. Για να βρεθούν οι ισχυρισμοί που ισχύουν για κάποιο άτομο o, ακολουθείται η ακόλουθη διαδικασία: Η ουρά αρχικοποιείται με τα φύλλα της ιεραρχίας (τις κλάσεις που δεν έχουν άλλες υποκλάσεις εκτός από τις τετριμμένες, δηλαδή τον εαυτό τους και το owl:Nothing ή $\bot$), και το σύνολο των ελεγμένων κλάσεων τίθεται ίσο με το κενό. Όσο η ουρά δεν είναι άδεια, εξάγεται η κεφαλή της, έστω A. Αν η A έχει ήδη ελεγχθεί, εξάγουμε το επόμενο στοιχείο από την ουρά και επαναλαμβάνουμε τη διαδικασία. Αν η A δεν έχει ελεγχθεί, ελέγχουμε τον ισχυρισμό o:A. Στο σημείο αυτό, ο αλγόριθμος διαφοροποιείται ανάλογα με τον ο ισχυρισμός αποτελεί συνέπεια της οντολογίας η όχι: Αν ναι, τότε προσθέτουμε τον ισχυρισμό στο σύνολο που επιστρέφει ο αλγόριθμος, μαζί με όλους τους ισχυρισμούς o:N, όπου N υπερκλάση της A. Σημειώνουμε επίσης την A και τις υπερκλάσεις της ως ελεγμένες. Αν όχι, σημειώνουμε την A ως ελεγμένη, και προσθέτουμε στο τέλος της ουράς τις μη-ελεγμένες υπερκλάσεις της A, για να ελεγχθούν αργότερα. Όταν η ουρά

αδειάσει, έχουν υπολογιστεί όλοι οι ισχυρισμοί εννοιών για το άτομο ο. Η διαδικασία αυτή επαναλαμβάνεται για όλα τα άτομα της οντολογίας, ώστε να ληφθεί το σύνολο των ισχυρισμών εννοιών.

Ακολουθεί ένα παράδειγμα για την επεξήγηση της λειτουργίας του αλγορίθμου και τη σύγκρισή του με την «αφελή» υλοποίηση. Θεωρούμε την οντολογία που αποτελείται από τα αξιώματα

Κόκκινο ⊑ Κρασί Λευκό ⊑ Κρασί

Κρασί ⊑ Ποτό Μπύρα ⊑ Ποτό

Φαΐ ⊓ Ποτό ⊑ ⊥ Κρασί ⊓ Μπύρα ⊑ ⊥

Κόκκινο ⊓ Λευκό ⊑ ⊥

και τους ισχυρισμούς

κοτόπουλο : Φαΐ

μοσχοφίλερο: Λευκό

«Αφελής» υλοποίηση: Σχηματίζονται και ελέγχονται όλοι οι δυνατοί ισχυρισμοί. Υπάρχουν 2 άτομα και 6 κλάσεις, επομένως γίνονται 12 έλεγχοι ισχυρισμών.

Βελτιστοποίηση: Η ταξινομημένη οντολογία φαίνεται στο επόμενο σχήμα.

Σχήμα 5: Ταξινομημένη οντολογία για τα παραδείγματα ισχυρισμών εννοιών

Τα φύλλα της είναι οι κλάσεις Φαΐ, Μπύρα, Κόκκινο και Λευκό. Εφαρμόζεται η διαδικασία που περιγράφηκε παραπάνω για κάθε άτομο.

κοτόπουλο:

Αρχικοποιείται η ουρά με τις κλάσεις-φύλλα. Οι ισχυρισμοί που ελέγχονται έχουν ως εξής:

κοτόπουλο:Φαΐ: αληθής. Η κλάση Φαΐ δεν έχει υπερκλάσεις, επομένως δε γίνονται άλλες ενέργειες εκτός από το να σημειωθεί ως ελεγμένη (η πράξη της επισήμανσης ως ελεγμένης της κλάσης που ελέγχεται κάθε φορά θα θεωρείται ότι υπονοείται εφ εξής).

κοτόπουλο:Μπύρα: ψεδής. Προστίθεται στην ουρά η κλάση Ποτό, ως υπερκλάση της κλάσης Μπύρα.

κοτόπουλο:Κόκκινο: ψευδής. Προστίθεται στην ουρά η κλάση Κρασί.

κοτόπουλο:Λευκό: ψευδής. Προστίθεται στην ουρά η κλάση Κρασί.

κοτόπουλο:Ποτό: ψευδής. Δεν έχει υπερκλάσεις, επομένως δεν προστίθεται τίποτα στην ουρά.

κοτόπουλο:Κρασί: ψευδής. Η υπερκλάση της, Ποτό, είναι ήδη ελεγμένη, επομένως δεν προστίθεται πάλι στην ουρά

κοτόπουλο:Κρασί: Η κλάση Κρασί είναι ήδη ελεγμένη, επομένως δε γίνεται τίποτα.

Στο σημείο αυτό η εκτέλεση του αλγορίθμου για το άτομο κοτόπουλο ολοκληρώνεται, καθώς η ουρά έχει αδειάσει.

μοσχοφίλερο:

Μετά τις αρχικοποιήσεις, ελέγχονται οι ακόλουθοι ισχυρισμοί:

μοσχοφίλερο:Φαΐ: ψευδής. Δεν υπάρχουν υπερκλάσεις, δε γίνονται άλλες ενέργειες.

μοσχοφίλερο:Μπύρα: ψευδής. Προστίθεται στην ουρά η κλάση Ποτό.

μοσχοφίλερο:Κόκκινο: ψευδής. Προστίθεται στην ουρά η κλάση Κρασί.

μοσχοφίλερο:Λευκό: αληθής. Προστίθενται στο σύνολο που θα επιστραφεί οι ισχυρισμοί {μοσχοφίλερο:Λευκό, μοσχοφίλερο:Κρασί, μοσχοφίλερο:Ποτό}, και σημειώνονται όλες αυτές οι κλάσεις ως ελεγμένες.

μοσχοφίλερο:Ποτό: η κλάση Ποτό είναι ελεγμένη, επομένως δε γίνεται τίποτα.

μοσχοφίλερο:Κρασί: η κλάση Κρασί είναι ελεγμένη, επομένως δε γίνεται τίποτα.

Στο σημείο αυτό, η ουρά έχει αδειάσει και ο αλγόριθμος σταματάει.

Συνολικά, ελέγχθηκαν 6 + 4 = 10 ισχυρισμοί, έναντι 12 με την «αφελή» υλοποίηση. Η διαφορά μπορεί να μην είναι ιδιαίτερη εντυπωσιακή, όμως πρέπει να σημειωθεί ότι πρόκειται για μία μικρή οντολογία, με λίγες σχετικά σχέσεις υπαγωγής, στην οποία η βελτιστοποίηση δεν έχει την καλύτερη απόδοση.

Η απόδοση του αλγορίθμου θα είναι βέλτιστη όταν όλες οι κλάσεις σχηματίζουν μία αλυσίδα, με την κάθε μία να είναι υποκλάση της επόμενης. Σε αυτή την περίπτωση θα υπάρχει πάντα μία κλάση στην ουρά, και η διαδικασία θα τελειώσει όταν βρεθεί ένας ισχυρισμός που είναι αληθής. Γενικότερα, όσο λιγότερο συσχετισμένες είναι μεταξύ τους οι κλάσεις, τόσο λιγότερο αποδοτικός είναι ο αλγόριθμος, γιατί δεν μπορεί να εκμεταλλευτεί πολλές σχέσεις υπαγωγής για να εξοικονομήσει ελέγχους. Επίσης, ο αλγόριθμος αποδίδει καλύτερα όταν οι ισχυρισμοί αφορούν κλάσεις που βρίσκονται σχετικά χαμηλά στην ιεραρχία (ιδανικά, όταν είναι φύλλα). Αυτό μπορεί να γίνει κατανοητό αν σκεφτεί κανείς ότι όσο νωρίτερα βρεθεί ένας αληθής ισχυρισμός, τόσο λιγότεροι έλεγχοι θα γίνουν στο κομμάτι της ιεραρχίας στο οποίο ανήκει η κλάση στην οποία αναφέρεται ο ισχυρισμός.

Ένα πλεονέκτημα του αλγορίθμου σε σχέση με την «αφελή» υλοποίηση είναι ότι δεν είναι τόσο ευαίσθητος στην προσθήκη νέων κλάσεων. Για παράδειγμα, αν προστεθεί μία νέα κλάση Εδώδιμο στην κορυφή της ιεραρχίας, δηλαδή ως υπερκλάση των Φαΐ και Ποτό, με την «αφελή» υλοποίηση θα πρέπει να ελεγχθεί ένας επιπλέον ισχυρισμός για κάθε άτομο, αφού δοκιμάζονται όλοι οι συνδυασμοί. Ωστόσο, με τη βελτιστοποίηση αυτή, δε θα υπάρξει κανένας νέος έλεγχος, γιατί οι νέοι ισχυρισμοί που συνεπάγονται θα ληφθούν έτσι κι αλλιώς πριν η κλάση Εδώδιμο ελεγχθεί. Αντίθετα, αν η νέα κλάση είναι φύλλο της ιεραρχίας, θα ελεγχθούν αναγκαστικά και νέοι ισχυρισμοί που την περιέχουν (ένας για κάθε άτομο, όπως και στην «αφελή» υλοποίηση).

3.3.3 Ισχυρισμοί ρόλων

Για κάθε ρόλο, αντί να ελέγχουμε όλους τους πιθανούς συνδυασμούς ατόμων, μπορούμε να εντοπίσουμε ποια άτομα συμμετέχουν στο ρόλο αυτό, περιορίζοντας έτσι τους συνδυασμούς που θα ελεγχθούν. Συγκεκριμένα, αν ισχύει ο ισχυρισμός ρόλου R(a,b), τότε θα ισχύουν και οι ισχυρισμοί εννοιών a: $\exists R$ και b: $\exists R^-$. Επομένως, τα μόνα άτομα που μπορούν να συμμετέχουν στο ρόλο R στην πρώτη ή τη δεύτερη θέση είναι αυτά που είναι στιγμιότυπα των κλάσεων $\exists R$ ή $\exists R^-$, αντίστοιχα.

Ο αλγόριθμος έχει ως εξής: Για κάθε ρόλο R, σχηματίζουμε δύο σύνολα candidate-1, candidate-2 από τα στιγμιότυπα των εννοιών $\exists R$ και $\exists R^-$, αντίστοιχα. Στη συνέχεια, ελέγχουμε μόνο τους ισχυρισμούς της μορφής (a,b):R, όπου a ∈ candidate-1 και b ∈ candidate-2. Το παραπάνω επαναλαμβάνεται για όλες τις ιδιότητες της οντολογίας.

Ως παράδειγμα, θεωρούμε μία οντολογία που περιγράφει το παρακάτω οικογενειακό δέντρο:

Σχημα 6: Οικογενειακό δέντρο για τα παραδείγματα ισχυρισμών ρόλων

Η οντολογία αποτελείται από τους ισχυρισμούς

έχειΠατέρα(νίκος,δημήτρης) έχειΠατέρα(μαργαρίτα,ματθαίος)

έχειΠατέρα(γιώργος, ματθαίος) έχειΠατέρα(μαρία, ματθαίος)

έχειΜητέρα(νίκος,μαργαρίτα) έχειΜητέρα(μαργαρίτα,βασιλική)

έχειΜητέρα(γιώργος, βασιλική) έχειΜητέρα(μαρία, βασιλική)

και τα αξιώματα υπαγωγής ρόλων

$$\text{έχειΜητέρα} \sqsubseteq \text{έχειΓονιό}$$

$$\text{έχειΠατέρα} \sqsubseteq \text{έχειΓονιό}$$

Θα μελετήσουμε τη συμπεριφορά της «αφελούς» υλοποίησης και της παραπάνω βελτιστοποίησης, περιγράφοντας το ποιοι ισχυρισμοί παράγονται και ελέγχονται.

«Αφελής» υλοποίηση: Για κάθε ρόλο, δοκιμάζονται οι συνδυασμοί όλων των ατόμων μεταξύ τους. Για παράδειγμα, για το ρόλο έχειΓονιό, θα δοκιμαστούν οι ισχυρισμοί έχειΓονιό(νίκος,νίκος), έχειΓονιό(νίκος,δημήτρης), έχειΓονιό(νίκος,μαργαρίτα), έχειΓονιό(νίκος,γιώργος), έχειΓονιό(νίκος,μαρία), έχειΓονιό(νίκος,βασιλική), έχειΓονιό(νίκος,ματθαίος), έχειΓονιό(δημήτρης,νίκος), έχειΓονιό(δημήτρης,δημήτρης) κ.ο.κ. Υπάρχουν 7 άτομα και 3 ρόλοι, επομένως παράγονται και δοκιμάζονται 49 ισχυρισμοί ανά ρόλο, δηλαδή συνολικά 149 ισχυρισμοί.

Βελτιστοποίηση: Ο κάθε ρόλος διαχειρίζεται διαφορετικά, βρίσκοντας πρώτα τα άτομα που συμμετέχουν σε αυτόν. Έχουμε:

έχειΓονιό:

candidate-1 = στιγμιότυπα της ∃έχειΓονιό = {νίκος, δημήτρης, μαργαρίτα, γιώργος, μαρία}, #candidate-1 = 5

candidate-2 = στιγμιότυπα της ∃έχειΓονιό$^-$ = {δημήτρης, μαργαρίτα, βασιλική, ματθαίος}, #candidate-2 = 4

Δοκιμάζονται μόνο συνδυασμοί μεταξύ των μελών του candidate-1 και των μελών του candidate-2, που είναι 20 το πλήθος.

έχειΠατέρα:

candidate-1 = στιγμιότυπα της ∃έχειΠατέρα = {νίκος, μαργαρίτα, γιώργος, μαρία}, #candidate-1 = 4

candidate-2 = στιγμιότυπα της ∃έχειΠατέρα$^-$ = {δημήτρης, ματθαίος}, #candidate-2 = 2

Με την ίδια λογική όπως και στην προηγούμενη περίπτωση, το πλήθος των συνδυασμών είναι 8.

έχειΜητέρα:

candidate-1 = στιγμιότυπα της ∃έχειΜητέρα = {νίκος, μαργαρίτα, γιώργος, μαρία}, #candidate-1 = 4

candidate-2 = στιγμιότυπα της ∃έχειΜητέρα⁻ = {μαργαρίτα, βασιλική}, #candidate-2 = 2

Όπως παραπάνω, το πλήθος των συνδυασμών είναι 8.

Στο σύνολο, επομένως, δοκιμάζονται 20 + 8 + 8 = 36 ισχυρισμοί, έναντι 147 με την «αφελή» υλοποίηση.

Συμπληρωματικά αναφέρεται ότι το πλήθος των ισχυρισμών που τελικώς ισχύουν είναι 16. Αυτό σημαίνει ότι η αφελής υλοποίηση έχει «επιτυχία» 16/147, δηλαδή περίπου 10.9% , σε αντίθεση με τη βελτιωμένη έκδοση, που έχει 16/36 = 44.4%.

Είναι ευνόητο ότι η συνεισφορά της βελτιστοποίησης εξαρτάται από το πόσα άτομα συμμετέχουν στους διάφορους ρόλους. Σε μία οντολογία με «πυκνούς» ισχυρισμούς ρόλων, όπου δηλαδή μεγάλο μέρος των ατόμων συμμετέχει σε κάθε ρόλο, η βελτιστοποίηση δε θα μειώσει ιδιαίτερα το χρόνο εκτέλεσης, καθώς το μέγεθος των συνόλων candidate-{1,2}, και άρα το πλήθος των ισχυρισμών που ελέγχονται, θα είναι μεγάλο. Αντίθετα, αν υπάρχουν σχετικά λίγα άτομα που συμμετέχουν σε ρόλους, ο αλγόριθμος θα βοηθήσει σημαντικά γιατί δε θα παραχθούν πολλοί ισχυρισμοί προς έλεγχο.

3.3.4 Επιπλέον Βελτιστοποιήσεις

Στο πλαίσιο των παραπάνω αλλαγών, οι οποίες παρουσιάζονται από τους Pan, Thomas, μπορούν να πραγματοποιηθούν και κάποιες άλλες παρεμβάσεις για να βελτιώσουν περισσότερο την απόδοση του αλγορίθμου. Συγκεκριμένα, εστιάζουμε στα εξής δύο σημεία:

Ισχυρισμοί ρόλων: Μπορούμε να εκμεταλλευτούμε τις ιδιότητες των συναρτησιακών ρόλων για να περιορίσουμε περαιτέρω την αναζήτηση των ισχυρισμών ρόλων που ισχύουν. Ένας ρόλος R καλείται συναρτησιακός (functional) όταν για οποιαδήποτε άτομα a, b, c ισχύει ότι:

$$R(a,b) \wedge R(b,c) \Rightarrow b = c$$

ή, αλλιώς,

$$\top \sqsubseteq \leq 1R.$$

Αν ένας ρόλος R είναι συναρτησιακός και ο ισχυρισμός R(a,b) έχει προστεθεί στο σύνολο συνεπαγωγών, τότε δεν μπορεί να υπάρξει κανένα άλλο άτομο $c \neq b$ για το οποίο να ισχύει ο ισχυρισμός R(a,c) (εφόσον ισχύει η υπόθεση μοναδικότητας των ονομάτων), και επομένως η αναζήτηση μπορεί να σταματήσει για το συγκεκριμένο άτομο. Η μόνη επιβάρυνση που επιφέρει η νέα αυτή έκδοση του αλγορίθμου είναι η εύρεση του συνόλου των συναρτησιακών ρόλων, που όμως μπορεί να γίνει απλά διατρέχοντας τους ρόλους της οντολογίας και ελέγχοντας απλώς αν είναι δηλωμένοι ως συναρτησιακοί.

Αλγόριθμος 3: Ισχυρισμοί ρόλων

Είσοδοι: O_1, N_P, N_I

Έξοδοι: Το υποσύνολο των ισχυρισμών ρόλων του ES(O_1,DL-Lite)

PA := ∅
functional := functional-properties(O_1) //*συναρτησιακοί ρόλοι*
for R ∈ NP:
    candidate-1 := instance-of(∃R) //*πιθανά μέλη για την πρώτη θέση του R*
    candidate-2 := instance-of(∃R^-) //*πιθανά μέλη για τη δεύτερη θέση του R*
    for a ∈ candidate-1:
        for b ∈ candidate-2:
            if (O_1 ⊨ R(a,b)) **then**:
                PA := PA ∪ {R(a,b), R^-(b,a)}
                if (R ∈ functional) **then**:
                    break
return PA

Σχήμα 7: Βελτιωμένος αλγόριθμος για τον υπολογισμό των ισχυρισμών ρόλων

Ως παράδειγμα της συνεισφοράς της νέας βελτιστοποίησης, θεωρούμε πάλι την οντολογία του οικογενειακού δέντρου που χρησιμοποιήθηκε προηγουμένως. Τα σύνολα και οι ισχυρισμοί που σχηματίζονται για κάθε ρόλο είναι οι ακόλουθοι: (θεωρούμε ότι τα στοιχεία ενός συνόλου διατρέχονται κατά τη σειρά με την οποία γράφονται)

έχειΓονιό: όπως ακριβώς και με την προηγούμενη έκδοση, αφού δεν είναι συναρτησιακός ρόλος. Ελέγχονται 20 ισχυρισμοί.

έχειΠατέρα: Τα σύνολα είναι όπως και παραπάνω:

candidate-1 = {νίκος, μαργαρίτα, γιώργος, μαρία}

candidate-2 = {δημήτρης, ματθαίος}

Αρχικά ελέγχεται ο ισχυρισμός έχειΠατέρα(νίκος,δημήτρης), που είναι αληθής. Επειδή ο ρόλος έχειΠατέρα είναι συναρτησιακός, δεν ελέγχονται άλλοι ισχυρισμοί με το άτομο νίκος στην πρώτη θέση, γιατί θα είναι σίγουρα ψευδείς. Επομένως, ο έλεγχος παρακάμπτει τον ισχυρισμό έχειΠατέρα(νίκος,ματθαίος) και συνεχίζει με τους ισχυρισμούς έχειΠατέρα(μαργαρίτα,δημήτρης) (ψευδής) και έχειΠατέρα(μαργαρίτα,ματθαίος) (αληθής). Εύκολα διαπιστώνεται ότι ελέγχονται 7 ισχυρισμοί.

Μία σημαντική παρατήρηση που πρέπει να γίνει σε αυτό το σημείο είναι ότι, σε αντίθεση με την προηγούμενη έκδοση του αλγορίθμου, η επιπλέον βελτιστοποίηση εξαρτάται από τη σειρά με την οποία διατρέχονται τα στοιχεία του συνόλου candidate-2. Αν προσπελαζόταν πρώτα το άτομο ματθαίος, θα γίνονταν λιγότεροι έλεγχοι, και συγκεκριμένα 5. Ωστόσο, σε ένα σύνολο δεν υπάρχει συγκεκριμένη σειρά διάσχισης, και άρα δεν γίνεται να εγγυηθεί κάτι τέτοιο, εκτός και αν χρησιμοποιηθεί άλλη δομή δεδομένων.

έχειΜητέρα: Όπως και με το ρόλο έχειΠατέρα, ελέγχονται 5 ή 7 ισχυρισμοί, ανάλογα με τη σειρά διάσχισης του συνόλου candidate-2.

Στην καλύτερη περίπτωση, συνολικά παράγονται 20 + 5 + 5 = 30 ισχυρισμοί, αντί για 36 με την απλή βελτιστοποίηση. Από αυτούς, οι 16 είναι σωστοί, δηλαδή ο αλγόριθμος έχει επιτυχία περίπου 53.3%. Παραπάνω βρέθηκε ότι η προηγούμενη έκδοση είχε επιτυχία 44.4%.

Εκτός από τους παράγοντες που επηρεάζουν την απόδοση της απλής βελτιστοποίησης (δηλαδή το πόσο «πυκνή» είναι η συμμετοχή των ατόμων στους ρόλους), και την (άγνωστη εκ των προτέρων) σειρά διάσχισης των συνόλων, είναι προφανές ότι η απόδοση αυτής της βελτιστοποίησης εξαρτάται άμεσα από το ποσοστό των ρόλων που είναι συναρτησιακοί. Συγκεκριμένα, έχουμε βέλτιστη απόδοση όταν η οντολογία περιέχει μόνο συναρτησιακούς ρόλους, ενώ, στην περίπτωση που κανένας ρόλος δεν είναι συναρτησιακός, η νέα βελτιστοποίηση δεν προσφέρει τίποτα σε σχέση με την προηγούμενη.

Για παράδειγμα, αν στην προηγούμενη οντολογία αγνοήσουμε το ρόλο έχειΓονιό, που δεν είναι συναρτησιακός, και επικεντρωθούμε μόνο στους άλλους δύο, η κατάσταση μεταβάλλεται ως εξής: Η προηγούμενη έκδοση ελέγχει 16 ισχυρισμούς, εκ των οποίων 8 είναι σωστοί (επιτυχία 50%), ενώ η νέα έκδοση παράγει, στην καλύτερη περίπτωση, 10 ισχυρισμούς, με επιτυχία 80%.

Ισχυρισμοί εννοιών: Μπορούμε να εκμεταλλευτούμε την ιδιότητα των ξένων κλάσεων (disjoint classes). Αν ένα άτομο a ανήκει σε μία κλάση C, τότε δεν μπορεί να ανήκει σε καμία κλάση η οποία είναι ξένη με την C. Επομένως, όταν προστίθεται στο σύνολο συνεπαγωγών ένας ισχυρισμός a:C, τότε μπορούμε να αποκλείσουμε όλες τις ξένες κλάσεις της C ως πιθανές κλάσεις του a.

Αλγόριθμος 4: Ισχυρισμοί εννοιών
Είσοδοι: ιεραρχία εννοιών H της O_1, σύνολο γενικών εννοιών CS, N_I
Έξοδοι: Το υποσύνολο των ισχυρισμών εννοιών του ES(O_1,DL-Lite)

```
CA := ∅
for a ∈ N_I:
        initialize(queue) //δημιουργία κενής ουράς
        checked := ∅
        push(queue, leaf-nodes(H)) //εκκίνηση από τα φύλλα της ιεραρχίας
        while (A := pop(queue)) :
                if A ∉ checked then:
                        if (O_1 ⊨ A(a)) then:
                                CA := CA ∪ {A(a)}
                                ancestors := superclasses(H,A)  //προσθήκη ισχυρισμών
                                for N ∈ ancestors:              //για υπερκλάσεις
```

```
                    CA := CA ∪ {N(a)}
                    checked := checked ∪ {N}
            disjoint := disjoint-classes(O1,A)
            for N ∈ disjoint:  //αποφυγή ελέγχου ξένων κλάσεων
                    checked := checked ∪ {N}
    else:
            push(queue, parents(H,A))
return CA
```

Σχήμα 8: Βελτιωμένος αλγόριθμος για τον υπολογισμό των ισχυρισμών εννοιών

Για την αξιολόγηση της βελτιστοποίησης αυτής, επανερχόμαστε στο παράδειγμα της οντολογίας των φαγητών και ποτών. Κατά την εκτέλεση της προηγούμενης έκδοσης του αλγορίθμου, δε λήφθηκε καθόλου υπόψη το γεγονός ότι οι κλάσεις Φαΐ και Ποτό είναι ξένες, με αποτέλεσμα να ελέγχονται αρκετοί ψευδείς ισχυρισμοί που θα μπορούσαν να έχουν απορριφθεί εξαρχής. Επαναλαμβάνουμε την εκτέλεση του αλγορίθμου, ενσωματώνοντας τη νέα βελτιστοποίηση του αλγορίθμου 4.

κοτόπουλο:

κοτόπουλο:Φαΐ: αληθής. Όλες οι κλάσεις που είναι ξένες με την κλάση Φαΐ σημειώνονται ως ελεγμένες.

κοτόπουλο:Μπύρα: η κλάση Μπύρα είναι ελεγμένη, άρα δε γίνεται τίποτα

κοτόπουλο:Κόκκινο: η κλάση Κόκκινο είναι ελεγμένη, άρα δε γίνεται τίποτα

κοτόπουλο:Λευκό: η κλάση Λευκό είναι ελεγμένη, άρα δε γίνεται τίποτα

Εδώ τελειώνει η εκτέλεση για το άτομο κοτόπουλο.

μοσχοφίλερο:

Όπως στην προηγούμενη έκδοση, ελέγχονται 4 ισχυρισμοί.

Συνολικά, ελέγχονται 1 + 4 = 5 ισχυρισμοί, δηλαδή οι μισοί απ' ό,τι με την προηγούμενη βελτιστοποίηση.

Πρέπει να σημειωθεί ότι, όπως και στον αλγόριθμο 3, έτσι και σε αυτή την περίπτωση έχει σημασία η σειρά διάσχισης των συνόλων. Εδώ, για παράδειγμα, αν ελεγχόταν πρώτα ο ισχυρισμός μοσχοφίλερο:Λευκό, θα ήταν και ο μόνος που θα ελεγχόταν. Έτσι, στην καλύτερη περίπτωση θα γίνονταν μόνο δύο έλεγχοι, που είναι και ο ελάχιστος δυνατός αριθμός, προκειμένου για δύο άτομα.

Η νέα βελτιστοποίηση έχει προφανώς αποτέλεσμα μόνο όταν υπάρχουν κλάσεις που δηλώνονται ως ξένες μεταξύ τους. Όσο περισσότερες τέτοιες σχέσεις υπάρχουν, τόσο καλύτερη μπορεί να είναι η απόδοση του αλγορίθμου 4. Μπορούμε να φανταστούμε ότι η βελτιστοποίηση λειτουργεί «κλαδεύοντας» υπο-δέντρα της ιεραρχίας κλάσεων, και περιορίζοντας έτσι τον αριθμό των ισχυρισμών που θα ελεγχθούν. Όταν δεν υπάρχει κανένα

αξίωμα ξένων κλάσεων, είναι λογικό να μην προσφέρει τίποτα παραπάνω από την προηγούμενη έκδοση.

Ένα άλλο σημείο στο οποίο υπερέχει από την προηγούμενη έκδοση είναι ότι μπορεί να είναι ακόμα λιγότερο ευαίσθητος σε αλλαγές στην ιεραρχία κλάσεων. Για παράδειγμα, ακόμα και αν προστεθεί μία νέα υποκλάση στην κλάση Μπύρα, είναι πιθανό να μην ελεγχθεί κανένας παραπάνω ισχυρισμός, αφού το συγκεκριμένο υπο-δέντρο που περιέχει τη νέα κλάση θα απορριφθεί λόγων των αξιωμάτων ξένων κλάσεων.

3.4 Ορθότητα και Πληρότητα

Προηγουμένως αναφέρθηκε ότι το πλεονέκτημα μίας σημασιολογικής έναντι μίας συντακτικής προσέγγισης είναι ότι η πρώτη διαδικασία είναι ορθή. Για να προσδιορίσουμε επακριβώς τη σημασία της δήλωσης αυτής, εισάγουμε τους παρακάτω συμβολισμούς και ορισμούς. Έστω μία οντολογία O και ένα ερώτημα q. Συμβολίζουμε με $S_{q,O}$ (αντίστοιχα, $S_{q,ES(O,DL\text{-}Lite)}$) το σύνολο των απαντήσεων του q πάνω στην οντολογία O (πάνω στο σύνολο ES(O,DL-Lite)). Θα λέμε ότι το $S_{q,ES(O,DL\text{-}Lite)}$ είναι ορθό αν $S_{q,ES(O,DL\text{-}Lite)} \subseteq S_{q,O}$, και ότι το $S_{q,ES(O,DL\text{-}Lite)}$ είναι πλήρες αν $S_{q,ES(O,DL\text{-}Lite)} \subseteq S_{q,O}$. Θα λέμε, τέλος, ότι το $S_{q,ES(O,DL\text{-}Lite)}$ είναι ορθό και πλήρες αν $S_{q,ES(O,DL\text{-}Lite)} = S_{q,O}$.

Ουσιαστικά, η ιδιότητα της ορθότητας εξασφαλίζει ότι κάθε απάντηση που προκύπτει από τη νέα οντολογία (το σύνολο ES(O,DL-Lite)) είναι απάντηση του ερωτήματος και στην αρχική, δηλαδή ότι η διαδικασία της προσέγγισης δεν εισάγει νέες συνεπαγωγές και συμπεράσματα. Από την άλλη, η πληρότητα εξασφαλίζει ότι κάθε απάντηση του ερωτήματος στην αρχική οντολογία είναι απάντησή του και στο νέο σύνολο, δηλαδή ότι η προσέγγιση δεν "αποκόπτει" συμπεράσματα από την αρχική οντολογία.

Από τον ορισμό του ES(O,DL-Lite), προκύπτει άμεσα ότι το $S_{q,ES(O,DL\text{-}Lite)}$ είναι ορθό για κάθε ερώτημα q και οντολογία O γραμμένη σε OWL DL. Αντίθετα, είναι μη-πλήρες στη γενική περίπτωση (ένα παράδειγμα παρουσιάζεται στο [31]). Ωστόσο, είναι πλήρες για οποιοδήποτε συζευκτικό ερώτημα χωρίς μη-διακεκριμένες μεταβλητές (δηλαδή στο οποίο όλες οι μεταβλητές που εμφανίζονται στο σώμα εμφανίζονται και στην κεφαλή του ερωτήματος). Ο περιορισμός αυτός δεν είναι τόσο περίεργος, καθώς και άλλα γνωστά συστήματα συλλογιστικής περιορίζονται στην παραπάνω κατηγορία ερωτημάτων. Ως σύνοψη των παραπάνω, μπορούμε να διατυπώσουμε το εξής συμπέρασμα: Για οποιαδήποτε οντολογία O γραμμένη σε OWL DL και για οποιοδήποτε συζευκτικό ερώτημα q που δεν έχει μη-διακεκριμένες μεταβλητές, το σύνολο $S_{q,ES(O,DL\text{-}Lite)}$ είναι ορθό και πλήρες, δηλαδή ισχύει ότι $S_{q,ES(O,DL\text{-}Lite)} = S_{q,O}$.

Μπορεί να δειχθεί εύκολα ότι οι ιδιότητες αυτές δε μεταβάλλονται με τις επιπλέον βελτιστοποιήσεις του αλγορίθμου. Εφόσον οι βελτιστοποιήσεις παράγουν ένα υποσύνολο των ισχυρισμών και αξιωμάτων που παράγονται από τον αλγόριθμο των Pan – Thomas, δεν επηρεάζεται η συνέπεια (καθώς ό,τι αξίωμα εισάγεται στην οντολογία θα εισαγόταν και χωρίς τις επιπλέον βελτιστοποιήσεις). Όσον αφορά στην πληρότητα, τα αξιώματα που δεν ελέγχονται είναι σίγουρο ότι δε συνεπάγονται από την αρχική οντολογία, επομένως δεν αποκόπτονται συμπεράσματα, και άρα δεν επηρεάζεται η πληρότητα.

4

Υλοποίηση

Για τον έλεγχο και την πληρέστερη μελέτη της διαδικασίας σημασιολογικής προσέγγισης, στα πλαίσια της εργασίας αυτής σχεδιάστηκε και υλοποιήθηκε ένα εργαλείο που αναλαμβάνει την ολοκλήρωση της διαδικασίας, από την κατασκευή της αντίστοιχης DL-Lite οντολογίας μέχρι την υποβολή ενός ερωτήματος και τη λήψη των αποτελεσμάτων. Στο κεφάλαιο αυτό περιγράφεται η δομή και η λειτουργία του.

4.1 Περιγραφή συστήματος

Το σύστημα υλοποιείται με μία αρχιτεκτονική κλήσης-και-επιστροφής (call-and-return). και η λειτουργία του μπορεί να αναλυθεί σε μία σειρά βημάτων. Σε κάθε βήμα, το κεντρικό τμήμα του συστήματος καλεί με τη σειρά τα άλλα υποσυστήματα, καθένα από τα οποία επιτελεί μία συγκεκριμένη λειτουργία, βασιζόμενο στα αποτελέσματα των προηγουμένων βημάτων.

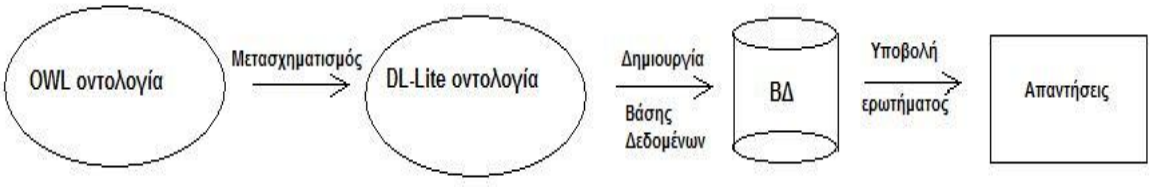
Για τη λειτουργία του συστήματος, ο χρήστης επιλέγει δύο πόρους. Πρώτον, μία οντολογία O_1, γραμμένη σε OWL DL ή κάποια υπογλώσσα της. Δεύτερον, ένα συζευκτικό ερώτημα Q πάνω στην οντολογία αυτή. Το σύστημα επιστρέφει το σύνολο των απαντήσεων του Q πάνω στην O_1, υλοποιώντας ένα μετασχηματισμό της σε DL-Lite. Κατά την εκτέλεση του συστήματος, δημιουργούνται και άλλοι πόροι, των οποίων η λειτουργία και χρήση θα παρουσιαστεί παρακάτω.

Όταν επιλεγούν τα απαραίτητα στοιχεία, ο χρήστης έχει την επιλογή να επεξεργαστεί ή όχι την O_1. Η προαιρετική αυτή επεξεργασία συνίσταται στην προσθήκη ενός ABox. Πολλές οντολογίες αποτελούνται μόνο από την περιγραφή των σχέσεων μεταξύ των εννοιών και των ρόλων, δηλαδή μόνο από ένα TBox. Για να απαντηθεί ένα ερώτημα, όμως, προφανώς χρειάζονται και κάποιοι ισχυρισμοί, οι οποίοι να προσδιορίζουν (άμεσα ή έμμεσα) κάποια άτομα ως απαντήσεις του ερωτήματος. Σε αντίθετη περίπτωση, κάθε ερώτημα θα οδηγήσει σε επιστροφή κενού συνόλου απαντήσεων. Επομένως, για να χρησιμοποιηθούν ως περιπτώσεις ελέγχου ορισμένες οντολογίες, προστίθεται σε αυτές ένα ABox που κατασκευάζεται με βάση το ερώτημα Q. Φυσικά, αν η οντολογία διαθέτει ήδη τους απαραίτητους ισχυρισμούς, μπορεί να παραμείνει αμετάβλητη. Σε κάθε περίπτωση, στο τέλος αυτού το βήματος η οντολογία O_1 περιέχει τους επιθυμητούς ισχυρισμούς για να μπορεί να απαντηθεί το ερώτημα.

Το επόμενο βήμα στη διαδικασία είναι ο μετασχηματισμός της O_1. Αυτός γίνεται σύμφωνα με τους αλγορίθμους σημασιολογικής προσέγγισης που παρουσιάστηκαν στο προηγούμενο κεφάλαιο και έχει ως αποτέλεσμα τη δημιουργία μίας νέας οντολογίας O_2, εκφρασμένης σε DL-Lite. Αυτό το βήμα είναι το ουσιαστικότερο στην όλη διαδικασία, καθώς αποτελεί την υλοποίηση της ιδέας του μετασχηματισμού γνώσης.

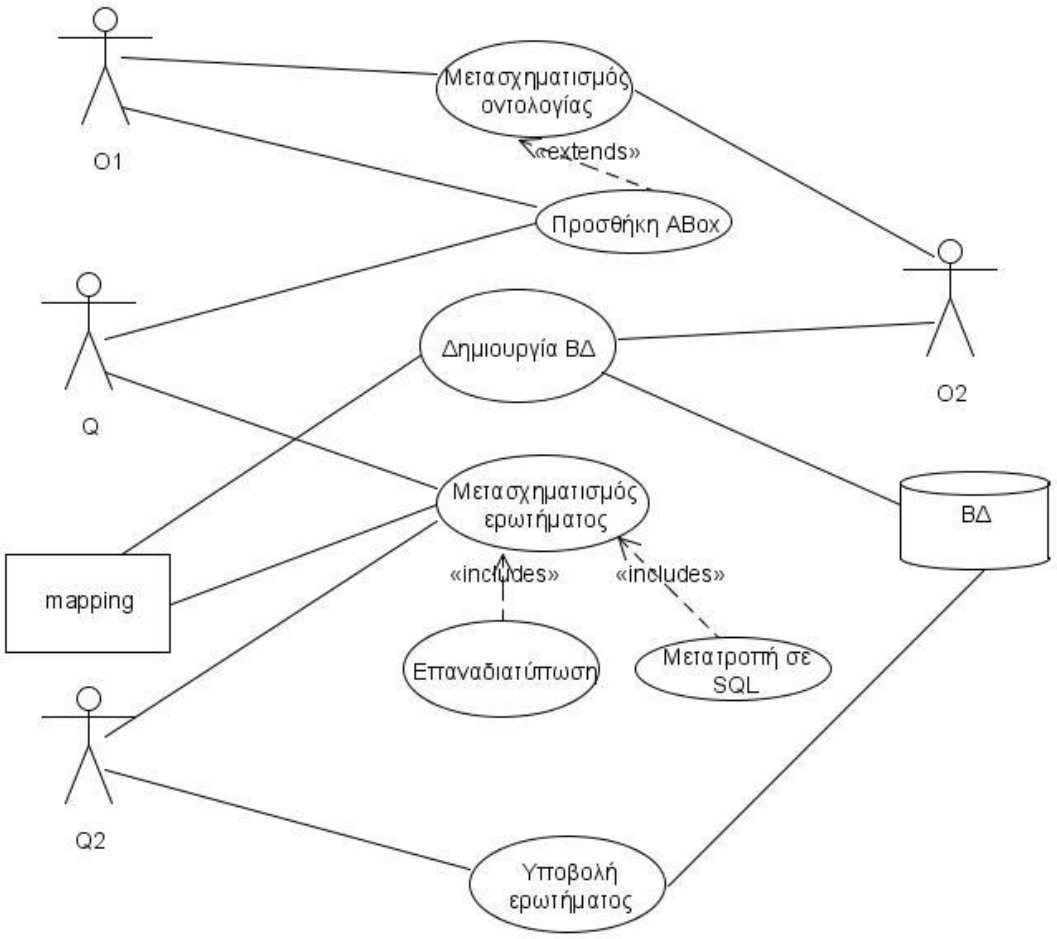
Με κατασκευασμένη την O_2, σειρά έχει η δημιουργία μίας σχεσιακής βάσης δεδομένων που θα χρησιμοποιηθεί για τη διεξαγωγή συλλογιστικής (στην προκειμένη περίπτωση, την απάντηση του ερωτήματος). Εκτός από τη βάση δεδομένων, στο βήμα αυτό δημιουργείται και ένα επιπλέον αρχείο, στο οποίο περιγράφεται η αντιστοίχηση (mapping) μεταξύ των περιεχομένων (εννοιών, ρόλων) της O_2 και των πινάκων της βάσης.

Στο σημείο αυτό, έχει ουσιαστικά ολοκληρωθεί η διαδικασία του μετασχηματισμού γνώσης. Ξεκινώντας από την O_1, η αρχική γνώση πήρε τη μορφή της λιγότερο εκφραστικής O_2, πριν καταλήξει να αποθηκευτεί στη βάση δεδομένων. Οι αλλαγές που υφίσταται κατά τη μετατροπή αυτή σχολιάζονται στο επόμενο κεφάλαιο. Σχηματικά, μπορούμε να αναπαραστήσουμε τους μετασχηματισμούς και τις εκφράσεις της αρχικής γνώσης όπως στο παρακάτω σχήμα.



Σχήμα 9: Μετασχηματισμοί της αρχικής γνώσης

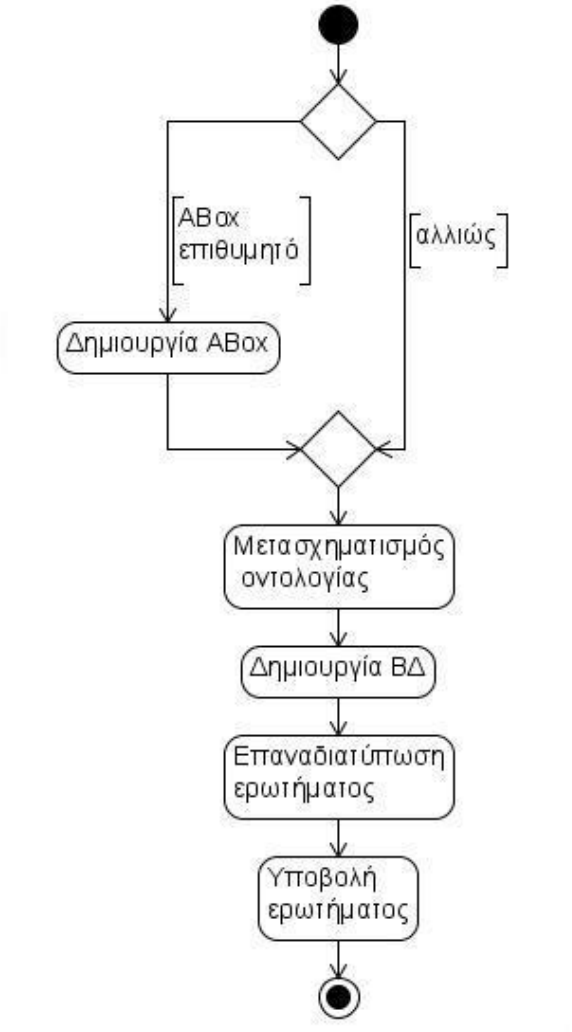
Μετά την ολοκλήρωση και του προηγούμενου βήματος, μπορεί να αρχίσει η διαδικασία απάντησης του ερωτήματος καθεαυτή. Πρώτα, το συζευκτικό ερώτημα Q μετασχηματίζεται, ώστε να επαναδιατυπωθεί ως ένα σύνολο ερωτημάτων. Η ενδιάμεση μορφή αυτή παράγεται αξιοποιώντας κατάλληλα το TBox της O_2, ώστε να κωδικοποιηθεί στο προκύπτον σύνολο ερωτημάτων η πληροφορία που περιέχει η γνώση για τις σχέσεις μεταξύ των εννοιών. Στη συνέχεια, από αυτό το σύνολο συζευκτικών ερωτημάτων σχηματίζεται ένα μοναδικό ερώτημα σε SQL, κάνοντας χρήση του αρχείου αντιστοιχήσεων που δημιουργήθηκε προηγουμένως. Το SQL ερώτημα Q_2 που προκύπτει κατ' αυτόν τον τρόπο υποβάλλεται στη βάση δεδομένων, για να ληφθούν οι απαντήσεις του, οι οποίες αποτελούν και τις απαντήσεις του αρχικού ερωτήματος Q πάνω στην αρχική οντολογία O_1 (όπως εξασφαλίζεται από τις ιδιότητες της ορθότητας και πληρότητας της διαδικασίας μετασχηματισμού γνώσης, που αναλύθηκαν στο Κεφάλαιο 3).



Σχήμα 10: Διάγραμμα περιπτώσεων χρήσης, παρουσιάζοντας τους πόρους (οντολογίες, ερωτήματα, αρχεία, βάση) ως δράστες. Ένα σενάριο χρήσης περιλαμβάνει όλες τις περιπτώσεις χρήσης που απεικονίζονται.

Συνοπτικά, η λειτουργία του συστήματος αποτελείται από τα εξής βήματα, που παρουσιάζονται και γραφικά στο σχήμα που ακολουθεί:

1. (προαιρετικά) Επέκταση της οντολογίας εισόδου με ένα ABox κατάλληλο για το ερώτημα που θα υποβληθεί.
2. Εφαρμογή του αλγορίθμου των Pan-Thomas για την κατασκευή μίας σημασιολογικής προσέγγισης της οντολογίας εισόδου.
3. Κατασκευή Βάσης Δεδομένων για την αποθήκευση της νέας οντολογίας.
4. Επεξεργασία του ερωτήματος και μετατροπή του σε ερώτημα SQL, υποβολή και λήψη απαντήσεων.



Σχήμα 11: Διάγραμμα δραστηριότητας του συστήματος

Ακολουθεί η περιγραφή της λειτουργίας των δομικών μονάδων-κλάσεων του συστήματος και η αλληλεπίδραση μεταξύ τους. Όπου αυτές χρησιμοποιούν εξωτερικά εργαλεία, γίνεται αναφορά σε αυτά και θα ακολουθήσει αναλυτικότερη παρουσίασή τους στο Κεφάλαιο 6.

4.2 Σημασιολογική προσέγγιση: η κλάση Approximator

Ο πυρήνας της υλοποίησης είναι η κλάση Approximator. Αυτή περιέχει τις απαραίτητες μεθόδους για την κατασκευή μίας DL-Lite οντολογίας που αποτελεί τη σημασιολογική προσέγγιση της OWL οντολογίας εισόδου. Ένα τυπικό παράδειγμα χρήσης της είναι το παρακάτω:

```
String infile = "file:/path/to/ontology/myOntology.owl" ;
String outfile = "file:/path/to/ontology/myOntology-out.owl" ;
String outlogical = "http://myontologies.net/myOntology-out" ;
Approximator approx = new Approximator(infile) ;
approx.transform(outfile, outlogical) ;
```

Κατά την αρχικοποίηση ενός αντικειμένου Approximator, δίνεται η τοποθεσία ενός αρχείου οντολογίας. Αυτή η οντολογία διατρέχεται ώστε να εντοπιστούν όλα τα ονόματα κλάσεων και ιδιοτήτων, που θα αποτελέσουν το αλφάβητο των ατομικών εννοιών και ρόλων της νέας οντολογίας. Με βάση αυτά, κατασκευάζονται τα σύνολα των βασικών και των γενικών εννοιών, όπως αυτές προσδιορίζονται από τη σύνταξη της DL-Lite. Στη συνέχεια πρέπει να κληθεί η μέθοδος transform του αντικειμένου, η οποία αναλαμβάνει να κατασκευάσει μία νέα, κενή οντολογία, να βρει το σύνολο συνεπαγωγών και τέλος να αποθηκεύσει τη νέα DL-Lite οντολογία. Η οντολογία αποθηκεύεται στη μορφή RDF/XML, σε θέση που προσδιορίζει ο χρήστης ως παράμετρο κατά την κλήση της transform.

Η εύρεση του συνόλου συνεπαγωγών είναι το κύριο μέρος της εργασίας και υλοποιείται από τη μέθοδο getEntailmentSet. Αυτή με τη σειρά της καλεί άλλες μεθόδους για τα διαφορετικά είδη αξιωμάτων, όπως φαίνεται επιγραμματικά στον παρακάτω ψευδοκώδικα:

```
ES = ∅;
ES = ES ∪ getPropertyAssertionAxioms() ;
ES = ES ∪ getClassAndClassAssertionAxioms() ;
```

Η getPropertyAssertionAxioms υλοποιεί τον αλγόριθμο για την εύρεση των ισχυρισμών ρόλων που περιγράφηκε στο προηγούμενο κεφάλαιο, ενώ η getClassAndClassAssertionAxioms κάνει την αντίστοιχη δουλειά για τα αξιώματα υπαγωγής εννοιών και τους ισχυρισμούς εννοιών. Η κάθε μία από αυτές υπολογίζει και επιστρέφει ένα σύνολο αξιωμάτων και στη συνέχεια αυτά προστίθενται στη νέα οντολογία, πριν αυτή

αποθηκευτεί. Ο λόγος που οι διαδικασίες εύρεσης των αξιωμάτων υπαγωγής και των ισχυρισμών εννοιών υλοποιούνται από την ίδια μέθοδο είναι ότι η πρώτη διαδικασία πραγματοποιεί την ταξινόμηση της οντολογίας, με βάση την οποία γίνεται η δεύτερη. Όπως αναφέρθηκε παραπάνω, η ταξινόμηση και ο έλεγχος των ισχυρισμών πρέπει να γίνει από κάποιο reasoner που να υποστηρίζει την πλήρη εκφραστικότητα της OWL, και στη συγκεκριμένη περίπτωση χρησιμοποιείται το πρόγραμμα Pellet. Αυτό παρέχει κατάλληλη διαπροσωπεία μέσω της οποίας γίνεται η επικοινωνία με την κλάση Approximator και η χρήση των απαραίτητων υπηρεσιών συλλογιστικής πάνω στην αρχική οντολογία.

4.3 Προσθήκη ABox: η κλάση SygeniaInterface

Όπως αναλύθηκε προηγουμένως, αν το ABox είναι κενό, δεν έχει νόημα η υποβολή ερωτημάτων και για αυτό το λόγο παρέχεται προαιρετικά η δυνατότητα ενσωμάτωσης ενός ABox στην οντολογία. Η δημιουργία του ABox γίνεται με χρήση της κλάσης PERFORMANSS του εργαλείου SyGENiA [33], το οποίο κατασκευάζει ένα ABox με βάση ένα συζευκτικό ερώτημα και μία οντολογία, εξασφαλίζοντας ότι το ερώτημα θα έχει απαντήσεις. Καθώς το SyGENiA αποθηκεύει το ABox σε ξεχωριστό αρχείο, αυτό πρέπει να προστεθεί στο αρχείο του TBox. Αυτό γίνεται απλώς διατρέχοντας τους ισχυρισμούς του ABox, προσθέτοντάς τους στο αρχικό TBox, και τέλος αποθηκεύοντας την προκύπτουσα οντολογία σε ένα νέο αρχείο, το οποίο πλέον περιέχει τόσο το TBox όσο και το ABox. Η διασύνδεση με το SyGENiA και η συγχώνευση του ABox και του TBox γίνονται με την κλάση SygeniaInterface.

4.4 Διασύνδεση με Βάση Δεδομένων: η κλάση CreateDB

Η συλλογιστική στη DL-Lite πραγματοποιείται, όπως αναφέρθηκε στην αντίστοιχη ενότητα, με την κατασκευή και την υποβολή ερωτημάτων σε μία σχεσιακή βάση δεδομένων. Η κλάση CreateDB επιτελεί, συγκεκριμένα, τις ακόλουθες εργασίες:

1. Δημιουργία ενός DL-Lite reasoner για την επιλεγμένη οντολογία.
2. Δημιουργία του σχήματος (schema) μίας σχεσιακής βάσης δεδομένων, βάσει του TBox. Το σχήμα αναπαριστά τις έννοιες και τους ρόλους της οντολογίας. Συγκεκριμένα, για κάθε έννοια υπάρχει ένας πίνακας με ένα πεδίο, ενώ για κάθε ρόλο δημιουργείται ένας πίνακας με δύο πεδία.
3. Αποθήκευση εγγραφών στη βάση δεδομένων, με βάση το ABox. Για κάθε ισχυρισμό (έννοιας ή ρόλου) εισάγεται μία εγγραφή στον αντίστοιχο πίνακα.

4. Δημιουργία ενός αρχείου αντιστοιχίσεων (mappings). Στο αρχείο αυτό περιγράφεται το ποιός πίνακας της βάσης δεδομένων αντιστοιχεί σε κάθε έννοια ή ρόλο της οντολογίας, με ένα κατάλληλο SQL ερώτημα SELECT.
5. Δημιουργία και χρήση ενός νέου RequiemInterface, για την κατάλληλη διαχείριση του ερωτήματος, και λήψη των απαντήσεων.

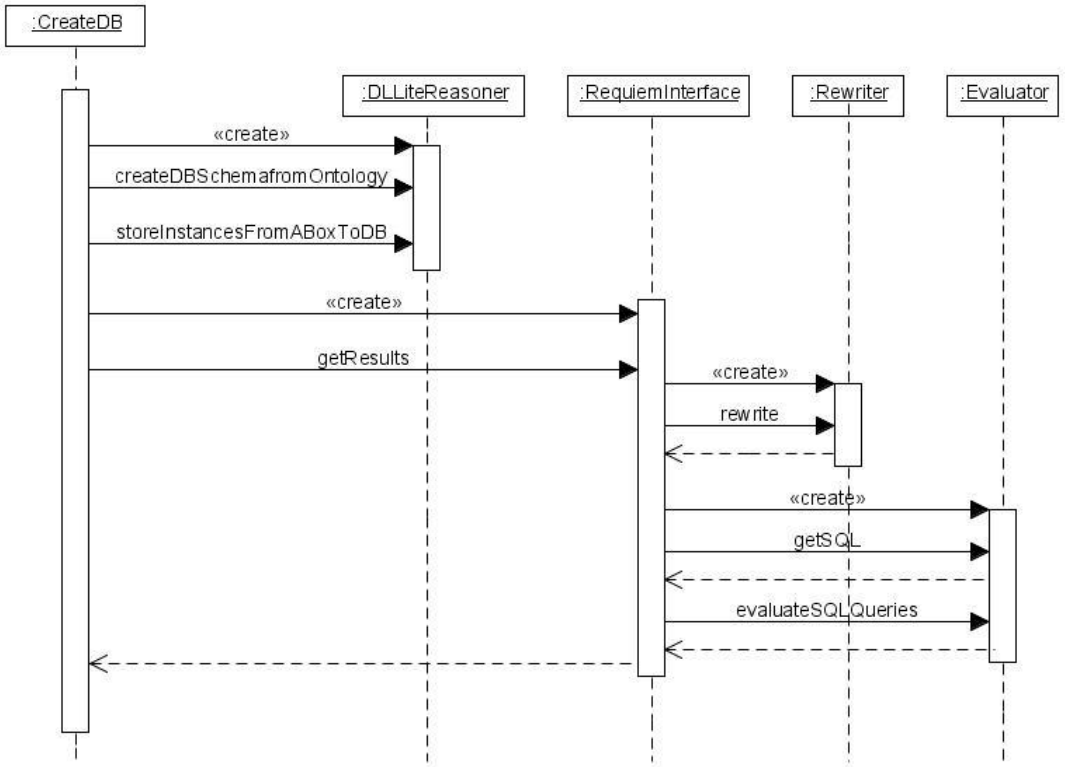
Για την ολοκλήρωση της παραπάνω διαδικασίας γίνεται χρήση του DL-Lite reasoner του Εργαστηρίου Επεξεργασίας Εικόνας, ο οποίος πραγματοποιεί την κατασκευή της βάσης δεδομένων (τόσο τον ορισμό του σχήματος όσο και την εισαγωγή των εγγραφών). Το σύστημα διαχείρισης που χρησιμοποιείται είναι το HyperSQL[34]. Η κλάση RequiemInterface που χρησιμοποιείται στο τελευταίο βήμα αναλύεται αμέσως παρακάτω.

4.5 Διασύνδεση με το REQUIEM: η κλάση RequiemInterface

Η κλάση αυτή αναλαμβάνει τη σύνδεση με το εργαλείο REQUIEM και τη χρήση κατάλληλων μεθόδων που αυτό προσφέρει για την αναδιατύπωση, μετατροπή και υποβολή του ερωτήματος. Η δημιουργία ενός αντικειμένου RequiemInterface απαιτεί τον προσδιορισμό ενός αρχείου οντολογίας και ενός ερωτήματος, με βάση τα οποία θα γίνουν όλες οι επόμενες διεργασίες. Η χρήση του αντικειμένου γίνεται με κλήση της μεθόδου getResults, προσδιορίζοντας μία βάση δεδομένων και ένα αρχείο αντιστοιχίσεων. Οι ενέργειες που πραγματοποιούνται μετά από μία τέτοια κλήση είναι οι ακόλουθες:

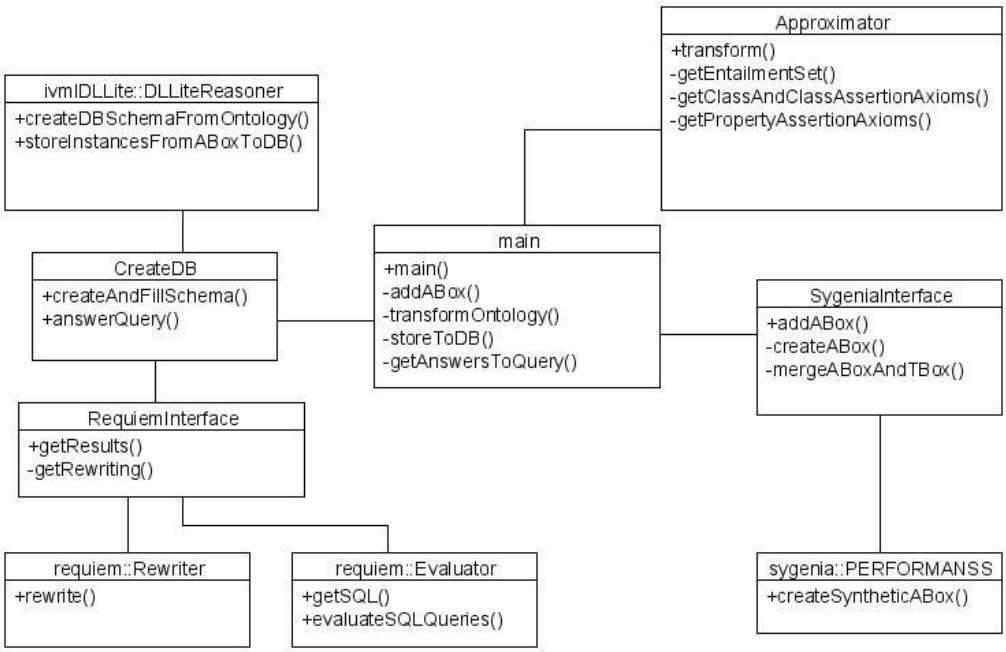
1. Επαναδιατύπωση του ερωτήματος. Γίνεται μία επεξεργασία στο ερώτημα, ώστε αυτό να μετασχηματιστεί σε ένα σύνολο από ερωτήματα, η ένωση των απαντήσεων των οποίων αποτελεί την απάντηση του αρχικού.
2. Μετατροπή σε SQL. Διατρέχονται όλα τα ερωτήματα-μέλη του παραπάνω συνόλου, και από τη διαδικασία αυτή δημιουργείται ένα και μοναδικό SQL ερώτημα. Ο σχηματισμός του ερωτήματος γίνεται βάσει των αντιστοιχήσεων που δημιουργήθηκαν και αποθηκεύτηκαν στο βήμα 4 της λειτουργίας της κλάσης CreateDB.
3. Υποβολή του ερωτήματος στη βάση δεδομένων. Το σύνολο των αποτελεσμάτων που λαμβάνεται είναι και η απάντηση του αρχικού ερωτήματος.

Για τη διεκπεραίωση των παραπάνω, χρησιμοποιούνται οι δυνατότητες που προσφέρει το REQUIEM, και πρωτίστως δύο κλάσεις. Η κλάση Rewriter πραγματοποιεί την επαναδιατύπωση του ερωτήματος, ενώ η Evaluator τη μετατροπή και αποτίμηση. Το τελικό αποτέλεσμα, δηλαδή το σύνολο των απαντήσεων, επιστρέφεται στον καλούντα.



Σχήμα 12:Απλοποιημένο ακολουθιακό διάγραμμα για τη διασύνδεση με τη βάση δεδομένων και την επικοινωνία με το REQUIEM

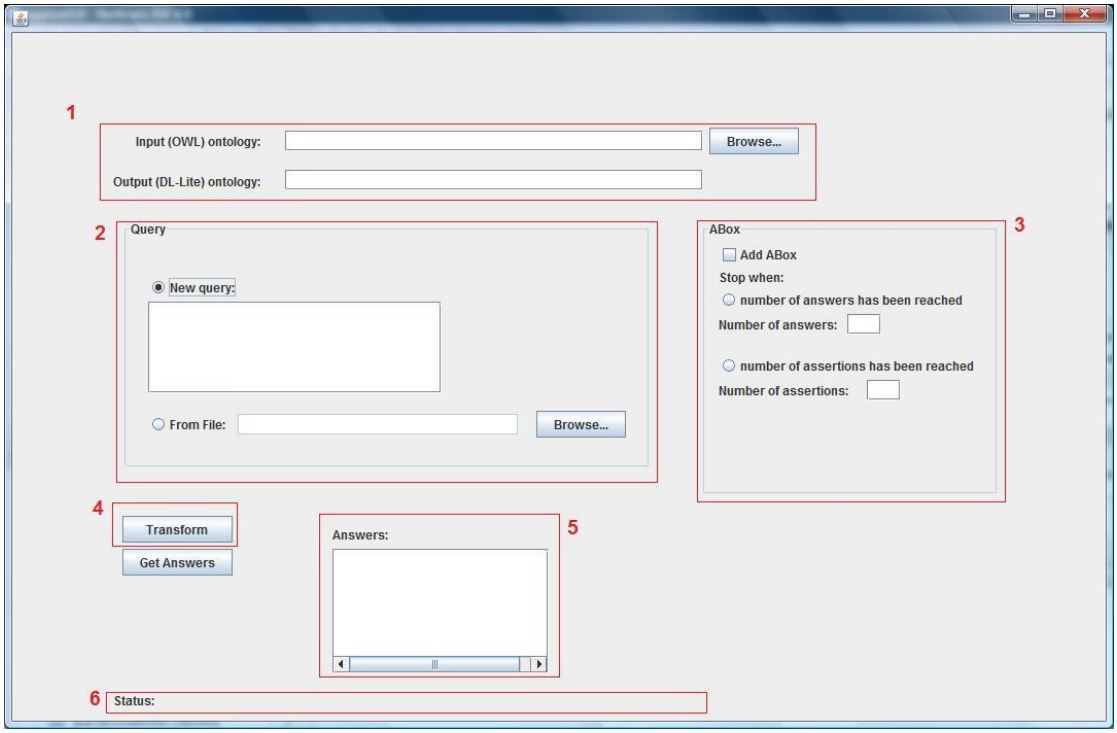
Στο παρακάτω σχήμα απεικονίζεται συνοπτικά η δομή του συστήματος και η διασύνδεση των διαφόρων κλάσεων, τόσο στο εσωτερικό του συστήματος, όσο και με κάποια από τα εξωτερικά εργαλεία.



Σχήμα 13: Απλοποιημένο διάγραμμα κλάσεων για το σύστημα

4.6 *Γραφικό Περιβάλλον*

Ως συμπλήρωμα του συστήματος, σχεδιάστηκε ένα γραφικό περιβάλλον για την καθοδήγηση του χρήστη, την επιλογή των απαραίτητων δεδομένων (τοποθεσία αρχικής οντολογίας, ερώτημα προς απάντηση) και την εμφάνιση των απαντήσεων.



Σχήμα 14: Η κεντρική οθόνη του γραφικού περιβάλλοντος

Η κεντρική οθόνη χωρίζεται χονδρικά σε δύο μέρη. Το πάνω μέρος δίνει στο χρήστη τη δυνατότητα προσδιορισμού των παραμέτρων που επιθυμεί, ενώ από το κάτω ελέγχεται η διεξαγωγή της διαδικασίας και φαίνονται τα αποτελέσματα. Πιο αναλυτικά, υπάρχουν οι ακόλουθες υπο-περιοχές:

1. Περιοχή προσδιορισμού της θέσης των οντολογιών εισόδου και εξόδου
2. Περιοχή εισαγωγής του ερωτήματος
3. Περιοχή ρύθμισης παραμέτρων του ABox
4. Κουμπί έναρξης μετασχηματισμού
5. Περιοχή λήψης απαντήσεων
6. Γραμμή κατάστασης, όπου εμφανίζονται ενημερωτικά μηνύματα προς το χρήστη.

Η λειτουργία της κάθε περιοχής και ο τρόπος χρήσης της περιγράφονται αναλυτικότερα αμέσως παρακάτω.

4.6.1 Προσδιορισμός θέσης οντολογιών

Σχήμα 15: Προσδιορισμός θέσης οντολογιών στο γραφικό περιβάλλον

Αρχικά, πρέπει να επιλεγεί η οντολογία που θα μετασχηματιστεί και θα χρησιμοποιηθεί για την απάντηση του ερωτήματος. Πατώντας το κουμπί Browse, εμφανίζεται ένα πλαίσιο επιλογής αρχείων (file chooser) , από το οποίο διαλέγουμε το αρχείο στο οποίο βρίσκεται η αρχική οντολογία που θέλουμε. Επιβεβαιώνοντας την επιλογή, η διαδρομή προς το αρχείο εμφανίζεται στο πεδίο "Input (OWL) ontology". Ταυτόχρονα, στο πεδίο "Output (DL-Lite ontology" τοποθετείται αυτόματα μία προτεινόμενη θέση για την αποθήκευση της οντολογίας που θα παραχθεί με το μετασχηματισμό. Η θέση αυτή είναι στον ίδιο φάκελο με την οντολογία εισόδου, αλλά με όνομα αρχείου που προκύπτει προσθέτοντας "-out" στο όνομα του αρχείου εισόδου, πριν την κατάληξη ".owl". Η θέση που προτείνεται με αυτό τον τρόπο δεν είναι δεσμευτική: αν επιθυμούμε η οντολογία εξόδου να αποθηκευτεί σε άλλη θέση, μπορούμε να αλλάξει το περιεχόμενο του δεύτερου πεδίου και να εισάγουμε την επιθυμητή τοποθεσία.

4.6.2 Προσδιορισμός ερωτήματος

Σχήμα 16: Προσδιορισμός ερωτήματος στο γραφικό περιβάλλον

Αφού επιλεγεί η οντολογία εισόδου, πρέπει να ορίσουμε το ερώτημα το οποίο θα απαντηθεί. Αυτό μπορεί να γίνει με δύο τρόπους, είτε γράφοντάς το εμείς οι ίδιοι, είτε καθορίζοντας ένα αρχείο το οποίο περιέχει το ερώτημα. Στην πρώτη περίπτωση, που είναι και αυτή που ενεργοποιείται εξ ορισμού, επιλέγουμε το κουμπί "New query" και γράφουμε το ερώτημα στο αντίστοιχο πλαίσιο κειμένου. Στη δεύτερη, επιλέγουμε το "From File" και, με το πλαίσιο επιλογής που εμφανίζεται πατώντας το κουμπί Browse, διαλέγουμε το αρχείο στο οποίο βρίσκεται το ερώτημά μας.

Και στις δύο περιπτώσεις, το ερώτημα πρέπει να είναι γραμμένο με συγκεκριμένη μορφή: Η κεφαλή του πρέπει να είναι της μορφής Q(?0,?1,…?n), όπου ?0, ?1 κ.ο.κ είναι οι μεταβλητές που παριστάνουν τις απαντήσεις προς επιστροφή. Με τον ίδιο τρόπο ορίζονται οι μεταβλητές και στο σώμα του ερωτήματος, το οποίο μπορεί να περιέχει και μεταβλητές που να μην εμφανίζονται στην κεφαλή. Το σώμα αποτελείται από μία σειρά ατόμων της μορφής C(?k) ή R(?k,?m), χωρισμένα με κόμμα (","), όπου C μία κλάση και R μια ιδιότητα. Για παράδειγμα, το ερώτημα

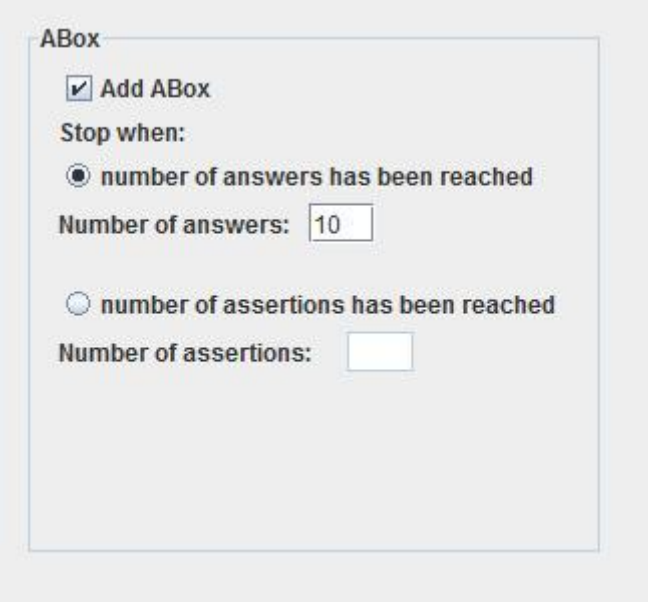
$$Q(x) \leftarrow worksFor(x,y) \wedge affiliatedOrganizationOf(y,z)$$

θα πρέπει να γραφεί (στο πλαίσιο κειμένου ή στο αρχείο) ως

$$Q(?0) <- worksFor(?0,?1), affiliatedOrganizationOf(?1,?2)$$

Στην περίπτωση που το ερώτημα βρίσκεται από αρχείο, το αρχείο μπορεί να περιέχει, εκτός της γραμμής του ερωτήματος, και σχόλια, τα οποία έχουν έκταση μίας σειράς και δηλώνονται με "//" στην αρχή της σειράς.

4.6.3 Ρύθμιση παραμέτρων του ABox



Σχήμα 17: Ρύθμιση παραμέτρων για τη δημιουργία του ABox στο γραφικό περιβάλλον

Στην παραπάνω εικόνα φαίνεται η περιοχή που σχετίζεται με τις ρυθμίσεις για την προσθήκη ABox στην οντολογία. Αρχικά, ορίζουμε αν θέλουμε να προστεθεί ένα ABox ή όχι, ενεργοποιώντας ή απενεργοποιώντας το κουτί επιλογής "Add ABox". Όπως είναι αναμενόμενο, οι υπόλοιπες ρυθμίσεις λαμβάνονται υπόψη μόνο αν το κουτί είναι επιλεγμένο. Μπορούμε να ελέγξουμε το μέγεθος του παραγόμενου ABox με δύο τρόπους. Πρώτον, να σταματήσουμε τη δημιουργία του ABox όταν έχει εξασφαλιστεί ότι θα υπάρχει ένας συγκεκριμένος αριθμός απαντήσεων στο ερώτημα. δεύτερον, να σταματήσουμε τη δημιουργία του ABox όταν ξεπεραστεί ένα συγκεκριμένο πλήθος ισχυρισμών σε αυτό. Δηλώνουμε την προτίμησή μας επιλέγοντας μία από τις δύο ενδείξεις "number of answers has been reached" (στην πρώτη περίπτωση) ή "number of assertions has been reached" (στη δεύτερη). Ταυτόχρονα, προσδιορίζουμε το πλήθος των απαντήσεων ή ισχυρισμών που θέλουμε να τεθεί ως όριο, εισάγοντας μία ακέραια τιμή στο αντίστοιχο πλαίσιο κειμένου.

4.6.4 Έναρξη μετασχηματισμού

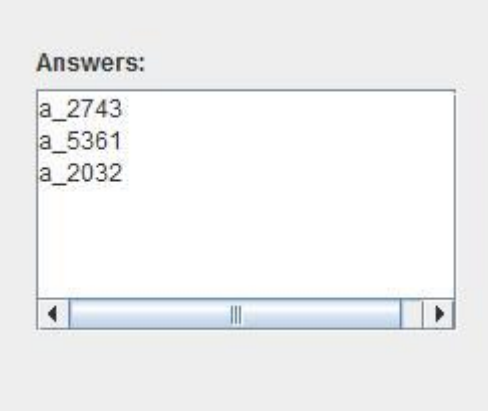
Σχήμα 18: Κουμπιά για την έναρξη μετασχηματισμού και τη λήψη απαντήσεων

Αν έχουμε προσδιορίσει όλες τις παραμέτρους, ακολουθώντας τα προηγούμενα βήματα, μπορούμε να ξεκινήσουμε τη διαδικασία μετασχηματισμού, πατώντας το κουμπί Transform. Αν το κουτί επιλογής "Add ABox" είναι ενεργοποιημένο, πρώτα δημιουργείται και ενσωματώνεται το ABox. Για το σκοπό αυτό, αν το ερώτημα έχει γραφεί στην κεντρική οθόνη (και δε βρίσκεται σε κάποιο αρχείο), δημιουργείται ένα αρχείο με το όνομα tempquery, στο οποίο περιέχεται το ερώτημα όπως ακριβώς δόθηκε. Με βάση το ερώτημα και την οντολογία, δημιουργείται ένα ABox, που αποθηκεύεται στο φάκελο της αρχικής οντολογίας. Τέλος, το ABox συγχωνεύεται με την αρχική οντολογία, και παράγεται ένα ακόμα αρχείο, επίσης στο φάκελο της αρχικής οντολογίας, που περιέχει την «επεκτεταμένη» οντολογία. Όταν ολοκληρωθεί αυτή η διαδικασία, ή αν είχαμε επιλέξει να μην προστεθεί ABox, ξεκινά η διαδικασία του μετασχηματισμού γνώσης. Ως αποτέλεσμα, δημιουργείται μία νέα DL-Lite οντολογία, η οποία, όπως έχει ήδη αναφερθεί, αποθηκεύεται στη θέση που καθορίζεται από το πλαίσιο κειμένου "Output (DL-Lite) ontology".

Αν κατά τη διάρκεια της διαδικασίας προκύψει κάποιο πρόβλημα (για παράδειγμα, αν η τιμή που έχει δοθεί ως όριο για τη δημιουργία του ABox δεν αναπαριστά ακέραιο αριθμό), η διαδικασία σταματά και εμφανίζεται ένα κατάλληλο μήνυμα στη γραμμή κατάστασης. Αν όλα πάνε καλά, εμφανίζεται, αντίστοιχα, το ενημερωτικό μήνυμα "Created DL-Lite ontology".

Σημειώνουμε ότι, σε αυτό το στάδιο, το ερώτημα λαμβάνεται υπόψη μόνο για τη δημιουργία του ABox. Αυτό σημαίνει ότι η διαδικασία μετασχηματισμού μπορεί να πραγματοποιηθεί ακόμα και αν δεν έχει προσδιοριστεί κάποιο ερώτημα. Ωστόσο, σε αυτή την περίπτωση πρέπει αναγκαστικά να απενεργοποιηθεί η δημιουργία του ABox (αφού η κατασκευή πρέπει να γίνει βάσει κάποιου ερωτήματος). Αυτό μπορεί να είναι χρήσιμο σε κάποιον που θέλει να λάβει απλώς το μετασχηματισμό της αρχικής οντολογίας σε DL-Lite, χωρίς να τον ενδιαφέρει εκείνη τη στιγμή η απάντηση κάποιου ερωτήματος. Επομένως, η χρήση του συστήματος μπορεί για κάποιους να ολοκληρωθεί εδώ, αν και υπάρχει ακόμα ένα βήμα που μπορεί να πραγματοποιηθεί.

4.6.5 Λήψη απαντήσεων



Σχήμα 19: Πλαίσιο απαντήσεων στο γραφικό περιβάλλον

Αφού ολοκληρωθεί επιτυχώς ο μετασχηματισμός της οντολογίας, απομένει μόνο να υποβάλλουμε το ερώτημα και να πάρουμε τις απαντήσεις. Αυτό γίνεται πατώντας το κουμπί "Get Answers" (βλ. Σχήμα 18), που έχει ως αποτέλεσμα την έναρξη μίας σειράς ενεργειών που περιγράφηκε σε προηγούμενες ενότητες, από τη δημιουργία της βάσης δεδομένων και το μετασχηματισμό του ερωτήματος μέχρι την υποβολή του στη βάση και τη λήψη των απαντήσεων. Οι απαντήσεις που λαμβάνονται παρουσιάζονται στο χρήστη στο πλαίσιο "Answers". Με την περάτωση και αυτού του βήματος, ολοκληρώνεται η λειτουργία του συστήματος.

5

Αξιολόγηση

Σε αυτό το κεφάλαιο περιέχονται σχόλια και παρατηρήσεις για τη συμπεριφορά του αλγορίθμου μετασχηματισμού οντολογιών, όπως αυτός υλοποιήθηκε σύμφωνα με τα προηγούμενα κεφάλαια.

5.1 Οντολογίες ελέγχου

Για την αξιολόγηση και τον έλεγχο του μετασχηματισμού, χρησιμοποιήθηκαν πέντε OWL οντολογίες, που παρουσιάζουν ποικιλία στην εκφραστικότητα και το πλήθος των αξιωμάτων που περιέχουν. Αυτές είναι:

- Generations (generations): Μία απλή οντολογία που περιγράφει ορισμένες οικογενειακές σχέσεις και περιλαμβάνει κάποιους ισχυρισμούς για μία συγκεκριμένη οικογένεια [35].
- Lehigh University Benchmark (univ-bench): Μία οντολογία σχετικά με το περιβάλλον του πανεπιστημίου. Χωρίς κανένα άτομο [36].
- Adolena (adolena): Μία οντολογία βιοϊατρικών όρων, με ABox που περιέχει ένα μόνο ισχυρισμό έννοιας. Ιδιαίτερα αυξημένη πολυπλοκότητα [37].
- Wines Ontology (wines): Οντολογία για το κρασί, μέρος των παραδειγμάτων της W3C για το πρότυπο της OWL [38].
- myGrid Domain Ontology (mygrid): Οντολογία βιοπληροφορικής και μοριακής βιολογίας, χωρίς ABox [39].

Μία συνοπτική παρουσίαση κάποιων βασικών χαρακτηριστικών της κάθε μίας βρίσκεται στον παρακάτω πίνακα:

Οντολογία	**Εκφραστι-κότητα**	**Κλάσεις**	**Ιδιότητες**	**Άτομα**	**Υπαγωγές**	**Ισοδυναμίες**	**Ισχυρισμοί κλάσεων**	**Ισχυρισμοί ιδιοτήτων**
generations	$\mathcal{ALCOIF}$	18	4	7	0	17	7	9
univ-bench	$\mathcal{ALEHI}$(**D**)	43	25	0	36	6	0	0
adolena	$\mathcal{SRIQ}$	141	16	1	192	4	1	0
wines	$\mathcal{ALEON}$(**D**)	169	18	226	243	0	187	246
mygrid	$\mathcal{ALCHIF}$	475	8	0	567	125	0	0

Πίνακας 2: Οντολογίες ελέγχου

5.2 Γενικές παρατηρήσεις

Όπως είναι φυσικό, δεδομένης της διαφορετικής σύνταξης και εκφραστικότητας των OWL DL και DL-Lite, κατά τη μετατροπή από τη μία στην άλλη γλώσσα αλλάζει το σύνολο των αξιωμάτων που αποτελούν την οντολογία. Στην ενότητα αυτή θα αναφερθούν ορισμένες παρατηρήσεις και σχόλια πάνω στις αλλαγές αυτές.

Αρχικά, σημειώνεται ότι η συγκεκριμένη υλοποίηση αφήνει εκτός της οντολογίας εξόδου όλες τις ιδιότητες δεδομένων της αρχικής οντολογίας. Ο λόγος για την επιλογή αυτή είναι ότι η DL-Lite δεν υποστηρίζει τύπους δεδομένων, επομένως οι ιδιότητες δεδομένων της OWL και τα σχετικά με αυτές αξιώματα δεν μπορούν να εκφραστούν σε αυτή. Στη συνέχεια, θα γίνει αναφορά μόνο σε ιδιότητες αντικειμένων και στις έννοιες και στα αξιώματα που σχετίζονται με αυτές.

Κάποιες περιγραφές κλάσεων, που δεν μπορούν να αποδοθούν επακριβώς σε DL-Lite, αντικαθίστανται από υπερκλάσεις τους. Έτσι, όλοι οι περιορισμένοι υπαρξιακοί περιορισμοί ($\exists$R.C) μετατρέπονται σε πλήρεις υπαρξιακούς περιορισμούς ($\exists$R.$\top$) στη νέα οντολογία. Ενδιαφέρον παρουσιάζουν ειδικά οι πλήρεις υπαρξιακοί περιορισμοί με ονοματικές έννοιες. Ένας ισχυρισμός έννοιας a:$\exists$R.{o} αναπαριστά ουσιαστικά έναν ισχυρισμό ρόλου (a,o):R και, εκτός από την παραπάνω μετατροπή, η νέα οντολογία περιέχει και αυτόν τον ισχυρισμό ρόλου. Για παράδειγμα, η οντολογία wines περιέχει τις δηλώσεις

```
SAUTERNE SubClassOf REGION value SAUTERNE-INDIVIDUAL

CHATEAU-D-YCHEM-SAUTERNE Type SAUTERNE
```

Στην DL-Lite οντολογία, βρίσκουμε τους εξής ισχυρισμούς, μεταξύ άλλων:

```
CHATEAU-D-YCHEM-SAUTERNE Type REGION some THING
```

CHATEAU-D-YCHEM-SAUTERNE REGION SAUTERNE-INDIVIDUAL

Στον πρώτο ισχυρισμό φαίνεται η αλλαγή του πλήρους υπαρξιακού περιορισμού σε περιορισμένο. Ο δεύτερος είναι ο ισχυρισμός ρόλου που αντιστοιχεί στον ισχυρισμό έννοιας.

Αντίστοιχες αλλαγές συμβαίνουν στους περιορισμούς ελάχιστης και ακριβούς πληθυκότητας. Για παράδειγμα, μία έκφραση $\geq 3R.C$ μετατρέπεται επίσης σε $\exists R.\top$, δηλαδή διατηρείται ο περιορισμός της ύπαρξης R-ακολούθων, αλλά χάνεται η πληροφορία για το πλήθος αυτών. Αυτό δεν είναι τόσο σημαντική απώλεια, καθώς στα συζευκτικά ερωτήματα δεν μπορεί να εκφραστούν περιορισμοί πληθυκότητας ούτως ή άλλως. Δε συμβαίνει κάποια αντίστοιχη μετατροπή, ωστόσο, με τους περιορισμούς μέγιστης πληθυκότητας, καθώς ένας ισχυρισμός της μορφής $a:\leq nR.C$ δεν εξασφαλίζει την ύπαρξη R-ακολούθων για το a, δηλαδή δε συνεπάγεται ότι $a:\exists R.\top$.

5.2.1 Αξιώματα που αποκόπτονται

Εξαιτίας της μειωμένης εκφραστικότητας της DL-Lite, ορισμένα αξιώματα χάνονται κατά τη μετατροπή. Μεταξύ αυτών είναι όσα περιλαμβάνουν περιορισμούς τιμής ($\forall R.C$), καθώς και τα αξιώματα συνόλου τιμών ιδιοτήτων. Παρότι δεν μπορούν να εκφραστούν σε DL-Lite αυτά καθαυτά, η πληροφορία τους διατηρείται στην τελική οντολογία, με τη μορφή των αντίστοιχων ισχυρισμών. Για παράδειγμα, και πάλι από την οντολογία wines:

ObjectProperty(GRAPE-SLOT domain(WINE))

CHIANTI SubClassOf GRAPE-SLOT value SANGIOVESE

οι οποίοι οδηγούν στην εισαγωγή του αξιώματος υπαγωγής

CHIANTI SubClassOf WINE

στη νέα οντολογία.

Παρομοίως, αν και η DL-Lite δεν υποστηρίζει την έκφραση αξιωμάτων ρόλων (μεταβατικότητα, συμμετρικότητα, κ.ά.), αυτά λαμβάνονται υπόψιν κατά τη μετατροπή. Για παράδειγμα, η οντολογία adolena περιέχει μία ιδιότητα hasPart που είναι δηλωμένη ως ανακλαστική, και ένα μοναδικό άτομο, Invacare. Στη νέα οντολογία, υπάρχει ο ισχυρισμός `Invacare hasPart Invacare`. Μάλιστα, επειδή η αρχική οντολογία δεν περιέχει κανέναν ισχυρισμό ρόλου, παρατηρείται το (ίσως αρχικά περίεργο) φαινόμενο του να εμφανίζονται ισχυρισμοί ρόλων χωρίς εμφανή προέλευση.

5.2.2 Αξιώματα που προστίθενται

Μέχρι τώρα είδαμε πώς ορισμένα αξιώματα αποκόπτονται στη νέα οντολογία, αλλά σε κάποιες περιπτώσεις αντικαθίστανται από άλλα, που κωδικοποιούν μερικώς την πληροφορία τους. Όπως αναφέρθηκε και στο Κεφάλαιο 3, όμως, ένα από τα οφέλη του μετασχηματισμού

γνώσης είναι ότι προσθέτει εκπεφρασμένα στη γνώση κάποια από τα συμπεράσματά της. Αυτό συμβαίνει και σε αυτή την εφαρμογή, όπως στις παραπάνω περιπτώσεις, ενώ επίσης ένα σημαντικό μέρος των αξιωμάτων που εισάγει η μετατροπή έχει να κάνει με τις ιεραρχίες των κλάσεων και των ρόλων. Πρόκειται για αξιώματα υπαγωγής και ισχυρισμούς που δεν υπήρχαν εκπεφρασμένα στην OWL DL οντολογία, που ουσιαστικά όμως περιγράφουν την ίδια ταξονομία, αλλά με «πλεονασμούς».

Ως παράδειγμα, αναφέρουμε την οντολογία adolena. Σε αυτή βρίσκουμε, μεταξύ άλλων, η ιεραρχία κλάσεων

$$\texttt{Vendor} \sqsubseteq \texttt{ServiceProvider} \sqsubseteq \texttt{NonPhysicalObject} \sqsubseteq \texttt{Endurant}$$

καθώς και ο ισχυρισμός `Invacare : Vendor`. Στην προκύπτουσα οντολογία, υπάρχουν επιπλέον εκπεφρασμένα τα αξιώματα που υπονοούνται από την παραπάνω ιεραρχία, δηλαδή:

$$\texttt{Vendor} \sqsubseteq \texttt{NonPhysicalObject}$$

$$\texttt{Vendor} \sqsubseteq \texttt{Endurant}$$

$$\texttt{ServiceProvider} \sqsubseteq \texttt{Endurant}$$

καθώς και οι ισχυρισμοί

`Invacare : ServiceProvider, Invacare : NonPhysicalObject, Invacare : Endurant`

Αν και προστίθενται αξιώματα και ισχυρισμοί, δεν αλλάζει καθόλου η σημασιολογία, καθώς η «νέα» αυτή γνώση είναι απλώς συμπεράσματα της αρχικής οντολογίας.

5.3 Βελτιστοποιήσεις

Αρχικά, συγκρίνονται οι χρόνοι εκτέλεσης του αλγορίθμου, πριν και μετά τις βελτιστοποιήσεις που χρησιμοποιήθηκαν. Τα αποτελέσματα παρουσιάζονται στον επόμενο πίνακα:

Οντολογία	**Χρόνος πριν (ms)**	**Χρόνος μετά (ms)**	**Διαφορά**
generations	328,5	342,1	-4,14%
univ-bench	1412,1	505,9	64,17%
adolena	7329,2	5081,2	30,67%
wines	58694,1	55695,5	5,11%
mygrid	-	163096,7	-

Πίνακας 3: Χρόνοι εκτέλεσης μετασχηματισμού με και χωρίς βελτιστοποιήσεις

Όπως φαίνεται, σε όλες σχεδόν τις οντολογίες ελέγχου, οι βελτιστοποιήσεις μείωσαν, σε κάποιες περιπτώσεις σημαντικά, το χρόνο εκτέλεσης. Εξαίρεση αποτελεί η οντολογία

ontologies, στην οποία η έκδοση του αλγορίθμου με τις βελτιστοποιήσεις απαιτεί λίγο περισσότερο χρόνο. Αυτό μπορεί να αποδοθεί στο μικρό μέγεθος της οντολογίας: επειδή το μέγεθος του συνόλου αξιωμάτων που παράγεται στην «αφελή» υλοποίηση είναι πολύ μικρό, τα αξιώματα μπορούν να ελεγχθούν γρήγορα και η διαδικασία διαρκεί λιγότερο από ό,τι στην περίπτωση των βελτιστοποιήσεων. Αυτό ίσως να υποδεικνύει ότι υπάρχει κάποιο κατώφλι μεγέθους για τις οντολογίες, κάτω από το οποίο οι επιπλέον ενέργειες που εκτελούνται στη βελτιστοποιημένη έκδοση είναι πιο χρονοβόρες από το κέρδος σε χρόνο λόγω των βελτιστοποιήσεων, με αποτέλεσμα η «αφελής» έκδοση να είναι γρηγορότερη. Πρέπει να σημειωθεί, εντούτοις, ότι η generations είναι πολύ μικρή και απλή οντολογία, και ότι η καθυστέρηση στη βελτιστοποιημένη έκδοση είναι της τάξης των ms.

Σε όλες τις άλλες περιπτώσεις, οι βελτιστοποιήσεις έχουν σαφώς θετικό αποτέλεσμα. Το ποσοστό βελτίωσης του χρόνου εκτέλεσης κυμαίνεται από περίπου 5% έως πάνω από 60%, ανάλογα με την οντολογία. Ίσως όμως το πιο σημαντικό αποτέλεσμα να είναι ότι, στην περίπτωση της mygrid, που είναι και η μεγαλύτερη οντολογία, η αφελής έκδοση έτρεξε για χρόνο υπερ-πενταπλάσιο αυτού της βελτιστοποιημένης, πριν τερματίσει λόγω έλλειψης μνήμης. Αυτό αποτελεί άλλο ένα σοβαρό επιχείρημα υπέρ των βελτιστοποιήσεων, ότι δηλαδή μπορούν να χειριστούν περιπτώσεις στις οποίες η «αφελής» υλοποίηση αποτυγχάνει.

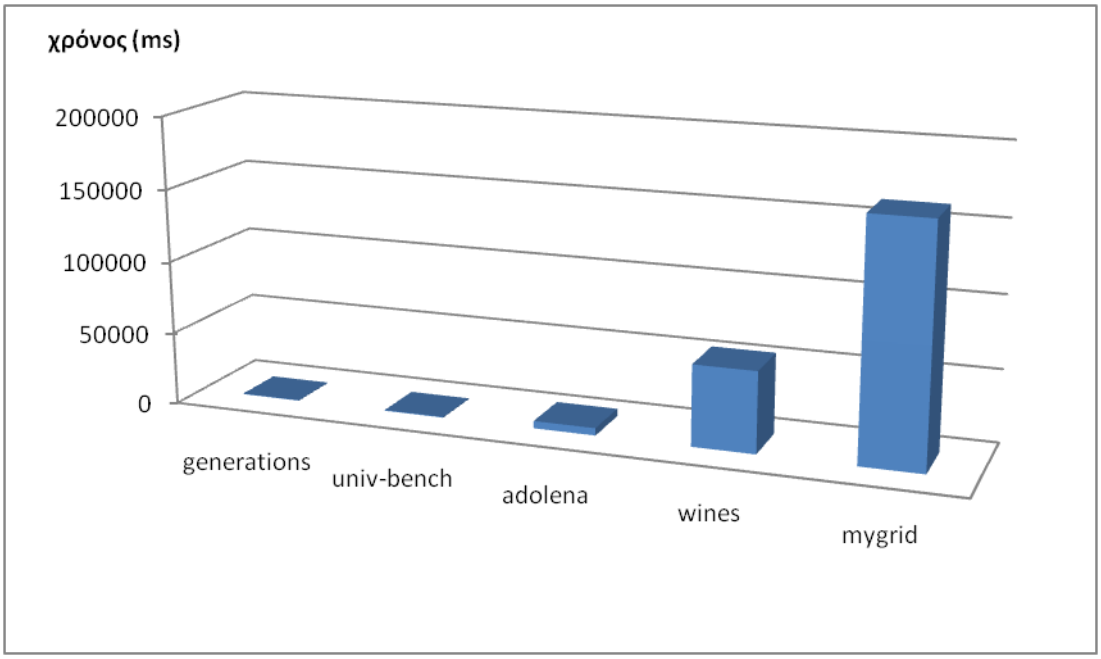
5.4 Χρόνος

Μία σημαντική παράμετρος που μελετήθηκε είναι, φυσικά, ο χρόνος εκτέλεσης του αλγορίθμου για τις διάφορες οντολογίες. Παρουσιάζει ενδιαφέρον η μελέτη και κατανόηση του πώς η δομή της οντολογίας επιδρά στο χρόνο εκτέλεσης. Για αυτό το σκοπό, μετρήθηκε ο συνολικός χρόνος εκτέλεσης του αλγορίθμου σημασιολογικής προσέγγισης και της αποθήκευσης της νέας οντολογίας. Ο μέσος όρος των χρόνων για τις οντολογίες ελέγχου φαίνεται στον παρακάτω πίνακα.

Οντολογία	**Εκφραστικότητα**	**Μέσος χρόνος (ms)**	**Μέγεθος αρχείου**
generations	$\mathcal{ALCOIF}$	342.1	13.5KB
univ-bench	$\mathcal{ALEHI}$(**D**)	505.9	14.1KB
adolena	$\mathcal{SRIQ}$	5081.2	60.8KB
wines	$\mathcal{ALEON}$(**D**)	55695.5	95.0KB
mygrid	$\mathcal{ALCHIF}$	163096.7	243KB

Πίνακας 4: Χρόνοι εκτέλεσης μετασχηματισμού για τις οντολογίες ελέγχου

Παρατηρούμε ότι ο χρόνος εκτέλεσης αυξάνεται καθώς αυξάνεται και το μέγεθος της οντολογίας, όπως ίσως ήταν αναμενόμενο, η αύξηση αυτή όμως δεν είναι αναλογική. Για παράδειγμα, η οντολογία wines είναι περίπου 1.5 φορές μεγαλύτερη από την adolena (ως προς το μέγεθος του αρχείου), αλλά απαιτεί 10 φορές περισσότερο χρόνο για το μετασχηματισμό της. Παρόμοια, η mygrid είναι 8 φορές μεγαλύτερη της adolena (και περιέχει περίπου τετραπλάσιο πλήθος αξιωμάτων κλάσεων), αλλά χρειάζεται πάνω από 30 φορές περισσότερο χρόνο, ενώ η adolena, περίπου 4 φορές μεγαλύτερη της univ-bench, απαιτεί δεκαπλάσιο χρόνο.



Σχήμα 20: Χρόνοι εκτέλεσης μετασχηματισμού για τις οντολογίες ελέγχου

Αυτό υποδεικνύει ότι το μέγεθος της οντολογίας (είτε ως μέγεθος αρχείου, είτε ως πλήθος αξιωμάτων) δεν είναι αρκετό για να μπορεί κανείς να κάνει υποθέσεις για το χρόνο που θα διαρκέσει ο μετασχηματισμός. Είναι λογικό να υποθέσουμε ότι σημαντικότερη επίδραση έχει η συγκεκριμένη δομή της οντολογίας, παρά το πλήθος των αξιωμάτων. Σε αυτό το σημείο αξίζει να σημειωθεί ότι μεγάλο μέρος του χρόνου εκτέλεσης, ιδιαίτερα στις wines και mygrid, αναλώνεται στην ταξινόμηση της οντολογίας, κάτι που σημαίνει ότι διαθέτουν αρκετά περίπλοκη ιεραρχία κλάσεων, που πιθανότατα συνεισφέρει στους υψηλούς χρόνους.

5.5 Πλήθος αξιωμάτων

Άλλη ενδιαφέρουσα παράμετρος είναι το πόσο αυξάνεται το μέγεθος της οντολογίας από τη διαδικασία του μετασχηματισμού. Όπως αναφέρθηκε σε προηγούμενη ενότητα, μπορεί κανείς

να παρατηρήσει ότι προστίθενται πολλά αξιώματα και ισχυρισμοί που είναι συνέπειες της αρχικής οντολογίας.

Οντολογία	Αξιώματα κλάσεων			Ισχυρισμοί κλάσεων			Ισχυρισμοί ιδιοτήτων		
	ΠΡΙΝ[4]	ΜΕΤΑ	ΑΥΞΗΣΗ	ΠΡΙΝ	ΜΕΤΑ	ΑΥΞΗΣΗ	ΠΡΙΝ	ΜΕΤΑ	ΑΥΞΗΣΗ
generations	34	100	66%	7	65	58%	9	15	6%
univ-bench	48	176	128%	0	0	-	0	0	-
adolena	200	10235	10035%	1	7	6%	0	1	-
wines	243	557	314%	187	1200	1013%	246	250	4%
mygrid	817	19364	18547%	0	0	-	0	0	-

Πίνακας 5: Πλήθος αξιωμάτων και ισχυρισμών πριν και μετά το μετασχηματισμό των οντολογιών ελέγχου

Είναι φανερό ότι το μέγεθος των οντολογιών αυξάνεται σημαντικά και ότι μεγάλο μέρος αυτής της αύξησης οφείλεται στα αξιώματα υπαγωγής που εισάγονται. Το γεγονός ότι οι ισχυρισμοί ιδιοτήτων δεν φαίνεται να αυξάνονται ιδιαίτερα μπορεί να αποδοθεί στο ότι οι οντολογίες ελέγχου περιέχουν σημαντικά μικρότερο αριθμό ιδιοτήτων απ' ό,τι κλάσεις, όπως φαίνεται και στον Πίνακα 2.

Είναι επίσης εμφανές ότι το ποσοστό της αύξησης των αξιωμάτων και ισχυρισμών δεν είναι σταθερό αλλά κυμαίνεται έντονα μέσα σε ένα μεγάλο εύρος τιμών. Τα αίτια αυτής της διακύμανσης θα πρέπει και πάλι να αναζητηθούν στη δομή των συγκεκριμένων οντολογιών, στις ιεραρχίες κλάσεων που ορίζουν. Για παράδειγμα, ενώ η οντολογία adolena έχει αρχικά λιγότερα αξιώματα κλάσεων από την wines, μετά το μετασχηματισμό καταλήγει με περισσότερα, καθώς το ποσοστό αύξησής τους είναι πάρα πολύ μεγαλύτερο αυτού των αξιωμάτων της wines, και παρότι η τελευταία έχει περισσότερες κλάσεις.

5.6 Αξιολόγηση με ερωτήματα

Ορισμένες από τις οντολογίες ελέγχου δεν περιέχουν άτομα, παρά μόνο κλάσεις και ιδιότητες. Για να μελετηθεί περισσότερο η συμπεριφορά του αλγορίθμου παρουσία ατόμων, κατασκευάστηκαν και ενσωματώθηκαν στην οντολογία univ-bench (που αρχικά δεν περιέχει κανένα άτομο) ορισμένα ABoxes, το καθένα κατασκευασμένο βάσει ενός ερωτήματος, όπως εξηγήθηκε στο προηγούμενο κεφάλαιο. Τα ερωτήματα που χρησιμοποιήθηκαν είναι τα εξής:

[4] Για τον υπολογισμό των αξιωμάτων κλάσεων πριν το μετασχηματισμό, κάθε αξίωμα ισοδυναμίας μέτρησε ως δύο αξιώματα.

	Σώμα ερωτήματος	Απαντήσεις
Q1	`Q(?0) <- Person(?0), worksFor(?0,?1), University(?1), hasAlumnus(?1,?0)`	10
Q2	`Q(?0) <- worksFor(?0,?1), affiliatedOrganizationOf(?1,?2)`	30
Q3	`Q(?0,?1) <- Person(?0), worksFor(?0,?1), Organization(?1)`	10
Q4	`Q(?0,?1) <- Person(?0), teacherOf(?0,?1), Course(?1)`	50
Q5	όπως το Q4	100
Q6	όπως το Q4	200

Πίνακας 6: Ερωτήματα που χρησιμοποιήθηκαν για την αξιολόγηση

Τα ερωτήματα Q1 – Q3 διαφέρουν ως προς τον αριθμό των κλάσεων και ρόλων που περιέχουν στο σώμα τους, καθώς και στον αριθμό των μεταβλητών που επιστρέφουν σε κάθε απάντηση. Τα Q4 – Q6 είναι το ίδιο ερώτημα, αλλά προσδιορίστηκε διαφορετικός αριθμός απαντήσεων κάθε φορά, ώστε να μελετηθεί η σχέση του χρόνου εκτέλεσης με τον αριθμό των ατόμων.

Ο αριθμός των ατόμων που εισήχθησαν και ο χρόνος εκτέλεσης του αλγορίθμου μετασχηματισμού παρουσιάζονται στον επόμενο πίνακα.

Ερώτημα	Άτομα	Χρόνος (ms)
Q1	13	891,7
Q2	88	2655,2
Q3	20	685,2
Q4	99	3514,1
Q5	197	11318,4
Q6	375	40374,9

Πίνακας 7: Χρόνοι εκτέλεσης μετασχηματισμού για τις οντολογίες που προκύπτουν βάσει των ερωτημάτων ελέγχου

Σημειώνεται ότι η διαφορά ανάμεσα στο πλήθος των ατόμων, σύμφωνα με τον Πίνακα 7, και το πλήθος που αναγράφεται στον Πίνακα 6, οφείλεται στο ότι ο Πίνακας 6 αναφέρεται σε αριθμό απαντήσεων του ερωτήματος, και όχι συνολικό αριθμό ατόμων που θα εισαχθούν. Επίσης, το SyGENiA έχει ένα μικρό περιθώριο σφάλματος κατά τη λειτουργία του, κάτι που εξηγεί τις άλλες αποκλίσεις.

Αρχικά παρατηρούμε ότι γενικά ο χρόνος εκτέλεσης αυξάνεται με την αύξηση του αριθμού των ατόμων. Μπορούμε να βρούμε μία εξαίρεση συγκρίνοντας τα Q1, Q3: ενώ το πρώτο εισάγει λιγότερο άτομα, οδηγεί σε μεγαλύτερο χρόνο εκτέλεσης. Αυτή η παρατήρηση φαίνεται να συμφωνεί με τα προηγούμενα συμπεράσματα, ότι δηλαδή το μέγεθος μίας οντολογίας δεν μπορεί να προσδιορίσει άμεσα την απόδοση του αλγορίθμου. Επίσης, η αύξηση δεν είναι ανάλογη του αριθμού των ατόμων. Αυτό είναι πιο εμφανές στην περίπτωση των Q4-Q6. Σε κάθε (περίπου) διπλασιασμό των ατόμων, ο χρόνος αυξάνεται περίπου 3-3.5 φορές.

Τα ερωτήματα Q4 - Q6 μπορούν να χρησιμεύσουν και για την αξιολόγηση μιας άλλης παραμέτρου, της συμπεριφοράς της βελτιστοποίησης για τον έλεγχο ισχυρισμών εννοιών (αλγόριθμος 4). Η συγκεκριμένη βελτιστοποίηση απαιτεί κάποιες κλάσεις να ορίζονται ως ξένες μεταξύ τους, αλλιώς δεν προσφέρει καμία βελτίωση. Τα ερωτήματα αναφέρονται σε άτομα που αναπαριστούν ανθρώπους και εργασίες, οπότε είναι (αρκετά) λογικό να θεωρήσουμε ότι οι αντίστοιχες κλάσεις είναι ξένες, και να δοκιμάσουμε τη συνεισφορά της βελτιστοποίησης.

Γι' αυτό το λόγο, μεταβάλλαμε την οντολογία univ-bench, δηλώνοντας τις κλάσεις Person και Work ως ξένες. Στη συνέχεια, παράγαμε νέα ABoxes για την τροποποιημένη οντολογία, για καθένα από τα ερωτήματα Q4 – Q6, ζητώντας τον ίδιο αριθμό απαντήσεων όπως και προηγουμένως, και μετρήσαμε εκ νέου το χρόνο εκτέλεσης του αλγορίθμου μετασχηματισμού. Ο επόμενος πίνακας περιέχει τα αποτελέσματα της σύγκρισης:

Ερώτημα	**Χρόνος πριν (ms)**	**Χρόνος μετά (ms)**	**Βελτίωση**
Q4	3514,1	3399	3,28%
Q5	11318,4	8106,1	28,38%
Q6	40374,9	26208	35,09%

Πίνακας 8: Χρόνοι εκτέλεσης πριν και μετά την τροποποίηση της οντολογίας univ-bench

Είναι σαφές ότι η νέα έκδοση της οντολογίας, δηλαδή με την προσθήκη του αξιώματος ξένων κλάσεων, οδηγεί σε καλύτερη απόδοση του αλγορίθμου, τουλάχιστον για τα συγκεκριμένα ερωτήματα. Αυτό μπορεί να αποδοθεί στη συνεισφορά της βελτιστοποίησης του αλγορίθμου 4, που εκμεταλλεύεται τις δηλώσεις ξένων κλάσεων για να περιορίσει το πλήθος των ισχυρισμών εννοιών που ελέγχονται. Η χρονική βελτίωση που προσφέρει δεν είναι τόσο μεγάλη στην περίπτωση του ερωτήματος Q4 (που εισάγει 99 άτομα), αυξάνεται όμως σημαντικά με την αύξηση του πλήθους των ατόμων.

5.7 Σύνοψη συμπερασμάτων αξιολόγησης

Το σημαντικότερο ίσως συμπέρασμα από τη μελέτη της συμπεριφοράς του συστήματος είναι η στενή εξάρτηση της επίδοσής του από τα χαρακτηριστικά της εκάστοτε οντολογίας. Μεταξύ όλων των περιπτώσεων μπορούμε να παρατηρήσουμε κάποια κοινά στοιχεία, όπως η αύξηση του μεγέθους της οντολογίας λόγω προσθήκης κυρίως αξιωμάτων υπαγωγής, όμως το μέγεθος και η έκταση των αλλαγών διαφέρει σημαντικά από τη μία στην άλλη οντολογία. Η επίδραση, επομένως, της δομής της οντολογίας στη συμπεριφορά του συστήματος φαίνεται να είναι ιδιαίτερα βαρύνουσα, κάτι που ίσως καθιστά δύσκολη την επιτυχημένη πρόβλεψη της επίδοσης βάσει μόνο απλών μετρικών μίας οντολογίας (π.χ. πλήθος κλάσεων, ιδιοτήτων, αξιωμάτων) και χωρίς γνώση της εσωτερικής δομής της.

Επιπλέον, οι παρατηρήσεις από τη διαδικασία αξιολόγησης τείνουν να επαληθεύσουν τη θεωρητική μελέτη, όσον αφορά τις βελτιστοποιήσεις. Επιβεβαιώνεται η συνεισφορά τόσο των βελτιστοποιήσεων των Pan – Thomas, όσο και των αλλαγών που προτάθηκαν σε αυτή την εργασία. Πιο συγκεκριμένα, εντοπίστηκε μία περίπτωση στην οποία είναι ιδιαίτερα εμφανής η απόδοση της βελτιστοποίησης του αλγορίθμου για την εύρεση των ισχυρισμών εννοιών. Η παρατήρηση αυτή είναι σύμφωνη με τη θεωρητική ανάλυση για τον αλγόριθμο που είχε παρουσιαστεί στο Κεφάλαιο 3.

6

Τεχνικές λεπτομέρειες

6.1 OWL API

Το OWL API [40] είναι ένα API για την αναπαράσταση και διαχείριση OWL οντολογιών σε Java, που αναπτύσσεται κατά κύριο λόγο από το University of Manchester [41]. Παρέχει έτοιμες κλάσεις και διαπροσωπείες (interfaces) που αντιπροσωπεύουν στοιχεία (όπως κλάσεις, ιδιότητες, άτομα, αξιώματα) μίας οντολογίας σε OWL, και που επιτρέπουν στο χρήστη να πραγματοποιήσει πληθώρα λειτουργιών, όπως δημιουργία νέων οντολογιών, αποθήκευση σε διάφορες μορφές, εισαγωγή νέων αξιωμάτων, διεξαγωγή συλλογιστικής κ.ά. Παρότι είναι σχεδιασμένο για την OWL, μπορεί να χρησιμοποιηθεί για την κατασκευή και διαχείριση οντολογιών γραμμένες σε άλλες, λιγότερο εκφραστικές, Περιγραφικές Λογικές, όπως σε DL-Lite (εξάλλου, η DL-Lite, όπως αναπτύχθηκε και περαιτέρω προηγουμένως, αποτελεί ουσιαστικά το προφίλ-υποσύνολο OWL QL της OWL 2). Το OWL API χρησιμοποιείται από πολλά προγράμματα και εργαλεία που διαχειρίζονται οντολογίες, όπως το Protégé [42] και το REQUIEM.

Το σύστημα που υλοποιήθηκε στα πλαίσια της διπλωματικής αυτής εργασίας χρησιμοποιεί την έκδοση 2.2.0 του OWL API, που υποστηρίζει την εκφραστικότητα της OWL 2. Το OWL API προσφέρει όλη την απαραίτητη υποστήριξη για το μετασχηματισμό οντολογιών, αλλά και για άλλα τμήματα του συστήματος. Στη συνέχεια θα αναλυθεί μέρος της

λειτουργικότητας που παρέχει. Η τελευταία έκδοση της πλήρους τεκμηρίωσης για τη διαπροσωπεία που ορίζει το OWL API μπορεί να βρεθεί στο [43].

6.1.1 Δημιουργία και αποθήκευση οντολογιών

Αρχικά, για να πραγματοποιηθεί οποιαδήποτε εργασία, πρέπει να ληφθεί ένα αντικείμενο OWLOntologyManager. Αυτό διαθέτει τις απαραίτητες μεθόδους για το φόρτωμα και την αποθήκευση οντολογιών. Μία οντολογία που είναι αποθηκευμένη σε ένα αρχείο μπορεί να φορτωθεί καλώντας τη μέθοδο loadOntologyFromPhysicalURI, η οποία δέχεται ως όρισμα το URI του αρχείου, που είναι της μορφής file:/path/to/file.owl. Το OWL API υποστηρίζει έναν αριθμό διαφορετικών μορφών σύνταξης για τις οντολογίες (μεταξύ αυτών, RDF/XML, OWL/XML, Manchester Syntax, OWL Functional Syntax). Συνεπώς, κατά το φόρτωμα, δοκιμάζονται διαφορετικοί αναλυτές (parsers), μέχρι κάποιος να επιτύχει. Εναλλακτικά, μπορεί να δημιουργηθεί μία νέα, κενή οντολογία με τη μέθοδο createOntology, προσδιορίζοντας το όνομά της μέσω ενός URI. Αντίστοιχη είναι η διαδικασία της αποθήκευσης, κατά την οποία μπορούν να καθοριστούν η μορφή (format) της οντολογίας, όπως επίσης και η θέση του αρχείου στο οποίο θα αποθηκευτεί. Αν αυτές οι παράμετροι δεν προσδιορίζονται, χρησιμοποιούνται εξ ορισμού τα χαρακτηριστικά που προσδιορίστηκαν κατά το φόρτωμα ή τη δημιουργία της οντολογίας.

Είναι σημαντικό να σημειωθεί ότι κάθε οντολογία στο OWL API συνοδεύεται από δύο URI: ένα φυσικό (physical URI), το οποίο προσδιορίζει τη θέση του αρχείου της οντολογίας, και ένα λογικό (logical ή ontology URI), το οποίο λειτουργεί ως όνομα της οντολογίας και δεν έχει αναγκαστικά φυσικό αντίκρισμα. Για να μπορέσει να χρησιμοποιείται το δεύτερο, πρέπει να οριστεί μία αντιστοίχιση μεταξύ αυτού και του φυσικού URI. Αυτό γίνεται με ένα αντικείμενο της κλάσης SimpleURIMapper, η οποία καθιερώνει ακριβώς αυτή την αντιστοιχία. Η αντιστοιχία στη συνέχεια πρέπει να ενσωματωθεί στον ontology manager, μέσω της μεθόδου addURIMapper.

Το παρακάτω απόσπασμα κώδικα περιλαμβάνει τις απαραίτητες ενέργειες για να φορτωθεί μία οντολογία, να δημιουργηθεί μία καινούρια και να αποθηκευτεί.

```
OWLOntologyManager manager = //δημιουργία ontology manager
                  OWLManager.createOntologyManager() ;
URI inURI = new URI("file:/user/home/ontology.owl") ;
OWLOntology ontology = //φόρτωμα οντολογίας
                  manager.loadOntologyFromPhysicalURI(inURI) ;
URI outOntologyURI =
      URI.create("http://my.ontologies.net/ontology-out.owl") ;
URI outPhysicalURI =
            URI.create("file:/user/home/ontology-out.owl") ;
```

```
SimpleURIMapper uriMapper = //δημιουργία αντιστοίχησης
            new SimpleURIMapper(ontologyURI, physicalURI) ;
manager.addURIMapper(uriMapper) ;
OWLOntology new_ontology = //δημιουργία κενής οντολογίας
                  manager.createOntology(ontologyURI) ;
manager.saveOntology(new_ontology) ; //αποθήκευση
```

Οι παραπάνω μέθοδοι παρουσιάζονται συνοπτικά στον παρακάτω πίνακα, μαζί με κάποιες που θα αναφερθούν στη συνέχεια.

Μέθοδος	**Λειτουργία**
createOntology	Δημιουργία κενής οντολογίας
loadOntologyFromPhysicalURI	Φόρτωμα ήδη υπάρχουσας οντολογίας
saveOntology	Αποθήκευση οντολογίας
addURIMapper	Προσθήκη αντιστοίχησης φυσικού-λογικού URI
getDataFactory	Λήψη data factory για πρόσβαση σε νέα στοιχεία
applyChange	Προσθήκη αξιώματος ή ισχυρισμού

Πίνακας 9: Βασικές μέθοδοι της διαπροσωπείας OWLOntologyManager

6.1.2 Πρόσβαση στα περιεχόμενα οντολογιών

Από τη στιγμή που μία οντολογία έχει φορτωθεί στον ontology manager, αντιπροσωπεύεται από ένα αντικείμενο OWLOntology. Το OWL API ορίζει διαπροσωπείες (και κλάσεις που τις υλοποιούν, αλλά δεν είναι ορατές στο χρήστη) για την αναπαράσταση κλάσεων, ιδιοτήτων και ατόμων μίας οντολογίας. Ορισμένες από αυτές αναφέρονται στον πίνακα που ακολουθεί.

Στοιχείο οντολογίας	**Στοιχείο OWL API**
Περιγραφή κλάσης (ανώνυμη ή μη)	OWLDescription
Επώνυμη (named) κλάση	OWLClass
Ιδιότητα αντικειμένων	OWLObjectProperty
Ιδιότητα δεδομένων	OWLDataProperty
Άτομο	OWLIndividual
Υπαρξιακός περιορισμός	OWLObjectSomeRestriction
Καθολικός περιορισμός	OWLObjectAllRestriction
Περιορισμός πληθικότητας	OWLObjectMinCardinalityRestriction, OWLObjectMaxCardinalityRestriction, OWLObjectExactCardinalityRestriction

Πίνακας 10: Αντιστοιχία στοιχείων οντολογιών και διαπροσωπειών του OWL API

Τα στοιχεία που ήδη περιέχονται σε μία οντολογία μπορούν να ληφθούν με τις μεθόδους getReferencedClasses, getReferencedObjectProperties και getReferencedIndividuals του αντίστοιχο αντικειμένου OWLOntology. Αυτά όμως είναι μόνο όσα στοιχεία έχουν όνομα. Για να αναφερθεί κάποιος σε ανώνυμες κλάσεις, π.χ. την ∃R.⊤, πρέπει να χρησιμοποιήσει ένα data factory. Αυτό αντιπροσωπεύεται με τη διαπροσωπεία OWLDataFactory και κατασκευάζεται από τη μέθοδο getOWLDataFactory του ontology manager που περιέχει την οντολογία. Το data factory παρέχει μεθόδους για την πρόσβαση σε σύνθετες έννοιες και ιδιότητες, καθώς και αξιώματα. Για παράδειγμα, η αναφορά στα ⊤ και ⊥ γίνεται με τις μεθόδους getOWLThing, getOWLNothing, για τη λήψη μίας αντίστροφης ιδιότητας χρησιμοποιείται η μέθοδος getOWLObjectPropertyInverse, για τη λήψη της άρνησης μίας κλάσης η μέθοδος getOWLObjectComplementOf, ενώ για τη λήψη ενός αξιώματος υπαγωγής η μέθοδος getOWLSubClassAxiom.

6.1.3 Εισαγωγή αξιωμάτων

Το OWL API παρέχει ακόμα τη δυνατότητα δημιουργίας νέων ισχυρισμών και αξιωμάτων και προσθήκη τους σε μία οντολογία. Όπως και με τις οντότητες που παρουσιάστηκαν στην προηγούμενη ενότητα, προσφέρει διαπροσωπείες για την αναπαράσταση αξιωμάτων.

Είδος αξιώματος	**Διαπροσωπεία OWL API**
Οποιοδήποτε αξίωμα	OWLAxiom
Υπαγωγή κλάσεων	OWLSubClassAxiom
Ισοδυναμία κλάσεων	OWLEquivalentClassesAxiom
Ξένων κλάσεων	OWLDisjointClassesAxiom
Ισχυρισμός κλάσης	OWLClassAssertionAxiom
Ισχυρισμός ιδιότητας αντικειμένων	OWLObjectPropertyAssertionAxiom

Πίνακας 11: Αντιστοιχία τύπων αξιωμάτων και διαπροσωπειών του OWL API

Ένα νέο αξίωμα μπορεί να ληφθεί χρησιμοποιώντας κατάλληλη μέθοδο του data factory, ανάλογα με το είδος του αξιώματος ή ισχυρισμού, και προσδιορίζοντας τις σχετικές κλάσεις, ιδιότητες ή άτομα που απαιτούνται. Τα ονόματα των μεθόδων είναι αντίστοιχα του είδους του αξιώματος που παράγεται, π.χ. getOWLEquivalentClassesAxiom, getOWLClassAssertionAxiom, κ.ο.κ. Στη συνέχεια, δημιουργείται ένα αντικείμενο της κλάσης AddAxiom με βάση το αξίωμα και την οντολογία. Τέλος, το αντικείμενο αυτό ενσωματώνεται στην οντολογία καλώντας τη μέθοδο applyChange του ontology manager. Στο παρακάτω παράδειγμα, θεωρούμε ότι η οντολογία είναι ήδη φορτωμένη στον manager

και ότι έχει ληφθεί το data factory, και προσθέτουμε ισχυρισμούς συνδυάζοντας όλα τα άτομα με όλες τις επώνυμες κλάσεις.

```
namedIndividuals = ontology.getReferencedIndividuals ;
namedClasses = ontology.getReferencedClasses
for (OWLIndividual o : namedIndividuals)
      for (OWLDescription A : namedClasses) {
            OWLAxiom ax = factory.getOWLClassAssertionAxiom(o, A) ;
            AddAxiom change = new AddAxiom(ontology, ax) ;
            manager.applyChange(change) ;
      }
```

Η διαδικασία εισαγωγής αξιωμάτων είναι κομβικής σημασίας στο σύστημα που υλοποιήθηκε. Το κομμάτι του μετασχηματισμού οντολογιών, μάλιστα, αποτελείται ουσιαστικά από τον κατάλληλο σχηματισμό αξιωμάτων, τον έλεγχό τους και την προσθήκη τους στη νέα οντολογία (μία ιδέα που, σε απλούστερη μορφή, αποτελεί ακριβώς τον πυρήνα της «αφελούς» υλοποίησης του μετασχηματισμού). Επιπρόσθετα, αν πραγματοποιηθεί προσθήκη ενός ABox πριν την υποβολή του ερωτήματος, πρέπει αυτό να ενσωματωθεί στην αρχική οντολογία, κάτι που γίνεται με τον ίδιο τρόπο: Λαμβάνονται όλοι οι ισχυρισμοί του νέου ABox, και προσθέτονται στην οντολογία όπως ακριβώς περιγράφηκε παραπάνω. Τέλος, η κανονικοποίηση της DL-Lite οντολογίας, που πραγματοποιείται από τον reasoner πριν την κατασκευή της βάσης δεδομένων, επίσης συνίσταται στην τροποποίηση και εισαγωγή αξιωμάτων και ισχυρισμών.

Εκτός από τα παραπάνω, το OWL API προσφέρει και άλλες διαπροσωπείες, από τις οποίες οι πιο σημαντικές για την εργασία είναι αυτές που έχουν να κάνουν με τη διεξαγωγή συλλογιστικής, και αναφέρονται στην αμέσως επόμενη ενότητα.

6.2 Pellet

Το Pellet [17] είναι ένα από τα πιο διαδεδομένα συστήματα συλλογιστικής. Προσφέρει τις συνήθεις υπηρεσίες συλλογιστικής (έλεγχος συνέπειας, συνεπαγωγής, ικανοποιησιμότητας, απάντηση ερωτημάτων) για την πλήρη εκφραστικότητα της OWL 2, κάνοντας χρήση αλγορίθμων tableau και υλοποιώντας μία σειρά βελτιστοποιήσεων για συγκεκριμένες περιπτώσεις που συνήθως συνεπάγονται αυξημένη πολυπλοκότητα. Εκτός των παραπάνω, προσφέρει τη δυνατότητα «διόρθωσης» οντολογιών των οποίων η εκφραστικότητα είναι μεγαλύτερη από την επιθυμητή (OWL Full αντί για OWL DL), ενώ επίσης παρέχει υποστήριξη για τα τρία προφίλ της OWL 2 [44]. Το Pellet είναι λογισμικό ανοιχτού κώδικα,

υλοποιημένο σε Java, που χρησιμοποιεί και μπορεί να συνδεθεί άμεσα με το OWL API. Η έκδοση που χρησιμοποιήθηκε είναι η 2.0.0.

Το κύριο τμήμα του συστήματος του Pellet είναι η κλάση Reasoner. Αυτή υλοποιεί πολλές διαπροσωπείες του OWL API σχετικές με τη συλλογιστική, μεταξύ αυτών τις OWLClassReasoner, OWLPropertyReasoner και OWLIndividualReasoner, που προδιαγράφουν συλλογιστική σε σχέση με κλάσεις, ιδιότητες και άτομα, αντίστοιχα, καθώς και την OWLReasoner, που συνδυάζει τις τρεις παραπάνω προδιαγραφές. Πριν από τη χρήση του συστήματος συλλογιστικής, πρέπει να ακολουθηθούν κάποια βήματα για την προετοιμασία του. Αρχικά, δημιουργείται ένα νέο αντικείμενο της κλάσης Reasoner., με βάση τον OWLOntologyManager που περιέχει την οντολογία. Στη συνέχεια, πρέπει να φορτωθεί η οντολογία, καλώντας τη μέθοδο loadOntology του νέου αντικειμένου Reasoner. Μετά από αυτό, το αντικείμενο Reasoner είναι έτοιμο για τη διεξαγωγή συλλογιστικής και μπορούν να κληθούν οι κατάλληλες μέθοδοί του, ανάλογα με την επιθυμητή υπηρεσία.

Στα πλαίσια της διπλωματικής, το Pellet χρησιμοποιείται κατά τη διαδικασία μετασχηματισμού της OWL οντολογίας. Συγκεκριμένα, γίνεται χρήση του για τους εξής σκοπούς:

- Ταξινόμηση της οντολογίας, με τη μέθοδο classify. Αυτό επιτρέπει τη χρήση των άλλων μεθόδων των σχετικών με την ιεραρχία κλάσεων και γίνεται κατά τη διάρκεια του υπολογισμού των αξιωμάτων υπαγωγής κλάσεων.
- Εύρεση των υποκλάσεων και υπερκλάσεων μίας κλάσης, με τις μεθόδους getSubClasses, getSuperClasses ή getDescendantClasses, getAncestorClasses. Η διαφορά των πρώτων δύο μεθόδων από τις άλλες δύο είναι ότι επιστρέφουν μόνο τις άμεσες υπο- ή υπερκλάσεις της κλάσης που δέχονται ως όρισμα, ενώ οι δύο τελευταίες επιστρέφουν και τις έμμεσες. Οι μέθοδοι αυτές εφαρμόζονται σε και επιστρέφουν μόνο επώνυμες κλάσεις. Για αυτό το λόγο απαιτείται η διαδικασία εισαγωγής ισοδύναμων επώνυμων κλάσεων για κάθε γενική κλάση, όπως περιγράφηκε στο Κεφάλαιο 3.
- Έλεγχος υπαγωγής δύο κλάσεων, με τη μέθοδο isSubClassOf. Μία κλήση της μορφής isSubClassOf(A,B) επιστρέφει αληθές αποτέλεσμα αν η A υπάγεται στη B, αλλιώς ψευδές. Χρησιμοποιείται για την εύρεση των αξιωμάτων υπαγωγής κλάσεων.
- Εύρεση των στιγμιότυπων μίας κλάσης, δηλαδή των ατόμων που ανήκουν σε αυτή, με τη μέθοδο getIndividuals. Αυτή επιστρέφει ένα σύνολο ατόμων που αποτελούν στιγμιότυπα της κλάσης-ορίσματος.
- Έλεγχος αν ένα άτομο είναι στιγμιότυπο μίας κλάσης ή αν ένα ζεύγος ατόμων είναι στιγμιότυπο μίας ιδιότητας. Αυτό γίνεται μέσω των μεθόδων hasType και hasObjectPropertyRelationship. Μία κλήση hasType(o,C) επιστρέφει αληθή τιμή αν

η οντολογία συνεπάγεται τον ισχυρισμό o:C, ενώ μία κλήση hasObjectPropertyRelationship(a,R,b) επιστρέφει αληθή τιμή αν η οντολογία συνεπάγεται τον ισχυρισμό (a,b):R. Χρησιμοποιείται για την εύρεση των ισχυρισμών εννοιών και των ισχυρισμών ρόλων.

- Έλεγχος του αν μία ιδιότητα είναι συναρτησιακή, με τη μέθοδο isFunctional. Χρησιμοποιείται στην εύρεση ισχυρισμών ρόλων, ως μέρος των βελτιστοποιήσεων.

6.3 REQUIEM

Το REQUIEM (REsolution-based QUery rewrIting for Expressive Models) [45] είναι ένα σύστημα που αναπτύχθηκε από το University of Oxford για το μετασχηματισμό ερωτημάτων. Δέχεται ως είσοδο ένα συζευκτικό ερώτημα και μία οντολογία, και επιστρέφει ένα νέο ερώτημα ή σύνολο ερωτημάτων που μπορεί να αποτιμηθεί σε μία βάση δεδομένων. Η ακριβής μορφή της εξόδου εξαρτάται από τη γλώσσα της οντολογίας, και το REQUIEM υποστηρίζει μέλη των οικογενειών της $\mathcal{EL}$ και της DL-Lite. Στην περίπτωση της DL-Lite, η έξοδος περιέχει ένα σύνολο συζευκτικών ερωτημάτων τα οποία πρέπει να υποβληθούν στη βάση. Με τα πακέτα που διανέμονται με την υλοποίηση, παρέχεται η δυνατότητα ελέγχου όλης της διαδικασίας, από την ανάλυση (parsing) του ερωτήματος μέχρι την αποτίμησή του.

Ο μετασχηματισμός ερωτημάτων συνίσταται στην εξής διαδικασία [46]: Δεδομένης μίας οντολογίας O και ενός συζευκτικού ερωτήματος Q, ζητείται να βρεθεί ένα ερώτημα Q_O τέτοιο ώστε οι απαντήσεις του Q πάνω στην οντολογία O και ένα οποιοδήποτε σύνολο δεδομένων (ABox) A να προκύπτουν από την αποτίμηση του Q_O πάνω στο A μόνο. Ουσιαστικά, δηλαδή, ένα μέρος της πληροφορίας της οντολογίας (συνήθως του TBox) κωδικοποιείται και ενσωματώνεται στο Q_O, που στη γενική περίπτωση είναι ένα σύνολο συζευκτικών ερωτημάτων. Αφού το χρήσιμο (για το ερώτημα Q) τμήμα της οντολογίας περιέχεται στο Q_O, δε χρειάζεται πλέον να συμβουλευτούμε την οντολογία κατά την απάντηση του ερωτήματος. Σημειώνεται ότι δε χρειάζεται να κωδικοποιηθεί όλη η πληροφορία της οντολογίας, αλλά μόνο εκείνο το τμήμα της που είναι σχετικό με την απάντηση του ερωτήματος. Τυπικά, αν με ans(Q,S) συμβολίσουμε το σύνολο των απαντήσεων του ερωτήματος Q πάνω στο σύνολο S, λέμε ότι το ερώτημα Q_O είναι μετασχηματισμός ή επαναδιατύπωση (rewriting) του ερωτήματος Q με βάση την οντολογία O αν $ans(Q,O \cup A) = ans(Q_O, A)$ για οποιοδήποτε A.

Ο αλγόριθμος που χρησιμοποιεί το REQUIEM παρουσιάζεται αναλυτικά στο [46], με το όνομα RQR (Resolution-based Query Rewriting). Ο αλγόριθμος εκτελεί κάποια ειδικά βήματα ανάλογα με τη γλώσσα στην οποία είναι γραμμένη η οντολογία. Για την περίπτωση της DL-Lite_R, αποτελεί βελτίωση ενός προηγούμενου αλγορίθμου που είχε προταθεί και χρησιμοποιηθεί για την ίδια διαδικασία [30]. Ο νέος αλγόριθμος υπερέχει του προηγούμενου

σε δύο σημεία: στο πώς χειρίζεται τους υπαρξιακούς περιορισμούς και στο πώς χειρίζεται αξιώματα που τους περιέχουν. Το αποτέλεσμα είναι ότι παράγει ισοδύναμους μετασχηματισμούς (δηλαδή που εξακολουθούν να δίνουν τις ίδιες απαντήσεις με το αρχικό ερώτημα), αλλά μικρότερου μεγέθους και συνήθως σε μικρότερο χρόνο.

Ο RQR δέχεται ως είσοδο μία οντολογία DL-Lite$_R$ O και ένα συζευκτικό ερώτημα Q, και εκτελεί τα παρακάτω βήματα.

1. Μετατροπή σε προτάσεις (clausification): Κάθε αξίωμα της O αντιστοιχίζεται σε μία πρόταση-ερώτημα, με βάση τον επόμενο πίνακα. Στο τέλος του βήματος, το ερώτημα Q και η οντολογία O έχουν μετατραπεί στο σύνολο προτάσεων (clauses) $\{Q\} \cup \Xi(O)$, όπου με $\Xi(O)$ συμβολίζεται το σύνολο των προτάσεων που λαμβάνονται από την O.

Πρόταση	**Αξίωμα DL-Lite$_R$**
$B(x) \leftarrow A(x)$	$A \sqsubseteq B$
$P(x,f(x)) \leftarrow A(x)$	$A \sqsubseteq \exists P$
$P(x,f(x)) \leftarrow A(x), B(f(x)) \leftarrow A(x)$	$A \sqsubseteq \exists P.B$
$P(f(x),x) \leftarrow A(x)$	$A \sqsubseteq \exists P^-$
$P(f(x),x) \leftarrow A(x), B(f(x)) \leftarrow A(x)$	$A \sqsubseteq \exists P^-.B$
$A(x) \leftarrow P(x,y)$	$\exists P \sqsubseteq A$
$A(x) \leftarrow P(y,x)$	$\exists P^- \sqsubseteq A$
$S(x,y) \leftarrow P(x,y)$	$P \sqsubseteq S, P^- \sqsubseteq S^-$
$S(x,y) \leftarrow P(y,x)$	$P \sqsubseteq S^-, P^- \sqsubseteq S$

Πίνακας 12: Αντιστοιχίες αξιωμάτων της οντολογίας και προτάσεων

Να σημειωθεί ότι οι συναρτήσεις που εισάγονται σε κάθε πρόταση είναι μοναδικές, δηλαδή συνδέονται μονοσήμαντα με το αξίωμα από το οποίο προήλθαν (παρόλο που στον παραπάνω πίνακα παριστάνονται όλες με το σύμβολο *f*)

2. Κορεσμός (saturation): Στο στάδιο αυτό, παράγονται νέες προτάσεις συνδυάζοντας τις ήδη υπάρχουσες. Συγκεκριμένα, για κάθε ζευγάρι προτάσεων C_1, C_2, προστίθενται όλα τα δυνατά αποτελέσματα της συνάρτησης resolve(C_1,C_2). Η λειτουργία της συνάρτησης resolve ορίζεται από ένα σύνολο από πρότυπα επαγωγής (inference templates), δηλαδή κανόνες που καθορίζουν τις προϋποθέσεις υπό τις οποίες μπορούν να συνδυαστούν δύο προτάσεις καθώς και τη μορφή της πρότασης που προκύπτει ως αποτέλεσμα. Κάποια από τα πρότυπα αυτά παρουσιάζονται, ενδεικτικά, στον ακόλουθο πίνακα.

Πρόταση 1	Πρόταση 2	Αποτέλεσμα
$C(x) \leftarrow B(x)$	$B(f(x)) \leftarrow A(x)$	$C(f(x)) \leftarrow A(x)$
$B(x) \leftarrow P(x,y)$	$P(x,f(x)) \leftarrow A(x)$	$B(x) \leftarrow A(x)$
$B(x) \leftarrow P(x,y)$	$P(f(x),x) \leftarrow A(x)$	$B(f(x)) \leftarrow A(x)$
$B(x) \leftarrow P(y,x)$	$P(x,f(x)) \leftarrow A(x)$	$B(f(x)) \leftarrow A(x)$
$B(x) \leftarrow P(y,x)$	$P(f(x),x) \leftarrow A(x)$	$B(x) \leftarrow A(x)$
$S(x,y) \leftarrow P(x,y)$	$P(x,f(x)) \leftarrow A(x)$	$S(x,f(x)) \leftarrow A(x)$
$S(x,y) \leftarrow P(x,y)$	$P(f(x),x) \leftarrow A(x)$	$S(f(x),x) \leftarrow A(x)$
$S(x,y) \leftarrow P(y,x)$	$P(x,f(x)) \leftarrow A(x)$	$S(f(x),x) \leftarrow A(x)$
$S(x,y) \leftarrow P(y,x)$	$P(f(x),x) \leftarrow A(x)$	$S(x,f(x)) \leftarrow A(x)$

Πίνακας 13: Πρότυπα επαγωγής για τη συνάρτηση resolve

Οι νέες προτάσεις που εισάγονται είναι πιθανό να μπορούν να συνδυαστούν με τις παλιές ή μεταξύ τους με εφαρμογή κάποιου προτύπου, δίνοντας με τη σειρά τους καινούριες. Το βήμα αυτό επαναλαμβάνεται έως ότου να μην προκύψουν νέες προτάσεις.

3. Ξεδίπλωμα (unfolding) και κλάδεμα (pruning): Στο βήμα αυτό επιτελείται μία σειρά από εργασίες για την τελική επεξεργασία του συνόλου των ερωτημάτων που έχει προκύψει. Αρχικά, απορρίπτονται όσες προτάσεις περιέχουν συναρτήσεις, με εφαρμογή της συνάρτησης ff. Αν N ένα σύνολο προτάσεων, ff(N) είναι το υποσύνολο των προτάσεων του N που δεν περιέχουν συναρτήσεις. Στη συνέχεια, στο προκύπτον σύνολο εφαρμόζεται η συνάρτηση unfold, η οποία συνδυάζει τις προτάσεις του συνόλου «ξεδιπλώνοντάς» τες τη μία μέσα στην άλλη, όπου αυτό είναι δυνατόν [47]. Για παράδειγμα, αν $N = \{A(x) \leftarrow B(x),\ B(x) \leftarrow C(x)\}$, τότε $unfold(N) = N \cup \{A(x) \leftarrow C(x)\}$. Τέλος, διατηρούνται μόνο τα ερωτήματα που έχουν την ίδια κεφαλή με το αρχικό ερώτημα Q, ενώ τα υπόλοιπα απορρίπτονται. Το σύνολο των ερωτημάτων που προκύπτει, και που είναι ερμηνεύεται ως ένωση αυτών των ερωτημάτων, αποτελεί το μετασχηματισμό του Q με βάση την O.

Συνοπτικά, ο αλγόριθμος παρουσιάζεται στο παρακάτω σχήμα.

Αλγόριθμος 5: RQR
Είσοδοι: Συζευκτικό ερώτημα Q, $DL\text{-}Lite_R$ οντολογία O
Έξοδος: Μετασχηματισμός του Q με βάση την O

R := {Q} ∪ Ξ(O) //*μετατροπή σε προτάσεις*

repeat
    for all $C_1, C_2 \in R$:
        $R := R \cup \text{resolve}(C_1,C_2)$ //*κορεσμός*
until δεν μπορούν να προστεθούν νέα ερωτήματα στο R
$Q_O := \{C \mid C \in \text{unfold}(\text{ff}(R)), C$ έχει την ίδια κεφαλή με το Q$\}$ //*ξεδίπλωμα και κλάδεμα*
return Q_O

Σχήμα 21: Αλγόριθμος RQR για DL-Lite$_R$ οντολογίες

Ως παράδειγμα για τη λειτουργία του αλγορίθμου, μπορούμε να θεωρήσουμε την οντολογία Ο, που αποτελείται μόνο από το αξίωμα

$$\text{Καθηγητής} \sqsubseteq \exists \text{διδάσκει}.\text{Μαθητής}$$

και το ερώτημα

$$Q(x) \leftarrow \textit{διδάσκει}(x,y) \wedge \textit{Μαθητής}(y)$$

Οι ενέργειες που γίνονται σε κάθε βήμα του αλγορίθμου είναι οι ακόλουθες:

Μετατροπή σε προτάσεις: Σύμφωνα με τις αντιστοιχίες του Πίνακα 9, το αξίωμα δίνει τις εξής δύο προτάσεις:

$$\textit{διδάσκει}(x,f(x)) \leftarrow \textit{Καθηγητής}(x)$$

$$\textit{Μαθητής}(f(x)) \leftarrow \textit{Καθηγητής}(x)$$

Κορεσμός: Με επαναλαμβανόμενες εφαρμογές της συνάρτησης resolve βάσει των προτύπων του Πίνακα 10, προσθέτονται σταδιακά οι προτάσεις:

$$Q(x) \leftarrow \textit{Καθηγητής}(x) \wedge \textit{Μαθητής}(f(x))$$

$$Q(x) \leftarrow \textit{διδάσκει}(x,f(x)) \wedge \textit{Καθηγητής}(x)$$

$$Q(x) \leftarrow \textit{Καθηγητής}(x)$$

Η επανάληψη σταματά σε αυτό το σημείο, γιατί δεν μπορούν να προστεθούν άλλες προτάσεις.

Ξεδίπλωμα και κλάδεμα: Πρώτα απορρίπτονται όλες οι προτάσεις που περιέχουν συναρτήσεις, επομένως από τις έξι παραπάνω που είχαν συγκεντρωθεί απομένουν μόνο οι

$$Q(x) \leftarrow \textit{διδάσκει}(x,y) \wedge \textit{Μαθητής}(y) \text{ και}$$

$$Q(x) \leftarrow \textit{Καθηγητής}(x)$$

Επειδή δεν μπορεί να εφαρμοστεί η διαδικασία του ξεδιπλώματος στις προτάσεις αυτές, απομένει μόνο να απορριφθούν όσες δεν έχουν ως κεφαλή *Q(x)*, που προφανώς σημαίνει ότι γίνονται δεκτές και οι δύο. Συνεπώς, ο μετασχηματισμός του Q με βάση την Ο αποτελείται από το σύνολο των παραπάνω δύο ερωτημάτων.

6.4 SyGENiA

Το SyGENiA [33] είναι ένα εργαλείο παραγωγής δεδομένων για οντολογίες του Σημασιολογικού Ιστού. Δέχεται μία οντολογία (γραμμένη σε RDF(S) ή OWL) και ένα συζευκτικό ερώτημα, και παράγει ένα ή περισσότερα ABoxes με τους απαραίτητους ισχυρισμούς ώστε το ερώτημα να έχει απαντήσεις. Σκοπός του είναι να διευκολύνει την αξιολόγηση των υπηρεσιών απάντησης συζευκτικών ερωτημάτων [48]. Συγκεκριμένα, αναπτύχθηκε για να μετρηθεί η πληρότητα της διαδικασίας απάντησης ερωτημάτων για διάφορα συστήματα συλλογιστικής. Η χρήση του έδωσε αποτελέσματα που αφενός διευκρινίζουν προηγούμενες έρευνες, προσδιορίζοντας πιο συγκεκριμένα σύνολα δεδομένων στα οποία παρατηρείται μη-πληρότητα, αφετέρου ανατρέπουν προηγούμενες εντυπώσεις: για παράδειγμα, στα σύνολα δεδομένα που παράγει παρατηρείται ορισμένες φορές μη-πλήρης απάντηση ερωτημάτων, ενώ σε προηγούμενες μελέτες το ίδιο σύστημα συλλογιστικής φαινόταν να είναι πλήρες.

Προσφέρει πολλές δυνατότητες παραμετροποίησης, μεταξύ των οποίων η επιλογή για το αν θα παραχθούν ένα ή περισσότερα ABoxes. Στην πρώτη περίπτωση, που είναι αυτή που χρησιμοποιείται στο σύστημα που αναπτύχθηκε, μπορεί επίσης θα ελεγχθεί το πότε θα σταματήσει η κατασκευή του ABox, ορίζοντας είτε ένα μέγιστο αριθμό απαντήσεων για το ερώτημα, είτε ένα μέγιστο αριθμό ισχυρισμών που θα περιέχει το ABox. Ένα παράδειγμα χρήσης του είναι το παρακάτω απόσπασμα από την κλάση SygeniaInterface, στο οποίο φαίνεται η αρχικοποίηση και η δημιουργία του ABox, έχοντας δεδομένες τις θέσεις της οντολογίας και του ερωτήματος.

```
//αρχικοποίηση
Configuration config = new Configuration() ;
config.varInstantiation = VariableInstantiationStrategy.INJECTIVE;
config.tbType = TestingBaseType.STRICT;

long totalNumberOfIndividuals = 3000;
PERFORMANSS syntheticGenerator =
      new PERFORMANSS(config, totalNumberOfIndividuals );

//συνθήκες τερματισμού
String terminationCondition = "TotalNumOfCertainAnswers" ;
long numberOfCertainAnswers = 4 ;

//δημιουργία ABox
syntheticGenerator.createSyntheticABox(ontologyFile, queryFile,
      terminationCondition, numberOfCertainAnswers) ;
```

6.5 HyperSQL

Το HyperSQL [34] (HSQL ή HSQLDB) είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS) ανοιχτού κώδικα γραμμένο σε Java. Χρησιμοποιείται σε πλήθος εφαρμογών, μεταξύ των οποίων το OpenOffice και το Mathematica. Στα πλαίσια της εργασίας, γίνεται χρήση του για τη δημιουργία της βάσης δεδομένων από την DL-Lite οντολογία, καθώς και για την εισαγωγή και άντληση στοιχείων από αυτήν για τη διεξαγωγή συλλογιστικής. Η έκδοση που χρησιμοποιείται είναι η 2.0.0.

7

Επίλογος

7.1 Σύνοψη και συμπεράσματα

Στα πλαίσια της παρούσας διπλωματικής εργασίας, αναπτύχθηκε μία υλοποίηση για τον αλγόριθμο μετασχηματισμού γνώσης από την OWL DL στην DL-Lite. Η υλοποίηση βασίστηκε στο [31], αλλά στη λύση που προτείνεται εκεί ενσωματώθηκαν και κάποιες νέες βελτιστοποιήσεις. Η υλοποίηση χρησίμευσε ως κεντρικό στοιχείο για την ανάπτυξη ενός συστήματος απάντησης ερωτημάτων, το οποίο σχεδιάστηκε, υλοποιήθηκε και ελέγχθηκε επιτυχώς χρησιμοποιώντας μία σειρά από διαθέσιμες οντολογίες διαφορετικής εκφραστικότητας.

Η αξιολόγηση του συστήματος έδειξε ότι μπορούμε να εξάγουμε κάποια απλά συμπεράσματα για τη συμπεριφορά του, όπως το ότι το μέγεθος της προκύπτουσας οντολογίας είναι σημαντικά μεγαλύτερο αυτού της αρχικής. Ωστόσο, φαίνεται πως δεν αρκεί να γνωρίζουμε αριθμητικά στοιχεία της οντολογίας για να προβλέψουμε την επίδοση του αλγορίθμου ως προς το χρόνο εκτέλεσης ή το ποσοστό αύξησης του μεγέθους. Αντίθετα, φαίνεται πως η επίδοση εξαρτάται έντονα από την εσωτερική δομή της κάθε οντολογίας.

Τα αποτελέσματα δείχνουν ότι η λύση του μετασχηματισμού γνώσης αποτελεί μία αποδοτική και εύχρηστη απάντηση στο πρόβλημα της υψηλής πολυπλοκότητας της συλλογιστικής με Περιγραφικές Λογικές, καθώς προσφέρει καλή επίδοση και επιθυμητές ιδιότητες, μεταξύ αυτών ορθότητα και πληρότητα της συλλογιστικής. Αυτό κάνει τη μέθοδο αυτή ιδιαίτερα

προσφερόμενη για χρήση σε συστήματα βασισμένα σε Περιγραφικές Λογικές, συμπεριλαμβανομένων και εφαρμογών του Σημασιολογικού Ιστού.

7.2 Μελλοντικές επεκτάσεις

Μία πολλά υποσχόμενη επέκταση των Περιγραφικών Λογικών είναι ο συνδυασμός τους με την ασαφή (fuzzy) λογική. Οι ασαφείς Περιγραφικές Λογικές μπορούν να βρουν εφαρμογή σε μεγάλη ποικιλία προβλημάτων, όπως σε θέματα επεξεργασίας εικόνας. Προσφέρονται ιδιαίτερα για συλλογιστική σε περιβάλλοντα όπου η γνώση δεν είναι πάντα σίγουρη, καθώς μπορούν να μοντελοποιήσουν αυτή την αβεβαιότητα, κάτι που ξεφεύγει από την εκφραστική δυνατότητα των κλασικών Περιγραφικών Λογικών. Ένα τέτοιο περιβάλλον είναι και ο ίδιος ο Σημασιολογικός Ιστός [49], κάτι που διευρύνει τις πιθανές χρήσεις αυτής της επέκτασης [50]. Έχουν προταθεί υλοποιήσεις και αναπτυχθεί συστήματα ασαφών Περιγραφικών Λογικών (όπως για παράδειγμα στα [51], [52]), χωρίς ωστόσο να υπάρχει ακόμα κάποιο πρότυπο (κατ' αντίστοιχα, π.χ., με αυτό της OWL για τις κλασικές γλώσσες αναπαράστασης οντολογιών).

Μία από τις προτεινόμενες χρήσεις τους είναι η γενίκευση της DL-Lite στην ασαφή Περιγραφική Λογική f-DL-Lite [53], στην οποία η απάντηση ερωτημάτων δεν συνίσταται πλέον απλώς στην επιστροφή συνόλων ατόμων, αλλά συνοδεύεται από ένα βαθμό βεβαιότητας για κάθε άτομο που επιστρέφεται. Η διαδικασία του μετασχηματισμού που αναπτύχθηκε στην εργασία μπορεί να επεκταθεί ώστε να υποστηρίζει και ασαφείς Περιγραφικές Λογικές, αυξάνοντας έτσι τη χρηστικότητά της και ταυτόχρονα συνεισφέροντας στην πιο αποδοτική χρήση των τελευταίων.

Μία άλλη πιθανή επέκταση που θα μπορούσε να μελετηθεί είναι η επιλεκτική εφαρμογή των βελτιστοποιήσεων κατά τη διαδικασία της μετατροπής. Το πόσο αποδοτικές είναι εξαρτάται από τη δομή της οντολογίας και το πλήθος και είδος των αξιωμάτων που περιέχει. Συνεπώς, αν μπορέσουν να προσδιοριστούν χαρακτηριστικά οντολογιών (π.χ. πλήθος κάποιου συγκεκριμένου είδους αξιωμάτων, σχέση του πλήθους των εννοιών με το πλήθος των ατόμων) για τα οποία οι διάφορες βελτιστοποιήσεις έχουν καλή ή κακή απόδοση, θα μπορούσε να ενσωματωθεί μία ευριστική διαδικασία που θα ελέγχει το ποιες από τις διαθέσιμες βελτιστοποιήσεις θα εφαρμοστεί στην υπό εξέταση οντολογία. Έτσι, θα μπορέσει να αποφευχθεί η εφαρμογή μεθόδων που δε θα έχουν καλή επίδοση, και θα βελτιωθεί η συμπεριφορά του συνολικού συστήματος.

8

Βιβλιογραφία

[1] Ronald J. Brachman and Hector J. Levesque, *Knowledge Representation and Reasoning*., 2004.

[2] Marvin Minsky, "A framework for representing knowledge," in *Mind Design*, John Haugeland, Ed.: MIT Press, 1981, pp. 95-128.

[3] Roger C. Schank and Robert P. Abelson, *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*., 1977.

[4] John F. Sowa, "Semantic Networks," in *Encyclopedia of Artificial Intelligence*, Stuart C Shapiro, Ed., 1987.

[5] Ronald J. Brachman and James G. Schmolze, "An Overview of the KL-ONE Knowledge Representation System," *Cognitive Science*, vol. 9, no. 2, pp. 171-216, 1985.

[6] Robert A. Kowalski, "Predicate logic as a programming language," *Information Processing*, no. 74, pp. 569-574, 1974.

[7] Tim Berners-Lee, James Hendler, and Ora Lasilla, "The Semantic Web," *Scientific American*, 2001.

[8] W3C. (2004, February) RDF Primer. [Online]. http://www.w3.org/TR/rdf-primer/

[9] W3C. (2004, February) RDF Vocabulary Description Language 1.0: RDF Schema. [Online]. http://www.w3.org/TR/rdf-schema/

[10] W3C. (2004, February) OWL Web Ontology Language Reference. [Online]. http://www.w3.org/TR/owl-ref/

[11] Marco Schaerf and Marco Cadoli, "Tractable Reasoning via Approximation," *Artificial Intelligence*, vol. 74, no. 2, 1995.

[12] Franz Baader, Deborah L. McGuinness, Danielle Nardi, and Peter F. Patel-Schneider, *The Description Logic Handbook: Theory, implementation, and applications*., 2002.

[13] Manfred Schmidt-Schauss and Gert Smolka, "Attributive Concept Descriptions with Complements," *Artificial Intellgience*, vol. 48, no. 1, pp. 1-26, 1991.

[14] Γιώργος Στοΐλος, Εισαγωγή στις Περιγραφικές Λογικές, 2007.

[15] Ian Horrocks and Ulrike Sattler, "A Description Logic with Transitive and Inverse Roles and Role Hierachies," *Journal of Logic and Computation*, vol. 9, no. 3, pp. 385-410, 1999.

[16] Franz Baader and Ulrike Sattler, "An Overview of Tableau Algorithms for Description Logics," *Studia Logica*, vol. 69, no. 1, pp. 5-40, 2001.

[17] Clark & Parsia. Pellet. [Online]. http://clarkparsia.com/pellet

[18] Fact++. factplusplus - Project Hosting on Google Code. [Online]. http://code.google.com/p/factplusplus/

[19] University of Oxford Information Systems Group. HermiT Reasoner. [Online]. http://hermit-reasoner.com/

[20] Stephen Tobies, "Complexity Results and Practical Algorithms for Logics in Knowledge Representation," LuFG Theoretical Computer Science, RWTH, Aachen, Germany, PhD Thesis 2001.

[21] Franz Baader, Sebastian Brand, and Carsten Lutz, "Pushing the EL Envelope," in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, 2005.

[22] Evgeny Zolin. Description Logic Complexity Navigator. [Online]. http://www.cs.man.ac.uk/~ezolin/dl/

[23] W3C. (2009, October) OWL 2 Web Ontology Language Document Overview. [Online]. http://www.w3.org/TR/owl2-overview/

[24] W3C. (2009, October) OWL 2 Web Ontology Language Profiles. [Online]. http://www.w3.org/TR/owl2-profiles/

[25] W3C. (2009, October) OWL 2 Web Ontology Language Primer. [Online]. http://www.w3.org/TR/owl2-primer/

[26] Γιώργος Στοΐλος, Γλώσσες Αναπαράστασης Γνώσης στο Σημασιολογικό Ιστό, 2007.

[27] Ian Horrocks and Peter Patel-Schneider, "Reducing OWL Entailment to Description Logic Satisfiability," *Web Semantics*, vol. 1, no. 4, 2004.

[28] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Riccardo Rosati, and Guido Vetere, "DL-Lite: Practical Reasoning for Rich DLs," in *Proceedings of the 2004 International Workshop on Description Logics*, 2004.

[29] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati, "DL-Lite: Tractable Description Logics for Ontologies," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 20, 2005, pp. 602-607.

[30] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati, "Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family," *Journal of Automated Reasoning*, vol. 39, no. 3, pp. 385-429, 2007.

[31] Jeff Z. Pan and Edward Thomas, "Approximating OWL-DL ontologies," in *Proceedings of the 22nd AAAI Conference*, 2007, pp. 1434-1439.

[32] Perry Groot, Heiner Stuckenschmidt, and Holger Wache, "Approximating Description Logic Classification for Semantic Web Reasoning," in *The Semantic Web: Research and Applications*., 2005, pp. 318-332.

[33] Giorgos Stoilos. sygenia- Project Hosting on Google Code. [Online]. http://code.google.com/p/sygenia/

[34] HSQLDB. [Online]. http://hsqldb.org/

[35] Matthew Horridge. Generations ontology. [Online]. http://protege.cim3.net/file/pub/ontologies/generations/generations.owl

[36] Lehigh University. Lehigh University Benchmark. [Online]. http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl

[37] Adolena ontology. [Online]. http://owl.cs.manchester.ac.uk/repository/download?ontology=http://ksg.meraka.co.za/adolena.owl&format=RDF/XML

[38] Wine ontology. [Online]. http://www.w3.org/TR/owl-guide/wine.rdf

[39] myGrid. myGrid ontology. [Online]. http://www.mygrid.org.uk/ontology/

[40] OWL API. [Online]. http://owlapi.sourceforge.net/

[41] University of Manchester. Information Management Group. [Online]. http://img.cs.man.ac.uk/

[42] Stanford University. The Protégé Ontology Editor and Knowledge Acquisition System. [Online]. http://protege.stanford.edu/

[43] Overview (The OWL API). [Online]. http://owlapi.sourceforge.net/javadoc/index.html

[44] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz, "Pellet: A practical OWL-DL reasoner," *Journal of Web Semantics*, vol. 5, no. 2, 2007.

[45] University of Oxford Computing Laboratory. REsolution-based QUery rewrIting for Expressive Models. [Online]. http://www.comlab.ox.ac.uk/projects/requiem/home.html

[46] Héctor Pérez-Urbina, Ian Horrocks, and Boris Motik, "Efficient Query Answering for OWL 2," in *Proceedings of the 8th International Semantic Web Conference*, 2009 , pp. 489-504.

[47] Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks, "Tractable Query Answering Under Description Logic Constraints," *Journal of Applied Logic*, vol. 8, no. 2, June 2010.

[48] Giorgos Stoilos, Bernardo Cuenca Grau, and Ian Horrocks, "How Incomplete is Your Semantic Web Reasoner? Systematic Analysis of the Completeness of Query Answering Systems," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[49] Giorgos Stoilos, Nikos Simou, Giorgos Stamou, and Stefanos Kollias, "Uncertainty and the Semantic Web," *IEEE Intelligent Systems*, vol. 21, no. 5, pp. 84-87, 2006.

[50] Giorgos Stoilos and Giorgos Stamou, "Extending Fuzzy Description Logics for the Semantic Web," in *3rd International Workshop of OWL: Experiences and Directions*, Innsbruck, 2007.

[51] Nikos Simou and Stefanos Kollias, "FiRE: A Fuzzy Reasoning Engine for Imprecise Knowledge," in *K-Space PhD Students Workshop*, Berlin, 2007.

[52] Giorgos Stoilos, Giorgos Stamou, Vassilis Tzouvaras, Jeff Z. Pan, and Ian Horrocks, "Fuzzy OWL: Uncertainty and the Semantic Web," in *International Workshop of OWL: Experiences and Directions*, Galway, 2005.

[53] Jeff Z. Pan, Giorgos Stamou, Giorgos Stoilos, Edward Thomas, and Stuart Taylor,

"Scalable Querying Services over Fuzzy Ontologies," in *Proceeding of the 17th international conference on World Wide Web*, Beijing, 2007.