ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Βελτιστοποιημένες μηχανές συλλογιστικής για Εκφραστικές Περιγραφικές Λογικές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΤΡΙΒΕΛΑ ΔΕΣΠΟΙΝΑΣ

Επιβλέπων : **Γιώργος Στάμου**
Λέκτορας Ε.Μ.Π.

Αθήνα, Μάιος 2009

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

Βελτιστοποιημένες μηχανές συλλογιστικής για Εκφραστικές Περιγραφικές Λογικές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΤΡΙΒΕΛΑ ΔΕΣΠΟΙΝΑΣ

Επιβλέπων : **Γιώργος Στάμου**
Λέκτορας Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26η Μαΐου 2009.

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

Αθήνα, Μάιος 2009

(Υπογραφή)

.....

Δέσποινα Τριβέλα

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

Περίληψη

Σκοπός της διπλωματικής εργασίας ήταν η ανάπτυξη μίας μηχανής συλλογιστικής για την περιγραφική λογική $\mathcal{ALC}$. Για το λόγο αυτό υλοποιήθηκε ο αλγόριθμος εξαγωγής συμπερασμάτων tableau.

Συγκεκριμένα, η μηχανή που κατασκευάστηκε, καλείται να επεξεργαστεί οντολογίες εκφραστικότητας συμβατής με αυτή της περιγραφικής λογικής $\mathcal{ALC}$. Τα αρχεία οντολογιών που αποτελούν την είσοδο της μηχανής, επεξεργάζονται για την κατασκευή της βάσης γνώσης. Ο τρόπος με τον οποίο οι κανόνες tableau εφαρμόζονται στα δεδομένα της γνώσης, αναπαρίσταται σχηματικά από δασική δομή.

Δόθηκε ιδιαίτερη προσοχή ώστε να χρησιμοποιηθούν τεχνικές βελτιστοποιήσης κατά την ανάπτυξη του αλγορίθμου. Οι τεχνικές αφορούν τη μείωση του χώρου αναζήτησης με σκοπό την ταχύτερη απόκριση, καθώς και τη μείωση των αυξημένων απαιτήσεων σε πόρους του συστήματος. Στην περίπτωση που το σύστημα δέχεται ως είσοδο οντολογίες υψηλότερης εκφραστικότητας, είναι σε θέση να τις επεξεργαστεί, αγνοώντας τα αξιώματα που συνθέτουν οι περισσότερο εκφραστικοί τελεστές.

Λέξεις κλειδιά:

άτομο, επέκταση, ερμηνεία, ετικέτα, ιδιότητα, ικανοποιησιμότητα, ισοδυναμία, ισχυρισμός, κατασκευαστές, κλάση, μοντέλο, μπλοκάρισμα, ξεδίπλωμα, ορολογία, περιοσμός τιμής, σημασιολογία, σύγκρουση, συμπερασμός, συμπλήρωμα, συναρτησιακός, συνεπαγωγή, συνέπεια, σώμα, τυπικό, υπαγωγή, υπαρξιακός περιορισμός.

Abstract

The purpose of this diploma thesis is the development of a reasoning machine for ALC description logics. For this purpose, we implemented the tableau algorithm.

In particular, the reasoner's goal is to process ontologies written in description languages that support the ALC expressiveness. The processing of input ontology files results to the construction of a knowledge base. The application of the tableau algorithm to the knowledge base data is represented by a forest model.

Care was taken in introducing optimization techniques while implementing the algorithm. These techniques aim to speeding up the resoner's response by minimizing the search space, as well as minimizing the machine's demands on system resources.

Should the input ontologies be described by more expressive languages, the reasoning machine is able to process them, by ignoring all the axioms that use constructors of higher expressiveness.

Keywords:

individual, expansion, interpretation, tag, property, satisfiability, equivalence, assertion, constructors, class, model, blocking, unfolding, terminology, value restriction, semantics, clash, inference, complement, functional, entailment, consistency, body, formal, subsumption, existential restriction.

Ευχαριστίες

Για την εκπόνηση της διπλωματικής εργασίας θα ήθελα να ευχαριστήσω κατά κύριο λόγο τον καθηγητή μου κ. Γ. Στάμου, που με την αμέριστη συμπαράσταση και τις πάντοτε καίριες και χρήσιμες συμβουλές του υπήρξε ο καθοδηγητής μου στην προσπάθεια αυτή. Επίσης, η εργασία αυτή δεν θα είχε έρθει εις πέρας χωρίς την πολύτιμη συμβολή και τις παρεμβάσεις του διδάκτορα Γ. Στοΐλου. Τέλος, ευχαριστώ θερμά τον συνάδελφο και φίλο Γιάννη για την υποστήριξή του.

Πίνακας περιεχομένων

1 *Εισαγωγή*

1.1 Μηχανές συλλογιστικής για Περιγραφικές Λογικές

Η αναπαράσταση γνώσης, ως κλάδος της τεχνητής νοημοσύνης, διαπραγματεύεται την απόδοση της πληροφορίας με συμβολικό τρόπο. Η δομή αυτή της πληροφορίας, επιτρέπει στις μηχανές συλλογιστικής περιγραφικών λογικών να επεξεργάζονται τη γνώση, με σκοπό όχι απλά την ανάκτηση δεδομένων, αλλά την εξαγωγή νέας γνώσης. Με άλλα λόγια, οι μηχανές συλλογιστικής, ασχολούνται με τον τρόπο που θα χειριστούν τη δεδομένη γνώση ώστε να καταλήξουν σε κάποιο συμπέρασμα. Η λειτουργία τους είναι μια υπολογιστική διαδικασία που αντιστοιχίζεται στη σκέψη.

Οι μηχανές συλλογιστικής ενδέχεται να χρησιμοποιηθούν με διαφορετικούς τρόπους ανάλογα με την εφαρμογή και τις απαιτήσεις του χρήστη. Σε κάθε περίπτωση, τίθεται κάποιο ερώτημα στο οποίο καλείται να απαντήσει, δεδομένης της γνώσης. Το ερώτημα διαφέρει από εκείνο που τίθεται σε μία βάση δεδομένων, καθώς αφορά την εξαγωγή κάποιου συμπεράσματος. Ο τρόπος με τον οποίο γίνεται η επεξεργασία της γνώσης και ο αλγόριθμος για την εξαγωγή των συμπερασμάτων διαφέρει από τη μία μηχανή συλλογιστικής στην άλλη.

Πρωταρχικό μέλημα αποτελεί ασφαλώς, η ορθότητα του αλγορίθμου. Δεδομένης της ορθότητας, τα στοιχεία που χαρακτηρίζουν τη μηχανή συλλογιστικής, είναι η ταχύτητα απόκρισης καθώς και οι απαιτήσεις της σε πόρους συστήματος. Όσο περισσότερο εκφραστική είναι η γλώσσα περιγραφικής λογικής την οποία χειρίζεται η μηχανή, τόσο περισσότερο αυξάνει η πολυπλοκότητα του αλγορίθμου που υλοποιεί. Επίσης, οι απαιτήσεις της σε πόρους συστήματος εξαρτώνται από το μέγεθος της γνώσης που καλείται να διαχειριστεί.

1.2 Αντικείμενο διπλωματικής

Σκοπός της διπλωματικής ήταν η κατασκευή μίας μηχανής συλλογιστικής για την περιγραφική λογική $\mathcal{ALC}$. Η λειτουργία της έγκειται στην εξαγωγή συμπεράσματος που αφορά τη συνέπεια ή την ασυνέπεια μίας οντολογίας.

Συγκεκριμένα, στο σύστημα εισάγεται ένα αρχείο οντολογίας, από την ανάγνωση του οποίου αντλούνται όλες εκείνες οι πληροφορίες που είναι απαραίτητες για την κατασκευή της βάσης γνώσης του συστήματος συλλογιστικής. Η βάση γνώσης περιλαμβάνει τους ορισμούς εννοιών καθώς επίσης και ένα σύνολο ισχυρισμών. Στόχος της μηχανής είναι ο έλεγχος της αλήθειας των ισχυρισμών δεδομένης της γνώσης του συστήματος. Ο αλγόριθμος που υλοποιεί τη συλλογιστική διαδικασία είναι ο αλγόριθμος εξαγωγής συμπεράσματος tableau. Στην περίπτωση που οι ισχυρισμοί είναι αληθείς στο σύνολό τους, η βάση γνώσης χαρακτηρίζεται συνεπής. Διαφορετικά, εάν τουλάχιστον ένας από τους ισχυρισμούς είναι ψευδής, η μηχανή αποφαίνεται για την ασυνέπεια της βάσης.

Μία από τις δυσκολίες της κατασκευής της μηχανής συλλογιστικής είναι, όπως προαναφέρμε, οι αυξημένες απαιτήσεις σε πόρους του συστήματος. Ο μεγάλος χώρος αναζήτησης που αναπτύσσεται κατά την εκτέλεση του αλγορίθμου, αυξάνει τις απαιτήσεις σε μνήμη και καθυστερεί την απόκριση της μηχανής. Για το σκοπό αυτό χρησιμοποιήθηκαν τεχνικές βελτιστοποίησης οι οποίες πρόκειται να περιγραφούν λεπτομερώς σε επόμενη ενότητα. Στόχος τους είναι η μείωση του χώρου αναζήτησης καθώς και ο περιορισμός των ενεργειών που εκτελεί ο αλγόριθμος στις απολύτως απαραίτητες, δηλαδή κατά απαίτηση του συστήματος.

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

Μελετήσαμε τις περιγραφικές λογικές με έμφαση στην εκφραστικότητα της $\mathcal{ALC}$, καθώς και την αναπαράσταση της γνώσης σε οντολογίες γραμμένες σε γλώσσα OWL.

Υλοποιήσαμε τον αλγόριθμο tableau.

Ενσωματώσαμε τεχνικές βελτιστοποίησης του αλγορίθμου.

Τέλος, αξιολογήσαμε την μηχανή για διαφορετικές οντολογίες εισόδου.

1.3 Οργάνωση κειμένου

Εργασίες σχετικές με το αντικείμενο της διπλωματικής παρουσιάζονται στο Κεφάλαιο 2. Στο Κεφάλαιο 3 παραθέτουμε το απαραίτητο θεωρητικό υπόβαθρο για την κατανόηση της λειτουργίας μίας μηχανής συλλογιστικής. Στο Κεφάλαιο 4 παρουσιάζουμε τις λειτουργίες του συστήματος, ενώ, στο Κεφάλαιο 5 πιο αναλυτικά, παρουσιάζουμε την αρχιτεκτονική του. Ο αλγόριθμος που συνθέτει τον κορμό του συστήματος αναπτύσσεται στο Κεφάλαιο 6. Στο Κεφάλαιο 7 αναφέρουμε τους ελέγχους που πραγματοποιήθηκαν και στο Κεφάλαιο 8 παραθέτουμε ιδέες και συμπεράσματα.

2 *Σχετικές εργασίες*

2.1 Συστήματα προσπέλασης οντολογιών

Το εργαλείο που χρησιμοποιήθηκε για την προσπέλαση των οντολογιών είναι το OWL API. Πρόκειται για μία Java διεπαφή για τη γλώσσα Web Ontology Language της W3C, OWL, που αναπτύχθηκε αρχικά στο πανεπιστήμιο του Manchester. Εστιάζει κυρίως σε οντολογίες γραμμένες σε OWL Lite, OWL DL και OWL 2. Επιτρέπει την προσπέλαση και τη σύνταξη οντολογιών και είναι δυνατό να χρησιμοποιηθεί από μηχανές συλλογιστικής όπως ο Pellet και ο Fact++.

Το OWL-API διαθέτει ένα μεγάλο αριθμό πακέτων. Αναφέρουμε παρακάτω τα σημαντικότερα από αυτά τα οποία μάλιστα χρησιμοποιήσαμε κατά την ανάπτυξη του συστήματος.

Ένα από τα σημαντικότερα πακέτα για την ανάγνωση της οντολογίας είναι το org.semanticweb.owl.model.

Οι οντολογίες, όπως θα δούμε αναλυτικά στο επόμενο κεφάλαιο, περιλαμβάνουν ένα σύνολο αξιωμάτων, μέσω των οποίων ορίζονται οι κλάσεις, οι ιδιότητες, καθώς και τα άτομα αυτής. Μέσω της κλάσης OWLOntology του πακέτου org.semanticweb.owl.model, μπορούμε να εξάγουμε τα αξιώματα καθώς και όλες τις οντότητες της οντολογίας. Οι έννοιες, οι ιδιότητες και τα άτομα της βάσης γνώσης αντιπροσωπεύονται από τις κλάσεις OWLClass, OWLObjectProperty και OWLIndividual αντίστοιχα.

Το OWL-API προσφέρει τις κλάσεις OWLSubClassAxiom, OWLEquivalentClassesAxiom για την ανάκτηση των αξιωμάτων υπαγωγής και ισοδύναμων κλάσεων. Τα αξιώματα αυτά είναι που ορίζουν τις έννοιες στην περιγραφική λογική $\mathcal{ALC}$. Αντίστοιχα, ορίζονται και οι ιδιότητες που συσχετίζουν δύο άτομα της γνώσης, με την κλάση OWLObjectPropertyAxiom.

Τα αξίωματα των ατόμων που υποστηρίζονται από την $\mathcal{ALC}$ σχετίζουν δύο άτομα μεταξύ τους, μέσω κάποιας ιδιότητας, OWLObjectPropertyAssertionAxiom, ή ορίζουν ένα άτομο ως υπόσταση μίας έννοιας, OWLClassAssertionAxiom.

Στο πακέτο org.semanticweb.owl.apibinding ανήκει και η κλάση OWLManager. Ο OWLManager μας επιτρέπει να φορτώσουμε την οντολογία ώστε έπειτα να την χειριστούμε και να την επεξεργαστούμε μέσω κάποιας μηχανής συλλογιστικής.

2.2 Συστήματα συλλογιστικής για Περιγραφικές Λογικές

Μία από τις πλέον γνωστές μηχανές συλλογιστικής είναι ο Pellet που υποστηρίζει την OWL DL. Πρόκειται για μία ανοικτού κώδικα εφαρμογή, υλοποιημένη σε Java. Παρέχει πληθώρα διεπαφών κάποια από τις οποίες υποστηρίζει και το OWL API.

Οι αλγόριθμοι συλλογιστικής που χρησιμοποιεί ο Pellet, βασίζονται στους αλγορίθμους tableau που αναπτύχθηκαν για εκφραστικές Περιγραφικές Λογικές. Η ορθότητα και πληρότητα της υπολλογιστικής διαδικασίας είναι εξασφαλισμένη για την OWL DL χωρίς nominals (SHIN(D)) και την OWL DL χωρίς αντίστροφες ιδιότητες (SHON(D)). Τέλος, ο συνδυασμός των αλγορίθμων που χρησιμοποιούνται εγγυάται την ορθότητα αλλά όχι και την πληρότητα της διαδικασίας όσον αφορά την OWL DL.

Από τα βασικά χαρακτηριστικά του Pellet, που τον διαφοροποιούν από άλλες μηχανές συλλογιστικής που θα αναφέρουμε παρακάτω, είναι η δυνατότητα να επεξεργάζεται προβλήματα συνεπαγωγής. Τα περισσότερα συστήματα συλλογιστικής διαπραγματεύονται προβλήματα ικανοποιησιμότητας και συνεπαγωγής ενώ δεν υποστηρίζουν άμεσα ερωτήματα συνεπαγωγής. Ο χειρισμός τους γίνεται έμεσα, αφού προηγουμένως αναχθούν σε πρόβλημα ικανοποιησιμότητας.

Άλλο βασικό χαρακτηριστικό είναι ο έλεγχος της ορθότητας των τύπων δεδομένων XML Schema ένος εγγράφου. Πιο αναλυτικά, ένα XML Schema σύστημα διαθέτει ένα σύνολο τύπων δεδομένων (datatypes) το οποίο μπορεί να επεκταθεί με την κατασκευή νέων τύπων που προκύπτουν από τον συνδυασμό των ήδη υπάρχοντων. Γενικά, ενδέχεται οι νέοι τύποι να γίνονται αποδεκτοί χωρίς να είναι αναγκαία ικανοποιήσιμοι. Ο Pellet αναλαμβάνει τον έλεγχο ικανοποιησιμότητας των νέων τύπων δεδομένων.

Τέλος, αναφέρουμε σχετικά με τον Pellet, τη δυνατότητα να ελέγχει εάν οι βάσεις γνώσης που είναι γραμμένες σε OWL Full, είναι δυνατό να μετατραπούν σε OWL DL. Συγκεκριμένα, το πρόβλημα παρουσιάζεται όταν οι συγγραφείς των xml/rdf εγγράφων προτίθενται να χρησιμοποιήσουν την OWL DL ενώ το αποτέλεσμα προκύπτει σε OWL Full. Κάτι τέτοιο είναι πολύ εύκολο να συμβεί για το λόγο ότι πρέπει να ικανοποιείται μία σειρά

από περιορισμούς των rfd γράφων. Ο Pellet ενσωματώνει μεθόδους για τον εντοπισμό των αρχείων σε OWL Full που όμως είναι δυνατό να κατασκευαστούν σε OWL DL και τα επιδιορθώνει.

Η μηχανή συλλογιστικής *FaCT++* αφορά την Περιγραφική Λογική SHOIQ (και κατά συνέπεια την OWL DL). Είναι υλοποιημένη σε C++ και είναι γνωστή για νέες τεχνικές βελτιστοποίησης που εισάγει κατά την εφαρμοφή του tableau αλγορίθμου σε εκφραστικές περιγραφικές λογικές.

Οι τεχνικές βελτιστοποίησης αφορούν την προεπεξεργασία της βάσης γνώσης όταν φορτώνεται στη μηχανή συλλογιστικής, ώστε να κανονικοποιηθεί και να αναπαρασταθεί με κατάλληλο τρόπο, καθώς και την καθεαυτή διαδικασία συλλογιστικής η οποία μπορεί να αφορά είτε τον έλεγχο ικανοποιησιμότητας είτε την ιεραρχία των εννοιών. Χαρακτηριστικά, αναφέρουμε ότι ο Fact++ στα πλαίσια των βελτιστοποιήσεων, ενσωματώνει ευριστικές συναρτήσεις με τις οποίες επιλέγεται κάθε φορά ο κανόνας που θα εκτελεστεί βάσει μιας σειράς παραμέτρων που σχετίζονται με το μέγεθος μίας έννοιας, ακόμα και τη συχνότητα χρησιμοποίησης τους. Οι τεχνικές αυτές βελτιώνουν σημαντικά τις επιδόσεις της μηχανής.

Η σημαντικότερη καινοτομία του Fact++ που τον διαχωρίζει από τις υπόλοιπες μηχανές είναι ότι εισάγει ένα νέο τρόπο εκτέλεσης των κανόνων tableau. Δηλαδή, δεν εφαρμόζονται στους κόμβους όπως συνήθως με φορά από πάνω προς τα κάτω, αλλά εισάγεται μία λίστα ToDo. Η ιδέα της λίστας είναι ότι οι κόμβοι εισάγονται σε αυτή κάθε φορά που προστίθεται μία νέα έννοια. Η λίστα ταξινομεί τα στοιχεία της με κάποια σειρά και εν συνεχεία εκτελούνται οι κανόνες tableau. Από την εφαρμογή των κανόνων, οι κόμβοι επεξεργάζονται και αφαιρούνται από τη λίστα.

Μία από τις διεπαφές που χρησιμοποιεί ο Fatc++ για τις OWL οντολογίες είναι η DIG interface (DL Implementation Group). Ο DIG χρησιμοποιείται επίσης και από τον Pellet.

Άλλο ένα παράδειγμα μηχανής συλλογιστικής είναι ο RACER , υλοποιημένος σε Lisp. Ο RACER μπορεί να χειρίζεται μεγάλου μεγέθους Aboxes σε συνδυασμό με εκφραστικά TBoxes. Παρέχει υπηρεσίες εξαγωγής συμπερασμάτων για προχωρημένες εφαρμογές με οντολογίες. Επίσης πέρα από την OWL υποστηρίζει την nRQL (new Racer Query Language) με την οποία μπορεί να απαντά σε εκφραστικά ερωτήματα (queries).

3 *Θεωρητικό υπόβαθρο*

3.1 Περιγραφικές Λογικές

Οι Περιγραφικές Λογικές είναι μία οικογένεια γλωσσών αναπαράστασης γνώσης που χρησιμοποιούν ένα δομημένο και αυστηρώς ορισμένο (τυπικό) τρόπο κατά την αναπαράσταση της γνώσης. Οι Περιγραφικές Λογικές αναπαριστούν τη γνώση ορίζοντας πρώτα τις έννοιες που δομούν τον 'κόσμο' (ορολογία), τις οποίες κατόπιν, χρησιμοποιούν για να αποδώσουν τις σχέσεις ανάμεσα στα αντικείμενα της γνώσης (του κόσμου). Ο όρος 'περιγραφικές' αφορά τον τρόπο με τον οποίο δίνονται οι ορισμοί της γνώσης, δηλαδή με περιγραφές εννοιών. Ο όρος ' λογικές' σχετίζεται με τη σημασιολογία της γλώσσας η οποία είναι βασισμένη στη μαθηματική Λογική.

Τα βασικά δομικά συστατικά μίας περιγραφικής λογικής είναι οι έννοιες όπως προαναφέρθηκε, οι ιδιότητες και τέλος τα άτομα, ή αλλιώς τα στιγμιότυπα- υποστάσεις των εννοιών. Η εκφραστικότητα της περιγραφικής λογικής καθορίζει τη δυνατότητά της να περιγράψει περισσότερο σύνθετες έννοιες. Καθορίζεται από τους κατασκευαστές που υποστηρίζει, οι οποίοι συνδυάζουν ατομικές- στοιχειώδεις έννοιες για τον ορισμό μίας νέας, σύνθετης. Εν γένει , η κάθε περιγραφική λογική υποστηρίζει το μικρότερο δυνατό σύνολο κατασκευαστών προκειμένου να μπορεί να αναπαραστήσει τις έννοιες που απαιτούνται ανάλογα με την εκάστοτε εφαρμογή.

Ένα ξεχωριστό χαρακτηριστικό των περιγραφικών λογικών είναι ότι παρέχουν τη δυνατότητα να εξαχθεί από τη βάση γνώσης, γνώση, η οποία δεν προκύπτει άμεσα από τους ορισμούς τον εννοιών αλλά μπορεί να εννοηθεί. Επιτρέπουν επίσης, την ιεράρχηση εννοιών και ατόμων του κόσμου.

Η ιεράρχηση των εννοιών ορίζει υποέννοιες, γεγονός που συνεισφέρει στην καλύτερη δόμηση της βάσης γνώσης. Επιπλέον, η συσχέτιση μεταξύ διαφορετικών εννοιών μέσω της

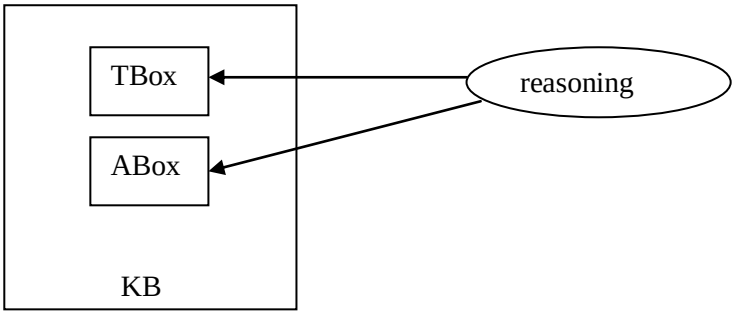
ιεράρχησής τους, είναι μία επιπρόσθετη χρησιμη πληροφορία για τη διαδικασία εξαγωγής συμπερασμάτων. Παράλληλα, η ιεράρχηση των ατόμων μπορεί να καθορίσει εάν ένα δοθέν άτομο θα είναι πάντα στιγμιότυπο μίας ορισμένης έννοιας.

Σε ό,τι αφορά τους μηχανισμούς συλλογιστικής για τις περιγραφικές λογικές, καλλούνται να αναλάβουν μία διαδικασία απόφασης. Η διαδικασία αυτή απαντά είτε θετικά είτε αρνητικά, και τερματίζει πάντοτε. Η απάντηση που προκύπτει από ένα μηχανισμό συλλογιστικής πρέπει να είναι εξασφαλισμένη και μάλιστα σε πεπερασμένο χρόνο. Αυτό δεν βεβαίως, δε σημαίνει ότι ο εν λόγω χρόνος μπορεί πάντοτε, να είναι μικρός ή έστω πρακτικά ανεκτός. Εγείρονται επομένως, τα δύο σημαντικότερα ζητήματα που αφορούν τους μηχανισμούς εξαγωγής συμπερασμάτων για τις περιγραφικές λογικές: η αποφασισιμότητα και η πολυπλοκότητα του αλγορίθμου που υλοποιούν.

Οι εκφραστικές περιγραφικές λογικές τείνουν να απαιτούν υψηλής πολυλπλοκότητας διαδικασίες εξαγωγής συμπερασμάτων. Από την άλλη πλευρά οι λιγότερο εκφραστικές λογικές ενδέχεται να είναι ανεπαρκείς στην αναπαράσταση των εννοιών κάποιας εφαρμογής. Γίνεται αντιληπτό ότι ένα από τα σημαντικότερα ζητήματα που αφορούν τους ερευνητές που εργάζονται στο χώρο της αναπαράστασης γνώσης, είναι η προσπάθεια εξισορρόπησης ανάμεσα στην εκφραστικότητα της περιγραφικής λογικής και την πολυπλοκότητα των προβλημάτων εξαγωγής συμπεράσματος.

Βάσει των όσων προαναφέρθηκαν, μπορούμε πιο εύκολα, να κατανοήσουμε τη δομή ενός συστήματος αναπαράστησης γνώσης το οποίο βασίζεται σε περιγραφικές λογικές. Ένα τέτοιο σύστημα αποτελείται από τη βάση γνώσης, τον κόσμο του συστήματος. Η βάση γνώσης ορίζεται με γλώσσες περιγραφικών λογικών και πάνω σε αυτή επενεργούν οι μηχανισμοί συλλογιστικής για την εξαγωγή συμπερασμάτων.

Τα συστατικά της βάσης γνώσης είναι δύο: το σώμα των ορολογιών (Terminology Box, TBox), όπου ορίζονται οι έννοιες που συνθέτουν τον κόσμο, και το σώμα των σχέσεων (Assertion Box, ABox), όπου δηλώνονται τα άτομα του κόσμου ως στιγμιότυπα εννοιών (type assertions) καθώς και οι σχέσεις μεταξύ των ατόμων (property assertions).



Ο τρόπος με τον οποίο οι μηχανισμοί συλλογιστικής επενργούν στη βάση γνώσης για την εξαγωγή συμπερασμάτων, καθώς επίσης και η φύση των ερωτημάτων στα οποία καλείται να απαντήσει ο μηχανισμός συλλογιστικής θα εξεταστούν σε επόμενη ενότητα.

3.1.1 Συντακτικό και σημασιολογία

Σε αυτό το τμήμα θα αναφερθούμε στην περισσότερο διαδεδομένη οικογένεια περιγραφικών λογικών την $\mathcal{AL}$ (*A*ttributive concept *L*anguage).

Οι περιγραφές των εννοιών ακολουθούν τους παρακάτω συντακτικούς κανόνες.

Έστω A μία ατομική έννοια και R ένας ατομικός ρόλος. Οι σύνθετες έννοιες C, D ορίζονται:

C, D → A |

$\top$ | (καθολική έννοια, universal concept)

$\bot$ | (κενή έννοια, bottom concept)

$\neg A$ | (άρνηση ατομικής έννοιας, atomic negation)

C⊓D | (τομή, intersection)

$\forall R.C$ | (καθολικός περιορισμός, value restriction)

$\exists R.\top$ |(περιορισμένος υπαρξιακός περιορισμός, limited existential restriction)

Για παράδειγμα, μπορούμε να περιγράψουμε την έννοια Γυναίκα ως εξής:

Γυναίκα ≡ Άνθρωπος ⊓ Θηλυκό

όπου οι έννοιες *Άνθρωπος* και *Θηλυκό* είναι ατομικές, ενώ η *Γυναίκα* είναι σύνθετη έννοια.

Επίσης Μητέρα είναι κάθε γυναίκα που έχει παιδί, δηλαδή

Μητέρα ≡ Γυναίκα ⊓ (ΞΈχειΠαιδί.Άνθρωπος).

Προκειμένου να αποδώσουμε τη σημασιολογία των *$\mathcal{AL}$-εννοιών*, θεωρούμε την ερμηνεία *I*, η οποία ορίζεται ως ένα ζεύγος $(\Delta^I, \cdot^I)$, όπου το σύνολο Δ^I είναι ο χώρος ερμηνείας (domain) και αποτελείται από τα αντικείμενα (objects), και η συνάρτηση $\cdot^I$, αντιστοιχίζει σε κάθε ατομική έννοια A ένα σύνολο $A^I \subseteq \Delta^I$ και σε κάθε ατομικό ρόλο R ένα σύνολο $R^I \subseteq \Delta^I \times \Delta^I$. Η συνάρτηση ερμηνείας επεκτείνεται και δίνει την ερμηνεία σε περιγραφές εννοιών.

$\top^I = \Delta^I$

$\bot^I = \emptyset$

$(C \sqcap D)^I = C^I \cap D^I$

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

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

Δύο έννοιες C, D είναι ισοδύναμες, $C \equiv D$, όταν για όλες τις ερμηνείες I ισχύει $C^{I} = D^{I}$.

Προσθέτουμε τώρα νέους κατασκευαστές στη γλώσσα με αποτέλεσμα να αυξηθεί η εκφραστικότητά της.

Η ένωση δύο εννοιών (Union, $\mathcal{U}$) συμβολίζεται ως $C \sqcup D$ και η ερμηνεία της έχει ως εξής:

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

Ο πλήρης υπαρξιακός περιορισμός (Full Existential Quantification, $\mathcal{E}$) συμβολίζεται ως $\exists R.C$ και η ερμηνεία του είναι:

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

Με αυτούς τους δύο επιπρόσθετους τελεστές η γλώσσα $\mathcal{AL}$ επεκτείνεται στην εκφραστικότερη $\mathcal{ALUE}$.

Εισάγουμε επίσης, την άρνησης σύνθετης έννοιας (Complement, $\mathcal{C}$):

$(\neg C)^{I} = \Delta^{I} \setminus C^{I}$.

Παρατηρούμε ότι με τον κατασκευαστή της άρνησης μπορούν να ορισθούν η ένωση και ο πλήρης υπαρξιακός περιορισμός. Δηλαδή, $C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$ και $(\exists R.C) \equiv \neg\forall R.\neg C$. Αντίστροφα, η ένωση και ο πλήρης υπαρξιακός περιορισμός μπορούν να ορίσουν την άρνηση σύνθετης έννοιας. Γενικά, οποιαδήποτε γλώσσα υποστηρίζει την άρνηση σύνθετης έννοιας θεωρούμε ότι υποστηρίζει και τους τελεστές της ένωσης και του πλήρους υπαρξιακού περιορισμού. Έτσι, η γλώσσα $\boldsymbol{\mathcal{ALUE}}$ είναι ισοδύναμη της $\boldsymbol{\mathcal{ALC}}$.

Η εκφραστικότητα αυξάνεται ακόμα περισσότερο όταν προσθέσουμε τους κατασκευαστές περιορισμού πληθυκότητας (Numerical restrictions, N). Ο κατασκευαστής αυτός συμβολίζεται ως εξής : $\leq nR,\ \geq nR$ όπου n είναι ένας θετικός ακέραιος αριθμός. Η ερμηνεία των κατασκευαστών εξηγείται παρακάτω:

$(\leq nR)^{I} = \{a \in \Delta^{I} \mid \left|\{b \mid (a, b) \in R^{I}\}\right| \leq n\}$,

$(\geq nR)^{I} = \{a \in \Delta^{I} \mid \left|\{b \mid (a, b) \in R^{I}\}\right| \geq n\}$

Επομένως η έννοια $\leq nR$ έχει ως ερμηνεία τα άτομα εκείνα που συμμετέχουν στη σχέση R με το πολύ n στο πλήθος άτομα. Αντίστοιχα η έννοια $\geq nR$ έχει ως ερμηνεία τα άτομα που συμμετέχουν στη σχέση R με τουλάχιστον n στο πλήθος άτομα.

Για παράδειγμα, ορίζουμε ως πολύτεκνο τον άνθρωπο εκείνο που έχει τρία ή περισσότερα παιδιά:

Πολύτεκνος ≡ Άνθρωπος ⊓ ($\geq 3 \mathit{έχειΠαιδιά}$).

Επίσης περιγράφουμε την έννοια Μητέρα που ορίσαμε πρωτύτερα με έναν ακόμα τρόπο:

Μητέρα ≡ $\theta\eta\lambda\upsilon\kappa\acute{o}$ ⊓ (≥ 1$\acute{\epsilon}\chi\epsilon\iota\Pi\alpha\iota\delta\acute{\iota}$)

Η γλώσσα που περιλαμβάνει όλους τους ανωτέρω κατασκευαστές είναι γνωστή ως $\boldsymbol{\mathcal{ALCN}}$.

Ο κατασκευαστής περιορισμού πληθυκότητας μετατρέπεται στον προσοντούχο κατασκευαστή πληθυκότητας (qualified number restriction, Q), εάν προσδιορίσουμε το σύνολο τιμών της ιδιότητας R. Συγκεκριμένα, ορίζουμε την έννοια $\geq nR.C$ ως εξής:

$(\geq nR.C)^{\mathrm{I}} = \{a \in \Delta^{\mathrm{I}} \mid |\{b|\ (a,b)\ \in\ R^{\mathrm{I}}\}| \geq n \wedge b \in C^{I}\}$.

Αντίστοιχα ορίζεται και ο $\leq nR.C$.

Η γλώσσα με την προσθήκη και αυτού του τελεστή είναι γνωστή ως $\boldsymbol{\mathcal{ALCQ}}$.

Στην πράξη αυτό σημαίνει ότι μπορούμε τώρα να καθορίσουμε όχι μόνο το πλήθος των ατόμων με τα οποία το άτομο συμμετέχει στη σχέση R, αλλά και τον τύπο τους, για την ακρίβεια, την έννοια στην οποία ανήκουν.

Για παράδειγμα, έστω ότι θέλουμε να περιγράψουμε έναν πολύτεκνο άνθρωπο με τουλάχιστον δύο κόρες. Αυτό γίνεται ως εξής:

Πολύτεκνος ⊓ (≥ 2$\acute{\epsilon}\chi\epsilon\iota\Pi\alpha\iota\delta\acute{\iota}.\theta\eta\lambda\upsilon\kappa\acute{o}$).

Μια μητέρα με δύο κόρες ακριβώς περιγράφεται:

Μητέρα ⊓ (≥ 2$\acute{\epsilon}\chi\epsilon\iota\Pi\alpha\iota\delta\acute{\iota}.\theta\eta\lambda\upsilon\kappa\acute{o}$) ⊓ (≤ 2$\acute{\epsilon}\chi\epsilon\iota\Pi\alpha\iota\delta\acute{\iota}.\theta\eta\lambda\upsilon\kappa\acute{o}$).

Ενώ μία μητέρα με ένα γιο περιγράφεται με δύο τρόπους:

Μητέρα ⊓ (∃$\acute{\epsilon}\chi\epsilon\iota\Pi\alpha\iota\delta\acute{\iota}.A\rho\sigma\epsilon\nu\iota\kappa\acute{o}$) ή

Μητέρα ⊓ (≤ 1$\acute{\epsilon}\chi\epsilon\iota\Pi\alpha\iota\delta\acute{\iota}.A\rho\sigma\epsilon\nu\iota\kappa\acute{o}$) ⊓ (≥ 1$\acute{\epsilon}\chi\epsilon\iota\Pi\alpha\iota\delta\acute{\iota}.A\rho\sigma\epsilon\nu\iota\kappa\acute{o}$)

3.1.2 Επεκτάσεις

Η εκφραστική δύναμη της περιγραφικής λογικής $\boldsymbol{\mathcal{ALCQ}}$ την οποία περιγράψαμε έως

τώρα, παρόλο που είναι αρκετά ισχυρή, ενδέχεται να μην επαρκεί σε ορισμένες εφαρμογές. Για το λόγο αυτό έχουν αναπτυχθεί πληθώρα νέων κατασκευαστών οι οποίοι χρησιμοποιούνται από συστήματα. Παράδειγμα των κατασκευαστών αυτών είναι οι κατασκευαστές ρόλων.

Θεωρούμε τα ονόματα ρόλων R, S, που αποτελούν αποτελούν περιγραφές ρόλων. Η τομή (intersection) $R \sqcap S$, η ένωση (union) $R \sqcup S$, το συμπλήρωμα (complement) $\neg R$, η σύνθεση (composition) $R \circ S$, το μεταβατικό κλείσιμο (transitive closure) R^{+}, και η

αντιστροφή (inverse) R^{-}, αποτελούν επίσης περιγραφές ρόλων. Η σημασιολογία τους δίνεται παρακάτω.

$$(R \sqcap S)^I = R^I \cap S^I$$

$$(R \sqcup S)^I = R^I \cup S^I$$

$$(\neg R)^I = \Delta^I \times \Delta^I \backslash R^I$$

$$(R \circ S)^I = \{(a,c) \in \Delta^I \times \Delta^I | \exists b.(\alpha,b) \in R^I \wedge (b,c) \in S^I\}$$

$$(R^+)^I = \bigcup_{i \geq 1} (R^I)^i$$

$$(R^-)^I = \{(b,\alpha) \in \Delta^I \times \Delta^I | (\alpha,b) \in R^I\}$$

Η εκφραστική δυνατότητα της γλώσσας αυξάνεται σημαντικά με τα αξιώματα ρόλων.

Όπως συμβαίνει με τις έννοιες έτσι και οι ρόλοι ανάλογα με την εκφραστικότητα της γλώσσας συμμετέχουν σε **αξιώματα υπαγωγής ρόλων** (role inclusion axioms). Μία ερμηνεία ικανοποιεί ένα αξίωμα υπαγωγής ρόλων $R \sqsubseteq S$, εάν ισχύει ότι $R^I \subseteq S^I$. Το αξίωμα αυτό σηλώνει ότι ο ρόλος R αποτελεί υπορόλο του S. Ουσιαστικά δημιουργούνται ιεραρχίες ρόλων (role hierarchies) .

Ένα παράδειγμα αξιώματος υπαγωγής είναι το εξής: έχειΚόρη⊑έχειΠαιδί. Είναι προφανές ότι ο ρόλος έχει Κόρη υπάγεται στο ρόλο έχειΠαιδί.

Ένα άλλο αξίωμα ρόλων είναι το **αξιώμα μεταβατικών ρόλων**, TR(R) (Transitive role axiom). Η σημασιολογία του ρόλου έχει ως εξής, αν $\{(\alpha^I, b^I), (b^I, c^I)\} \subseteq R^I$ τότε $(\alpha^I, C^I) \subseteq R^I$. Θεωρούμε για παράδειγμα, τον ρόλο έχειΑδερφό και ορίζουμε TR(έχειΑδερφό). Οπότε αν το σώμα ισχυρισμών περιέχει τους ισχυσμούς: $\mathcal{A}$={(Κώστας, Γιώργος):έχειΑδερφό, (Γιώργος, Πάνος):έχειΑδερφό} τότε συνεπάγεται ο ισχυρισμός (Κώστας, Πάνος):έχειΑδερφό.

Η εκφραστικότητα των γλωσσών περιγραφικής λογικής αυξάνεται ακόμα περισσότερο με τον κατασκευαστή **ονοματική έννοια** (nominal concept). Ο κατασκευαστής αυτός, ορίζει έννοιες απαριθμώντας ρητά τα μέλη τους-άτομα. Πιο συγκεκριμένα, έστω α ένα άτομο. Η έννοια {α} ερμηνεύεται ως $\{\alpha\}^I = \{\alpha^I\}$, είναι δηλαδή το υποσύνολο του Δ^I που περιλαμβάνει μόνο το άτομο α. Χαρακτηριστικό παράδειγμα είναι η έννοια που περιγράφει τις ημέρες της εβδομάδας, δηλαδή, ΜέρεςΕβδομάδας≡{Δευτέρα}⊔{Τρίτη}⊔{Τετάρτη}⊔ ... ⊔{Κυριακή}.

Τέλος, αναφέρουμε την έννοια των **τύπων δεδομένων** (datatypes). Εάν p είναι το όνομα ενός τύπου δεδομένων, για παράδειγμα integer, το όνομα αυτό αντιστοιχεί σε μία συγκεκριμένη ερμηνεία, στο παράδειγμά μας στο σύνολο των ακέραιων αριθμών. Πιο τυπικά, ένα σύστημα τύπων είναι ένα ζευγάρι (Δ_D, Φ_D), όπου το σύνολο Φ_D είναι ο σύνολο των ονομάτων και Δ_D είναι ο χώρος ερμηνείας των ονομάτων αυτών. Οι τύποι δεδομένων

χρησιμοποιούνται σε συνδυασμό με τον υπαρξιακό ή καθολικό τελεστή για να δημιουργήσουν μία έννοια. Για παράδειγμα, η έννοια ∃έχειΒάρος. >60 περιγράφει το σύνολο των ανθρώπων με βάρος μεγαλύτερο από 60 κιλά.

Ανάλογα με τις επεκτάσεις της γλώσσας διαμορφώνεται και η ονομασία της, οπότε αναφερόμαστε στην $\boldsymbol{\mathcal{ALCQ}}$ που υποστηρίζει αντιστροφή ρόλων με την ονομασία $\boldsymbol{\mathcal{ALCQI}}$**.** Σημειώνουμε επίσης, την περιγραφική λογική ***S*** (standard) η οποία έχει επεκτάσεις όπως ***SI***, η οποία υποστηρίζει την αντιστροφή ρόλων, ***SHI***, που υποστηρίζει και την ιεραρχία ρόλων, ***SHOIQ***, η οποία υποστηρίζει τους ονοματικούς τύπους, την αντιστροφή ρόλων καθώς και τους προσοντούχους περιορισμούς πληθυκότητας.

3.1.3 Αξιώματα ορολογίας

Στην προηγούμενη ενότητα περιγράψαμε τη σύνταξη των γλωσσών της οικογένειας της $\boldsymbol{\mathcal{ALC}}$ **.** Κάνοντας χρήση των κατασκευαστών (constructors) περιγράψαμε σύνθετες έννοιες από τις ατομικές. Σε αυτή την ενότητα θα παρουσιάσουμε τον τρόπο με τον οποίο ορίζονται οι σύνθετες έννοιες, τα αξιώματα δηλαδή της ορολογίας της βάσης γνώσης.

Τα αξιώματα των εννοιών είναι δύο ειδών. Τα αξιώματα ισοδυναμίας (equivalent axioms) και τα αξιώματα υπαγωγής (subclass axioms).

Έστω οι έννοιες *C,D.*

$C \sqsubseteq D$, είναι ένα αξίωμα υπαγωγής όπου μία ερμηνεία το ικανοποιεί εάν ισχύει ότι $C^I \subseteq D^I$

$C \equiv D$, αξίωμα ισοδυναμίας όπου μία ερμηνεία το ικανοποιεί όταν $C^I = D^I$.

Σε αυτό το σημείο μπορούμε να ορίσουμε το σώμα ισχυρισμών (Terminological Box, *T*) ως ένα σύνολο από αξιώματα (υπαγωγής και ισοδυναμίας).

Επιπλέον, λέμε ότι μία ερμηνεία *I* ικανοποιεί το σύνολο *T* εάν ικανοποιεί όλα τα αξιώματα που ανήκουν σε αυτό. Αντίστροφα, εάν μία ερμηνεία ικανοποιεί το σύνολο των αξιωμάτων του σώματος *T* τότε λέμε ότι ικανοποιεί και το *T.*

Η ερμηνεία ενός αξιώματος ή ενός σώματος ορολογίας αποτελεί *μοντέλο* του αξιώματος ή του σώματος ορολογίας αντίστοιχα. Επομένως δύο αξιώματα είναι ισοδύναμα αν και μόνον αν έχουν το ίδιο μοντέλο.

Το σύνολο των αξιωμάτων του TBox ουσιαστικά αποτελεί ένα σύνολο από ορισμούς-περιγραφές εννοιών. Οι σύνθετες έννοιες ορίζονται χρησιμοποιώντας τις ατομικές έννοιες ή αλλιώς τις πρωταρχικές έννοιες οι οποίες δεν ορίζονται.

Στην περίπτωση που κάποια έννοια ορίζεται από μία περιγραφή που περιλαμβάνει και την ίδια την έννοια, τότε λέμε ότι το αξίωμα που την ορίζει είναι κυκλικό. Με άλλα λόγια ένα αξίωμα είναι κυκλικό όταν το αρίστερο μέλος του αξιώματος εμφανίζεται και στο δεξί. Ένα σώμα ορολογίας που δεν περιλαμβάνει κυκλικά αξιώματα, λέγεται *απλό-ακυκλικό (simple)*. *Γενικευμένο* ή *κυκλικό (general, cyclic)* λέγεται όταν περιλαμβάνει κυκλικά αξιώματα ισοδυναμίας ή γενικευμένα αξιώματα υπαγωγής. Ένα αξίωμα υπαγωγής ονομάζεται γενικευμένο όταν η έννοια του αριστερού μέλους δεν είναι πρωταρχική. Τα γενικευμένα αξιώματα υπαγωγής θα μας απασχολήσουν και στη συνέχεια, καθώς επηρεάζουν σημαντικά τον αλγόριθμο στους μηχανισμούς συλλογιστικής.

3.1.4 Αξιώματα ισχυρισμών

Το σώμα των ισχυρισμών (Assertional Box, *A*) διαισθητικά παρουσιάζει μία κατάσταση του κόσμου ο οποίος περιγράφεται στο σώμα της ορολογίας. Περιγράφει άτομα του κόσμου τα οποία είτε αποτελούν την υπόσταση μίας έννοιας είτε σχετίζονται με άλλα μέσω των σχέσεων του κόσμου. Πιο συγκεκριμένα, αποτελείται από ένα σύνολο αξιωμάτων που έχουν την ακόλουθη μορφή:

α:C (ισχυρισμός έννοιας, concept assertion) ή

(α, b):R (ισχυρισμός ρόλων, role assertion)

Το πρώτο αξίωμα δηλώνει ένα άτομο ως υπόσταση μίας έννοιας C και αποτελεί έναν ισχυρισμό έννοιας. Το δεύτερο είναι ένας ισχυρισμός ιδιότητας και δηλώνει τη σχέση R που συνδέει τα δύο άτομα α,b (role assertion).

Χρειάζεται τώρα να επεκτείνουμε τον ορισμό της ερμηνείας $I = (\Delta^I, \cdot^I)$ ώστε να αντιστοιχίζει όχι μόνο έννοιες, αλλά και κάθε άτομο α σε ένα στοιχείο $a^I \in \Delta^I$. Θεωρούμε ότι διαφορετικά ονόματα αντιπροσωπεύουν και διαφορετικά άτομα. Έτσι λοιπόν για τα α, b θα έχουμε ότι $a^I \neq b^I$. Η ερμηνεία I ικανοποιεί τον ισχυρισμό έννοιας α:C εάν $a^I \in C^I$, και τον ισχυρισμό ρόλου (α, b):R εάν $(a, b)^I \in R^I$. Μία ερμηνεία I ικανοποιεί το ABox εάν ικανοποιεί κάθε ισχυρισμό του. Σε αυτήν την περίπτωση, η ερμηνεία αποτελεί μοντέλο του ABox. Τέλος μία ερμηνεία ικανοποιεί έναν ισχυρισμό ή ολόκληρο το ABox με βάση το σώμα ορολογίας TBox, εάν εκτός από μοντέλο του ισχυρισμού ή του σώματος ισχυρισμών, είναι και μοντέλο ολόκληρου του σώματος ορολογίας.

3.2 Βελτιστοποιημένοι αλγόριθμοι συλλογιστικής για την ALC

Ένα σύστημα αναπαράστασης γνώσης βασισμένο σε περιγραφικές λογικές μπορεί να παρέχει διαφορετικές υπηρεσίες συλλογιστικής. Όπως προείπαμε, ο σκοπός της βάσης

γνώσης δεν είναι η απλή αποθήκευση της πληροφορίας. Βασιζόμενοι στην αποθηκευμένη γνώση, η οποία έχει αποκτήσει σημασιολογία, μπορούμε με διαδικασίες εξαγωγής συμπερασμάτων να παράγουμε νέα. Όπως θα εξηγήσουμε στη συνέχεια, όλες οι διαδικασίες εξαγωγής συμπεράσματος καταλήγουν στον έλεγχο της συνέπειας του σώματος ισχυρισμών.

3.2.1 Υπηρεσίες εξαγωγής συμπεράσματος για έννοιες

Είναι γνωστό ότι ένα σώμα ορολογίας αποτελείται από ένα σύνολο αξιωμάτων τα οποία ορίζουν έννοιες. Μία έννοια καταλαβαίνουμε διαισθητικά ότι έχει νόημα, όταν η ερμηνεία της δεν είναι ένα κενό σύνολο, αλλά ικανοποιεί τα αξιώματα του σώματος ορολογίας. Μία έννοια η ερμηνεία της οποίας είναι μη κενό σύνολο, λέγεται *ικανοποιήσιμη*. Σε διαφορετική περίπτωση πρόκειται για *μη ικανοποιήσιμη* έννοια.

Ένα άλλο πρόβλημα εξαγωγής συμπεράσματος εκτός από τον έλεγχο της ικανοποιησιμότητας μιας έννοιας, είναι ο *έλεγχος υπαγωγής έννοιας* σε κάποια άλλη. Μία έννοια C υπάγεται σε μία άλλη D, εάν η ερμηνεία της C αποτελεί υποσύνολο της ερμηνείας της D.

Τέλος, οι υπηρεσίες εξαγωγής συμπεράσματος ενδέχεται να εστιάσουν στον έλεγχο που αφορά την ισοδυναμία εννοιών ή τις ξένες μεταξύ τους έννοιες.

Παρακάτω παρουσιάζουμε πιο τυπικά όσα ειπώθηκαν για τους ελέγχους που μπορούν να απασχολήσουν τις μηχανές συλλογιστικής και αφορούν έννοιες.

Ικανοποιησιμότητα έννοιας (satisfiability): μία έννοια C είναι ικανοποιήσιμη με βάση το σώμα ορολογίας TBox, εάν υπάρχει μοντέλο *I* του TBox τέτοιο ώστε η C^I να μην είναι το κενό σύνολο. Προφανώς, σε αυτή την περίπτωση το *I* είναι και μοντέλο της C.

Υπαγωγή έννοιας σε άλλη (subsumption): μία έννοια C υπάγεται στην έννοια D με βάση το TBox, εάν $C^I \subseteq D^I$ για κάθε μοντέλο *I* του TBox. Όταν συμβαίνει κάτι τέτοιο γράφουμε $T \models C \sqsubseteq D$.

Ισοδυναμία εννοιών(equivalence): Δύο έννοιες είναι ισοδύναμες με βάση το TBox εάν $C^I = D^I$ για κάθε μοντέλο *I* του TBox. Σε αυτήν την περίπτωση συμβολίζουμε $T \models C \equiv D$.

Ξένες έννοιες (disjointness): Δύο έννοιες είναι ξένες μεταξύ τους με βάση το TBox, εάν οι ερμηνείες τους είναι ξένα μεταξύ τους σύνολα, δηλαδή εάν $C^I \cap D^I = \emptyset$, για κάθε μοντέλο του TBox.

Οι παραπάνω σχέσεις μεταξύ των εννοιών ισχύουν και για την περίπτωση που το σώμα ορολογίας TBox, είναι κενό.

Αναγωγή σε πρόβλημα υπαγωγής

Η πιο συνήθης υπηρεσία εξαγωγής συμπεράσματος που παρέχει ένα σύστημα περιγραφικής λογικής, αφορά την υπαγωγή εννοιών. Στην πραγματικότητα, αυτή η υπηρεσία είναι επαρκής καθώς όλα τα υπόλοιπα προβλήματα μπορούν να εξαχθούν σε πρόβλημα υπαγωγής.

Θεωρούμε το πρόβλημα ικανοποιησιμότητας μίας έννοιας C. Το πρόβλημα αυτό ανάγεται σε πρόβλημα υπαγωγής έαν παρατηρήσουμε την ισοδυναμία μεταξύ των παρακάτω προτάσεων:

Μία έννοια C είναι μη ικανοποιήσιμη ανν η C υπάγεται στην κενή έννοια $\perp$.

Θεωρούμε τώρα το πρόβλημα της ισοδυναμίας δύο εννοιών. Ανάγεται και αυτό σε πρόβλημα υπαγωγής:

Δύο έννοιες C, D είναι ισοδύναμες εάν η C υπάγεται στην D και η έννοια D υπάγεται στην C.

Τέλος, για το πρόβλημα των ξένων εννοιών έχουμε ότι:

Δύο έννοιες C, D είναι ξένες ανν η τομή C⊓ D υπάγεται στην κενή έννοια $\perp$.

Αναγωγή σε πρόβλημα μη ικανοποιησιμότητας

Εάν το σύστημα περιγραφικής λογικής επιτρέπει τη χρήση του τελεστή άρνησης τότε όλα τα προβλήματα που αφορούν έννοιες μπορούν να αναχθούν σε πρόβλημα μη ικανοποιησιμότητας μιας έννοιας. Πιο αναλυτικά, για δύο έννοιες C, D, έχουμε ότι:

Η C υπάγεται στην έννοια D ανν η σύνθετη έννοια C ⊓ ¬D είναι μη ικανοποιήσιμη.

Οι έννοιες C, D είναι ισοδύναμες ανν οι σύνθετες έννοιες (C⊓ ¬ D) και (¬ C⊓ D) είναι μη ικανοποιήσιμες.

Οι έννοιες C, D είναι ξένες μεταξύ τους ανν η C ⊓ D είναι μη ικανοποιήσιμη.

Η παρατήρηση ότι όλα τα προβλήματα ανάγονται σε προβλήματα ελέγχου ικανοποιησιμότητας δημιούργησε νέους αλγορίθμους συλλογιστικής για περιγραφικές λογικές (tableau αλγόριθμοι). Μάλιστα μηχανές συλλογιστικής όπως ο RACE και ο FaCT ανάγουν όλα τα προβλήματα σε έλεγχο ικανοποιησιμότητας.

Σημειώνουμε επίσης, ότι προκείμενου ο έλεγχος ικανοποιησιμότητας να είναι το βασικό πρόβλημα κατά την εξαγωγή συμπερασμάτων, πρέπει η γλώσσα περιγραφικής λογικής που χρησιμοποιείται να υποστηρίζει την άρνηση σύνθετης έννοιας. Διαφορετικά δεν είναι δυνατή η αναγωγή όλων των προβλημάτων σε έλεγχο ικανοποιησιμότητας παραμόνο εάν αυξηθεί αρκετά η πολυπλοκότητα της διαδικασίας της αναγωγής.

Στην οικογένεια των $\mathcal{AL}$- γλωσσών το πρόβλημα της υπαγωγής θέτει το άνω όριο της πολυπλοκότητας αφού όλα τα υπόλοιπα προβλήματα ανάγονται σε αυτό, ακόμα και αν η

γλώσσα δεν υποστηρίζει πλήρη άρνηση, άρνηση σύνθετης έννοιας. Από την άλλη πλευρά το κατώτατο όριο της πολυπλοκότητας ορίζει το πρόβλημα της ικανοποιησιμότητας αφού αποτελεί μια ειδική περίπτωση όλων των υπόλοιπων προβλημάτων. Η πρόταση αυτή παρουσιάζεται παρακάτω.

Έστω μία έννοια C. Οι ακόλουθες προτάσεις είναι ισοδύναμες:

Η C είναι μη ικανοποιήσιμη.

Η C υπάγεται στην $\bot$.

Η C και η $\bot$ είναι ισοδύναμες.

Η C και η $\top$ είναι ξένες έννοιες.

Συμπεραίνουμε λοιπόν, ότι το άνω όριο της πολυπλοκότητας για το πρόβλημα της υπαγωγής, αποτελεί ανώτατο όριο πολυπλοκότητας και για τα αντίστοιχα προβλήματα της ικανοποιησιμότητας, της ισοδυναμίας και των ξένων εννοιών. Αντίστοιχα, το κάτω όριο της πολυπλοκότητας για το πρόβλημα της ικανοποιησιμότητας αποτελεί και κατώτατο όριο πολυπλοκότητας στα αντίστοιχα προβλήματα της υπαγωγής, της ισοδυναμίας και των ξένων εννοιών.

3.2.2 Εξάλειψη του σώματος ορολογίας

Στην πράξη, όταν εκτελείται κάποια διαδικασία εξαγωγής συμπεράσματος που

αφορά μία έννοια, για παράδειγμα ο έλεγχος ικανοποιησιμότητας μίας έννοιας, ο έλεγχος γίνεται με βάση το σώμα ορολογίας, δηλαδή λαβάνοντας υπόψη τα αξιώματα που υπάρχουν σε αυτό. Πολλές φορές είναι πιο εύκολο η διαδικασία εξαγωγής συμπεράσματος να γίνεται στα πλαίσια ενός κενού σώματος ορολογίας. Διαισθητικά, μπορούμε να κατανοήσουμε ότι ο έλεγχος για παράδειγμα, της ικανοποιησιμότητας μιας έννοιας, γίνεται πιο έυκολα όταν εξετάζεται η καθε αυτή έννοια χωρίς να γίνονται αναφορές στο TBox.

Στην πραγματικότητα, εάν το σώμα ορολογίας είναι απλό, ή αλλιώς ακυκλικό, όλα τα προβλήματα με βάση το σώμα ορολογίας μπορούν να αναχθούν σε προβλήματα με βάση το κενό σώμα ορολογίας. Για το σκοπό αυτό, θεωρούμε μία επέκταση $\mathcal{T}'$ της αρχικής ορολογίας $\mathcal{T}$. Η επέκταση $\mathcal{T}'$ προκύπτει εάν στο $\mathcal{T}$ αντικαταστήσουμε κάθε εμφάνιση μίας σύνθετης έννοιας A με τον ορισμό της, που αποτελείται από πρωταρχικές-ατομικές έννοιες μόνο. Η επέκταση $\mathcal{T}'$ είναι ισοδύναμη με το αρχικό σώμα $\mathcal{T}$.

Για παράδειγμα θεωρούμε το παρακάτω σώμα ορολογίας:

Μητέρα ≡ Γυναίκα ⊓ (ΞΈχειΠαιδί.Άνθρωπος).

Γυναίκα ≡ Άνθρωπος ⊓ Θηλυκό

Έστω ότι θέλουμε να εξετάσουμε την ικανοποιησιμότητα της έννοιας *Μητέρα⊓ Άνθρωπος*. Δεν έχουμε παρά να αντικαταστήσουμε κάθε εμφάνιση της έννοιας *Μητέρα* με τον ορισμό της. Οπότε η έννοια γίνεται *Γυναίκα ⊓* (∃ *ΈχειΠαιδί.Άνθρωπος) ⊓ Άνθρωπος*. Η έννοια *Γυναίκα* είναι και αυτή σύνθετη και με τη σειρά της αντικαθίσταται από τον ορισμό της. Η έννοια περιγράφεται τώρα ως εξής: *Άνθρωπος ⊓ Θηλυκό⊓* (∃ *ΈχειΠαιδί.Άνθρωπος).* Σε αυτό το σημείο ο έλεγχος για την ικανοποιησιμότητα της έννοιας γίνεται με βάση το κενό TBox.

Συνοψίζοντας, η επέκταση των εννοιών με βάση το ακυκλικό TBox, επιτρέπει την εξάλειψη του TBox και την εξαγωγή συμπερασμάτων χωρίς να το λαμβάνουμε υπόψη. Αυτό σημαίνει ότι το πρόβλημα εξαγωγής συμπεράσματος με βάση το TBox ανάγεται στο ίδιο πρόβλημα το TBox κενό. Από τη σκοπιά της πολυπλοκότητας, η διαδικασία της επέκτασης κοστίζει, καθώς στην χειρότερη περίπτωση η νέα έννοια θα είναι εκθετική στο μέγεθος του TBox. Από την άλλη πλευρά, η επέκταση ενδέχεται να είναι αναπόφευκτη καθώς η πολυπλοκότητα κατά την εξαγωγή συμπεράσματος με βάση το TBox είναι συγκριτικά πολύ μεγαλύτερη.

3.2.3 Υπηρεσίες εξαγωγής συμπεράσματος για σώματα ισχυρισμών

Όπως έχουμε ήδη αναφέρει αρκετές φορές, η βάση γνώσης ενός συστήματος

αποτελείται από το σώμα της ορολογίας και το σώμα των ισχυρισμών. Το σώμα της ορολογίας πρέπει να περιγράφει έννοιες ικανοποιήσιμες και άρα να διαθέτει ένα μοντέλο. Ατυό όμως δεν αρκεί. Πρέπει και το σώμα ισχυρισμών να είναι συνεπές.

Ένα σώμα ισχυρισμών είναι συνεπές με βάση ένα σώμα ορολογίας, εάν υπάρχει ερμηνεία που να αποτελεί ταυτόχρονα μοντέλο του σώματος ορολογίας και του σώματος ισχυρισμών. Λέμε απλά ότι το σώμα ισχυρισμών ABox, είναι συνεπές εννοώντας ότι είναι συνεπές με βάση το TBox.

Για πράδειγμα, θεωρούμε το TBox που παραθέσαμε παραπάνω:

Μητέρα ≡ Γυναίκα ⊓ (∃ΈχειΠαιδί.Άνθρωπος).

Γυναίκα ≡ Άνθρωπος ⊓ Θηλυκό

και το ABox, $\mathcal{A}$={ Ελένη:Μητέρα, Ελένη: Γυναίκα }.

Η έννοια *Μητέρα* έχει ως ερμηνεία το άτομο *Ελένη* όπως η έννοια *Γυναίκα.* Το μοντέλο του TBox αποτελεί και μοντέλο του ABox και άρα πρόκειται για ένα συνεπές σώμα ισχυρισμών.

Ο έλεγχος συνέπειας του ABox με βάση το TBox μπορεί να αναχθεί όπως και στην περίπτωση των εννοιών, σε έλεγχο συνέπειας ενός επεκταμένου ABox με βάση το κενό TBox. Ορίζουμε την επέκταση $\mathcal{A}$', με βάση το TBox, του ABox $\mathcal{A}$, το σώμα ισχυρισμών

που προκύπτει εάν αντικαταστήσουμε κάθε σύνθετη έννοια που βρίσκεται στους ισχυρισμούς εννοιών, για παράδειγμα στον *Ελένη:Μητέρα*, με την περιγραφή της, έτσι όπως προκύπτει από το TBox. Δηλαδή , στο παράδειγμά μας παίρνουμε τον ισχυρισμό *Ελένη: Άνθρωπος ⊓ Θηλυκό⊓* (∃ *ΈχειΠαιδί.Άνθρωπος)*. Τελικά το $\mathcal{A}'$ είναι συνεπές εάν και το $\mathcal{A}$ είναι συνεπές.

Σε ότι αφορά το ABox, μπορούν να τεθούν διάφορα προβλήματα εξαγωγής συμπεράσματος. Μια περίπτωση είναι εάν ένας ισχυρισμός είναι δυνατό να προκύψει από το ABox. Ένας ισχυρισμός έστω α, απορρέει από το ABox εάν κάθε μοντέλο του ABox ικανοποιεί και τον ισχυρισμό α. Συμβολίζουμε με $\mathcal{A} \models \alpha$. Εαν ο α είναι ισχυρισμός ρόλων, τότε ο έλεγχος γίνεται έυκολα μιας και δεν υπάρχουν τελεστές για να συνθέσουν σύνθετους ρόλους. Στην περίπτωση που ο α είναι ισχυρισμός έννοιας μπορούμε να ανάγουμε το πρόβλημα στον έλεγχο συνέπειας του ABox. Η αναγωγή αυτή υποστηρίζεται από την παρακάτω πρόταση:

$\mathcal{A} \models \{x: C\}$ ανν το $\mathcal{A} \cup \neg\{x: C\}$ είναι ασυνεπές.

Σημειώνουμε ότι και ο έλεγχος για την ικανοποιησιμότητα μίας έννοιας ανάγεται σε πρόβλημα συνέπειας του ABox.

Η έννοια C είναι ικανοποιήσιμη ανν το $\mathcal{A}=\{x:C\}$ είναι συνεπές.

Ένα άλλο πρόβλημα εξαγωγής συμπεράσματος που αφορά το ABox, είναι το πρόβλημα ανάκτησης (*retrieval problem*). Δεδομένου ενός σώματος ισχυρισμών $\mathcal{A}$, και μίας έννοιας C, το πρόβλημα που τίθεται είναι η εύρεση όλων των ατόμων α, τα οποία ανήκουν στην έννοια C ή μπορούμε να συμπεράνουμε ότι ανήκουν σε αυτή, δηλαδή $\mathcal{A} \models (\alpha: C)$.

Το πρόβλημα πραγματοποίησης (*realization problem*) θέτει στο ABox το ερώτημα για την εύρεση των περισσότερο ειδικών εννοιών C, τέτοιων ώστε δεδομένου ενός ατόμου α, να ισχύει ότι $\mathcal{A} \models (\alpha: C)$. Με τον όρο 'περισσότερο ειδικές έννοιες' εννοούμε εκείνες που βρίσκονται πιο χαμηλά στην ιεράχηση, δηλαδή στο αριστερό σκέλος των αξιωμάτων υπαγωγής. Αυτό το πρόβλημα βρίσκει εφαρμογή για παράδειγμα, σε ένα σύστημα παραγωγής φυσικής γλώσσας όπου αναζητείται το πιο συγκεκριμένο αντικείμενο που ενδέχεται να εμφανίζεται σε μία συζήτηση.

3.2.4 Αλγόριθμοι εξαγωγής συμπερασμάτων

Όπως είδαμε παραπάνω, οι περιγραφικές λογικές που υποστηρίζουν την πλήρη άρνηση (full negation) επιτρέπουν την εφαρμογή αλγορίθμων συλλογιστικής που ονομάζονται *tableau*. Σε αυτή την περίπτωση υπενθυμίζουμε, ότι όλα τα προβλήματα ανάγονται στον έλεγχο ικανοποιησιμότητας. Ωστόσο, υπάρχουν αρκετές περιπτώσεις συστημάτων περιγραφικών λογικών που δεν υποστηρίζουν την πλήρη άρνηση. Η αναγωγή των προβλημάτων γίνεται σε αυτή την περίπτωση στο πρόβλημα ελέγχου υπαγωγής. Οι

αλγόριθμοι συλλογιστικής που εφαρμόζονται σε αυτά τα συτήματα είναι γνωστοί ως δομικοί αλγόριθμοι υπαγωγής (*structural subsumption algorithms*).

Αλγόριθμοι υπαγωγής

Οι αλγόριθμοι αυτοί ακολουθούν δύο φάσεις. Στην πρώτη φάση, οι περιγραφές των εννοιών κανονικοποιούνται και έπειτα, συγκρίνονται με τη σύνταξη των κανονικών μορφών.

Για παράδειγμα, θεωρούμε τη γλώσσα περιγραφικής λογικής $\mathcal{FL}_0$, η οποία υποστηρίζει την τομή, (C⊓D), και τον πλήρη υπαρξιακό περιορισμό $(\exists R.C)$. Επιπλέον, μπορούν να εισαχθούν η κενή έννοια ($\perp$), η άρνηση ατομικής έννοιας ($\neg A$), όπως και ο κατασκευαστής περιορισμού πληθυκότητας ($\geq nR.C$). Η περιγραφή μίας $\mathcal{FL}_0$- έννοιας είναι σε κανονική μορφή όταν και μόνο όταν έχει την ακόλουθη σύνταξη:

$$A_1 \sqcap \ldots \sqcap A_m \sqcap \forall R_1.C_1 \sqcap \ldots \sqcap \forall R_n.C_n$$

όπου τα A_1,..., A_m είναι ονόματα εννοιών, τα R_1,..., R_n είναι ονόματα ρόλων και τα C_1,..., C_n, είναι $\mathcal{FL}_0$–έννοιες σε κανονική μορφή.

Έστω $A_1 \sqcap \ldots \sqcap A_m \sqcap \forall R_1.C_1 \sqcap \ldots \sqcap \forall R_n.C_n$ η κανονική μορφή της $\mathcal{FL}_0$- έννοιας C, και $B_k \sqcap \ldots \sqcap B_k \sqcap \forall S_1.D_1 \sqcap \ldots \sqcap \forall S_l.D_l$ η κανονική μορφή της $\mathcal{FL}_0$- έννοιας D. Η έννοια C υπάγεται στη D, $C \sqsubseteq D$ ανν ισχύουν τα παρακάτω:

Για κάθε i, 1≤i≤k, υπάρχει j, 1≤j≤m, τέτοιο ώστε $B_i = A_j$

Για κάθε i, 1≤i≤ l, υπάρχει j, 1≤j≤n, τέτοιο ώστε $S_i = R_j$ και $C \sqsubseteq D$.

Από τις παραπάνω προτάσεις είναι προφανές ότι το πρόβλημα ελέγχου της υπαγωγής δύο εννοιών για την περιγραφική λογική $\mathcal{FL}_0$, αντιμετωπίζεται με έναν αναδρομικό αλγόριθμο, ο οποίος χαρακτηρίζεται μάλιστα από πολυπλοκότητα πολυωνυμικού χρόνου. Ο αλγόριθμος είναι ορθός και πλήρης (η πληρότητα εξασφαλίζεται από την αντιστροφή της ισοδυναμίας).

Εάν η $\mathcal{FL}_0$ επεκταθεί ώστε να υποστηρίζει την κενή έννοια θα έχουμε ότι μία περιγραφή $\mathcal{FL}_\perp$- έννοιας είναι σε κανονική μορφή εάν και μόνο αν είναι είτε η κενή έννοια $\perp$, είτε έχει την ακόλουθη σύνταξη:

$$A_1 \sqcap \ldots \sqcap A_m \sqcap \forall R_1.C_1 \sqcap \ldots \sqcap \forall R_n.C_n$$

όπου τα A_1,..., A_m είναι ονόματα εννοιών διαφορετικά από την κενή έννοια, τα R_1,..., R_n είναι ονόματα ρόλων και τα C_1,..., C_n, είναι $\mathcal{FL}_\perp$–έννοιες σε κανονική μορφή. Ο αλγόριθμος υπαγωγής είναι όπως πριν με την παρατήρηση ότι η κενή έννοια υπάγεται σε όλες τις υπό λοιπες.

Παρόμοιοι αλγόριθμοι εφαρμόζονται και στις επεκτάσεις της $\mathcal{FL}_0$ με κατασκευαστές ατομικής άρνησης καθώς και περιορισμού πληθυκότητας. Προφανώς οι γλώσσες αυτές αποτελούν υπογλώσσες της οικογένειας των $\mathcal{AL}$- γλωσσών. Καθώς η εκφραστικότητα της

$\mathcal{FL}_0$ αυξάνεται με την προσθήκη νέων κατασκευαστών και πλησιάζει όλο και περισσότερο την $\mathcal{ALCN}$, οι αλγόριθμοι υπαγωγής παύουν να είναι πλήρεις. Συγκεκριμένα, με την εισαγωγή των κατασκευαστών της πλήρους άρνησης, της ένωσης και του πλήρους υπαρξιακού περιορισμού, εφαρμόζονται οι tableau αλγόριθμοι.

Tableau αλγόριθμοι

Οι αλγόριθμοι αυτοί αναγάγουν τα όλα προβλήματα στον έλεγχο της ικανοποιησιμότητας. Μια έννοια είναι ικανοποιήσιμη όταν η ερμηνεία της δεν είναι το κενό σύνολο. Έτσι λοιπόν, ένας αλγόριθμος tableau, καλείται να αποφασίσει εάν υπάρχει μοντέλο προκειμένου να αποδείξει την ικανοποιησιμότητα ή μη μιας έννοιας.

Σε γενικές γραμμές, οι αλγόριθμοι αυτοί ακολουθούν τους παρακάτω κανόνες:

Για κάθε υπαρξιακό περιορισμό ο αλγόριθμος εισάγει ένα νέο άτομο το οποίο πρέπει να κανοποιεί τον συγκεκριμένο περιορισμό.

Ο κατασκευαστής περιορισμού τιμής χρησιμοποιείται σε συνδυασμό με υπάρχουσες σχέσεις ρόλων μεταξύ των ατόμων ώστε να εισάγει νέους περιορισμούς που αφορούν τα άτομα.

Στην περίπτωση της ένωσης εννοιών ο αλγόριθμος δοκιμάζει τυχαία μία από αυτές και οπισθοδρομεί στην περίπτωση που ένα άτομο ανήκει σε δύο προφανώς συγκρουόμενες έννοιες.

Τέλος, στην περίπτωση του περιορισμού πληθυκότητας, πρέπει να γίνει καταμέτρηση των διαφορετικών ατόμων που ικανοποιούν τον περιορισμό του κατασκευαστή.

Σε αυτό το σημείο θα παρουσιάσουμε τον αλγόριθμο tableau για τη γλώσσα $\mathcal{ALC}$. Εξετάζουμε τη γενική περίπτωση όπου ο αλγόριθμος εφαρμόζεται σε μία βάση γνώσης με σκοπό να αποφανθεί για τη συνέπεια ή μη της βάσης. Όπως γίνεται κατανοητό από τα παρακάτω, το πρόβλημα αναγάγεται και πάλι σε έλεγχο ικανοποιησιμότητας. Δηλαδή, έστω μία βάση γνώσης $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ και μία έννοια C.

Η έννοια C είναι ικανοποιήσιμη με βάση τη βάση γνώσης $\mathcal{K}$, δηλαδή $\mathcal{K} \vDash C$, ανν η βάση γνώσης $\mathcal{K} = (\mathcal{T}, \mathcal{A} \cup \{x: C\})$ είναι συνεπής.

Έχουμε ήδη αναφέρει ότι ο αλγόριθμος αποφασίζει για τη συνέπεια της βάσης γνώσης, ουσιαστικά για την ικαννοποιησιμότητα των εννοιών, κατασκευάζοντας ένα μοντέλο. Εάν το μοντέλο δεν είναι δυνατό να κατασκευαστεί, σημαίνει ότι η βάση γνώσης είναι ασυνεπής.

Ο αλγόριθμος ξεκινά με ένα σώμα ισχυρισμών $\mathcal{A}$. Αναλύοντας τους περιορισμούς του $\mathcal{A}$ και συνυπολογίζοντας τα αξιώματα του σώματος ορολογίας, κατασκευάζει ένα δάσος οι αρχικοί κόμβοι του οποίου αντιστοιχούν στα άτομα που παρουσιάζονται στους ισχυρισμούς του $\mathcal{A}$. Ξεκινώντας από κάθε έναν από αυτούς τους κόμβους (root nodes), κατασκευάζεται

ένα δένδρο. Εάν τα δένδρα είναι πεπερασμένα, τότε υπάρχει μοντέλο και η βάση γνώσης είναι συνεπής.

Πιο αναλυτικά, κατά την κατασκευή ενός μοντέλου, εάν αυτό υπάρχει, ο αλγόριθμος επενεργεί σε μία δομή που ονομάζεται δάσος πληρότητας (*completion forest*). Πρόκειται για έναν κατευθυνόμενο γράφο οι κόμβοι και οι ακμές του οποίου σημειώνονται με μία ετικέτα. Από κάθε κόμβο ξεκινά μία δενδρική δομή, δένδρο πληρότητας (*completion tree*). Η ετικέτα $\mathcal{L}(x)$ ενός κόμβου x περιλαμβάνει το σύνολο των εννοιών στις ερμηνείες των οποίων ανήκει το άτομο x. Η ετικέτα $\mathcal{L}(< x, y >)$ μίας ακμής $< x, y >$, αποτελείται από τους ρόλους που συσχετίζουν τα δύο άτομα x, y.

Ξεκινώντας, από τη βάση γνώσης $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, το δάσος $\mathcal{F}_{\mathcal{A}}$ αρχικοποιείται με έναν ριζικό κόμβο x_α (*root node*) και μία ετικέτα $\mathcal{L}(x_\alpha) = \{C | \alpha : C \in \mathcal{A}\}$, για κάθε άτομο α του $\mathcal{A}$, και με μία ακμή $< x_\alpha, x_b >$ $\mathcal{L}(< x_\alpha, x_b >) = \{R | (\alpha, b) : R \in \mathcal{A}\}$, για κάθε ζεύγος ατόμων (α,b) που συσχετίζονται με ρόλους στο $\mathcal{A}$. Έπειτα, ο αλγόριθμος συνεχίζει με την εφαρμογή των κανόνων επέκτασης (*expansion rules*) οι οποίοι αποσυνθέτουν τις έννοιες της κάθε ετικέτας. Οι κανόνες του tableau- αλγορίθμου παρουσιάζονται παρακάτω.

- Κανόνας -⊓: **Εάν** $(C_1 \sqcap C_2) \in \mathcal{L}(x)$ **και** $\{C_1, C_2\} \notin \mathcal{L}(x)$
  τότε η ετικέτα γίνεται $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$.
- Κανόνας -⊔: **Εάν** $(C_1 \sqcup C_2) \in \mathcal{L}(x)$ **και** $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
  τότε η ετικέτα γίνεται $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C\}$, όπου $C \in \{C_1, C_2\}$.
- Κανόνας –∃: ***Εάν*** $\exists R.C \in \mathcal{L}(x)$ ***και*** δεν υπάρχει κόμβος y, τέτοιος ώστε
  $R \in \mathcal{L}(< x, y >)$ και $C \in \mathcal{L}(y)$
  τότε δημιουργείται ένας νέος κόμβος y με ετικέτα $\mathcal{L}(y) = \{C\}$ και ακμή με $\mathcal{L}(< x, y >) = \{R\}$.
- Κανόνας -∀: ***Εάν*** $\forall R.C \in \mathcal{L}(x)$ ***και*** υπάρχει κόμβος y, τέτοιος ώστε
  $R \in \mathcal{L}(< x, y >)$ και $C \notin \mathcal{L}(y)$
  τότε ο κόμβος y αποκτά ετικέτα $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$.
- Κανόνας -⊑: **Εάν** $C_1 \sqsubseteq C_2 \in \mathcal{T}$ **και** $C_2 \sqcup \neg C_1 \notin \mathcal{L}(x)$
  τότε $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_2 \sqcup \neg C_1\}$,

Η εφαρμογή των κανόνων τερματίζεται στην περίπτωση που εντοπιστεί κάποια αντίφαση (clash). Για την $\mathcal{ALC}$ η αντίφαση ορίζεται από την ύπαρξη δύο συμπληρωματικών εννοιών $C, \neg C$. Εάν για παράδειγμα σε κάποιο κόμβο x, έχουμε ότι $\{C, \neg C\} \in \mathcal{L}(x)$ τότε προφανώς, το δάσος περιλαμβάνει κάποια ασυνέπεια και άρα δεν υπάρχει μοντέλο. Στην περίπτωση που δεν εντοπίζεται κάποια σύγκρουση εννοιών, ο αλγόριθμος τερματίζει όταν

δεν εφαρμόζεται άλλος κανόνας. Σε αυτή την περίπτωση το δάσος αντιπροσωπεύει ένα μοντέλο της βάσης γνώσης, η οποία είναι συνεπής.

Στο σημείο αυτό, είναι απαραίτητο να επισημάνουμε την μη-ντετερμινιστική επιλογή που εισάγει ο κανόνας της ένωσης. Κατά την εφαρμογή του κανόνα της ένωσης, επιλέγεται με τρόπο μη ντετερμινιστικό η μία έννοια. Στην περίπτωση που η επιλογή της έννοιας οδηγήσει σε σύγκρουση, πρέπει ο αλγόριθμος να μπορεί να οπισθοδρομήσει έτσι ώστε να αναπτυχθεί ένα δένδρο με βάση την εναλλακτική επιλογή έννοιας. Αρκεί μία από τις μη ντερμινιστικές επιλογές να καταλήγει σε κάποιο μοντέλο, δηλαδή να μην οδηγεί σε αντίφαση, ώστε να αποφανθούμε για τη συνέπεια της βάσης. Εάν όλες οι επιλογές οδηγούν σε σύγκρουση τότε δεν υπάρχει μοντέλο και η βάση γνώσης είναι ασυνεπής.

Σημειώνουμε επίσης, ότι στον αλγόριθμο υπάρχουν δύο μη ντετερμινιστικές επιλογές διαφορετικής φύσεως. Ο κανόνας της ένωσης εισάγει την έννοιας της τυχαίας επιλογής η οποία ενδέχεται είτε να οδηγήσει σε μοντέλο είτε όχι. Εάν όχι, ο αλγόριθμος πρέπει να οπισθοδρομήσει. Η σειρά με την οποία ο αλγόριθμος επιλέγει να εφαρμόζει τους κανόνες, αποτελεί μία διαφορετικής φύσεως μη ντετερμινιστική επιλογή η οποία επηρεάζει μεν το κόστος του χρόνου εκτέλεσης, αλλά δεν ενδέχεται να εξαναγκάσει τον αλγόριθμο σε οπισθοδρόμηση.

Μπλοκάρισμα

Ο αλγόριθμος έτσι όπως περιγράφηκε εφαρμόζεται σε βάσεις γνώσεις με ακυκλικά σώματα ορολογίας. Στην περίπτωση των σωμάτων ορολογίας που υποστηρίζουν γενικευμένα αξιώματα υπαγωγής, πρέπει να εισαχθεί στον αλγόριθμο η τεχνική *μπλοκαρίσματος (blocking).*

Η παρουσία των αξιωμάτων που περιέχουν κύκλους έχει ως αποτέλεσμα να επαναλαμβάνεται η εφαρμογή κανόνων κατά την εκτέλεση του αλγορίθμου, οι οποίοι όμως δεν προσφέρουν καμία καινούρια πληροφορία. Παρουσιάζονται δηλαδή, ‘κύκλοι΄ στην εκτέλεση των κανόνων. Η τεχνική του μπλοκαρίσματος εισάγεται προκειμένου να επιλύσει ακριβώς αυτό το πρόβλημα και ο αλγόριθμος να τερματίζει.

Σύμφωνα με την τεχνική του μπλοκαρίσματος, κατά την εφαρμογή των κανόνων ελέγχεται εάν οι έννοιες στην ετικέτα ενός ατόμου x, αποτελούν υποσύνολο των εννοιών άλλου ατόμου y, με το οποίο το x συνδέεται μέσω κάποιων ρόλων. Στην περίπτωση που συμβαίνει κάτι τέτοιο, παύει η εκτέλεση των κανόνων αφού δεν προσφέρει κάποια νέα πληροφορία. Λέμε ότι το άτομο y μπλοκάρει το x. Το δένδρο που αναπτύσσεται κατά την εκτέλεση των κανόνων και αντιπροσωπεύει το μοντέλο της βάσης, είναι πεπερασμένο. Στην πραγματικότητα, είναι άπειρο εάν αντικαταστήσουμε τον κόμβο στον οποίο έχει γίνει μπλοκάρισμα με το υποδένδρο που αντιστοιχεί στην ατέρμονη επανάληψη των κανόνων.

Σημειώνουμε σε αυτό το σημείο, ότι το πρόβλημα συνέπειας για τις $\mathcal{ALC}$ βάσεις γνώσης χωρίς γενικευμένα αξιώματα υπαγωγής είναι πολυωνυμικού χώρου (PSpace). Είναι μάλιστα NPSpace λόγω του μη ντεντερμινιστικού κανόνα της ένωσης, αλλά όλα τα NPSpace προβλήματα ανάγονται σε PSpace. Στην περίπτωση που η βάση γνώσης περιλαμβάνει και κυκλικά αξιώματα το ίδιο πρόβλημα γίνεται εκθετικού χρόνου (EXPTime-complete), καθώς δεν είναι γνωστό πόσοι κύκλοι επανάληψης κανόνων θα λάβουν χώρα εως ότου γίνει το μπλοκάρισμα.

Τερματισμός, ορθότητα, πληρότητα

Οι κανόνες επέκτασης αναπτύσσουν τις έννοιες στις ετικέττες των κόμβων. Οι έννοιες αυτές απορρέουν από τη βάση γνώσης και είναι σαφώς πεπερασμένες. Επιπλέον, πεπερασμένο είναι και το πλάτος του δένδρου, καθώς οι νέοι κόμβοι που δημιουργούνται κατά την εκτέλεση του δένδρου, διαθέτουν ένα άνω φράγμα, αφού το πλήθος τους δεν είναι δυνατό να ξεπερνά τον αριθμό των υπαρξιακών τελεστών. Πεπερασμένο όμως είναι και το μήκος του δένδρου, κάτι που εξασφαλίζεται από την τεχνική του μπλοκαρίσματος.

Από τα παραπάνω γίνεται κατανοητό ότι ο αλγόριθμος τερματίζει.

Η ορθότητα του αλγορίθμου προκύπτει από το γεγονός ότι κατασκευάζεται ένα μοντέλο για τη δεδομένη βάση γνώσης καθώς αναπτύσσεται το δάσος πληρότητας, (completion forest). Στην περίπτωση των κόμβων που έχουν μπλοκαριστεί τους αντικαθιστούμε με το αντίστοιχο άπειρο υποδένδρο και έτσι προκύπτει το εν λόγω μοντέλο.

Τέλος, ο αλγόριθμος είναι πλήρης διότι εάν η βάση γνώσης είναι συνεπής ο αλγόριθμος θα εξετάσει όλες τις μη ντετερμινιστικές επιλογές προκειμένου να βρει το μοντέλο.

Τεχνικές βελτιστοποίησης

Στην περίπτωση που η βάση γνώσης δεν περιλαμβάνει κυκλικά αξιώματα, είναι δυνατό το πρόβλημα της συνέπειας της βάσης γνώσης να αναχθεί σε πρόβλημα συνέπειας του σώματος ισχυρισμών μόνο, με κενό δηλαδή σώμα ορολογίας. Κάτι τέτοιο επιτυγχάνεται εάν αντικαταστήσουμε κάθε έννοια με την περιγραφή της όπως προκύπτει από το σώμα ορολογίας. Η ίδια διαδικασία επαναλαμβάνεται για όλες τις έννοιες που εμφανίζονται στο ABox και η περιγραφή τους ορίζεται στο TBox, εώς ότου το TBox να είναι κενό. Η τεχνική αυτή ονομάζεται *ξεδίπλωμα (unfolding)* του σώματος ορολογίας. Όταν εφαρμόζεται η τεχνική αυτή, ο κανόνας της υπαγωγής προφανώς δεν εκτελείται.

Είναι εύκολα κατανοητό ότι η τεχνική του ξεδιπλώματος αυξάνει εκθετικά το μέγεθος του ABox. Για το λόγο αυτό, είναι προτιμότερο να εφαρμόζεται κατά απαίτηση, οκνηρό ξεδίπλωμα (*lazy unfolding*). Με άλλα λόγια, η εκάστοτε έννοια αντικαθίσταται από την περιγραφή της μόνο όταν πρόκειται να χρησιμοποιηθεί στη διάρκεια της εκτέλεσης του

προγράμματος. Έτσι λοιπόν, ο κανόνας της υπαγωγής εγκαταλείπεται και το οκνηρό ξεδίπλωμα (lazy unfolding) υλοποιείται με δύο κανόνες:

- **Εάν** $A \equiv C \in \mathcal{T},\ A \in \mathcal{L}(x)$ **και** $C \notin \mathcal{L}(x)$

  τότε $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C\}$

- **Εάν** $A \equiv C \in \mathcal{T},\ \neg A \in \mathcal{L}(x)$ **και** $\neg C \notin \mathcal{L}(x)$

  τότε $\mathcal{L}(x) = \mathcal{L}(x) \cup \{\neg C\}$

Μία άλλη τεχνική βελτιστοποιήσης που χρησιμοποείται είναι η οπισθοδρόμηση βάσει εξαρτημένου συνόλου (*dependency directed backtracking*). Σύμφωνα με την τεχνική αυτή, η οπισθοδρόμηση, η οποία λαμβάνει χώρα όταν ανιχνευθεί κάποια αντίφαση (clash) ενώ έχει ήδη εφαρμοσθεί ο μη ντετερμινιστικός κανόνας της ένωσης, γίνεται μέσω τον κατάλληλων κόμβων.

Πιο αναλυτικά, όταν ο αλγόριθμος ανιχεύσει αντίφαση είτε τερματίζει, είτε όπως έχουμε ήδη περιγράψει, οπισθοδρομεί προκειμένου να εξετάσει την εναλλακτική επιλογή έτσι όπως προκύπτει από τον κανόνα της ένωσης. Σκοπός είναι η οπισθοδρόμηση να γίνεται απ'ευθείας σε κόμβους όπου έχει γίνει μία μη ντετερμινιστική επιλογή. Σε διαφορετική περίπτωση, εξετάζονται με τη σειρά όλοι οι προγενέστεροι κόμβοι χωρίς κανένα αποτέλεσμα. Με την βελτιστοποιημένη οπισθοδρόμηση, οι κόμβοι αυτοί παραλείπονται, και γίνεται η μεταπήδηση σε κόμβους όπου έχει ήδη εφαρμοσθεί ο κανόνας της ένωσης και άρα ενέχουν εναλλακτική επιλογή.

3.3 Σημασιολογικός Ιστός

Ο Σημασιολογικός Ιστός αποτελεί μία επέκταση του Παγκόσμιου Ιστού με πρωταρχικό σκοπό την αυτοματοποιήση των υπηρεσιών του που αφορούν αιτήματα χρηστών και μηχανών. Η αυτοματοποίηση των λειτουργιών επιτυγχάνεται με τροποποίηση της αναπαράστασης της πληροφορίας ώστε να αποκτήσει τυπικό ορισμό. Ο τυπικός ορισμός της πληροφορίας και άρα η καθορισμένη σημασιολογία που αποκτά, την καθιστά κατανοητή από της μηχανές που την επεξεργάζονται.

Η ανάγκη για μία τυπική αναπαράσταση της πληροφορίας του σημασιολογικού ιστού καθιστά τις γλώσσες αναπαράστασης γνώσης το κατάλληλο εργαλείο για το σκοπό αυτό. Παράλληλα, υπάρχει και η ανάγκη συμβατότητας με τον τρόπο αναπαράστασης της πληροφορίας στον υπόλοιπο ιστό, που γίνεται με τη χρήση της γλώσσας xml. Η γλώσσα που έχει αναπτυχθεί για τον σκοπό αυτό είναι η OWL (Web Ontology Language).

Η OWL ανήκει στην οικογένεια των γλωσσών αναπαράστασης γνώσης και είναι βασισμένη στις περιγραφικές λογικές. Η σημασιολογία της αντιστοιχίζεται άμεσα με αυτή

των περιγραφικών λογικών. Αυτή η συσχέτιση, επιτρέπει τη χρήση μηχανών συλλογιστικής οι οποίες έχουν ήδη αναπτυχθεί για εκφραστικές περιγραφικές λογικές, σε εφαρμογές που χρησιμοποιούν την OWL.

Η OWL επιτρέπει την επεξεργασία της πληροφορίας από εφαρμογές οι οποίες διαχειρίζονται το περιεχόμενο και τη σημασιολογία της. Επιπλέον, διευκολύνει την κατανόηση της πληροφορίας από τις μηχανές του διαδικτύου περισσότερο από ότι το μοντέλο RDF/XML και αυτό γιατί έχει μεγαλύτερη εκφραστική δύναμη και καθορισμένη σημασιολογία. Η γλώσσα OWL έτσι όπως αναπτύχθηκε από τον οργανισμό προτυποποίησης W3C διαθέτει τρεις μορφές: OWL Lite, OWL DL, OWL Full.

Η *OWL Lite* είναι χαμηλής εκφραστικότητας γλώσσα. Χρησιμοποιείται σε εξειδικευμένες εφαρμογές όπου δεν απαιτείται υψηλή εκφραστικότητα αλλά δίνεται έμφαση στην ταχύτερη εξαγωγή συμπερασμάτων. Η περιγραφική λογική με την οποία παρέχει την ίδια εκφραστικότητα είναι η *SHIN(D).*

Η *OWL DL* παρέχει την ίδια εκφραστικότητα με τη γλώσσα *SHOIN(D),* τη μέγιστη εκφραστικότητα που προσφέρει η γλώσσα OWL, χωρίς αυτό να την καθιστά υπολογιστικά αδύναμη.

Η *OWL Full* διαθέτει το ίδιο συντακτικό λεξιλόγιο με την *OWL DL,* ενώ ταυτόχρονα το συντακτικό μοντέλο και η σημασιολογία της αποτελούν επέκταση του RDF/XML μοντέλου. Πρόκειται για μία μη αποφασίσιμη γλώσσα.

Μία οντολογία γραμμένη σε OWL μπορεί να θεωρηθεί ως ένα σώμα ορολογίας (Tbox) όπου οι έννοιες της ορολογίας αντιστοιχούν στις κλάσεις της οντολογίας και οι ρόλοι-σχέσεις στις ιδιότητες της οντολογίας. Η οντολογία αποτελείται από ένα σύνολο αξιωμάτων που ορίζουν τις κλάσεις, τις ιδιότητες και τις μεταξύ τους σχέσεις. Οι κλάσεις μπορούν να κατασκευαστούν από άλλες απλούστερες με τη χρήση κατασκευαστών. Οι κατασκευαστές της OWL και η σημασιολογία αυτών αντιστοιχίζονται άμεσα με αυτούς των εκφραστικών λογικών.

Κατασκευαστής	Σύνταξη DL
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$
complementOf	$\neg C$
oneOf	$\{x_1 \ldots x_2\}$
allValuesFrom	$\forall R.C$
someValuesFrom	$\exists R.C$
hasValue	$\exists R.\{x\}$
minCardinality	$(\geq nR)$
maxCardinality	$(\leq nR)$
inverseOf	R^-

Πίνακας κατασκευαστών της OWL

Αξίωμα	Σύνταξη περιγραφικής λογικής
subClassOf	$C_1 \sqsubseteq C_2$
equivalentClass	$C_1 \equiv C_2$
subPropertyOf	$R_1 \sqsubseteq R_2$
equivalentProperty	$R_1 \equiv R_2$
disjointWith	$C_1 \sqsubseteq \neg C_2$
sameAs	$\{x_1\} \equiv \{x_2\}$
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$
TransitiveProperty	$TR(R), R$ μεταβατικός ρόλος
FunctionalProperty	$\top \sqsubseteq (\leq 1R)$
InverseFunctionalProperty	$\top \sqsubseteq (\leq 1R^-)$
SymmetricProperty	$R \equiv R^-$

Πίνακας OWL αξιωμάτων

4 *Αρχιτεκτονική Συστήματος*

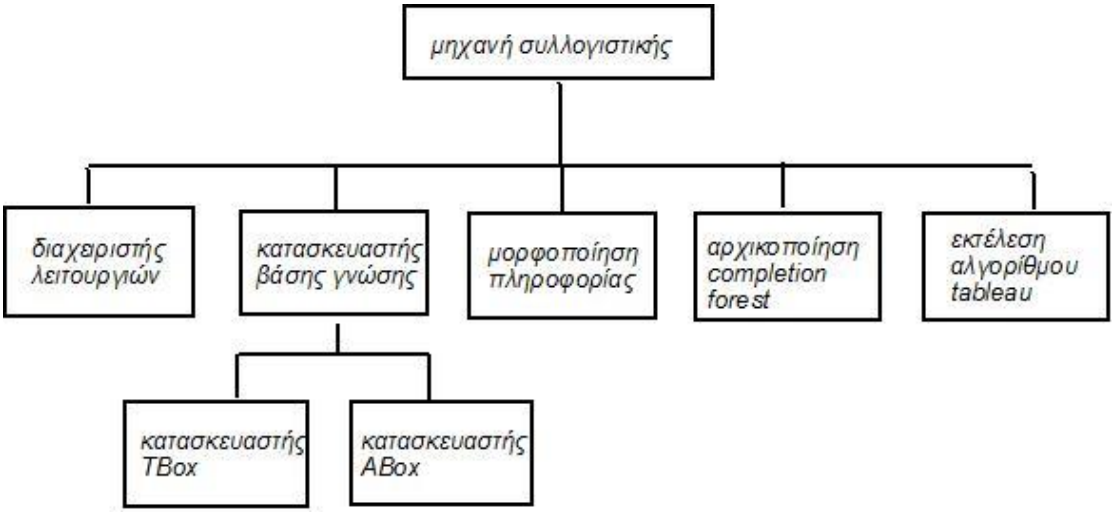
Στο κεφάλαιο αυτό παρουσιάζουμε την αρχιτεκτονική του συστήματος και αναλύουμε τις απαιτήσεις για τις λειτουργίες του.

4.1 Γενική αρχιτεκτονική

Η μηχανή συλλογιστικής που κατασκευάστηκε δέχεται ως είσοδο αρχεία οντολογιών, τα οποία χειρίζεται με σκοπό να ανακτήσει όλες τις απαραίτητες πληροφορίες για την κατασκευή της βάσης γνώσης. Πρόκειται για αρχεία *.owl από τα οποία ανακτώνται όλα τα αξιώματα που είναι συμβατά με την εκφραστικότητα της περιγραφικής λογικής $\mathcal{ALC}$.

Οι πληροφορίες που αντλούνται από την οντολογία μέσω του OWL-API, τροποποιούνται έτσι ώστε να αποκτήσουν μία ενιαία μορφή η οποία διευκολύνει το χειρισμό τους. Το υποσύστημα που αναλαμβάνει την ανάκτηση της πληροφορίας και την μορφοποίησή της, επικοινωνεί με ένα δεύτερο, το οποίο είναι υπεύθυνο για την κατασκευή της βάσης γνώσης. Για το σκοπό αυτό, δομείται το σώμα ορολογίας της γνώσης και έπειτα το σώμα των ισχυρισμών. Πρωτού αποθηκευθούν τα αξιώματα στο σώμα ορολογίας, γίνεται η μετατροπή της μορφής της πληροφορίας, έτσι όπως αυτή ανακτάται μέσω του OWL API. Με τον ίδιο τρόπο, το υποσύστημα- κατασκευαστής της βάσης γνώσης, επικοινωνεί με το υποσύστημα μορφοποίησης της πληροφορίας κατά την κατασκευή του σώματος ισχυρισμών. Τέλος, δεδομένης της βάσης γνώσης, αρχικοποιείται το δάσος πληρότητας (completion forest) και εκτελείται ο αλγόριθμος tableau. Η έναρξη της λειτουργίας των επιμέρους υποσυστημάτων καθώς και η αρχική πρόσβαση στα δεδομένα της οντολογίας, αποτελούν έργο του υποσυστήματος διαχείρησης των λειτουργιών.

Παρακάτω, παρουσιάζουμε και σχηματικά τον τρόπο με τον οποίο η μηχανή συλλογιστικής που κατασκευάστηκε διαχωρίζεται στα επιμέρους υποσυστήματα.



4.2 Περιγραφή Λειτουργιών

Σε αυτό το σημείο, παρουσιάζουμε πιο αναλυτικά, τις λειτουργίες που εκτελεί το σύστημά μας. Παραθέτουμε τα επιμέρους υποσυστήματα, τις περιγραφές αυτών, καθώς και διάγραμμα ροής που περιγράφει τη λειτουργία τους.

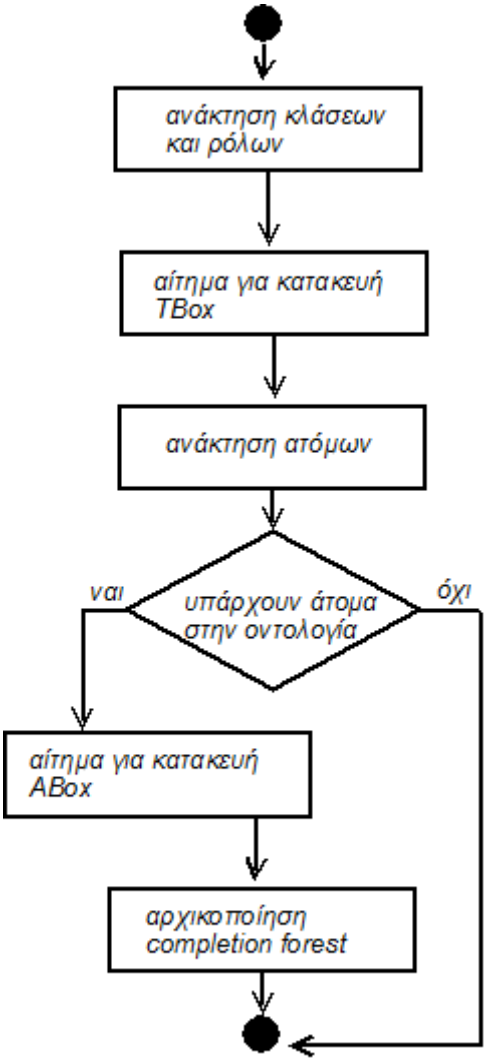
4.2.1 Υποσύστημα διαχείρησης λειτουργιών

Το σύστημά μας ανακτά από την οντολογία τις απαραίτητες πληροφορίες για την κατασκευή της βάσης γνώσης κάνοντας χρήση του OWL-API. Έτσι λοιπόν, χωρίς να διατρέχει όλη την οντολογία, μπορεί να εξάγει το σύνολο των κλάσεων, των ιδιοτήτων, των αντικειμένων και γενικά, όλα τα απαιτούμενα δεδομένα που προορίζονται για τη βάση γνώσης.

Η πρώτη και βασική λειτουργία που επιτελεί, είναι να φορτώσει την οντολογία από το δοσμένο URI και να δημιουργήσει έναν διαχειριστή (manager) ο οποίος είναι υπεύθυνος για το χειρισμό της. Έπειτα, μέσω των κλάσεων του OWL-API, γίνεται η συλλογή των κλάσεων της οντολογίας οι οποίες είναι απαραίτητες για το υποσύστημα που υλοποιεί τη βάση γνώσης και συγκεκριμένα το σώμα της ορολογίας. Εν συνεχεία, συλλέγονται οι ιδιότητες και το σύνολο των ατόμων, εάν αυτά υπάρχουν, προκειμένου να κατασκευαστεί το σώμα των ισχυρισμών.

Τέλος, αφού έχει ήδη οριστεί η βάση γνώσης, δίνεται το έναυσμα την εκτέλεση του αλγορίθμου tableau με την αρχικοποίηση του δάσους πληρότητας.

Παρουσιάζουμε το σύστημα σχηματικά, σε μορφή διαγράμματος ροής.



διαχειριστής λειτουργών συσήματος

4.2.2 Μορφοποίηση πληροφορίας

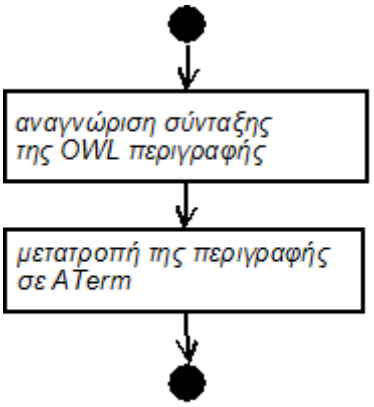
Το υποσύστημα αυτό επεξεργάζεται τα δεδομένα της οντολογίας έτσι όπως

προκύπτουν από τη χρήση του OWL-API. Συγκεκριμένα, οι κλάσεις, οι ιδιότητες, τα άτομα και όλες οι λοιπές οντότητες που ανακτώνται, έχουν τύπο που καθορίζεται από τις κλάσεις του OWL-API. Για παράδειγμα, οι κλάσεις της οντολογίας, ή αλλιώς οι έννοιες της βάσης γνώσης μας, είναι τύπου OWLClass.

Σκοπός του υποσυστήματος αυτού, είναι τα δεδομένα της οντολογίας να αποκτήσουν μία ενιαία μορφή πριν εισαχθούν στη βάση γνώσης. Ο κοινός τρόπος αναπαράστασης των δεδομένων προσφέρει συνοχή καθώς επιτρέπει τον ενιαίο χειρισμό τους, και διευκολύνει την μετέπειτα επεξεργασία τους κατά την εκτέλεση του αλγορίθμου. Η αναπαράστασή τους, γίνεται μέσω των ATerm (Annotated Terms). Ο εν λόγω τύπος δεδομένων, χρησιμοποιείται κατά κύριο λόγο σε περιπτώσεις δενδρικής αναπαράστασης δεδομένων και επιτρέπει την

εκτέλεση ενεργειών όπως είναι για παράδειγμα, το μαρκάρισμα των όρων και η κατασκευή σύνθετων, νέων όρων από τους ήδη υπάρχοντες. Η χρησιμότητα της μορφοποίησης αυτής γίνεται περισσότερο κατανοητή παρακάτω.

Το υποσύστημα που αναλαμβάνει τη μορφοποίηση της πληροφορίας, δέχεται ως είσοδο τα δεδομένα της οντολογίας αμέσως πριν αυτά αποθηκευθούν στη βάση γνώσης. Έτσι, για παράδειγμα, αφού ο κατασκευαστής του σώματος των ισχυρισμών συλλέξει τα άτομα της οντολογίας, τα στέλνει στο υποσύστημα μορφοποίησης, πρωτού εισάγει στη βάση γνώσης τους ισχυρισμούς στους οποίους συμμετέχουν.



μορφοποίηση περιγραφών

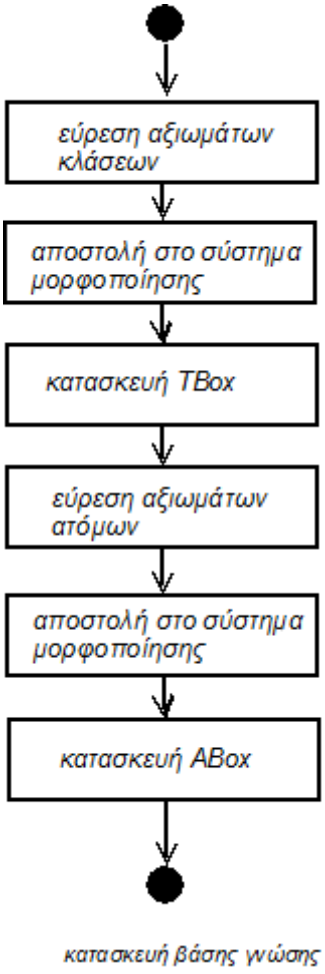
4.2.3 Κατασκευή βάσης γνώσης

Η κατασκευή της βάσης γνώσης ξεκινά από το σώμα ορολογίας. Το υποσύστημα διαχείρησης των λειτουργιών παρέχει τις κλάσεις της οντολογίας. Από τις κλάσεις αυτές και εκμεταλλευόμενος τις μεθόδους του OWL-API, ο κατασκευαστής του σώματος ορολογίας αντλεί από την οντολογία όλες τις περιγραφές που αφορούν τις κλάσεις και κατασκευάζει την ορολογία της βάσης γνώσης. Πιο συγκεκριμένα, αντλούνται τα αξιώματα ισοδυναμίας κλάσεων, υπαγωγής και ξένων εννοιών. Τα αξιώματα αυτά αποτελούν τις περιγραφές των $\mathcal{ALC}$ εννοιών και αφού αποκτήσουν την κατάλληλη μορφή, δηλαδή αφού γίνουν ATerm όροι, αποθηκεύονται στο TBox.

Εν συνεχεία, το υποσύστημα λαμβάνει ως είσοδο τις ιδιότητες της οντολογίας. Η είσοδος προκύπτει και αυτή τη φορά, από το υποσύστημα διαχείρησης λειτουργιών. Για κάθε μία από τις ιδιότητες εντοπίζεται το πεδίο ορισμού και το πεδίο τιμών που συνθέτουν τον ορισμό της. Ο ορισμός αφού αποκτήσει ATerm τύπο, αποθηκεύεται στο σώμα της ορολογίας.

Τέλος, το εν λόγω υποσύστημα καλείται να κατασκευάσει το σώμα ισχυρισμών. Για κάθε ένα από τα άτομα της οντολογίας, που λαμβάνει ως είσοδο από τον διαχειριστή των λειτουργιών, εντοπίζει τις σχέσεις τύπων και τις σχέσεις ρόλων στις οποίες συμμετέχει. Για άλλη μία φορά, αλληλεπιδρά με το υποσύστημα που αναλαμβάνει τη μετατροπή των όρων σε ATerm και αποθηκεύει τους ισχυρισμούς στο ABox.

Σχηματικά, το σύστημα παρουσιάζεται παρακάτω.



κατασκευή βάσης γνώσης

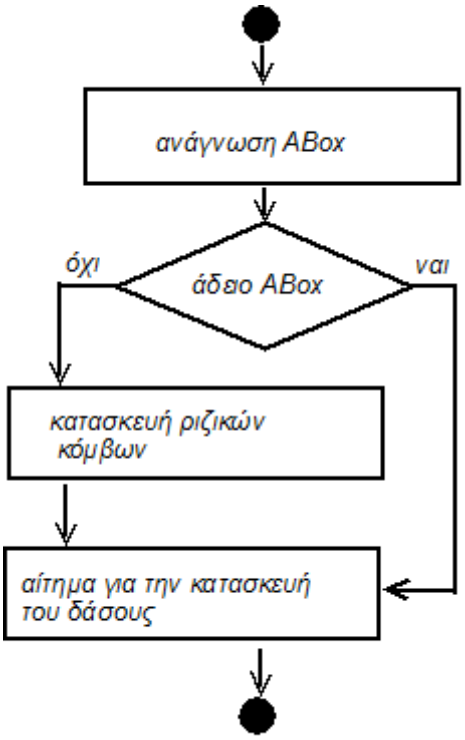
4.2.4 Αρχικοποίηση completion forest

Το σύστημα είναι έτοιμο να προχωρήσει στην εκτέλεση του αλγορίθμου tableau από τη στιγμή που θα κατασκευαστεί η βάση γνώσης του κόσμου μας. Για το σκοπό αυτό αρχικοποιείται το δένδρο πληρότητας. Για κάθε άτομο που έχει καταχωρηθεί στο σώμα ισχυρισμών δημιουργείται ένας ριζικός κόμβος, ο οποίος σημειώνεται με το σύνολο των εννοιών στις οποίες ανήκει το άτομο. Για παράδειγμα, εάν στο ABox περιλαμβάνεται ο ισχυρισμός *Γιώργος:Πατέρας*, τότε δημιουργείται ο ριζικός κόμβος με το όνομα *Γιώργος* και με ετικέτα *Πατέρας*. Εάν μάλιστα, το ABox περιλαμβάνει και τον ισχυρισμό *Γιώργος*: *Επαγγελματίας*, τότε στην ετικέτα του κόμβου *Γιώργος* προστίθεται και η έννοια *Επαγγελματίας*.

Στη συνέχεια, διατρέχονται οι ισχυρισμοί που αφορούν τις σχέσεις ρόλων (role assertions) στις οποίες συμμετέχουν τα άτομα των ριζικών κόμβων. Με άλλα λόγια, για κάθε ένα από τα άτομα των κόμβων που κατασκευάστηκαν, γίνεται έλεγχος εάν συμμετέχει σε κάποια σχέση ρόλων οπότε και θα σχετίζεται με κάποιο άλλο άτομο- κόμβο. Για κάθε

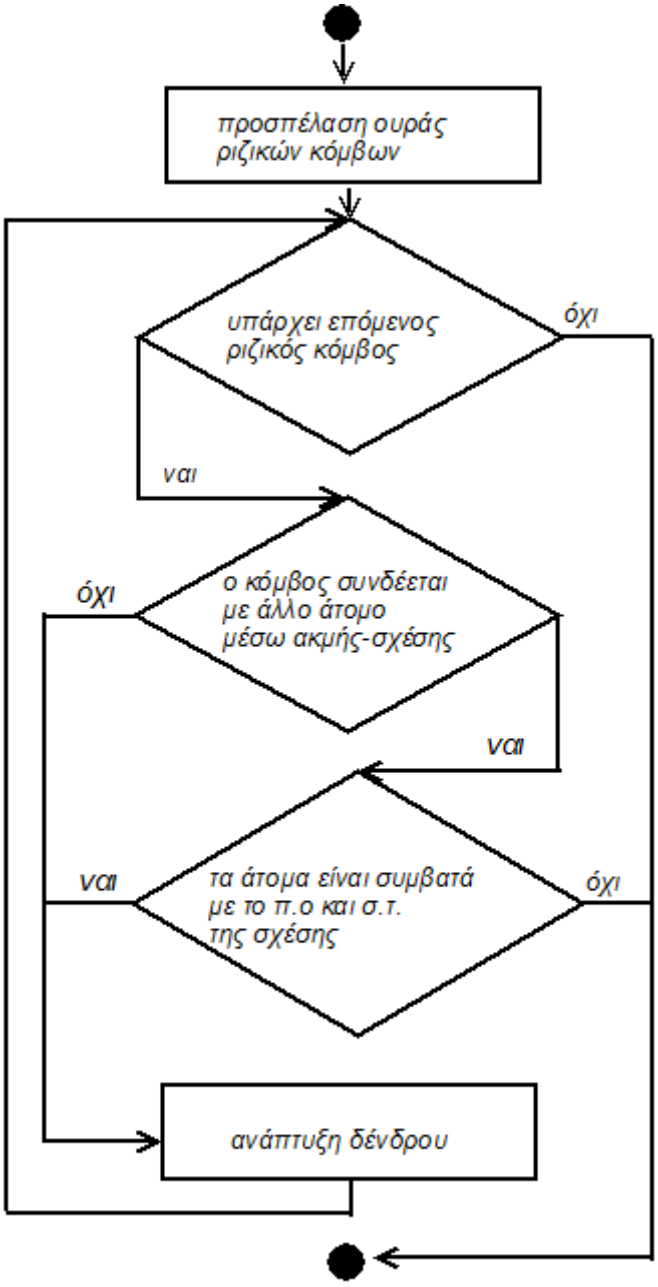
διαφορετικό άτομο με το οποίο σχετίζεται κατασκευάζεται μία ακμή που συνενώνει τους δύο κόμβους μεταξύ τους. Η ακμή αυτή φέρει ετικέτα με το ρόλο του ισχυρισμού.

Τέλος, δίνεται η εντολή για την εκτέλεση του tableau αλγορίθμου.



4.2.5 Εκτέλεση tableau αλγορίθμου

Το υποσύστημα αυτό έχει ως σκοπό την εφαρμογή των κανόνων tableau για την ανάπτυξη του δάσους πληρότητας. Δέχεται ως είσοδο τους ριζικούς κόμβους από τους οποίους ξεκινά και την ανάπτυξη των επιμέρους δένδρων. Στη διάθεση του υποσυστήματος είναι όπως πάντα η βάση γνώσης, από την οποία αντλεί τις περιγραφές των εννοιών και τα αξιώματα στα οποία συμμετέχουν προκειμένου να εφαρμόσει τους κανόνες.



εκτέλεση αλγορίθμου tableau

5 *Σχεδίαση Συστήματος*

Στο κεφάλαιο αυτό παρουσιάζουμε πιο αναλυτικά τη σχεδίαση της μηχανής συλλογιστικής που αναπτύχθηκε.

5.1 Δομή συστήματος

Το σύστημα αναπτύχθηκε σε Java και αποτελείται από δεκαοκτώ κλάσεις. Η κλάση που χειρίζεται ως είσοδο το αρχείο της οντολογίας και εκκινεί τις λειτουργίες του συστήματος είναι η Start. Φορτώνει την οντολογία από το δοσμένο URI και ορίζει τον manager-διαχειριστή αυτής. Ανακτά από την οντολογία τις κλάσεις, τις οποίες στέλνει στην ClassAxiomConstructor.java για την κατασκευή των περιγραφών των εννοιών που θα αποθηκευθούν στο TBox.java. Σημειώνουμε σε αυτό το σημείο, ότι το σώμα της ορολογίας TBox, καθώς και το σώμα των ισχυρισμών ABox, αναπαρίστανται από τις ομώνυμες κλάσεις. Οι περιγραφές των εννοιών γίνονται είτε μέσω αξιωμάτων υπαγωγής, είτε με αξιώματα ισοδυναμίας. Τα αντικείμενα της κλάσης ClassDefinition.java αποδίδουν τους ορισμούς αυτούς.

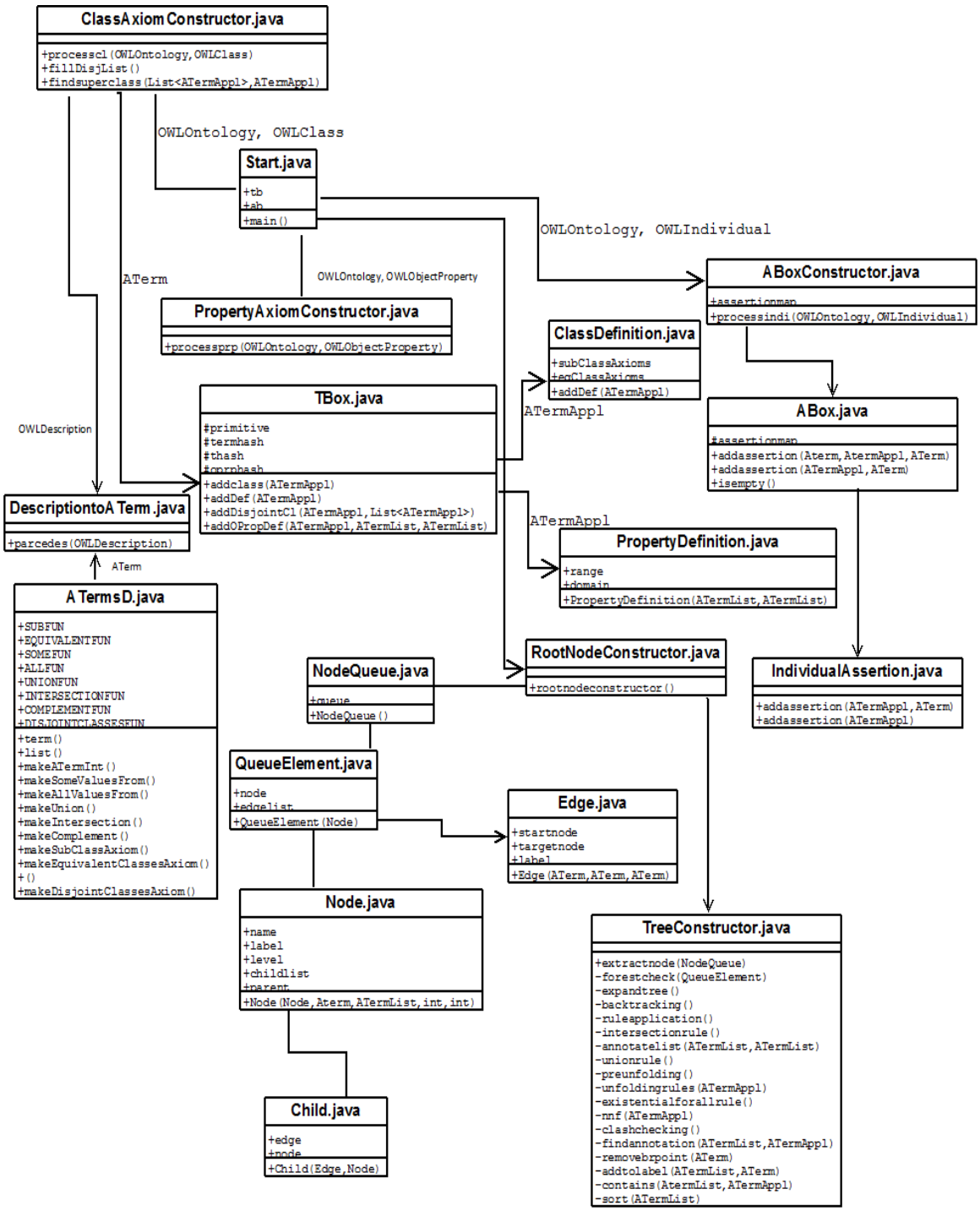
Εν συνεχεία, η Start ανακτά από την οντολογία τις ιδιότητες της οντολογίας, τις οποίες στέλνει στην κλάση PropertyAxiomConstructor.java, για την κατασκευή του ορισμού των ιδιοτήτων. Ο ορισμός της ιδιότητας περιλαμβάνει το πεδίο ορισμού και το σύνολο τιμών και αναπαρίσταται από την κλάση PropertyDefinition.java. Η PropertyAxiomConstructor.java αποθηκεύει τους εν λόγω ορισμούς στο TBox.

Τέλος, η Start.java, ανακτά τα άτομα της οντολογίας τα οποία στέλνει ως παράμετρο στην μέθοδο της ABoxConstructor.java, με σκοπό την κατασκευή του σώματος των ισχυρισμών. Οι ισχυρισμοί του ABox είναι είτε σχέσεις τύπων, είτε σχέσεις ρόλων. Η αναπαράστασή τους γίνεται μέσω της κλάσης IndividualAssertions.java.

Η αποθήκευση των αξιωμάτων και ορισμών στα TBox και ABox γίνεται αφού προηγουμένως η DescriptiontoATerm.java μετατρέψει τις OWL περιγράφες σε ATerm όρους. Η DescriptiontoATerm.java αναγνωρίζει τον τύπο της OWL περιγραφής και καλεί τις μεθόδους της ATermsD.java για την κατασεκυή του αντίστοιχου ATerm τύπου.

Όλες οι προαναφερθείσες κλάσεις αφορούν την κατασκευή της βάσης γνώσης από την οντολογία εισόδου. Οι υπόλοιπες κλάσεις του συστήματος αφορούν την ανάπτυξη του δάσους πληρότητας κατά την εκτέλεση των κανόνων tableau. Συγκεκριμένα, η RootNodeConstructor.java διατρέχοντας τα αξιώματα του ABox, αρχικοποιεί το δάσος καθώς κατασκευάζει μία ουρά από τους αρχικούς-ριζικούς κόμβους. Η ουρά αναπαρίσταται από την κλάση NodeQueue.java και αποτελείται από αντικείμενα της κλάσης QueueElement.java. Τα στοιχεία της ουράς έχουν δύο πεδία, τον κόμβο, Node, και μία λίστα από ακμές, Edge. Η RootNodeConstructor.java, αφού αρχικοποιήσει το δάσος εκκινεί την ανάπτυξη του αλγορίθμου. Η κλάση που εφαρμόζει τους κανόνες tableau είναι η TreeConstructor.java, και είναι ίσως η μεγαλύτερης σημασίας κλάση του συστήματος.

Παρακάτω παρουσιάζουμε ένα απλοποιημένο uml διάγραμμα των κλάσεων.


ClassAxiomConstructor.java
+processcl(OWLOntology,OWLClass)
+fillDisjList()
+findsuperclass(List<ATermAppl>,ATermAppl)
OWLOntology, OWLClass
Start.java
+tb
+ab
+main()
OWLOntology, OWLIndividual
ABoxConstructor.java
+assertionmap
+processindi(OWLOntology,OWLIndividual)
ATerm
OWLOntology, OWLObjectProperty
PropertyAxiomConstructor.java
+processprp(OWLOntology,OWLObjectProperty)
ClassDefinition.java
+subClassAxioms
+eqClassAxioms
+addDef(ATermAppl)
ATermAppl
TBox.java
#primitive
#termhash
#thash
#opprphash
+addclass(ATermAppl)
+addDef(ATermAppl)
+addDisjointCl(ATermAppl,List<ATermAppl>)
+addOPropDef(ATermAppl,ATermList,ATermList)
ABox.java
#assertionmap
+addassertion(Aterm,AtermAppl,ATerm)
+addassertion(ATermAppl,ATerm)
+isempty()
OWLDescription
DescriptiontoA Term.java
+parcedes(OWLDescription)
ATerm
ATermAppl
PropertyDefinition.java
+range
+domain
+PropertyDefinition(ATermList,ATermList)
ATermsD.java
+SUBFUN
+EQUIVALENTFUN
+SOMEFUN
+ALLFUN
+UNIONFUN
+INTERSECTIONFUN
+COMPLEMENTFUN
+DISJOINTCLASSESFUN
+term()
+list()
+makeATermInt()
+makeSomeValuesFrom()
+makeAllValuesFrom()
+makeUnion()
+makeIntersection()
+makeComplement()
+makeSubClassAxiom()
+makeEquivalentClassesAxiom()
+()
+makeDisjointClassesAxiom()
NodeQueue.java
+queue
+NodeQueue()
RootNodeConstructor.java
+rootnodeconstructor()
IndividualAssertion.java
+addassertion(ATermAppl,ATerm)
+addassertion(ATermAppl)
QueueElement.java
+node
+edgelist
+QueueElement(Node)
Edge.java
+startnode
+targetnode
+label
+Edge(ATerm,ATerm,ATerm)
Node.java
+name
+label
+level
+childlist
+parent
+Node(Node,Aterm,ATermList,int,int)
TreeConstructor.java
+extractnode(NodeQueue)
-forestcheck(QueueElement)
-expandtree()
-backtracking()
-ruleapplication()
-intersectionrule()
-annotatelist(ATermList,ATermList)
-unionrule()
-preunfolding()
-unfoldingrules(ATermAppl)
-existentialforallrule()
-nnf(ATermAppl)
-clashchecking()
-findannotation(ATermList,ATermAppl)
-removebrpoint(ATerm)
-addtolabel(ATermList,ATerm)
-contains(AtermList,ATermAppl)
-sort(ATermList)
Child.java
+edge
+node
+Child(Edge,Node)

5.2 Περιγραφή Κλάσεων

Στις παρακάτω ενότητες δίνουμε μία περιεκτική περιγραφή των κλάσεων του συστήματος.

1.2.2 5.2.1 Κλάση Start.java

Η κλάση Start όπως έχουμε ήδη αναφέρει, συντονίζει κατά κάποιο τόπο τις λειτουργίες των επιμέρους υποσυστημάτων.

Αρχικά, στην κλάση αυτή και με την εκκίνηση της λειτουργίας του συστήματος, κατασκευάζεται ένα αντικείμενο της κλάσης TBox και ένα αντικείμενο της κλάσης ABox. Πρόκειται για ένα κενό σώμα ορολογίας και σώμα ισχυρισμών, τα οποία στη συνέχεια συμπληρώνονται βάσει της οντολογίας που δίνεται ως είσοδος.

Δημιουργείται επομένως, ένα διαχειριστής του αρχείου .owl το οποίο αποτελεί την οντολογία εισόδου. Με τη βοήθεια του διαχειριστή, φορτώνεται η οντολογία ώστε να γίνει δυνατή η ανάγνωσή της. Για το σκοπό αυτό χρησιμοποιούνται οι κλάσεις του OWL-API. Εν συνεχεία, συγκεντρώνονται όλες τις κλάσεις που περιέχονται στην οντολογία. Για κάθε μία από τις κλάσεις, η Start επικοινωνεί με την ClassAxiomConstructor.java στέλνοντας ως παράμετρο στη μέθοδο processcl(OWLOntology, OWLClass), την κλάση, καθώς και ολόκληρη την οντολογία, για την περαιτέρω επεξεργασία της. Τέλος, καλεί τη μέθοδο fillDisjList προκειμένου να συμπληρωθεί η λίστα στο σώμα ορολογίας με τις ξένες μεταξύ τους έννοιες.

Επόμενη λειτουργία της Start, είναι η συλλογή των ιδιοτήτων της οντολογίας. Η οντολογία διαθέτει διαφόρων τύπων ιδιότητες ατόμων. Ο τύπος που αντιστοιχεί στις σχέσεις της $\mathcal{ALC}$, είναι ο OWLObjectProperty. Συγκεκριμένα, η OWL2 σαφώς υποστηρίζει ιδιότητες που συσχετίζουν άτομα με κάποιο τύπο δεδομένων. Όμως, οι τύποι δεδομένων δεν υποστηρίζονται από την $\mathcal{ALC}$ και για το λόγο αυτό δεν ανακτώνται από την οντολογία. Κάθε μία από τις ιδιότητες περνά ως παράμετρος στη μέθοδο processprp της κλάσης PropertyAxiomConstructor.

Σε αυτό το σημείο, η κατασκευή του σώματος ορολογίας έχει ολοκληρωθεί και απομένει η κατασκευή του σώματος ισχυρισμών. Όμοια με τα προηγούμενα, γίνεται η συλλογή των ατόμων της οντολογίας και κάθε ένα από αυτά αποτελεί παράμετρο κατά την κλήση της μεθόδου processindi της κλάσης ABoxConstructor.

Τέλος, κατασκευάζεται ένα αντικείμενο της RootNodeConstructor προκειμένου να αρχικοποιηθεί το δάσος πληρότητας.

1.2.3 5.2.2 Κλάση ClassAxiomConstructor.java

Σκοπός της κλάσης αυτής είναι η κατασκευή των περιγραφών των εννοιών της βάσης γνώσης. Η μέθοδος processcl(OWLOntology, OWLClass) δέχεται ως παράμετρο την οντολογία και κάθε μία από τις κλάσεις που ανήκουν σε αυτή. Χρησιμοποιώντας τις μεθόδους του OWL-API, ανιχνεύει το σύνολο των αξιωμάτων στα οποία συμμετέχουν οι κλάσεις. Τα αξιώματα αυτά είναι αξιώματα υπαγωγής, ισοδυναμίας και ξένων εννοιών. Έτσι, μόλις για παράδειγμα, ανιχνευθεί ένα αξίωμα υπαγωγής στο οποίο η κλάση συμμετέχει ως υποκλάση, εξάγεται η περιγραφή της υπερκλάσης. Η περιγραφή περνά ως παράμετρος στη μέθοδο parcedes(OWLDescription) της DescriptiontoATerm.java προκειμένου ο τύπος της να μετατραπεί από OWLDescription σε ATermAppl. Στη συνέχεια, καλείται η μέθοδος makeSubClassAxiom της ATermsD.java με παραμέτρους την κλάση και την υπερκλάση της σε μορφή ATermAppl. Το αποτέλεσμα είναι η δημιουργία του αξιώματος υπαγωγής σε μορφή ATermAppl όρου. Σε αυτή τη μορφή το αξίωμα αποθηκεύεται στο TBox. Η ίδια διαδικασία ακολουθείται και για τα αξιώματα ισοδυναμίας και τα αξιώματα των ξένων εννοιών. Γίνεται δηλαδή, η ανάκτηση των απαραίτητων στοιχείων από την οντολογία, η αναγνώρισή τους και η μετατροπή του τύπου τους από την parcedes(OWLDescription) της DescriptiontoATerm.java. Τέλος, το αντίστοιχο αξίωμα κατασκευάζεται σε μορφή ATermAppl όρου από την ATermsD.java. Στην περίπτωση που η κλάση δεν περιγράφεται από κάποιο αξίωμα, αποτελεί μία πρωταρχική έννοια της $\mathcal{ALC}$, οπότε και αποθηκεύεται ως τέτοια στο TBox , αφού προηγουμένως μετατραπεί σε ATermAppl.

Η μέθοδος fillDisjList() συμπληρώνει τις λίστες των ξένων εννοιών. Κάθε έννοια του σώματος ορολογίας αντιστοιχίζεται σε μία λίστα από έννοιες που είναι ξένες προς αυτή. Η μέθοδος fillDisjList() αναλαμβάνει να συμπληρώσει τη λίστα μίας έννοιας με το συμπλήρωμά της. Επιπλέον, στη λίστα προστίθεται το συμπλήρωμα των ισοδύναμων εννοιών καθώς και των εννοιών που συνιστούν την υπερκλάση στα αξιώματα υπαγωγής στα οποία συμμετέχει η εν λόγω έννοια. Για παράδειγμα, εάν στο σώμα ορολογίας εμφανίζεται το αξίωμα υπαγωγής : $A \sqsubseteq B$, τότε στη λίστα των ξένων εννοιών της A προστίθεται το συμπλήρωμα $\neg A$, καθώς και το συμπλήρωμα της έννοιας B.

1.2.4 5.2.3 Κλάση PropertyAxiomConstructor.java

Η κλάση αυτή κατασκευάζει τους ορισμούς των σχέσεων της βάσης γνώσης. Σε κάθε κλήση της, η μέθοδος processprop(OWLOntology, OWLObjectProperty), ανακτά από την οντολογία το πεδίο ορισμού και το σύνολο τιμών της ιδιότητας της παραμέτρου. Το πεδίο ορισμού ανακτάται από την οντολογία με χρήση μεθόδων του OWL-API και έχει τη μορφή Owl περιγραφής, με τύπο OWLDescription. Όπως συμβαίνει και με την ClassAxiomConstructor.java, η περιγραφή του πεδίου ορισμού αποκτά τύπο ATermAppl

μετά την κλήση της parcedes(OWLDescription) της DescriptiontoATerm.java. Αφού συμβεί το ίδιο και με το σύνολο τιμών, προστίθεται ο ορισμός της ιδιότητας στο σώμα της ορολογίας.

1.2.5 5.2.4 Κλάση ClassDefinition.java

Η ClassDefinition αναπαριστά τον ορισμό έννοιας στο σώμα της ορολογίας. Αποτελείται από δύο πεδία, μία λίστα με τις ισοδύναμες έννοιες, και άλλη μία με τις έννοιες που ανήκουν στο δεξιό σκέλος των αξιωμάτων υπαγωγής.

Η μέθοδος addDef(ATermAppl) δέχεται ως παράμετρο ένα αξίωμα με τύπο ATermAppl, έτσι όπως έχει κατασκευαστεί από την ClassAxiomConstructor.java. Εκμεταλλευόμενη τη δομή του ATermAppl όρου, αναγνωρίζει εάν πρόκειται για αξίωμα υπαγωγής ή ισοδυναμίας. Ανάλογα με την περίπτωση, εξάγεται η ισοδύναμη έννοια ή η έννοια του δεξιού σκέλους του αξιώματος υπαγωγής, και προστίθεται στην αντίστοιχη λίστα. Με τον τρόπο αυτό συμπληρώνεται η περιγραφή μίας έννοιας στο σώμα της ορολογίας.

1.2.6 5.2.5 Κλάση PropertyDefinition.java

Αντίστοιχα με την ClassDefinition, τα αντικείμενα της PropertyDefinition αναπαριστούν τον ορισμό μίας σχέσης της βάσης γνώσης. Αποτελείται από τα πεδία range και domain που συνιστούν το σύνολο τιμών και το πεδίο ορισμού αντίστοιχα. Τα πεδία αυτά περνάνε ως παράμετροι στον κατασκευαστή της κλάσης, με τύπο ATerm ο οποίος κατασκευάζεται από την PropertyAxiomConstructor.java.

1.2.7 5.2.6 Κλάση TBox.java

Αναπαριστά το σώμα ορολογίας της βάσης γνώσης στο οποίο παρουσιάζονται οι περιγραφές των εννοιών. Το πρώτο πεδίο, primitive, αποτελεί το σύνολο των πρωταρχικών εννοιών, δηλαδή τις έννοιες εκείνες που δεν περιγράφονται στη βάση γνώσης. Το επόμενο πεδίο, αντιστοιχίζει σε κάθε έννοια την περιγραφή της με τύπο ClassDefinition, και ουσιαστικά αντιπροσωπεύει το σώμα ορολογίας. Στο συγκεκριμένο πεδίο συγκεντρώνονται όλα τα αξιώματα που αφορούν την υπαγωγή και την ισοδυναμία των εννοιών. Ακολουθεί ο πίνακας thash, όπου η κάθε έννοια αντιστοιχίζεται σε μία λίστα με ξένες έννοιες προς αυτή. Το τελευταίο πεδίο περιγράφει τους ορισμούς των σχέσεων που παρουσιάζονται στη βάση γνώσης μας. Κάθε σχέση αντιστοιχίζεται στον ορισμό της PropertyDefinition, όπου ορίζεται το σύνολο τιμών και το πεδίο ορισμού της.

Οι μέθοδοι της TBox.java, εξυπηρετούν στην προσθήκη των διάφορων ορισμών σε κάθε ένα από τα προαναφερθέντα πεδία. Συγκεκριμένα, η addclass δέχεται ως παράμετρο μία πρωταρχική κλάση και την προσθέτει στο πρώτο πεδίο. Η κλήση της μεθόδου γίνεται από την ClassAxiomConstructor.java, η οποία αφού προηγούμενως εντοπίσει την πρωταρχική κλάση

στην οντολογία, μετατρέπει τον τύπο της σε ATerm από OWLDescription και καλεί την addclass.

Παρόμοια διαδικασία ακολουθείται προκειμένου να προστεθεί στο πεδίο του σώματος της ορολογίας ο ορισμός μίας κλάσης. Η ClassAxiomConstructor.java καλεί την μέθοδο addDef με παράμετρο το αξίωμα που πρόκειται να προστεθεί το οποίο έχει ήδη μετατραπεί σε όρο ATermAppl. Η addDef ανιχνεύει, εκμεταλλευόμενη τη δομή του ATermAppl τύπου, εάν η κλάση που συμμετέχει στο αξίωμα έχει ήδη καταχωρηθεί στο σώμα της ορολογίας. Στην περίπτωση που υπάρχει ήδη μία περιγραφή της εν λόγω κλάσης, το νέο αξίωμα που προέκυψε λαμβάνεται υπ' όψιν στην περιγραφή της. Εάν η κλάση δεν έχει οριστεί πρωτύτερα στο σώμα της ορολογίας, κατασκευάζεται ο ορισμός της, ως αντικείμενο της κλάσης ClassDefinition.java και το νέο ζευγάρι κλάση-ορισμός προστίθεται στο αντίστοιχο πεδίο της TBox.

Η ενημέρωση του πεδίου thash γίνεται μέσω της μεθόδου addDisjCl. Καλείται από την ClassAxiomConstructor.java με παραμέτρους μία κλάση και μία λίστα κλάσεων η οποία αντιπροσωπεύει τη λίστα ξένων εννοιών που πρόκειται να προστεθεί.

Τέλος, η ενημέρωση του πίνακα των ορισμών των σχέσεων γίνεται μέσω της addOPropDef.

1.2.8 5.2.7 Κλάση ABoxConstructor.java

Πρόκειται για τον κατασκευαστή του σώματος ορολογίας. Διαθέτει τη μέθοδο processindi η οποία καλείται από την Start.java για κάθε ένα άτομο της οντολογίας, με παραμέτρους την οντολογία και το άτομο. Αρχικά το άτομο μετατρέπεται σε όρο ATerm με μία κλήση της ATermsD.java. Έπειτα, η ABoxConstructor.java εντοπίζει στην οντολογία το σύνολο των αξιωμάτων στα οποία συμμετέχει το άτομο. Τα αξιώματα ενδέχεται να είναι είτε ισχυρισμοί τύπων, είτε ισχυρισμοί σχέσεων. Στην περίπτωση του ισχυρισμού τύπου, η OWL περιγραφή του, στέλνεται ως παράμετρος στην parcedes της DescriptiontoATerm με σκοπό να αποκτήσει ATerm μορφή. Στη δεύτερη περίπτωση, που πρόκειται για ισχυρισμό σχέσης, εξαγάγεται η σχέση καθώς και το άτομο το οποίο συμμετέχει σε αυτή. Ο τύπος της σχέσης, OWLObjectPropertyExpression και του ατόμου OWLIndividual, μετατρέπονται σε ATerm όρους. Τέλος, οι όροι ATerm αποστέλλονται ως παράμετροι στη μέθοδο addassertion της ABox.java, προκειμένου να ενημερωθεί το σώμα ισχυρισμών.

1.2.9 5.2.8 Κλάση IndividualAssertions.java

Πρόκειται για την κλάση η οποία αναπαριστά το σύνολο των ισχυρισμών στους οποίους συμμετέχει ένα άτομο. Το πεδίο typeassertion αποτελεί μία λίστα από έννοιες στις οποίες ανήκει το άτομο. Το πεδίο objectpropertyassertion περιλαμβάνει τις σχέσεις καθώς και το άτομο με το οποίο συμμετέχει στη σχέση αυτή. Έτσι για παράδειγμα, εάν στην οντολογία

εμφανίζεται ο ισχυρισμός τύπου: Βασίλης:Πατέρας, και ο ισχυρισμός σχέσης: (Βασίλης, Κλεοπάτρα):έχειΠαιδί, τότε στο άτομο Βασίλης θα αντιστοιχεί ένα αντικείμενο της κλάσης IndividualAssertions.java, με πεδίο typeassertion την έννοια Πατέρας, και πεδίο objectpropertyassertion τη σχέση έχειΠαιδί που αντιστοιχίζεται στο άτομο Κλεοπάτρα.

Η μέθοδος addassertion με παράμετρο έννοια με τύπο ATerm, αναλαμβάνει την προσθήκη της έννοιας στην λίστα typeassertion του ατόμου. Εάν η addassetion κληθεί με δύο παραμέτρους, οι οποίες αναπαριστούν μία σχέση και ένα άτομο, τότε δημιουργεί μία αντιστοιχία μεταξύ τους και τις προσθέτει στο πεδίο objectpropertyassertion.

1.2.10 5.2.9 Κλάση ABox.java

Όπως έχει προαναφερθεί η ABox.java αντιπροσωπεύει το σώμα ισχυρισμών. Το πεδίο της κλάσης assertionmap, είναι μία δομή πίνακα η οποία αντιστοιχίζει το κάθε άτομο που εμφανίζεται στην οντολογία σε ένα αντικείμενο της κλάσης IndividualAssertions.java. Πρόκειται ουσιαστικά για το σώμα ισχυρισμών της βάσης γνώσης.

Σε ό,τι αφορά τις μεθόδους, η addassertion καλείται από την ABoxConstructor.java με σκοπό την προσθήκη νέων ισχυρισμών που αφορούν ένα άτομο. Οι παράμετροι της μεθόδου διαφέρουν ανάλογα με το είδος του ισχυρισμού. Στην περίπτωση που ένα άτομο ανήκει σε έναν ισχυρισμό τύπου, η ABoxConstructor.java καλεί την addassertion με παραμέτρους το άτομο και την έννοια στην οποία ανήκει. Εάν το άτομο εμφανίζεται για πρώτη φορά, η addassertion κατασκευάζει ένα αντικείμενο της IndividualAssertions.java και το αντιστοιχίζει στο άτομο. Στη συνέχεια, καλεί την ομώνυμη μέθοδο της IndividualAssertions.java και φροντίζει ώστε η έννοια να προστεθεί στη λίστα εννοιών που αντιστοιχεί στο άτομο. Στην περίπτωση που το άτομο ανήκει σε ισχυρισμό σχέσης, η addassertion καλείται με παραμέτρους το άτομο, τη σχέση, καθώς και το άτομο με το οποίο συμμετέχει σε αυτή. Όμοια με πριν, καλείται η μέθοδος της IndividualAssertions.java και ενημερώνεται ο πίνακας που περιλαμβάνει τους ισχυρισμούς σχέσης του ατόμου.

1.2.11 5.2.10 Κλάση DescriptiontoATerm.java

Η κλάση αυτή όπως έχουμε ήδη αναφέρει, διαθέτει τη μέθοδο parcedes η οποία μετατρέπει τις OWL περιγραφές σε ATermAppl όρους. Πιο συγκεκριμένα, καλείται με παράμετρο μία OWLDescription, δηλαδή μία περιγραφή, η οποία προκύπτει από την ανάγνωση της οντολογίας μέσω των κλάσεων του OWL API. Σκοπός της μεθόδου είναι να ανιχνεύσει τη δομή της OWLDescription.

Οι περιγραφές της OWL 2 οι οποίες είναι συμβατές με την εκφραστικότητα της $\mathcal{ALC}$ είναι οι ακόλουθες:

Class Expressions :=

- *class*
- *ObjectUnionOf*
- *ObjectIntersectionOf*
- *ObjectComplementOf*

Όπου τα ανωτέρω ορίζονται ως εξής:

ObjectUnionOf:= 'UnionOf' '(' ClassExpression ClassExpression {ClassExpression} ')'

ObjectIntersectionOf:= 'IntersectionOf' '(' ClassExpression ClassExpression {ClassExpression} ')'

ObjectComplementOf:= 'ComplementOf' '(' ClassExpression ')'

Object Property Expressions:=

- *ObjectAllValuesFrom*
- *ObjectSomeValuesFrom*

ObjectAllValuesFrom:= 'AllValuesFrom' '(' ObjectPropertyExpression ClassExpression ')'

ObjectSomeValuesFrom:= 'SomeValuesFrom' '(' ObjectPropertyExpression ClassExpression ')'

Η parcedes ανιχνεύει τον τύπο της έκφρασης και την αποσυνθέτει στα συστατικά της μέρη τα οποία μετατρέπει σε όρους ATerm. Η έκφραση στη συνέχεια, ανασυντίθεται και παίρνει τη μορφή ATermAppl. Τον όρο ATermAppl επιστρέφει η parcedes ως αποτέλεσμα της κλήσης της. Για παράδειγμα, έστω ότι ανιχνεύεται ότι η παράμετρος της μεθόδου είναι μία περιγραφή τύπου ObjectUnion, πρόκειται δηλαδή για μία ένωση κλάσεων. Διακρίνονται τα συστατικά μέρη της ένωσης. Στη συνέχεια κάθε ένα από αυτά εξετάζεται εάν αποτελεί μία OWL κλάση ή εάν πρόκειται για μία σύνθετη OWL περιγραφή (OWL Description) οπότε και καλείται αναδρομικά η parcedes. Η αναδρομική κλήση της μεθόδου είναι απαραίτητη καθώς όπως φαίνεται από τον ορισμό της OWL περιγραφής, στην έκφραση ObjectUnion ενδέχεται να υπάρχουν εμφωλευμένες περισσότερες από μία OWL περιγραφές. Η αναδρομική κλήση συνεχίζεται εώς ότου βρεθούν οι επιμέρους κλάσεις που απαρτίζουν την ένωση. Εν συνεχεία, οι κλάσεις μετατρέπονται σε όρους με τύπο ATerm. Όλη η έκφραση αποκτά τύπο ATermAppl με μία κλήση στην ATermsD.java.

1.2.12 5.2.11 Κλάση ATermsD.java

Είναι η κλάση υπεύθυνη για την κατασκευή των ATerm όρων. Οι μέθοδοι που διαθέτει κατασκευάζουν ATerm όρους δοθέντος ενός String, μέθοδος term, επίσης κατασκευάζουν

λίστες τύπου ATermList δοθέντων λιστών με ATerm όρους, μέθοδος list, και ATermInt όρους από ακεραίους τύπου interger, μέθοδος makeATermInt.

Σχετικά με τον τύπο ATermAppl αναφέρουμε ότι εκφράζει όρους ATerm στους οποίους εφαρμόζεται μία συνάρτηση η AFun. Η συνάρτηση αυτή διαθέτει συγκεκριμένο όνομα και καθορισμένο αριθμό παραμέτρων. Έτσι, για παράδειγμα, εκφράζουμε τον τελεστή της ένωσης κλάσεων με την AFun UNIONFUN, η οποία ορίζουμε να δέχεται μία μοναδική παράμετρο. Η παράμετρός της αναφέρεται στη λίστα των κλάσεων που συνθέτουν την ένωση. Η UNIONFUN εφαρμόζεται στην ATermList των κλάσεων της ένωσης, με τη μέθοδο makeUnion. Η makeUnion καλεί την κατασκευάστρια συνάρτηση των ATermAppl όρων, makeAppl. Αποτέλεσμα είναι ένας όρος ATermAppl.

Με παρόμοιο τρόπο κατασκευάζονται ATermAppl όροι προκειμένου να παραστήσουν την τομή κλάσεων, την εφαρμογή του καθολικού καθώς και του υπαρξιακού τελεστή σε εκφράσεις, το συμπλήρωμα κλάσης, καθώς και τα αξιώματα υπαγωγής και ισοδυναμίας κλάσεων. Χαρακτηριστικά αναφέρουμε ότι η μέθοδος makeEquivalentClassesAxiom δέχεται ως παράμετρο μία OWLClass την οποία μετατρέπει σε ATerm, και μία λίστα ATermList με τις ισοδύναμες κλάσεις. Κατασκευάζει έναν ATermAppl όρο ο οποίος προκύπτει από την εφαρμογή την συνάρτησης EQUIVALENTFUN στην κλάση με τύπο ATerm και στη λίστα των ισοδύναμων κλάσεων με τύπο ATermList. Προφανώς, η EQUIVALENTFUN είναι μία συνάρτηση AFUN με δύο ορίσματα.

1.2.13 5.2.12 Κλάση RootNodeConstructor.java

Η συγκεκριμένη κλάση διαθέτει τη μέθοδο rootnodeconstructor η οποία καλείται από την Start.java με σκοπό να κατασκευάσει τους ριζικούς κόμβους του completion forest το οποίο πρόκειται να αναπτυχθεί.

Αρχικά, γίνεται έλεγχος του σώματος ισχυρισμών και συγκεκριμένα του πεδίου assertionmap, προκειμένου να διαπιστωθεί εάν είναι άδειο ή περιλαμβάνει άτομα. Εάν είναι άδειο τυπώνεται αντίστοιχο μήνυμα και η διαδικασία τερματίζεται. Εάν υπάρχουν άτομα, κατασκευάζεται ένα αντικείμενο της κλάσης NodeQueue.java, το οποίο αναπαριστά την ουρά που περιλαμβάνει τους ριζικούς κόμβους του δάσους. Στη συνέχεια, για κάθε ένα από τα άτομα του σώματος των ισχυρισμών, κατασκευάζεται ένας κόμβος τύπου Node. Ο κόμβος θα αποτελέσει τμήμα του στοιχείου της ουράς με τύπο QueueElement. Τέλος, καλείται η μέθοδος extractnode της κλάσης TreeNodeConstructor για την ανάπτυξη του δάσους ξεκινώντας από την ουρά των ριζικών κόμβων.

1.2.14 5.2.13 Κλάση NodeQueue.java

Η κλάση περιλαμβάνει το πεδίο queue που είναι μία λίστα με στοιχεία τύπου QueueElement. Αντιπροσωπεύει όπως προαναφέρθηκε μία ουρά με τους ριζικούς κόμβους του δάσους πληρότητας.

1.2.15 5.2.14 Κλάση QueueElement.java

Αντιπροσωπεύει τα επιμέρους στοιχεία που συνθέτουν την ουρά NodeQueue. Διαθέτει δύο πεδία node και edgelist. Το πρώτο αφορά τον κόμβο τύπου Node, και το δεύτερο τη λίστα με τις ακμές που ξεκινούν από τον εν λόγω κόμβο.

Ο κατασκευαστής της κλάσης δέχεται ως παράμετρο ένα στοιχείο τύπου Node από το οποίο παίρνει την τιμή του το πρώτο πεδίο της κλάσης. Στη συνέχεια, γίνεται η αναζήτηση στο σώμα ισχυρισμών, των σχέσεων στις οποίες αντιστοιχεί το άτομο για το οποίο κατασκευάστηκε ο κόμβος. Για κάθε μία από τις σχέσεις κατασκευάζεται μία ακμή τύπου Edge. Το αντικείμενο της Edge.java προστίθεται στη λίστα του πεδίου edgelist.

1.2.16 5.2.15 Κλάση Node.java

Το πρώτο πεδίο της κλάσης, name, είναι το όνομα του κόμβου το οποίο ταυτίζεται με το όνομα του ατόμου για το οποίο κατασκευάστηκε ο κόμβος. Το δεύτερο πεδίο, label, περιλαμβάνει τις έννοιες στις οποίες ανήκει το άτομο έτσι όπως προκύπτει από το πεδίο assertionmap της ABox.java. Το πεδίο level χαρακτηρίζει το επίπεδο του δάσους στο οποίο ανήκει ο κόμβος. Τέλος, η λίστα childlist είναι μία κενή λίστα στην οποία πρόκειται να συμπεριληφθούν οι κόμβοι που παράγονται από τον συγκεκριμένο, κατά την ανάπτυξη του δένδρου.

1.2.17 5.2.16 Κλάση Child.java

Τα αντικείμενα της κλάσης αυτής αναπαριστούν τους νέους κόμβους που προκύπτουν από τους ριζικούς κατά την ανάπτυξη του δάσους πληρότητας. Χαρακτηρίζονται από τον κόμβο πατέρα, node και την ακμή που τους συνδέει, edge.

1.2.18 5.2.17 Κλάση Edge.java

Τα αντικείμενα της κλάσης αντιπροσωπεύουν τις ακμές που συνδέουν τους κόμβους δύο ατόμων τα οποία συμμετέχουν σε μία σχέση. Παράμετροι στον κατασκευαστή της κλάσης είναι τα δύο άτομα στη μορφή ATerm όρου, καθώς και η σχέση στην ίδια μορφή.

1.2.19 5.2.18 Κλάση TreeConstructor.java

Οι μέθοδοι της κλάσης TreeConstructor.java, αναπτύσσουν το δάσος πληρότητας υλοποιώντας τους κανόνες tableau.

Αρχικά, η μέθοδος extractnode εξάγει τα στοιχεία της ουράς rootnodequeue η οποία περνά ως παράμετρος. Για κάθε ένα από τα στοιχεία της ουράς καλεί την forestcheck. Η μέθοδος αυτή, εξετάζει τη σχέση στην οποία συμμετέχει το άτομο του στοιχείου της ουράς. Ελέγχει εάν το πεδίο ορισμού της σχέσης καθώς και το σύνολο τιμών της, είναι συμβατά με τα άτομα τα οποία συμμετέχουν σε αυτή. Εάν όχι, το σύστημα ανιχνεύει ασυνέπεια. Σε διαφορετική περίπτωση, γίνεται η επιστροφή στην extractnode, οπότε και ξεκινά η ανάπτυξη του δένδρου από τον εκάστοτε ριζικό κόμβο. Η εφαρμογή των κανόνων και η ανάπτυξη του δένδρου γίνεται από τη μέθοδο ruleapplication. Πριν όμως, την κλήση της ruleapplication, εξάγεται το πεδίο label του κόμβου, ο οποίος αντιστοιχείζεται στο τρέχον στοιχείο της ουράς NodeQueue. Τα δύο πεδία newlabel και label της κλάσης TreeNodeConstructor.java, με τύπο ATermList, αρχικοποιούνται με την ετικέτα, label, του εν λόγω κόμβου.

Η ruleapplication καλεί με τη σειρά της, μία σειρά από μεθόδους οι οποίες εφαρμόζουν τους κανόνες tableau. Αρχικά, καλείται η intersectionrule, η οποία εφαρμόζει τον κανόνα της ένωσης σε όλα τα στοιχεία κατά μήκος της ετικέτας του κόμβου. Σε αυτό το σημείο, όπως και μετά από κάθε εφαρμογή κανόνα, γίνεται κλήση της μεθόδου clashchecking η οποία ανιχνεύει εαν υπάρχουν αντιφάσεις στην ετικέτα του συγκεκριμένου κόμβου. Εν συνεχεία, καλεί την unionrule για την εφαρμογή του κανόνα της ένωσης. Ακολουθεί νέος έλεγχος συνέπειας. Στο σημείο αυτό, πρέπει να αναφέρουμε ότι στην περίπτωση που η μη ντετερμινιστική επιλογή που πραγματοιήθηκε κατά την εκτέλεση του κανόνα της ένωσης, οδηγήσει σε αντίφαση, υπάρχει η δυνατότητα οπισθοδρόμησης στην ετικέτα εκείνη που αποτελεί την εναλλακτική επιλογή. Η διαδικασία της άμεσης οπισθοδρόμησης εξασφαλίζεται με κλήση της μεθόδου backtracking. Έπειτα, καλείται η μέθοδος preunfolding, η οποία αντικαθιστά τις έννοιες που εμφανίζονται στην ετικέτα με τις ισοδύναμές τους, ή με τις έννοιες στο δεξιό σκέλος των αξιωμάτων υπαγωγής. Για τη διαδικασία του ξεδιπλώματος των εννοιών, είναι προφανές ότι η preunfolding ανατρέχει στο σώμα ορολογίας. Σημειώνουμε επίσης, ότι στην περίπτωση που εμφανίζεται το συμπλήρωμα μίας έννοιας, τότε και αυτό αντικαθίσταται από το συμπλήρωμα των ισοδύναμων εννοιών. Στη συνέχεια, η ruleapplication καλει για ακόμη μία φορά τον έλεγχο συνέπειας και ξεκινά εκ νέου την εφαρμογή των ανωτέρω κανόνων με τη σειρά που περιγράφηκε. Εάν κανένας κανόνας δεν εφαρμόζεται, γίνεται κλήση της μεθόδου existentialforallrule προκειμένου να εκτελεσθούν οι κανόνες του καθολικού και υπαρξιακού τελεστή. Στην πραγματικότητα, οι δύο κανόνες εκτελούνται παράλληλα, αφού κάθε φορά που εντοπίζεται ο υπαρξιακός τελεστής για τον οποίο πρόκειται να κατασκευαστεί ένας νέος κόμβος παιδί, ελέγχεται εάν υπάρχει καθολικός τελεστής που να αφορά την ίδια σχέση. Εάν υπάρχει, τότε, στην ετικέτα του νέου κόμβου προστίθεται και η έννοια που βρίσκεται στην έκφραση του καθολικού τελεστή.

Σε αυτό το σημείο, έχουν εφαρμοστεί οι κανόνες tableau που αφορούν κάποιο ριζικό κόμβο. Απομένει να εφαρμοσθούν οι κανόνες στους κόμβους παιδία αυτού. Για το λόγο αυτό καλείται η epxandtree μέθοδος, η οποία αναλαμβάνει την ανάπτυξη του δένδρου που ξεκινά από τον τρέχοντα ριζικό κόμβο. Σημειώνουμε ότι τα παιδιά ενός κόμβου ανήκουν στο πεδίο του childlist, το οποίο συμπληρώνεται κατά την εκτέλεση του κανόνα του υπαρξιακού τελεστή. Η expandtree καλεί τον εαυτό της προκειμένου να εξελιχθεί το κατά βάθος ξεδίπλωμα του δένδρου. Αφού ολοκληρωθεί η εκτέλεση των κανόνων που αφορούν το δένδρο πληρότητας του ριζικού κόμβου, η ίδια διαδικασία επαναλαμβάνεται από την extractnode εώς ότου εξαχθούν όλοι οι κόμβοι της ουράς rootnodequeue. Σε κάθε επανάληψη γίνεται έλεγχος ασυνέπειας στις ετικέττες του δένδρου, καθώς και έλεγχος ασυνέπειας των ριζικών κόμβων με τα πεδία ορισμού και τιμών της σχέσης που τους συνδέει. Στην περίπτωση που ανιχνευθεί σύγκρουση, η λειτουργία του συστήματος τερματίζεται με μήνυμα ασυνέπειας. Διαφορετικά, η διαδικασία ολοκληρώνεται και ακολουθεί αντίστοιχο μήνυμα για την συνέπεια της οντολογίας.

6 *Υλοποίηση*

Στο κεφάλαιο αυτό παρουσιάζουμε αναλυτικά ζητήματα που αφορούν την υλοποίηση του συστήματος.

6.1 Λεπτομέρειες υλοποίησης

Η κλάση TreeConstructor.java αποτελεί τον πυρήνα της μηχανής συλλογιστικής και αυτό γιατί υλοποιεί τον αλγόριθμο tableau. Παρακάτω δίνουμε μία περιγραφή αυτής.

6.1.1 Ανάπτυξη δάσους πληρότητας

Στο ξεκίνημα της εφαρμογής του αλγορίθμου καλείται η μέθοδος ***extractnode*** με παράμετρο την ουρά από τους ριζικούς κόμβους του completion forest. Οι κόμβοι εξάγονται διαδοχικά, μέσω ενός βρόχου επανάληψης και για κάθε έναν από αυτούς αναπτύσσεται το αντίστοιχο δένδρο πληρότητας, completion tree. Πριν όμως αναπτυχθεί το δένδρο, γίνεται έλεγχος εάν ο εν λόγω κόμβος σχετίζεται με κάποιον άλλο. Στην περίπτωση που ισχύει κάτι τέτοιο, η μέθοδος forestcheck ελέγχει εάν οι ετικέτες των κόμβων, δηλαδή οι έννοιες στις οποίες ανήκει το άτομο που παριστά ο κόμβος, είναι συμβατές με το πεδίο ορισμού και το σύνολο τιμών της σχέσης μέσω της οποίας συνδέονται.

Η ***forestcheck*** αντλεί την edgelist του ριζικού κόμβου. Για κάθε μία από τις ακμές, edge, μέσω των οποίων συνδέεται με άλλο ριζικό κόμβο, κάνει την εξής έλεγχο: διατρέχει το tbox και εντοπίζει το πεδίο ορισμού, domain, της σχέσης καθώς και το σύνολο τιμών, range. Εν συνεχεία, από το αντικείμενο της κλάσης Edge, αντλεί το πεδίο που αφορά τον έτερο ριζικό κόμβο, targetnode. Για τον targetnode, αναζητά στο ABox τη λίστα με τις έννοιες των οποίων αποτελεί υπόσταση. Σε αυτό το σημείο, έχουν γίνει γνωστές οι ετικέτες των ριζικών κόμβων που συνδέονται μεταξύ τους. Συγκρίνονται οι έννοιες της ετικέτας του αρχικού ριζικού κόμβου με αυτές που ανήκουν στο πεδίο ορισμού της σχέσης και έπειτα η ετικέτα του

κόμβου στόχου, με το σύνολο τιμών της σχέσης. Εάν εντοπιστεί κάποια αντίφαση η μεταβλητή consistent λαμβάνει την τιμή false. Προφανώς, σε αυτή την περίπτωση είναι άσκοπο να αναπτυχθεί το δένδρο πληρότητας και έτσι το σύστημα τερματίζει με μήνυμα ασυνέπειας.

6.1.2 Ανίχνευση ασυνέπειας

Στο σημείο αυτό, πριν περιγράψουμε την ανάπτυξη του δένδρου πληρότητας που ξεκινά από το ριζικό κόμβο, είναι σημαντικό να περιγράψουμε τον τρόπο με τον οποίο ελέγχονται οι συγκρούσεις των εννοιών (clashes). Η μέθοδος που είναι υπεύθυνη για την ανίχνευση συγκρούσεων είναι η ***clashcheking***. Όταν καλείται χωρίς παραμέτρους, έχει ως σκοπό να εξετάσει την ετικέτα του τρέχοντος κόμβου η οποία αναπαρίσταται από τη μεταβλητή label της κλάσης. Η label έχει τύπο ATermList, προκειται δηλαδή για έναν Aterm όρο. Οι έννοιες όπως είναι ήδη γνωστό εκφράζονται και αυτές με τύπο ATermAppl. Για κάθε έννοια της label που εξάγεται, εξάγεται από το tbox η αντίστοιχη λίστα ξένων εννοιών. Έπειτα, ελέγχεται εάν κάποια από τις ξένες έννοιες ανήκει στη label. Στην περίπτωση που συμβαίνει κάτι τέτοιο, ανιχνεύεται clash, αφού το άτομο που αναπαριστά ο τρέχον κόμβος ανήκει ταυτόχρονα σε δύο συμπληρωματικές έννοιες.

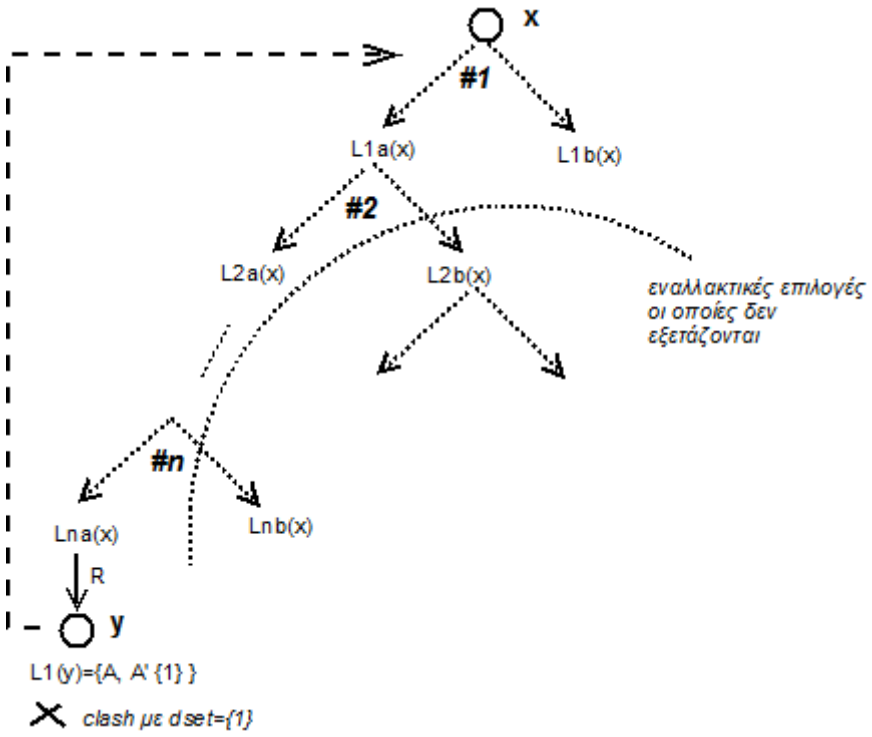
Η clashchecking είναι δυνατό να κληθεί με δύο παραμέτρους. Οι παράμετροι είναι τύπου ATermList και αντιπροσωπεύουν ετικέτες, για τις οποίες θέλουμε να ελέγξουμε εάν περιλαμβάνουν αντιφατικές έννοιες. Η κλήση της μεθόδου με τέτοιες παραμέτρους, γίνεται από την forestckeck που περιγράφηκε πρωτύτερα, για τον έλεγχο συμβατότητας μιας ετικέτας, με τύπο ATermList, με το πεδίο ορισμού μίας σχέσης, με τύπο επίσης ATermList. Ο τρόπος που γίνεται ο έλεγχος είναι παρόμοιος με τα προηγούμενα με τη διαφορά ότι κάθε φορά που εξάγεται μία έννοια από τη label και αντλείται από το tbox η αντίστοιχη λίστα ξένων εννοιών, γίνεται έλεγχος εάν κάποια από αυτές ανήκει όχι στην ίδια την ετικέτα, αλλά σε αυτή που αντιπροσωπεύει η δεύτερη παράμετρος.

Στην περίπτωση που ανιχνευθεί κάποια αντίφαση, το σύστημα δεν τερματίζει απ' ευθείας με μήνυμα ασυνέπειας. Εξετάζει προηγουμένως, όλες τις εναλλακτικές επιλογές που ενδέχεται να εμφανιστούν κατά την ανάπτυξη του δένδρου πληρότητας. Οι εναλλακτικές επιλογές προκύπτουν από την εφαρμογή του μη ντετερμινιστικού κανόνα της ένωσης. Κάθε φορά που εκτελείται ο κανόνας, εισάγεται στην ετικέτα η πρώτη από τις έννοιες που εμφανίζεται στην αλυσίδα της ένωσης, ενώ δημιουργείται μία επιπρόσθετη, εναλλακτική ετικέτα με την ένωση των υπόλοιπων εννοιών της αλυσίδας. Η ετικέτα που δεν επιλέγεται φυλάσσεται στον πίνακα branchingpoint, όπου αντιστοιχίζεται με τον αύξοντα αριθμό του κανόνα της ένωσης. Οι έννοιες που εξαρτώνται από τη δεδομένη επιλογή, μαρκάρονται με τον εκάστοτε αύξοντα αριθμό.

Επιστρέφοντας, στην περιγραφή της μεθόδου clashchecking, μπορούμε τώρα, να εξηγήσουμε πως κάθε φορά που ανιχνεύεται κάποια αντίφαση μεταξύ δύο εννοιών, εξάγεται από αυτές η πληροφορία που αφορά τον αύξοντα αριθμό του κανόνα της ένωσης από τον οποίο εξαρτώνται. Η πληροφορία που περιλαμβάνει τους αύξοντες αριθμούς έχει τη μορφή anntotation του ATermAppl. Οι annotations των αντιφατικών εννοιών αποθηκεύονται σε μία λίστα ATermList με το όνομα dset. Η μεταβλητή αυτή, αναπαριστά το σύνολο εξάρτησης (dependency set), ή με άλλα λόγια πληροφορεί για το σύνολο των εναλλακτικών ετικετών από τις οποίες μπορεί να εξαρτάται η συγκεκριμένη αντίφαση. Με αυτό τον τρόπο ο αλγόριθμος οπισθοδρομεί εξετάζοντας τις συγκεκριμένες ετικέτες που καταδεικνύει το dset, χωρίς να καθυστερεί με τον έλεγχο των υπόλοιπων μη καθοριστικών επιλογών.

Η οπισθοδρόμηση πραγματοποιείται από τη μέθοδο ***backtracking***. Η πρώτη λειτουργία που εκτελείται είναι η ταξινόμηση του συνόλου dset. Με αυτό τον τρόπο η οπισθοδρόμηση ξεκινά από την εναλλακτική ετικέτα που εμφανίστηκε τελευταία και είναι μαρκαρισμένη με τον μεγαλύτερο αύξοντα αριθμό.

Παρουσιάζουμε σχηματικά μία περίπτωση οπισθοδρόμησης:



Στο παράδειγμά μας θεωρούμε την ετικέτα του κόμβου x: $\mathcal{L}(x) = \{(\forall R.\neg A \sqcup \forall R.\neg B), C_1 \sqcup D_1, C_2 \sqcup D_2, \ldots, C_n \sqcup D_n, \exists R.A\}$. Η αντίφαση προκύπτει όταν εφαρμόζεται ο κανόνας του υπαρξιακού τελεστή στην έκφραση $\exists R.A$, ενώ προηγουμένως, κατά την πρώτη

εφαρμογή του κανόνα της ένωσης στην έκφραση $\forall R.\neg A \sqcup \forall R.\neg B$, έχει επιλεγεί η $\forall R.\neg A$ έναντι της εναλλακτικής $\forall R.\neg B$. Η οπισθοδρόμηση έχει νόημα να γίνει απ΄ευθείας στην ετικέτα όπου έγινε η μη ντετερμινιστική επιλογή που μόλις περιγράψαμε. Ο έλεγχος των μη ντετερμινιστικών επιλογών που προκύπτουν κατά την εφαρμογή του κανόνα της ένωσης στις υπόλοιπες εκφράσεις, $C_n \sqcup D_n$, δεν επιφέρει κανένα αποτέλεσμα, μόνο καθυστερεί τον αλγόριθμο.

Επιστρέφοντας στην περιγραφή του αλγορίθμου, αναφέρουμε ότι η μεταβλητή label αποκτά νέα τιμή που εκφράζει την νέα ετικέτα στην οποία γίνεται αρχικά η οπισθοδρόμηση. Απομένει να εφαρμοσθούν οι κανόνες tableau στις έννοιες της νέας label, έτσι ώστε να ανιχνευθεί εάν υπάρχει αντίφαση ή όχι. Στην περίπτωση που ανιχνευθεί εκ νέου αντίφαση (clash) γίνεται και πάλι οπισθοδρόμηση εάν αυτό είναι δυνατό, δηλαδή εάν το σύνολο dset δεν είναι κενό.

Η μέθοδος backtracking καλείται αναδρομικά εώς ότου ολοκληρωθεί η εφαρμογή των κανόνων tableau στις έννοιες της ετικέτας χωρίς να εντοπιστεί αντίφαση. Το σύνολο dset που αφορά την εκάστοτε διαδικασία οπισθοδρόμησης αποθηκεύεται σε κάθε κλήση της μεθόδου σε μία νέα βοηθητική λίστα. Με αυτό τον τρόπο η πληροφορία δεν χάνεται κάθε φορά που η διαδικασία της οπισθοδρόμησης διακόπτεται από μία νέα. Στην περίπτωση που όλες η επιλογές οδηγήσουν σε αντίφαση, η backtracking επιστρέφει τιμή false.

6.1.3 Ανάπτυξη δένδρου πληρότητας

Στο σημείο αυτό, έχουμε ήδη περιγράψει τον τρόπο με τον οποίο γίνεται έλεγχος για την ύπαρξη αντίφασης. Απομένει να περιγράψουμε τη μέθοδο με την οποία αναπτύσσεται το δένδρο που ξεκινά από έναν ριζικό κόμβο. Η μέθοδος ***expandtree*** εξάγει τα παιδιά του εκάστοτε ριζικού κόμβου και για κάθε ένα από αυτά ξεκινά να αναπτύσσει το δένδρο κατά βάθος. Η διαδικασία έχει ως εξής: εξάγεται το πεδίο childlist του τρέχοντα κόμβου που εξετάζεται. Από τη λίστα childlist εξάγονται όλα τα παιδιά του κόμβου μέσω βρόχου επανάληψης. Κάθε φορά ο κόμβος- παιδί που εξετάζεται, γίνεται και τρέχον κόμβος με την ενημέρωση της μεταβλητής currentnode. Εν συνεχεία, η μεταβλητή label, που εκφράζει την ετικέτα του κόμβου που εξετάζει το σύστημα τη δεδομένη στιγμή, παίρνει την τιμή του αντίστοιχου πεδίου του currentnode. Καλείται η μέθοδος ruleapplication για την εφαρμογή των κανόνων. Εάν η μέθοδος επιστρέψει τιμή true, σημαίνει ότι οι κανόνες εφαρμόζονται στις έννοιες της label χωρίς να ανιχνευθεί αντίφαση. Η expandtree καλεί εκ νέου τον εαυτό της με σκοπό να αναπτύξει το δένδρο που ξεκινά από το νέο κόμβο-παιδί. Η ίδια διαδικασία επαναλαμβάνεται και το δένδρο αναπτύσσεται κατά βάθος, έως ότου δημιουργηθεί κόμβος ο οποίος έχει κενό το πεδίο childlist (κάτι τέτοιο συμβαίνει όταν η ετικέτα του κόμβου δεν περιμβάνει καθόλου εκφράσεις με υπαρξιακό τελεστή).

Προφανώς, κατά την εκτέλεση των κανόνων ενδέχεται να προκύψει clash. Στην περίπτωση που η μεταβλητή dset είναι κενή και δεν υπάρχει δυνατότητα οπισθοδρόμησης, το σύστημα τερματίζει με μήνυμα ασυνέπειας. Εάν το σύνολο dset δεν είναι κενό τότε καλείται η backtracking. Εάν η οπισθοδρόμηση αποτύχει τότε και πάλι το σύστημα τερματίζει. Εάν είναι επιτυχής, διακρίνουμε δύο περιπτώσεις: στην πρώτη, η ετικέτα στην οποία γίνεται η οπισθοδρόμηση αποτελεί μία εναλλακτική ετικέτα του τρέχοντος κόμβου. Η expandtree καλείται εκ νέου για το ξεδίπλωμα του δένδρου που ξεκινά από τους νέους κόμβους- παιδιά, εάν αυτά υπάχρουν. Στη δεύτερη περίπτωση, η μεταπήδηση γίνεται σε ετικέτα η οποία αντιστοιχεί σε κόμβο υψηλότερου επιπέδου, δηλαδή σε κόμβο που δημιουργήθηκε προγενέστερα του τρέχοντος. Η περίπτωση αυτή παρουσιάζει κάποια ιδιαιτερότητα, καθώς η μέθοδος expandtree πρέπει να κάνει τη μεταπήδηση στο νέο κόμβο και να αγνοήσει τους κόμβους- αδέλφια του τρέχοντος. Επιλέγεται δηλαδή μία ετικέτα, η οποία δεν καταλήγει στη δημιουργία τις τρέχουσας childlist που εξετάζεται, και άρα αυτή θα πρέπει να αγνοηθεί. Για το λόγο αυτό, η expandtree με μία μεταβλητή σημαία, εξέρχεται από το βρόχο επανάληψης που αφορά το δεδομένο τρέχοντα βρόχο και τα αδέλφια του. Στη συνέχεια, καλείται εκ νέου για τον νέο τρέχοντα κόμβο.

Σημειώνουμε ότι η μέθοδος backtracking ενημερώνει τη μεταβλητή jumpedtolevel με το επίπεδο του δένδρου στο οποίο γίνεται η μεταπήδηση.

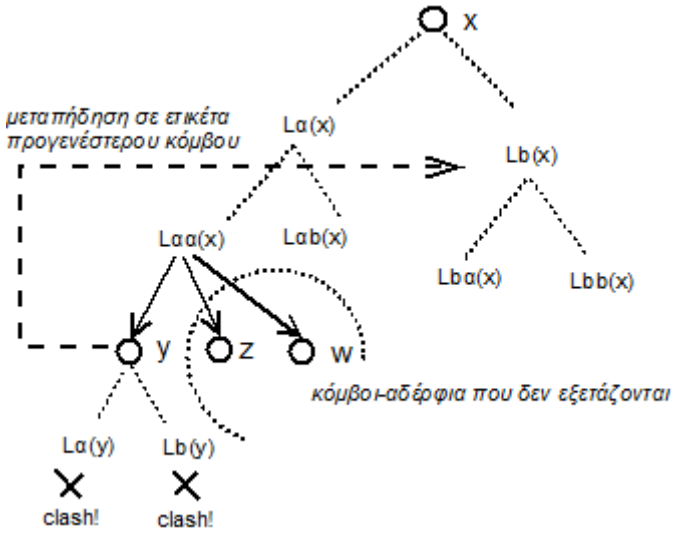
Παραθέτουμε τον κώδικα της expandtree και μια σχηματική αναπαράσταση περίπτωσης λειτουργίας της, για να γίνει περισσότερο κατανοητή.

```
private boolean expandtree(){
List<Child> chlist=currentnode.childlist;//i lista me ta paidia tou trexontos komvou pou prepei na exetastoun
boolean succeeded=true;
if (chlist.isEmpty()) {succeeded=true;return true;}
int nodestochecknum=chlist.size();
jumpedtolevel=currentnode.level;
for(int i=0;i<nodestochecknum;i++){
        currentnode=chlist.get(i).node;
        label=currentnode.label;
        newlabel=newlabel.getEmpty();
        newlabel=newlabel.concat(label);
        boolean applied=ruleapplication();
        if (!applied){//vrethike clash
```

```
                    if (dset.isEmpty()){//den einai dynato na sumvei backtracking

                        consistent=false;
                        return false;
                    }
                    else {//periptwsi pou exei entopistei clash kai to dset den einai empty opote kai ginetai backtracking
                        boolean successful=backtracking();
                        if (successful){
                        if (jumpedtolevel<currentnode.level){succeeded=false;}
                            else {succeeded=true;}
                        }
                        else {consistent=false; return false;}
                    }
                }
                if (!succeeded) break;//den exetazontai oi komvoi aderfia
                succeeded=expandtree();//by depth xediplwma dendrou

            }//egine elegxos gia ton currennode k ola tou ta paidia-sunexeia me epomeno root node*/
            if (!succeeded){
                if (!consistent)return false;
                else if (jumpedtolevel<currentnode.level){
                    while (jumpedtolevel<currentnode.parent.level){//anixneuetai o komvos o opoios anikei sto epipedo pou tha ginei h metapidisi
                        currentnode=currentnode.parent;}
                     expandtree();
                    }
            }
            return true;
        }
```



6.1.4 Εφαρμογή κανόνων tableau

Οι κανόνες tableau υλοποιούνται με μία σειρά από μεθόδους, τις intersectionrule, unionrule, existentialforallrule. Η σειρά εκτέλεσης των κανόνων ρυθμίζεται από την ruleapplication μέθοδο η οποία καλεί τις παραπάνω συναρτήσεις. Είναι πολύ σημαντικό να αναφέρουμε ότι η μηχανή συλλογιστικής που αναπτύχθηκε εφαρμόζει την τεχνική του οκνηρού ξεδιπλώματος (lazy unfolding), έτσι όπως αυτό περιγράφηκε στο 3^{ο} κεφάλαιο για την περίπτωση απουσίας γενικευμένων αξιωμάτων υπαγωγής.

Όπως είπαμε και στο 3^{ο} κεφάλαιο, το ξεδίπλωμα κατά απαίτηση υλοποιείται με την εφαρμογή δύο κανόνων. Κάθε έννοια που εμφανίζεται στην ετικέτα ενός κόμβου, αντικαθίσταται από το δεξιό σκέλος του αξιώματος υπαγωγής ή ισοδυναμίας στο οποίο συμμετέχει. Εάν εμφανίζεται το συμπλήρωμα της έννοιας, αντικαθίσταται από το συμπλήρωμα του δεξιού σκέλους του αξιώματος στο οποίο συμμετέχει. Η μέθοδος που υλοποιεί τους κανόνες είναι η ***preunfolding***. Με ένα βρόχο επανάληψης διατρέχει την ετικέτα label, και καλεί την μέθοδο ***unfoldingrules*** για κάθε μία από τις έννοιες, ATermAppl, της label. Αρχικά, η unfoldingrules εξετάζει εάν η έννοια ανήκει στον πίνακα termhash του TBox οπότε και θα συμμετέχει σε κάποιο αξίωμα ισοδυναμίας ή υπαγωγής. Ανιχνεύεται το αξίωμα και εξάγεται το δεξιό σκέλος αυτού, που συνιστά έναν ATermAppl όρο. Ο όρος μαρκάρεται με τις annotations της έννοιας που πρόκειται να αντικαταστήσει. Τέλος, αποθηκεύεται στη label αφού προηγουμένως εξασφαλιστεί ότι δεν ανήκει ήδη σε αυτή, ώστε να αποφευχθούν οι διπλές εμφανίσεις εννοιών. Η διαφορά ανάμεσα στους δύο τύπους αξιωμάτων υπαγωγής και ισοδυναμίας, εντοπίζεται στις περισσότερες από μία έννοιες που ίσως ανήκουν στο δεξιό σκέλος του αξιώματος της ισοδυναμίας. Έτσι, στην περίπτωση της ισοδυναμίας, η έννοια αντικαθίσταται με το σύνολο των ισοδύναμων εννοιών οι οποίες είναι συγκεντρωμένες σε μία

λίστα ATermList. Οι έννοιες εξάγονται από την ATermList, μαρκάρονται μία προς μία με τις annotations, για να εισαχθούν τέλος, στη label. Στην περίπτωση της υπαγωγής το δεξιό σκέλος αποτελείται από μία έννοια ή καλύτερα έκφραση, και όχι λίστα εννοιών.

Υπάρχει όμως, το ενδεχόμενο, ο όρος ο οποίος καλείται να αντικατασταθεί, να μην ανιχνεύεται στο termhash του TBox. Σε αυτή την περίπτωση, η unfoldingrules εξετάζει εάν ο όρος αποτελεί συμπλήρωμα έννοιας, οπότε ακολουθείται η ίδια διαδικασία που περιγράφηκε παραπάνω. Γίνεται η αναζήτηση των αξιωμάτων στα οποία συμμετέχει η έννοια απαλλαγμένη όμως, από τον τελεστή της άρνησης. Στις εκφράσεις που πρόκειται να την αντικαταστήσουν εφαρμόζεται ο τελεστής της άρνησης πρωτού αυτές προστεθούν στη label.

Ένα άλλο ενδεχομενο για το οποίο ο όρος δεν ανιχνεύεται στο termhash του tbox είναι η περίπτωση στην οποία εκφράζει μία πρωταρχική-ατομική έννοια η οποία δεν περιγράφεται στο σώμα της ορολογίας. Προφανώς, στην περίπτωση αυτή δεν εφαρμόζονται οι κανόνες unfolding. Τέλος, εάν ο όρος που δεν ανιχνεύεται στο σώμα ορολογίας, δεν είναι ατομική έννοια, αλλά ούτε και συμπλήρωμα έννοιας, εξετάζεται εάν είναι δυνατό να μετατραπεί σε κανονική μορφή άρνησης με κλήση της μεθόδου nnf. Με αυτό τον τρόπο εξασφαλίζουμε για παράδειγμα, το ξεδίπλωμα, lazy unfolding, μίας έκφρασης όπου ο τελεστής της άρνησης εφαρμόζεται σε τομή εννοιών.

Σημειώνουμε ότι, γενικά, οι επιτρεπτές μορφές ενός όρου της label είναι η τομή εννοιών, η ένωση, η έκφραση υπαρξιακού και καθολικού τελεστή, και το συμπλήρωμα έννοιας. Στο σημείο που εφαρμόζεται ο κανόνας unfoldingrules έχει ήδη εξαντληθεί η εφαρμογή των κανόνων της ένωσης και της τομής, ενώ ο κανόνας υπαρξιακού τελεστή εφαρμόζεται τελευταίος.

Η ***intersectionrule*** εξάγει έναν προς έναν τους ATermAppl όρους της label και εξετάζει εάν αποτελούν τομή εννοιών. Η περίπτωση αυτή ανιχνεύεται γιατί η σύναρτηση AFun που εφαρμόζεται στον ATermAppl όρο είναι η *INTERSECTIONFUN*. Η *INTERSECTIONFUN* εφαρμόζεται σε μία ATermList, τη λίστα των εννοιών της τομής. Οι έννοιες εξάγονται από τη λίστα, μαρκάρονται με τις annotations του αρχικού ATermAppl όρου, και εισάγονται στη label. Εάν τελικά ανιχνευθεί τομή και εφαρμοστεί ο κανόνας η intersectionrule επιστρέφει στη ruleapplication την κάλεσε την τιμή true.

Η ***unionrule*** ανιχνεύει τους όρους ATermAppl της label που συνιστούν ένωση εννοιών από την AFun που έχει τιμή το πεδίο *UNIONFUN* της ATermsD.java. Όπως και η intersectionrule εφαρμόζει τον κανόνα της ένωσης κατά μήκος της label, για κάθε έναν από τους όρους της. Ο τρόπος με τον οποίο πραγματοποιείται ο μη ντερμινιστικός κανόνας, περιγράφηκε αναλυτικά πρωτύτερα, για αυτό δεν θα αναφέρουμε περαιτέρω λεπτομέρειες.

Η ruleapplication καλεί την intersectionrule και εάν αυτή επιστρέψει τιμή true σημαίνει ότι ανιχνεύθηκε τομή εννοιών και εφαρμόστηκε ο κανόνας. Η ετικέτα έχει λοιπόν τροποποιηθεί και είναι σκόπιμο να εξεταστεί εάν παρουσιάζεται κάποια αντίφαση. Καλείται επομένως, η clashchecking. Εν συνεχεία, καλείται η unionrule και γίνεται νέος έλεγχος για clash εάν η unionrule επιστρέψει true. Εκτελούνται οι κανόνες του οκνηρού ξεδιπλώματος, (lazy unfolding) και ακολουθείται εκ νέου η ίδια σειρά εκτέλεσης κανόνων μέχρις ότου όλες οι μέθοδοι επιστρέψουν τιμή false, και άρα έχει εξαντληθεί η εφαρμογή τους. Στο σημείο αυτό καλείται η existentialforallrule.

Η μέθοδος ***existentialforallrule*** υλοποιεί τους κανόνες υπαρξιακού και καθολικού τελεστή. Η λογική της σύμπτυξης των δύο κανόνων σε μία μέθοδο έγκειται στο ότι ο κανόνας του καθολικού τελεστή δεν εφαρμόζεται, παραμόνο όταν έχει δημιουργηθεί κατάλληλος κόμβος από την εκτέλεση του κανόνα του υπαρξιακού τελεστή.

Η διαδικασία που ακολουθείται από τη μέθοδο έχει ως εξής: αρχικά, η label διατρέχεται για τον εντοπισμό των καθολικών τελεστών. Έπειτα, οι σχέσεις στις οποίες εφαρμόζεται ο τελεστής αποθηκεύονται στον πίνακα foralltable και αντιστοιχίζονται με τις έννοιες συμπληρώματα της σχέσης, property fillers. Εν συνεχεία, η label διατρέχεται για μία δεύτερη φορά με σκοπό τον εντοπισμό των υπαρξιακών τελεστών. Μόλις ανιχνεύεται ο τελεστής εξάγεται η σχέση στην οποία εφαρμόζεται, μεταβλητή prop, και η έννοια συμπλήρωμα της σχέσης, μεταβλητή concept. Αξίζει να σημειώσουμε ότι, εάν η έννοια προστεθεί σε κάποια ετικέτα θα μαρκαριστεί προηγουμένως με τις annotations της έννοιας από την οποία προήλθε. Επανερχόμαστε τώρα στον τρέχοντα κόμβο και εξάγουμε τη λίστα με τα παιδία κόμβους αυτού. Εάν βεβαίως αυτά υπάρχουν, εξετάζουμε μέσω ποιας σχέσης συνδέονται με τον τρέχοντα κόμβο. Στην περίπτωση που η σχέση ταυτίζεται με την prop τότε, εξάγεται και η ετικέτα του κόμβου παιδιού και ελέγχεται εάν περιλαμβάνει την concept. Στην περίπτωση που η concept ήδη είναι καταχωρημένη σε κάποια ετικέτα ο κανόνας δεν εφαρμόζεται. Από την άλλη μεριά, εάν έχουν ελενχθεί όλα τα παιδιά του τρέχοντος κόμβου χωρίς κάποιο από αυτά να συνδέεται μέσω της prop και ταυτόχρονα η ετικέτα του να περιλαβάνει την concept, τότε, εφαρμόζεται ο κανόνας του υπαρξιακού τελεστή. Δημιουργούνται νέα αντικείμενα των κλάσεων Node και Edge, τα οποία αποτελούν τις παραμέτρους του κατασκευαστή της Child. Ο νέος κόμβος που κατασκευάζεται πρόκειται να προστεθεί στην childlist του currentnode. Σημειώνουμε επίσης, ότι η δημιουργείται μία νέα λίστα τύπου ATermList προκειμένου να αναπαραστήσει την ετικέτα του νέου κόμβου. Η ετικέτα περιλαβάνει αρχικά την έννοια concept. Πρωτού όμως, σταλεί ως παράμετρος στον κατασκευαστή της Node με σκοπό να συμπληρώσει το πεδίο label του νέου κόμβου, συνενώνεται με τη λίστα των εννοιών η οποία αντιστοιχεί στη σχέση prop σύμφωνα με τον

πίνακα foralltable. Με τον τρόπο αυτό εφαρμόζεται και ο κανόνας που αφορά τον καθολικό τελεστή.

Κατά την επιστροφή στην ruleapplication και στην περίπτωση που η μέθοδος existentialforallrule επιστρέψει τιμή true, καλείται εκ νέου η clashchecking. Στο σημείο αυτό έχουν εφαρμοσθεί έως τέλους οι κανόνες tableau στις έννοιες της label. Εάν δεν έχει ανιχνευθεί κάποιο clash, ή έχει αντιμετωπιστεί με οπισθοδρόμηση, η ruleapplication επιστρέφει τιμή true. Το σύστημα συνεχίζει τη λειτουργία του είτε με νέα κλήση της ruleapplication που αφορά την ετικέτα των κόμβων παιδιών, εάν υπάρχουν, είτε με την εξαγωγή νέου ριζικού κόμβου και την κλήση της ruleapplication για την ετικέτα αυτού.

6.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Προαπαιτούμενα για την εγκατάσταση της μηχανής συλλογιστικής σε υπολογιστή είναι οποιαδήποτε πλατφόρμα ανάπτυξης λογισμικού, όπως eclipse και netbeans, η java 1.6 καθώς και οι πρόσθετες βιβλιοθήκες owlapi-bin.jar, owlapi-src.jar, για το OWL-API, και η aterm-java-1.6.jar.

Ο έλεγχος του συστήματος πραγματοποιήθηκε στο περιβάλλον eclipse και σε υπολογιστή μνήμης 4GB, με επεξεργαστή Intel Core Duo, συχνότητας 2.4 GHz.

Σημειώνουμε ότι, οι οντολογίες που χρησιμοποιήθηκαν στην αξιολόγηση του συστήματος, δομούνται είτε σε αρχεία .owl είτε σε αρχεία .rdf. Σε κάθε περίπτωση, είσοδος της μηχανής είναι το URI της οντολογίας. Ενδέχεται όμως, το αρχείο εισόδου να μην βρίσκεται στο σκληρό δίσκο του υπολογιστή. Σε αυτή την περίπτωση ο χρήστης επιλέγει κάποια οντολογία του διαδικτύου και άρα θα πρέπει να υπάρχει πρόσβαση σε αυτό.

7 Έλεγχος

Στο παρόν κεφάλαιο παραθέτουμε τον τρόπο αξιολόγησης της μηχανής συλλογιστικής που κατασκευάστηκε.

7.1 Μεθοδολογία ελέγχου

Ο έλεγχος του συστήματος πραγματοποιήθηκε με τη χρήση διαφορετικών αρχείων οντολογίας στην είσοδο του συστήματος. Ο χρήστης πληκτρολογεί το URI της οντολογίας που επιθυμεί και εν συνεχεία, η μηχανή συλλογιστικής αποφαίνεται για τη συνέπειά της. Εξετάσαμε την απόκριση της μηχανής συλλογιστικής σε οντολογίες διαφορετικού περιεχομένου και έκτασης. Στη συνέχεια συγκρίναμε τα αποτελέσματα με αυτά που προκύπτουν από τις μηχανές συλλογιστικής Pellet και Fact++. Αναφέρουμε στην επόμενη ενότητα τα χαρακτηριστικότερα παραδείγματα οντολογιών εισόδου.

7.2 Αναλυτική παρουσίαση ελέγχου

1.2.20 7.2.1 Απλές οντολογίες εισόδου

Αρχικά, για έναν πρώτο έλεγχο της ορθότητας της μηχανής που αναπτύξαμε, τοποθετήσαμε στην είσοδο αρχεία οντολογιών περιορισμένης έκτασης. Ένα τέτοιου τύπου, απλό παράδειγμα οντολογίας είναι το αρχείο inconsistent1.owl. Στο αρχείο περιγράφεται η οντολογία με το εξής περιεχόμενο: $\mathcal{T} = \{A, B, C \equiv A \sqcap B, D \sqsubseteq \forall R.C, E \sqsubseteq \exists R.\neg A\}$, $\mathcal{A} = \{x\colon A, x\colon D, x\colon E\}$. Προφανώς, από την εκτέλεση του αλγορίθμου tableau προκύπτει το ασυνεπές σώμα ισχυρισμών $\mathcal{A} = \{x\colon A, x\colon \neg A\}$.

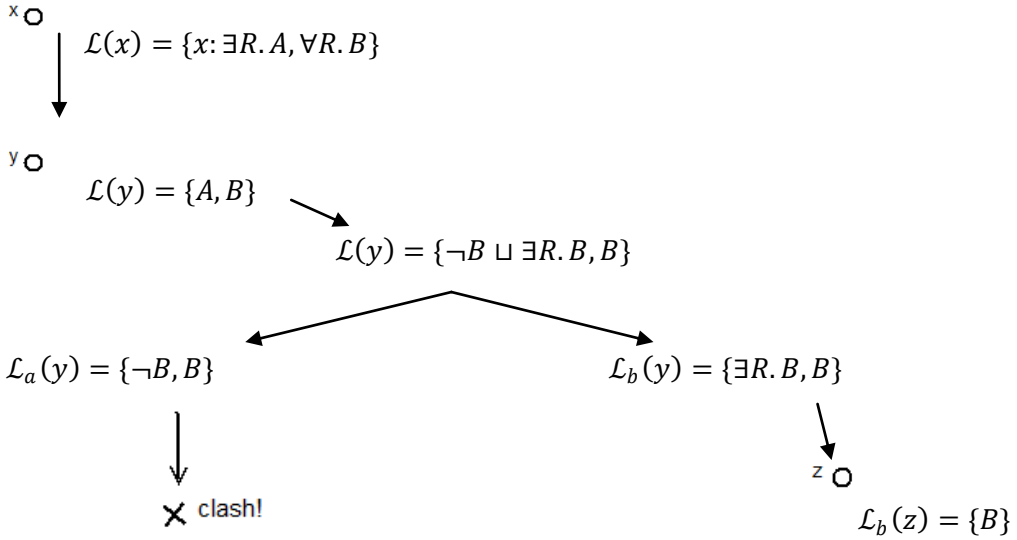
Έτσι λοιπόν, όταν ο χρήστης εισάγει τη διαδρομή στην οποία βρίσκεται το inconsistent1.owl, για παράδειγμα

file:/Users/depi/workspace/testcases/testcases/inconsistent1.owl, και λαμβάνει το μήνυμα inconsistent από το σύστημα.

Το σύστημα αποκρίνεται με κατασκευή της βάσης γνώσης από το αρχείο εισόδου σε 203 milliseconds ενώ ο έλεγχος της συνέπειας της βάσης γνώσης πραγματοποιείται σε 33 milliseconds. Τα αποτελέσματα για τους Pellet και Fact++ είναι 31 και 64 milliseconds αντίστοιχα.

Στη συνέχεια χρησιμοποιούμε την πλατφόρμα Protégé προκειμένου να πραγματοποιήσουμε τον ίδιο έλεγχο στο αρχείο οντολογίας της εισόδου, χρησιμοποιώντας διαφορετικές μηχανές συλλογιστικής. Η απόκριση του Pellet προκύπτει σε χρόνο 15 milliseconds. Ο Fact++ ο οποίος συνδέεται με το Protégé μέσω του DIG αποκρίνεται σε χρόνο 33 milliseconds ενώ χρόνος διασύνδεσής του στο σύστημα είναι 45 milliseconds.

Ένα δεύτερο απλό παράδειγμα απλής οντολογίας είναι το αρχείο consistent3.owl. Η οντολογία που περιγράφει είναι η ακόλουθη: $\mathcal{T} = \{B, A \sqsubseteq \neg B \sqcup \exists R.B\}$, $\mathcal{A} = \{x: \exists R.A, \forall R.B\}$. Η εφαρμογή των κανόνων tableau παρουσιάζεται σχηματικά ως εξής:



Η μηχανή αποκρίνεται με το μήνυμα consistent. Ο χρόνος εκτέλεσης του αλγορίθμου συλλογιστικής για την πραγματοποίηση του ελέγχου ικανοποιησιμότητας είναι 1millisecond. Ο Pellet ανταποκρίνεται στον ίδιο χρόνο και ο Fatc++ σε 18 milliseconds. Η κατάσκευη της βάσης γνώσης από το σύστημά μας γίνεται σε 203 milliseconds, ενώ ο χρόνος για τη σύνδεση του Fact++ με το αρχείο οντολογίας μέσω της διεπαφής DIG διαρκεί μόλις 45 milliseconds.

1.2.21 7.2.2 Σύνθετες οντολογίες εισόδου

Πέραν των απλών οντολογιών που κατασκευάσαμε για μια πρώτη αξιολόγηση της μηχανής συλλογιστικής, χρησιμοποιήσαμε στη συνέχεια, πιο σύνθετες οντολογίες με σαφώς περισσότερες κλάσεις και ισχυρισμούς. Ορισμένες από τις οντολογίες τις οποίες

χρησιμοποιήσαμε απ' ευθείας από το διαδίκτυο δεν διέθεταν σώμα ισχυρισμών. Σε αυτές τις περιπτώσεις το σύστημα ανταποκρίθηκε με αντίστοιχο μήνυμα.

Η οντολογία aeo_v2_10.owl αποτελεί ένα τέτοιο παράδειγμα. Το περιεχόμενό της αφορά στοιχεία της IAAF για το στίβο. Ο χρήστης πληκτρολογεί τη διαδρομή στην οποία βρίσκεται η συγκεκριμένη οντολογία, πληκτρολογήσαμε file:/Users/depi/workspace/testcases/aeo/aeo_v2_10.owl, και η απόκριση του συστήματος είναι: `empty Abox..`

Για το αρχείο file:/Users/depi/workspace/testcases/aeo/aeo_v3_1.owl, το οποίο διαθέτει σώμα ισχυρισμών, η απόκριση εξόδου είναι: `consistent`

Ένας επιπλέον έλεγχος πραγματοποιήθηκε με οντολογίες που εντοπίσαμε στην ιστοσελίδα του πανεπιστημίου του Poznan της Πολωνίας. Οι οντολογίες έχουν οικονομολογικό περιεχόμενο και αφορούν τράπεζες. Οι κλάσεις τους περιγράφουν τις έννοιες των δανείων, των πιστωτικών καρτών, λογαριασμών, ενώ είναι καταχωρημένες και πληροφορίες που αφορούν τους πελάτες της τράπεζας, όπως οι οφειλές τους και η κατάσταση του λογαριασμού τους.

Έτσι εάν ο χρήστης πληκτρολογήσει το URI: http://www.cs.put.poznan.pl/alawrynowicz/goldDLP.owl η μηχανή φορτώνει την εν λόγω οντολογία και απαντά με `consistent` σε χρόνο 421 milliseconds. Ο Pellet δίνει την ίδια απάντηση σε 250 milliseconds και ο Fact++ σε 16 milliseconds.

Ίδιο είναι το αποτέλεσμα με το αρχείο http://www.cs.put.poznan.pl/alawrynowicz/financial.owl, με τη διαφορά ότι η συγκεκριμένη οντολογία έχει μεγαλύτερο σώμα ισχυρισμών με αποτέλεσμα το σύστημα να καθυστερεί ελαφρώς περισσότερο κατά την εξαγωγή του συμπεράσματος συνέπειας. Η απόκριση προκύπτει σε 23.084 sec. Ο Pellet αποκρίνεται σε χρόνο 9.064 και ο Fact++ σε 21.715 sec.

Για τον έλεγχο της αξιοπιστίας του συστήματος εξετάστηκαν και άλλα αρχεία οντολογιών όπως το http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl, το οποίο περιλαμβάνει κλάσεις που περιγράφουν ένα πανεπιστήμιο, τα τμήματά του, τα μαθήματα, τον διευθυντή και τη διαβάθμιση του προσωπικού. Μια σύντομη παρουσίαση της ιεραρχίας των κλάσεων της οντολογίας έχει ως εξής:

Organization

College

Department

Institute

Program

ResearchGroup

University

Person

Chair

Dean

Director

Employee

AdministrativeStaff

Faculty

Lecturer

PostDoc

Professor

ClericalStaff

SystemsStaff

GraduateStudent

Student

ResearchAssistant

UndergraduateStudent

TeachingAssistant

Publication

Article

Book

Manual

Software

Specification

UnofficialPublication

Schedule

Work

Course

Research

Η οντολογία δεν περιλαμβάνει σώμα ισχυρισμών. Το μήνυμα εξόδου του συστήματος είναι `empty Abox..` με χρόνο απόκρισης 250 milliseconds.

Όταν συνδέσουμε τον Pellet 1.5.2 η απόκριση προκύπτει σε χρόνο 312 milliseconds. Για την αντίστοιχη περίπτωση ο Fact++ αποκρίνεται σε 93 milliseconds.

Στη συνέχεια, εκτελούμε επιπλέον ελέγχους στην εν λόγω οντολογία οι οποίοι αφορούν την ικανοποιησιμότητα εννοιών. Σημειώνουμε ότι το σύστημα που κατασκευάσαμε αποφαίνεται για τη συνέπεια ή μη, μίας βάσης γνώσης κατασκευάζοντας ένα δάσος πληρότητας. Προκειμένου να αρχικοποιηθεί το δάσος είναι απαραίτητη η ύπαρξη ατόμων στο σώμα ισχυρισμών. Έτσι λοιπόν, ο έλεγχος που αφορά την ικανοποιησιμότητα έννοιας γίνεται εάν προσθέσουμε στο σώμα των ισχυρισμών ένα άτομο που να ανήκει σε αυτή.

Ο έλεγχος για την ικανοποιησιμότητα της έννοιας: $ResearchAssistant \sqcap \neg Student$ ισοδυναμεί με τον έλεγχο της συνέπειας του ABox $A = A \cup \{x: (ResearchStudent \sqcap \neg Student)\}$. Δημιουργούμε ένα νέο αρχείο file:/Users/depi/workspace/testcases/univ-bench2.owl το οποίο περιλαμβάνει σώμα ισχυρισμών.

Το ερώτημα που θέτουμε στη μηχανή είναι εάν είναι δυνατό κάποιο άτομο που δεν είναι φοιτητής να είναι βοηθός ερευνητή. Από την ιεραρχία των εννοιών που παρουσιάστηκε παραπάνω είναι σαφές ότι κάτι τέτοιο είναι αδύνατο. Πράγματι, η απόκριση του συστήματος στο νέο αρχείο είναι `inconsistent` σε χρόνο 250 milliseconds. Ο Pellet αποφαίνεται για την ασυνέπεια της ίδιας οντολογίας σε χρόνο 125 milliseconds, και ο Fact++ σε χρόνο 62 milliseconds.

Στη συνέχεια, κατασκευάζουμε ένα άτομο το οποίο αποτελεί στιγμιότυπο της έννοιας Student, John:Student, και ένα στιγμιότυπο της έννοιας Course, math:Course. Θέτουμε το ερώτημα εάν είναι δυνατόν ο μαθητής John να διδάσκει το μάθημα math, δηλαδή εάν A⊨{(John, math): teacherOf}. Το νέο σώμα ισχυρισμών είναι ασυνεπές αφού από τους περιορισμούς στο σώμα της ορολογίας ένας μαθητής δεν επιτρέπεται να διδάσκει. Το σύστημα αποκρίνεται σε χρόνο 281 milliseconds. Η απάντηση από τον Pellet προκύπτει σε χρόνο 156 milliseconds και από τον Fact++ σε 138 milliseconds.

8 *Επίλογος*

8.1 Σύνοψη και συμπεράσματα

Συνοψίζοντας, καταλήγουμε ότι η μηχανή που αναπτύχθηκε αποφαίνεται με αξιοπιστία για τη συνέπεια ή μη, μιας οντολογίας. Οι τεχνικές βελτιστοποίησης που χρησιμοποιήθηκαν, όπως το ξεδίπλωμα κατά απαίτηση (lazy unfolding) και η οπισθοδρόμηση με σύνολο εξάρτησης (dependency directed backtracking), συμβάλλουν στην καλύτερή της επίδοση. Βεβαίως οι απαιτήσεις σε χρόνο εκτέλεσης και μνήμη συστήματος παραμένουν αυξημένες.

8.2 Μελλοντικές επεκτάσεις

Υπάρχουν αρκετές δυνατότητες επέκτασης της μηχανής που κατασκευάστηκε. Οι επεκτάσεις αφορούν τόσο τις επιλογές που μπορεί να έχει ο χρήστης του συστήματος όσο και τις δυνατότητες που μπορεί να υποστηρίζει η ίδια η μηχανή.

Σε κάποια μελλοντική επέκταση, ο χρήστης θα μπορούσε ενδεχομένως, να θέσει απ΄ευθείας ένα ερώτημα στη μηχανή, χωρίς να είναι αναγκασμένος να το ενσωματώσει στο αρχείο της οντολογίας. Βεβαίως, σε ό,τι αφορά τη μηχανή, μπορεί να εξελιχθεί ώστε να υποστηρίζει περιγραφικές λογικές υψηλότερης εκφραστικότητας, καθώς και γενικευμένα αξιώματα υπαγωγής. Τέλος, μία ανεπτυγμένη διεπαφή θα διευκόλυνε τη χρήση του συστήματος, κάτι στο οποίο δεν δόθηκε ιδιαίτερη προσοχή καθώς δε συνάγει άμεσα με τους σκοπούς της διπλωματικής.

9 *Βιβλιογραφία*

[1] Ronald Branchman, Hector Levesque. Knowledge Representation and Reasoning, Elsevier, 2004.

[2] Γ. Στοΐλος. Εισαγωγή στις Περιγραφικές Λογικές. Σημειώσεις, 2007.

[3] Franz Baader, Ian Horrocks, Ulrike Sattler, In Frank van Harmelen, Vladimir Lifschitz, Bruce Porter. Description Logics, Hanbook of Knowledge Representation. Elsevier, 2007.

[4] Franz Baader, Uli Sattler. An Overview of Tableau Algorithms for Description Logics. Studia Logic, vol.69, 2000.

[5] Franz Baader, Ian Horrocks, Ulrike Sattler. Description Logics (Chapter 3). Elsevier, 2007.

[6] Franz Baader, Werner Nutt. Basic Description Logics (Chapter 2). Cambridge University Press, 2003.

[7] Ian Horrocks. Implementation and Optimisation Techniques (Chapter 9). Cambridge University Press, 2003

[8] Ian Horrocks, Ulrike Sattler. A Description Logic with Transitive and Inverse Role Hierarchies. Oxford University Press, 1999.

[9] Dmitry Tsarkov, Ian Horrocks, Peter F. Patel-Schneider. J. of Automated Reasoning, 39(3):277-316, 2007.

[10] Dmitry Tsarkov, Ian Horrocks, Peter F. Patel-Schneider. Optimizing Terminological Reasoning for Expressive Description Logics. Springer Science + Business Media B.V. 2007.

[11] Γ. Στοΐλος. Γλώσσες Αναπαράστασης Γνώσης στο Σημασιολογικό Ιστο. Σημειώσεις, 2007.

[12] Dmitry Tsarkov, Ian Horrocks. Fact Description Logic Reasoner: System Description. The University of Manchester, 2006.

[13] Bijan Parsia, Evren Sirin. Pellet: an OWL DL Reasoner. MINDSWAP Research Group, University of Maryland, 2004.

[14] Zhengxiang Pan. Benchmarking DL Reasoners Using Realistic Ontologies.

Bell Labs Research and Lehigh University, 2006.

[15] Rogers Cadenhead, Laura Lemay, Πλήρες Εγχειρίδιο της Java 2, Τρίτη Έκδοση, Μόσχος Γκιούρδας, 2003.

[16] http://owlapi.sourceforge.net/

[17] http://www.w3.org/TR/owl2-profiles/

[18] http://www.w3.org/TR/owl-features/

[19] http://www.mindswap.org/2003/pellet/

[20] http://clarkparsia.com

[21] http://owl.man.ac.uk/factplusplus/

[22] http://www.sts.tu-harburg.de/~r.f.moeller/racer/

[23] http://protege.stanford.edu/

[24] http://protege.stanford.edu/doc/users.html#tutorials

[25] http://protege.cim3.net/cgi-bin/wiki.pl?ProtegeOntologiesLibrary

[26] http://www.co-ode.org/ontologies/

[27] http://km.aifb.uni-karlsruhe.de/projects/owltests/

[28] http://semanticweb.org/wiki/

[29] http://java.sun.com/j2se/1.4.2/docs/api/

[30] http://homepages.cwi.nl/~daybuild/daily-books/technology/aterm-guide/aterm-guide.html

[31] http://owl.cs.manchester.ac.uk/2007/05/api/javadoc/org/semanticweb/owl/