Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων
Και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Διασύνδεση με μηχανές συλλογιστικής για ασαφείς περιγραφικές λογικές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΚΩΝΣΤΑΝΤΙΝΟΥ Ν. ΣΙΣΜΑΝΗ

Αθήνα, Ιούλιος 2009

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων & Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Διασύνδεση με μηχανές συλλογιστικής για ασαφείς περιγραφικές λογικές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΚΩΝΣΤΑΝΤΙΝΟΥ Ν. ΣΙΣΜΑΝΗ

ΕΠΙΒΛΕΠΩΝ: Γιώργος Στάμου
Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 22η Ιουλίου 2009.

...	...	...
Γ. Στάμου	Στ. Κόλλιας	Α.Σταφυλοπάτης
Λέκτορας Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2009

. . .

ΚΩΝΣΤΑΝΤΙΝΟΣ Ν. ΣΙΣΜΑΝΗΣ
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

ΠΕΡΙΛΗΨΗ

Το όραμα για τον Σημασιολογικό Ιστό, όπως εκφράζεται από τον Tim Berners Lee, είναι ενός ιστού όπου οι πληροφορίες είναι προσβάσιμες όχι μόνο στους ανθρώπους, αλλά επίσης σε αυτοματοποιημένες εφαρμογές. Πράκτορες, που διασχίζουν τον ιστό πραγματοποιώντας χρήσιμες εργασίες, όπως η λεπτομερέστερη αναζήτηση, η ανακάλυψη νέων πηγών πληροφοριών και το φιλτράρισμα τους. Ο αυτοματισμός αυτών των εργασιών προϋποθέτει την αναβάθμιση του παγκόσμιου ιστού από το στάδιο του «αναγνώσιμου» από τις μηχανές στο στάδιο που θα ονομάζαμε «κατανοήσιμο από τις μηχανές». Η κεντρική ιδέα είναι ο ορισμός και η διασύνδεση της πληροφορίας κατά τέτοιο τρόπο, έτσι ώστε να επιτρέπεται, η ερμηνεία της με σαφήνεια από ένα λογισμικό και όχι απλά η αυτονόητη ερμηνεία της από ανθρώπους.

Κλειδί για την υλοποίηση της ιδέας του «κατανοήσιμου από τις μηχανές» αποτελεί η ανάπτυξη γλωσσών με αυστηρά ορισμένη σημασιολογία· όπως είναι οι Περιγραφικές Λογικές. Επιπλέον η ανάγκη διαχείρισης της εγγενής ασάφειας, που διακρίνει την πολυμεσική πληροφορία, καθιστά την έρευνα των Ασαφών επεκτάσεων των Περιγραφικών Λογικών, ιδιαίτερα σημαντική.

Η χρήση των μηχανών συλλογιστικής Περιγραφικών Λογικών γίνεται όλο και πιο ευρεία για την συλλογιστική επάνω σε πληροφορίες του Σημασιολογικού Ιστού και σε πολλές ακόμη εφαρμογές. Ιδιαίτερα χρήσιμος είναι ο ορισμός ενός κοινού τρόπου διασύνδεσης, που θα επέτρεπε σε απομακρυσμένες εφαρμογές να αλληλεπιδρούν με διαφορετικές μηχανές συλλογιστικής. Η διασύνδεση DIG για συστήματα Περιγραφικών Λογικών αποτελεί μια προσπάθεια ορισμού ενός προτύπου που παρέχει αυτήν ακριβώς τη δυνατότητα. Οι εργασίες μας για την επέκταση του προτύπου DIG, με την υποστήριξη των Ασαφών Περιγραφικών Λογικών και η υλοποίηση του σε μια μηχανή ασαφούς συλλογιστικής, αποτελούν το αντικείμενο της εργασίας αυτής.

Λέξεις Κλειδιά : Περιγραφικές Λογικές, Ασαφείς Περιγραφικές Λογικές, αβεβαιότητα, ασάφεια, συλλογιστική, έννοια, ρόλος, αξίωμα, ισχυρισμός, σώμα ορολογίας, σώμα ισχυρισμών, συνεπαγωγή, ικανοποιησιμότητα, διασύνδεση DIG Interface, HTTP, XML.

ABSTRACT

The Semantic Web vision as articulated by Tim Berners-Lee, is of a web in which resources are accessible not only to humans, but also to automated processes "agents" roaming the web performing useful tasks such as improved search, resource discovery, information brokering and information filtering. The automation of tasks depends on elevating the status of the web from machine-readable to something we might call machine-understandable. The key idea is to have data on the web defined and linked in such a way that its meaning is explicitly interpretable by software processes rather than just being implicitly interpretable by humans.

Thus the provision of languages with well-defined semantics is seen as key to enabling the notion of machine-understandability, and the latest generation of knowledge representation languages, such as the Description Logics have placed much emphasis on such semantics. The necessity of handling the inherent imprecision in multimedia object representation lead to the emerging "fuzzy" Description Logics.

Description logic reasoners are becoming more widely used for reasoning about resources on the Semantic Web. To allow client tools to interact with different reasoners in a standard way, a common standard interface is highly desirable. The DIG Description Logic Interface is a proposed standard that provides just this capability. We extend the proposed standard with the capability of handling fuzzy knowledge , and we implement a DIG adapter for a fuzzy reasoner. These efforts are the subject of this report.

Keywords : Description Logics,Fuzzy Description Logics,uncertainty,reasoner, fuzziness, concept, role, axiom, assertion, terminological box,assertional box, knowledge base, reasoning service, entailment, satisfiability,DIG Interface, HTTP, XML

Κεφάλαιο 1

Εισαγωγή

Πρώτοι οι μαθηματικοί, από την εποχή του Αριστοτέλη καταπιάστηκαν με το πρόβλημα της τυπικής (μαθηματικής) αναπαράστασης της ανθρώπινης γνώσης· έτσι ώστε η γνώση να περιγράφεται με μονοσήμαντη και πλήρως καθορισμένη ερμηνεία. Σήμερα η Επιστήμη των Υπολογιστών και ιδιαίτερα ο τομέας της Τεχνητής Νοημοσύνης ασχολείται με την αναπαράσταση της ανθρώπινης γνώσης σε ένα Υπολογιστικό Σύστημα με σκοπό την αξιοποίηση της, με τη διενέργεια σύνθετων ενεργειών που θα πλησιάζουν σε ποιότητα αυτών της ανθρώπινης συλλογιστικής και την εξαγωγή ευφυών αποτελεσμάτων.

Αρκετά Υπολογιστικά Συστήματα χρησιμοποιούν κάποιο είδος γλώσσας αναπαράστασης γνώσης για τη βελτίωση των δυνατοτήτων τους. Παραδείγματα είναι οι εφαρμογές βάσεων δεδομένων, όπου γνώση χρησιμοποιείται για τη βελτίωση της ακρίβειας της ανάκλησης πληροφορίας (information retrieval), ή οι εφαρμογές επεξεργασίας πολυμεσικών κειμένων. Το πιο δημοφιλές παράδειγμα χρήσης γλωσσών αναπαράστασης γνώσης είναι ο Παγκόσμιος Ιστός (World Wide Web), όπου γνώση χρησιμοποιείται για τη βελτίωση των δυνατοτήτων των πρακτόρων (agents)· μια από τις προσπάθειες για την ανάπτυξη του Σημασιολογικού Ιστού (Semantic Web)

Κεντρικό ρόλο στην ανάπτυξη των εφαρμογών αναπαράστασης γνώσης κατέχουν οι Περιγραφικές Λογικές [2]. Οι Περιγραφικές Λογικές (ΠΛ) αποτελούν μια οικογένεια γλωσσών αναπαράστασης γνώσης βασισμένων σε έννοιες - μοναδιαία κατηγορήματα και ρόλους - δυαδικά κατηγορήματα, σε αντίθεση με την Λογική Πρώτης Τάξης (First-Order Logic) και τον Λογικό Προγραμματισμό (Logic Programming) που χρησιμοποιούν γενικευμένα κατηγορήματα ν-οστού βαθμού. Η ομοιότητα των ΠΛ με το αντικειμενοστραφές μοντέλο, που τις καθιστά ιδιαίτερα εύχρηστες, και η ισορροπία που διατηρούν ανάμεσα στην δυνατότητα εκφραστικότητας και στην αποφασισιμότητα είναι δύο από κύρια τα χαρακτηριστικά τους στα οποία οφείλεται η ευρεία διάδοση τους.

Αν και χρησιμοποιούνται με μεγάλο βαθμό επιτυχίας σε διαφορετικές εφαρμογές, η χρήση των ΠΛ σε εφαρμογές για την ανάκτηση πολυμεσικής πληροφορίας αναδεικνύουν την ανάγκη επέκτασης τους για την διαχείριση της εγγενής ανακρίβειας και αβεβαιότητας που διακρίνει τις διαδικασίες ανάκλησης και αναπαράστασης γνώσης από πολυμεσικά αντικείμενα. Σε αυτές τις περιπτώσεις ένα αίτημα ανάκλησης πληροφορίας δεν μπορεί να ικανοποιηθεί με έναν απόλυτο τρόπο. Η σχετικότητα ενός πολυμεσικού αντικειμένου με τα δεδομένα του αιτήματος μπορεί να επαληθευτεί μόνο ως προς έναν βαθμό. Η επέκταση των ΠΛ με τη δυνατότητα διαχείρισης της ασάφειας βασίζεται στην ασαφή συνολοθεωρία και οδήγησε στην ανάπτυξη των Ασαφών ΠΛ. Η ενασχόληση με τις ασαφείς επεκτάσεις των γλωσσών που έχουν προταθεί για την αναπαράσταση

γνώσης στο Σημασιολογικό Ιστό απαιτεί την μελέτη πρωτότυπων αλγορίθμων για την εξαγωγή συμπερασμάτων σε πολύ εκφραστικές ΠΛ.

Ταυτόχρονα με το ερευνητικό ενδιαφέρον που συγκεντρώνουν οι ασαφείς ΠΛ, μεγάλη προσπάθεια γίνεται για την ανάπτυξη και υλοποίηση στην πράξη μηχανών συλλογιστικής και εφαρμογών τους που θα παρέχουν αποδεκτή υπολογιστική απόδοση σε ρεαλιστικά προβλήματα. Με δεδομένο το συνεχώς αυξανόμενο ενδιαφέρον για την ανάπτυξη εφαρμογών στο Σημασιολογικό Ιστό, γίνεται αντιληπτό πόσο σημαντική και αναγκαία είναι για τον προγραμματιστή σημασιολογικών συστημάτων η ύπαρξη ενός εύκολου τρόπου πρόσβασης σε διαφορετικές μηχανές συλλογιστικής. Ένας ενιαίος και πρότυπος τρόπος για την πρόσβαση σε μία συλλογιστική μηχανή (reasoner), θα βοηθούσε επιπλέον, στην ανάπτυξη συστημάτων περιγραφικών λογικών, καθώς οι μηχανικοί και προγραμματιστές θα μπορούν να επικεντρωθούν στην ανάπτυξη της ίδιας της μηχανής συλλογιστικής, απαλλαγμένοι από το φόρτο της υποστήριξης των ποικίλων υπηρεσιών αναπαράστασης που μπορεί να απαιτούνται από την κάθε εφαρμογή. Ένα τέτοιο πρότυπο διασύνδεσης με διαφορετικές μηχανές συλλογιστικής ορίζει η διασύνδεση DIG από την ομάδα Description Logic Implementation Group[1].Αν και το ακρωνύμιο αυστηρά ορίζει το όνομα της ομάδας που είναι υπεύθυνη για την ανάπτυξη · με αυτό το όνομα θα αναφερόμαστε και στο ίδιο το πρότυπο.

Το πρότυπο DIG περιγράφει τη διασύνδεση ανάμεσα σε μία μηχανή συλλογιστικής και στις πιθανές εφαρμογές της, επιτυγχάνοντας την ανεξάρτητη ανάπτυξη τους, για παράδειγμα ως προς την γλώσσα υλοποίησης ή ακόμη και ως προς την τοποθεσία εκτέλεσης τους. Η διασύνδεση DIG προτείνει ένα εύκολο τρόπο χρήσης εναλλακτικών reasoner από τις εφαρμογές συστημάτων ΠΛ χωρίς να χρειάζεται καμία μετατροπή τους. Διευκολύνεται έτσι το πειραματικό έργο των ερευνητών των ΠΛ αλλά και της γενικότερης εφαρμογής των συστημάτων ΠΛ στην πράξη.

Σκοπό της εργασίας αποτελεί η επέκταση και η υλοποίηση της πρότυπης διασύνδεσης DIG με την υποστήριξη των ασαφών Περιγραφικών Λογικών και των μηχανών συλλογιστικής τους. Για την επίτευξη του μελετούμε την εκφραστική δυνατότητα του προτύπου DIG και την επεκτείνουμε με τα απαραίτητα συντακτικά στοιχεία για την περιγραφή ασαφών οντολογιών. Στη συνέχεια υλοποιούμε την διασύνδεση στην γλώσσα προγραμματισμού Java, και επεκτείνουμε με αυτή, τη μηχανή συλλογιστικής ασαφούς περιγραφικής λογικής $fuzzy - SHIN$, FiRE.

Η εργασία αυτή οργανώνεται ως εξής :

- Στη συνέχεια της Εισαγωγής παρουσιάζουμε συνοπτικά τα χαρακτηριστικά των κλασσικών Περιγραφικών Λογικών. Επιπλέον αναφερόμαστε στις ασαφείς επεκτάσεις των ΠΛ και στις υπηρεσίες συλλογιστικής τους.

- Ακολούθως στο κεφάλαιο 2 παρουσιάζεται αναλυτικά το πρότυπο διασύνδεσης με μηχανές συλλογιστικής DIG. Περιγράφουμε τις υποθέσεις στις οποίες βασίζεται, το πρωτόκολλο επικοινωνίας και την γλώσσα περιγραφής γνώσης που προτείνει .

- Στο κεφάλαιο 3 μελετούμε αναλυτικότερα τις δομές του προτύπου, που ενδιαφερόμαστε να επεκτείνουμε με την υποστήριξη της ασάφειας και ακολούθως παρουσιάζουμε τις επεκτάσεις μας προς αυτή την κατεύθυνση.

- Στο κεφάλαιο 4 γίνεται μια διεξοδική ανάλυση της υλοποίησης της διασύνδεσης, των δομών και των υπηρεσιών που αυτή διαθέτει.

[1]http://dl.kr.org/dig/

- Στο κεφάλαιο 5 παρουσιάζουμε τις τελικές σκέψεις μας για το πρότυπο και παρουσιάζουμε θέματα για μελλοντική έρευνα.

1.1 Περιγραφικές Λογικές

Οι Περιγραφικές Λογικές (ΠΛ) (Description Logics - DLs)[2] αποτελούν μια οικογένεια γλωσσών αναπαράστασης γνώσης, είναι βασισμένες στην λογική, και έχουν σχεδιαστεί για την καταγραφή γνώσης και την διενέργεια εργασιών συλλογιστικής επάνω σε αυτή με ένα δομημένο και κατανοητό τρόπο. Οι ΠΛ είναι μέλη μιας ευρύτερης οικογένειας γλωσσών αναπαράστασης γνώσης, που ονομάζονται γλώσσες περιγραφής (description languages). Ουσιαστικά προέκυψαν από την συστηματική προσπάθεια ορισμού αυστηρής σημασιολογίας, για τη γλώσσα των Σημασιολογικών Δικτύων (Semantic Networks) [7] και τη γλώσσα των Πλαισίων (Frames) [5]· οι οποίες δεν διέθεταν κάποια μαθηματική θεωρία για την απόδοση τυπικής σημασιολογίας. Οι γλώσσες αυτές βασίζονται σε ένα γραφικό μοντέλο που θυμίζει πολύ το αντικειμενοστραφές μοντέλο των κλάσεων και των ιδιοτήτων· κάτι που τις καθιστά προσιτές στην κατανόηση και στη χρήση και από μη ειδικούς του χώρου.

Οι ΠΛ προσφέρουν ένα σύνολο από κατασκευαστές (constructors) για την δημιουργία εννοιών (concepts) και ρόλων (roles). Έννοιες και ρόλοι συμμετέχουν μαζί με αντικείμενα σε αξιώματα (axioms) και ισχυρισμούς (assertions) και συνθέτουν βάσεις γνώσεις (knowledge bases)· πάνω στις οποίες μπορούμε να εφαρμόσουμε εργασίες συλλογιστικής, για την εξαγωγή νέας γνώσης από την ήδη περιγραφείσα. Ένα ακόμη βασικό χαρακτηριστικό των ΠΛ, συνυπεύθυνο για την ευρεία διάδοση τους, είναι ότι στην πλειοψηφία τους είναι αποφασίσιμες (decidable).Οι ΠΛ αποτελούνται από ένα αλφάβητο (alphabet), ένα συντακτικό (syntax) και μια σημασιολογία (semantics).

Το αλφάβητο ορίζεται από ένα σύνολο ατομικών εννοιών (atomic concepts) (**C**), ένα σύνολο ατομικών ρόλων (atomic roles) (**R**) και από ένα σύνολο ατόμων (individuals) (**I**). Οι ατομικές έννοιες και ρόλοι αποτελούν τα δομικά στοιχεία της γλώσσας. Διαισθητικά μια έννοια C αναπαριστά ένα σύνολο αντικειμένων με κοινές ιδιότητες, όπως οι κλάσεις στο αντικειμενοστραφές μοντέλο δεδομένων. Αντίθετα ένας ρόλος R αναπαριστά μια δυαδική σχέση ανάμεσα σε δύο αντικείμενα.

Οι ΠΛ παρέχουν ένα σύνολο από κατασκευαστές, οι οποίοι επενεργούν πάνω στις ατομικές έννοιες και ρόλους και συντελούν στην κατασκευή περισσότερο πολύπλοκων εννοιών και εκφράσεων που ονομάζονται περιγραφές εννοιών (concept descriptions) ή σύνθετες έννοιες (complex concepts). Από το σύνολο των κατασκευαστών που χρησιμοποιούνται ορίζεται η ΠΛ. Για παράδειγμα το σύνολο των κατασκευαστών $\neg, \sqcap, \sqcup, \exists, \forall$ ορίζει την ΠΛ ALC. Δομικά στοιχεία των ΠΛ αποτελούν οι έννοιες $\top$ και $\bot$,που ονομάζονται καθολική έννοια (universal concept) και κενή έννοια (bottom concept), αντίστοιχα. Οι έννοιες $\forall R.C$ και $\exists R.C$ ονομάζονται περιορισμός τιμής (value restriction) και υπαρξιακός περιορισμός existensial restriction, αντίστοιχα. Η ΠΛ ALC περιέχει λοιπόν έννοιες της μορφής :

$$C, D \rightarrow \top | \bot | \neg C | C \sqcup D | C \sqcap D | \forall R.C | \exists R.C$$

Με την εισαγωγή νέων κατασκευαστών στην ΠΛ επιτυγχάνεται ο ορισμός περισσότερο σύνθετων εννοιών. Οι έννοιες της μορφής $\geq pR$ και $\leq pR$ ονομάζονται περιορισμοί πληθυκότητας και συμβολίζονται με το γράμμα N. Από αυτούς ο $\geq pR$ ονομάζεται ως περιορισμός το-λιγότερο (at - least) και ο $\leq pR$ περιορισμός το-περισσότερο (at - most). Στην περίπτωση που επιτρέπουμε στην μεταβλητή p να πάρει μόνο τις τιμές 0

και 1 προκύπτουν οι συναρτησιακοί περιορισμοί πληθυκότητας, που συμβολίζονται με το γράμμα F. Οι έννοιες της μορφής $\{o\}$ ονομάζονται ονοματικές έννοιες (nominal concepts) και η δυνατότητα ορισμού τους συμβολίζεται με το γράμμα O. Η ΠΛ μπορεί επίσης να διαθέτει τον κατασκευαστή για τον ορισμό αντίστροφων ρόλων (inverse roles), που συμβολίζεται με το γράμμα I. Στην περίπτωση που υποστηρίζονται οι παραπάνω κατασκευαστές έχει οριστεί η γλώσσα $ALCOIN$. Με την εκφραστική δυνατότητα της γλώσσας αυτής μπορούμε να περιγράψουμε έννοιες του κόσμου μας, όπως η έννοια των ανθρώπων που έχουν ακριβώς τρία παιδιά, γράφοντας

$$Human \sqcap \geq 3 hasChild \sqcap \leq 3 hasChild,$$

την έννοια των ελαττωματικών μηχανών γράφοντας

$$Machine \sqcap \exists hasPart.MachinePart \sqcap \forall hasPart.FaultyPart,$$

την έννοια των ημερών της εβδομάδας, γράφοντας

$$Sunday \sqcup Monday \sqcup \ldots \sqcup Saturday$$

ή την ιδιότητα $hasParent$ γράφοντας $hasChild^{-}$.

Ιδιαίτερο χαρακτηριστικό των γλωσσών περιγραφικής λογικής είναι η υψηλή τυπικότητα που τις διακρίνει. Τα δομικά στοιχεία τους ερμηνεύονται με έναν τυπικό, μαθηματικό τρόπο, έτσι ώστε το νόημα μιας δήλωσης να επιδέχεται μοναδικής ερμηνείας, κάτι που δεν συμβαίνει με τις φυσικές γλώσσες.

Οι ΠΛ διαθέτουν μια μοντελοθεωρητική σημασιολογία (model-theoretic semantics), που ορίζεται με τη χρήση ερμηνειών. Μια ερμηνεία (interpretation) I αποτελείται από ένα χώρο ερμηνείας (domain of interpretation) Δ^I και από μια συνάρτηση ερμηνείας (interpretation function) $\cdot^I$. Ο χώρος ερμηνείας αποτελεί ένα σύνολο αντικειμένων (objects) και η συνάρτηση ερμηνείας απεικονίζει :

- κάθε άτομο $a \in I$ σε ένα αντικείμενο $a^I \in \Delta^I$
- κάθε ατομική έννοια $CN \in C$ σε ένα υποσύνολο $CN^I \subseteq \Delta^I$
- κάθε ατομικό ρόλο $RN \in R$ σε μια δυαδική σχέση $RN^I \subseteq \Delta^I \times \Delta^I$

Η συνάρτηση ερμηνείας μπορεί να επεκταθεί παρέχοντας την ερμηνεία για πιο πολύπλοκες έννοιες και ρόλους. Όπως γίνεται αντιληπτό κάθε έννοια της ΠΛ ερμηνεύεται ως ένα υποσύνολο του Δ^I. Για παράδειγμα η έννοια $\top$ ερμηνεύεται ως το σύνολο που περιέχει όλα τα αντικείμενα του χώρου ερμηνείας, ενώ η έννοια $\bot$ ερμηνεύεται ως το κενό σύνολο. Η έννοια $C \sqcap D$ ερμηνεύεται ως το σύνολο που προκύπτει από την τομή των ερμηνειών των εννοιών C και D. Η ερμηνεία της έννοιας $\forall R.C$ περιέχει το σύνολο των αντικειμένων του Δ^I τα οποία αν συμμετέχουν στον ρόλο R^I με κάποιο άλλο αντικείμενο, τότε το αντικείμενο αυτό ανήκει στην ερμηνεία της έννοιας C δηλαδή στο σύνολο C^I.Με αντίστοιχο τρόπο παρέχεται η ερμηνεία κάθε έννοιας που συνθέτουν οι κατασκευαστές μιας ΠΛ.

Εκτός από την χρήση των κατασκευαστών εννοιών πολύπλοκες έννοιες μπορούμε να ορίζουμε με τη βοήθεια αξιωμάτων ορολογίας (terminological axioms) τα οποία συνθέτουν ένα σώμα ορολογίας. Ένα σώμα ορολογίας (terminological box TBox) T, είναι ένα πεπερασμένο σύνολο από αξιώματα υπαγωγής εννοιών (general inclusion axioms) της μορφής $C \subseteq D$ και αξιώματα ισοδυναμίας εννοιών (general equivalence axioms) $C \equiv D$, όπου οι C και D είναι έννοιες της ΠΛ. Τα αξιώματα υπαγωγής εννοιών

στα οποία η έννοια του αριστερού μέλους είναι μια πολύπλοκη έννοια ονομάζονται αξιώματα υπαγωγής γενικευμένων εννοιών (general concept inclusions - GCIs) ή απλά γενικευμένα αξιώματα. Στην περίπτωση που ένα TBox περιέχει αξιώματα στα οποία μια έννοια το αριστερού μέλους ορίζεται είτε άμεσα είτε έμμεσα από τον εαυτό της, περιέχεται δηλαδή κάποιος κύκλος, τότε καλούμε το TBox κυκλικό. Ένα Tbox που δεν περιέχει γενικευμένα και κυκλικά αξιώματα, καλείται απλό.

Μια ερμηνεία ικανοποιεί satisfies ένα αξίωμα $C \subseteq D$ όταν $C^I \subseteq D^I$, ενώ ικανοποιεί ένα αξίωμα $C \equiv D$ αν $C^I \equiv D^I$. Μια ερμηνεία I ικανοποιεί ένα σώμα ορολογίας T αν ικανοποιεί όλα τα αξιώματα στο T· λέμε τότε ότι η I είναι μοντέλο (model) του T.

Με αντίστοιχο τρόπο οι ΠΛ παρέχουν μηχανισμούς για την περιγραφή αξιωμάτων ρόλων, που οργανώνονται σε σώματα ρόλων. Ένα σώμα ρόλων (role box RBox) R, είναι ένα πεπερασμένο σύνολο από αξιώματα μεταβατικών ρόλων (transitive role axioms), της μορφής $Trans(R)$ και αξιώματα υπαγωγής ρόλων role inclusion axioms, της μορφής $R \subseteq S$. Με τα αξιώματα αυτά μπορούμε να δηλώσουμε για παράδειγμα ότι οι ρόλοι $hasPart$ και $ancestor$ είναι μεταβατικοί γράφοντας, $Trans(hasPart)$ και $Trans(ancestor)$, αντίστοιχα, ότι ο ρόλος $hasChild$ είναι υπο-ρόλος του ρόλου $hasOffspring$, γράφοντας $hasParent \subseteq hasOffspring$,ότι ο ρόλος $hasParent$ είναι αντίστροφος του ρόλου $hasChild$, γράφοντας $hasParent \subseteq hasChild^-$.

Μια ερμηνεία I ικανοποιεί ένα αξίωμα $Trans(R)$ αν, για κάθε $x, y, z \in \Delta^I$ $\{\langle x, y\rangle, \langle y, z\rangle\} \subseteq R^I \rightarrow \langle x, z\rangle \in R^I$ και ικανοποιεί ένα αξίωμα $R \subseteq S$ αν $R^I \subseteq S^I$. Με αυτό τον τρόπο ορίζεται μια ιεραρχία ρόλων. Μια ερμηνεία I ικανοποιεί ένα σώμα ορολογίας (RBox R) αν ικανοποιεί κάθε αξίωμα του· τότε η ερμηνεία I καλείται μοντέλο του R.

Οι ΠΛ παρέχουν επιπλέον τη δυνατότητα δήλωσης αξιωμάτων ατόμων, δηλαδή σχέσεων στιγμιότυπου ανάμεσα σε άτομα και έννοιες ή σε ζεύγη ατόμων και ρόλους. Τα αξιώματα ατόμων συνθέτουν ένα σώμα ισχυρισμών (assertional box ABox), που συμβολίζεται με A και είναι ένα πεπερασμένο σύνολο από ισχυρισμούς (assertions) της μορφής $a : C$, οι οποίοι ονομάζονται ισχυρισμοί εννοιών (concept assertions), ή της μορφής $(a, b) : R$, που ονομάζονται ισχυρισμοί ρόλων (role assertions) και της μορφής $a = b$ και $a \neq b$ που δηλώνουν αν δύο άτομα είναι ταυτόσημα ή όχι.

Μια ερμηνεία I ικανοποιεί τον ισχυρισμό $a : C$ αν $a^I \in C^I$, τον ισχυρισμό $(a, b) : R$ $(a^I, b^I) \in R^I$ και τη σχέση $a = b$ $(a \neq b)$ αν $a^I = b^I$ $(a^I \neq b^I)$. Μια ερμηνεία ικανοποιεί ένα σώμα ορολογίας A αν ικανοποιεί κάθε αξίωμα του· σε αυτήν την περίπτωση λέμε ότι η ερμηνεία I αποτελεί μοντέλο του A.

Τα σώματα ορολογίας T, ρόλων R και ισχυρισμών A συνθέτουν μια βάση γνώσης (knowledge base) $\Sigma = \langle T, R, A.\rangle$. Μια ερμηνεία I ικανοποιεί μια βάση γνώσης Σ αν ικανοποιεί κάθε αξίωμα της· στην περίπτωση αυτή λέμε ότι η I είναι μοντέλο της Σ.

Οι ΠΛ παρέχουν υπηρεσίες εξαγωγής νέας γνώσης από την ήδη περιγραφείσα, που ονομάζονται υπηρεσίες συλλογιστικής (inference services). Συνοπτικά συγκεντρώνουμε τις υπηρεσίες συλλογιστικής τους :

Ικανοποιησιμότητα μιας ΒΓ : Μια ΒΓ Σ είναι ικανοποιήσιμη αν και μόνο αν (ανν) υπάρχει μοντέλο I για την Σ.

Ικανοποιησιμότητα μιας έννοιας : Η έννοια C είναι ικανοποιήσιμη με βάση την (μ.β.τ.) Σ ανν υπάρχει μοντέλο I της Σ τέτοιο ώστε (τ.ω.) $C^I \neq \emptyset$.

Υπαγωγή Εννοιών : Η C υπάγεται (subsumed) στην D μ.β.τ. Σ ανν για κάθε μοντέλο της Σ έχουμε $C^I \subseteq D^I$.

Συνέπεια ενός σώματος ισχυρισμών : Ένα σώμα ισχυρισμών A είναι συνεπές (Consistent) αν υπάρχει κάποιο μοντέλο για το A.

Λογική Συνεπαγωγή : Δοθέντος ενός αξιώματος εννοιών, ρόλων ή ενός ισχυρισμού, φ, η Σ συνεπάγεται λογικά (entails) το φ, γράφοντας $\Sigma \models \varphi$ ανν για κάθε μοντέλο I της Σ έχουμε ότι η I ικανοποιεί το φ.

Για πολλές από τις υπηρεσίες συλλογιστικής, απαιτείται να ελέγξουμε όλα τα μοντέλα μιας βάσης γνώσης· αυτό είναι αδύνατο. Τα περισσότερα από τα προβλήματα συλλογιστικής ανάγονται σε απλούστερα. Συγκεκριμένα ισχύει ότι :

- Η έννοια C είναι ικανοποιήσιμη μ.β.τ. Σ ανν το σώμα ισχυρισμών $\{a : C\}$ είναι συνεπές μ.β.τ Σ για κάποιο τυχαίο α.
- Η C υπάγεται στην D μ.β.τ. Σ ανν το σώμα ισχυρισμών $\{a : C \sqcap \neg D\}$ είναι μη - συνεπές μ.β.τ Σ.
- Μια $\Sigma = \langle T, R, A\rangle$ συνεπάγεται έναν ισχυρισμό $a : C$ ανν η $\Sigma = \langle T, R, A \sqcup \{a : \neg C\}\rangle$ είναι μη - ικανοποιήσιμη.

Οι υπηρεσίες συλλογιστικής βασίζονται σε αλγόριθμους· που ονομάζουμε αλγόριθμους συλλογιστικής. Η ανάπτυξη αποφασίσιμων αλγορίθμων για τα παραπάνω προβλήματα είναι αρκετά δύσκολη και γίνεται ακόμη δυσκολότερη όσο αυξάνεται η εκφραστικότητα της γλώσσας αναπαράστασης γνώσης. Οι διαδικασίες ,με τις οποίες διαχειρίζονται την γνώση οι αλγόριθμοι συλλογιστικής, σχετίζονται με την προ-επεξεργασία του σώματος ορολογίας, το οποίο μπορεί να απλοποιηθεί ή να εξαλειφθεί εντελώς, και το πρόβλημα να αναχθεί μ.β.τ. κενό TBox. Η διαδικασία αυτή ονομάζεται ξεδίπλωμα (unfolding) για τα απλά TBox ή εσωτερίκευση (internalization) για τα κυκλικά και γενικευμένα σώματα ορολογίας· στην πρώτη περίπτωση το σώμα ορολογίας εξαλείφεται εντελώς, ενώ στην δεύτερη αντικαθίσταται με ένα μόνο μόνο αξίωμα. Οι διαδικασίες αυτές βασίζονται σε αλγόριθμους πινάκων (tableaux algorithms), για την απλοποίηση περίπλοκων εκφράσεων σε πιο απλές, των οποίων η ικανοποιησιμότητα είναι προφανής.

1.2 Ασαφείς Περιγραφικές Λογικές

Η χρήση των ΠΛ σε ορισμένες εφαρμογές, έχει δείξει ότι σε πολλές περιπτώσεις θα επιθυμούσαμε την επέκταση των περιγραφικών και συλλογιστικών τους δυνατοτήτων. Ειδικότερα ο τομέας της ανάκτησης και αναπαράστασης της πολυμεσικής πληροφορίας, αναδεικνύει την ανάγκη επέκτασης των ΠΛ, για την διαχείριση της εγγενής ανακρίβειας και αβεβαιότητας που διακρίνει τα πολυμεσικά αντικείμενα Η αναπαράσταση κάποιων εγγενώς ασαφών εννοιών, όπως είναι οι έννοιες «σκούρος» ή «στρογγυλός», είναι αδύνατη με τις κλασσικές ΠΛ. Η αναπαράσταση των πολυμεσικών αντικειμένων και οι διαδικασίες συλλογιστικής επάνω σε αυτά θα πρέπει να διακρίνονται από ένα βαθμό αβεβαιότητας. Η επέκταση των ΠΛ με ικανότητες αναπαράστασης και διαχείρισης ασαφούς γνώσης, έχει ως βάση την ασαφή συνολοθεωρία (fuzzy set theory). Αντίθετα από την κλασσική συνολοθεωρία στην οποία ένα αντικείμενο ανήκει ή όχι σε ένα σύνολο, στην ασαφή συνολοθεωρία ένα αντικείμενο μπορεί να ανήκει σε ένα ασαφές σύνολο σε ένα βαθμό συμμετοχής ανάμεσα στο 0 και στο 1.

Όπως στις κλασσικές, στις ασαφείς ΠΛ ορίζουμε αρχικά ένα αλφάβητο από ατομικές ασαφείς έννοιες $\mathbf{C}$, ατομικούς ασαφείς ρόλους $\mathbf{R}$ και άτομα $\mathbf{I}$. Οι ασαφείς έννοιες

και ρόλοι ορίζονται συντακτικά με τον ίδιο ακριβώς τρόπο που ορίζονται οι αντίστοιχες κλασσικές έννοιες και ρόλοι. Αυτό συμβαίνει γιατί περιγραφικά, οι ασαφείς επεκτάσεις μιας ΠΛ αναφέρονται στο βαθμό συμμετοχής ενός ατόμου σε μια κλάση, δηλαδή στο επίπεδο των ισχυρισμών και όχι σε αυτό των κλάσεων και των ρόλων.

Αν και ο ορισμός των ασαφών εννοιών και ρόλων δεν διαφέρει από των αντίστοιχων κλασσικών, η σημασιολογία τους σε μια ασαφή ΠΛ είναι πολύ διαφορετική. Τα δομικά στοιχεία μιας ΠΛ θα πρέπει να διαθέτουν μια ασαφή ερμηνεία. Η σημασιολογία της ασαφούς γλώσσας ορίζεται με τη βοήθεια των ασαφών ερμηνειών (fuzzy interpretations)[10]. Μια ασαφή ερμηνεία αποτελείται από ένα ζευγάρι $I = (\Delta^I, \cdot^I)$, όπου ο χώρος ερμηνείας Δ^I είναι ένα μη κενό σύνολο αντικειμένων και $\cdot^I$ είναι μια ασαφής συνάρτηση ερμηνείας (fuzzy interpretation function), η οποία απεικονίζει :

- ένα άτομο $a \in I$ σε ένα στοιχείο $a^I \in \Delta^I$,
- μια ατομική έννοια $A \in C$ σε μια συνάρτηση συμμετοχής $A^I : \Delta^I \to [0, 1]$,
- και έναν ατομικό ρόλο R σε μια συνάρτηση συμμετοχής της μορφής $R^I : \Delta^I \times \Delta^I \to [0.1]$.

Ένα αντικείμενο ή ένα ζεύγος αντικειμένων μπορεί να ανήκει σε μια έννοια ή έναν ρόλο αντίστοιχα σε οποιαδήποτε βαθμό ανάμεσα στο 0 και στο 1. Η σχέση Κυκλικός^I(Πρόσωπο^I)=0.7, σημαίνει ότι το Πρόσωπο είναι κυκλικό με βαθμό ίσο με 0.7. Οι συναρτήσεις ερμηνείας μπορούν να επεκταθούν ερμηνεύοντας πολύπλοκες έννοιες και ρόλους. Η εφαρμογή διαφορετικών κατασκευαστών μπορεί να οδηγήσει στην σύνθεση πολύπλοκων εννοιών· όπως στις κλασσικές ΠΛ . Η ερμηνεία των ασαφών εννοιών και ρόλων βασίζεται στους τελεστές της ασαφούς συνολοθεωρίας. . Για παράδειγμα η ερμηνεία της ένωσης δύο εννοιών $C \sqcup D$ είναι $(C \sqcup D)^I(a) = u((C^I(a), D^I(a)))$, που σημαίνει ότι ο βαθμός συμμετοχής του αντικειμένου a στην ερμηνείας της ένωσης των δύο εννοιών ισούται με την σ- νόρμα των βαθμών συμμετοχής του αντικειμένου στα ασαφή σύνολα C^I και D^I.

Οι ασαφείς ΠΛ διαθέτουν σύνολα από αξιώματα εννοιών και ρόλων· κατά τρόπο αντίστοιχο με τις κλασσικές. Ένα ασαφές σώμα ορολογίας αποτελείται από ασαφή αξιώματα υπαγωγής $C \subseteq D$ και ισοδυναμίας $C \equiv D$ ασαφών εννοιών. Μια ασαφή ερμηνεία I ικανοποιεί το αξίωμα

$$C \subseteq D \text{ αν } \forall a \in \Delta^I, C^I(a) \leq D^I(a)$$

ή ικανοποιεί το αξίωμα

$$C \equiv D \text{ όταν } C^I(a) = D^I(a).$$

Μια ασαφής ερμηνεία ικανοποιεί ένα ασαφές σώμα αξιωμάτων T όταν ικανοποιεί κάθε του αξίωμα· τότε την αποκαλούμε μοντέλο του T. Αντίστοιχα με τις κλασσικές ΠΛ, αξιώματα υπαγωγής ασαφών ρόλων $R \subseteq S$ και αξιώματα μεταβατικών ασαφών ρόλων $Trans(R)$ συνθέτουν ένα ασαφές σώμα ρόλων (RBox). Η ερμηνεία των αξιωμάτων ασαφών ρόλων στηρίζεται αντίστοιχα στην χρήση τελεστών της ασαφούς συνολοθεωρίας.

Ένα ασαφές σώμα ισχυρισμών (assertional box) ABox είναι ένα πεπερασμένο σύνολο από ασαφή αξιώματα ατόμων τα οποία αποκαλούμε ασαφούς ισχυρισμούς. Ένας ασαφής ισχυρισμός (fuzzy assertion)[10] είναι μια δήλωση της μορφής $(a : C), ((a, b) : R) \bowtie n$, όπου $\bowtie = (\geq, >, \leq, <), a = b, a \neq b$ για $a, b \in I$. Μια ερμηνεία I ικανοποιεί :

τον ισχυρισμό $a : C \geq n$ αν $C^I(a^I) \geq n$,
τον ισχυρισμό $a : C \leq n$ αν $C^I(a^I) \leq n$,
τον ισχυρισμό $(a, b) : R \geq n$ αν $R^I(a^I, b^I) \geq n$,
τον ισχυρισμό $(a, b) : R \leq n$ αν $R^I(a^I, b^I) \leq n$,
τον ισχυρισμό $a = b$ αν $a^I = b^I$,
τον ισχυρισμό $a \neq b$ αν $a^I \neq b^I$

Μια ασαφής ερμηνεία ικανοποιεί ένα ασαφές σώμα ισχυρισμών A όταν ικανοποιεί όλους τους ασαφείς ισχυρισμούς στο A· τότε την αποκαλούμε μοντέλο του A.

Οι υπηρεσίες εξαγωγής συμπερασμάτων (inference services) ή υπηρεσίες συλλογιστικής που μπορούμε να υποστηρίξουμε σε ασαφείς ΠΛ είναι :

Ικανοποιησιμότητα μιας ασαφούς ΒΓ : Μια ΒΓ Σ είναι ικανοποιήσιμη αν και μόνο αν υπάρχει μοντέλο I για την Σ.

Ικανοποιησιμότητα μιας έννοιας : Η έννοια C είναι n - ικανοποιήσιμη (μ.β.τ.) Σ ανν υπάρχει μοντέλο I της Σ στο οποίο υπάρχει κάποιο $a \in \Delta^I$ τέτοιο ώστε $C^I(a) = n$ και $n \in [0, 1]$ [6].

Υπαγωγή εννοιών : Η C υπάγεται (subsumed) στην D μ.β.τ. Σ ανν για κάθε μοντέλο I της Σ έχουμε $\forall D \in \Delta^I, C^I(d) \leq D^I(d)$.

Συνέπεια ασαφούς σώματος ισχυρισμών : Ένα ασαφές σώμα ισχυρισμών A είναι συνεπές (consistent) μ.β.τ το σώμα ορολογίας T και το σώμα ρόλων R αν υπάρχει μοντέλο I των T και R το οποίο ικανοποιεί κάθε ισχυρισμό του A.

Λογική Συνεπαγωγή : Δοθέντος ενός αξιώματος εννοιών, ρόλων ή ενός ισχυρισμού, Ψ, η Σ συνεπάγεται λογικά (entails) το Ψ, γράφοντας $\Sigma \models \Psi$ ανν όλα τα μοντέλα I της Σ ικανοποιούν το Ψ.

Κεφάλαιο 2

Η διασύνδεση DIG

Η υλοποίηση εφαρμογών για συστήματα Περιγραφικής Λογικής (ΠΛ), συχνά φέρνει τον προγραμματιστή αντιμέτωπο με ένα συναρτησιακό interface, οριζόμενο από μια Lisp τύπου σύνταξη μικρότερης ή μεγαλύτερης πολυπλοκότητας, ανάλογα με το υπό υλοποίηση σύστημα.

Η γλώσσα Lisp εξακολουθεί να είναι η πιο διαδεδομένη για την ανάπτυξη και υλοποίηση συστημάτων Περιγραφικής Λογικής. Εξάλλου η πρώτη προσπάθεια καθορισμού ενός προτύπου KRSS [12] για τα συστήματα ΠΛ το 1993 είχε σαν βάση μια σύνταξη κατά πολύ όμοια με την Lisp. Η εξάρτηση αυτή, με δεδομένο το ενδιαφέρον που παρουσιάζει σήμερα η ανάπτυξη υπηρεσιών στο World Wide Web, αποτελεί ένα σημαντικό περιορισμό για τους προγραμματιστές εφαρμογών συστημάτων Περιγραφικών Λογικών, που χρησιμοποιούν άλλες γλώσσες προγραμματισμού όπως η Java ή είναι συνηθισμένοι στην ανάπτυξη λογισμικού με κατανεμημένη αρχιτεκτονική.

Σε ένα κατανεμημένο περιβάλλον, ένα σύστημα συλλογιστικής ΠΛ θα πρέπει να θεωρηθεί ως ένα απομονωμένο κομμάτι του οποίου οι λεπτομέρειες υλοποίησης ή ακόμη και η ακριβής θέση εκτέλεσης είναι κρυμμένες από την ίδια την εφαρμογή. Αυτή η προσέγγιση έχει πολλά πλεονεκτήματα :

- Είναι δυνατή η χρήση οποιασδήποτε γλώσσας προγραμματισμού.
- Είναι δυνατό να οριστεί μια κοινή διασύνδεση προγραμματισμού API μεταξύ των εφαρμογών και του συστήματος ΠΛ.
- Μπορεί να οριστεί ένας μηχανισμός τοπικής ή απομακρυσμένης επικοινωνίας, ανάμεσα στις εφαρμογές και το σύστημα ΠΛ.
- Εναλλακτικά μπορεί να χρησιμοποιηθούν διαφορετικά συστήματα ΠΛ χωρίς καμία μεταβολή στην ίδια την εφαρμογή.

Στο παρελθόν την προσέγγιση αυτή ακολούθησε η ανάπτυξη του CORBA-FaCT συστήματος [9]. Η δημιουργία ενός CORBA[1] inteface για τον reasoner FaCT επέτρεψε την επικοινωνία του με εφαρμογές όπως το OilEd[8] και το ICOM[1]. Το CORBA-FaCT παρουσίαζε κάποιες αδυναμίες· κληρονομώντας από τον Fact reasoner, στον οποίο είναι βασισμένο, την έλλειψη υποστήριξης σωμάτων ορολογίας (A-box), την αδυναμία υποστήριξης του στέρεου χώρου (concrete domain), ενώ απουσίαζε η δυνατότητα αναγνώρισης του reasoner και των υπηρεσιών διαθέσιμων υπηρεσιών συλλογιστικής του, στην εφαρμογή- πελάτη. Παρόμοια προσπάθεια δημιουργίας ενός

[1]Common Object Requesting Broker Architecture

ενδιάμεσου interface ανάμεσα στο σύστημα ΠΛ και στις εφαρμογές του έγινε για το σύστημα RACER, που προσέφερε ένα, βασισμένο σε sockets, interface. Η προσπάθεια βασίστηκε με επιτυχία πάνω σε μια πλήρη γλώσσα περιγραφής εννοιών, με αρνητική ωστόσο συνέπεια, τη μεταφορά μεγάλου προγραμματιστικού βάρους προς τη μεριά του προγραμματιστή των εφαρμογών.

Το Description Logic Implementation Group (DIG)[2] είναι ένα μικρό αυτοδιάθετο σύνολο από προγραμματιστές, που δημιουργήθηκε με σκοπό την ανταλλαγή εμπειριών υλοποίησης, αλλά και την προώθηση κοινών αρχιτεκτονικών ανάπτυξης συστημάτων ΠΛ. Μία από τις κύριες δραστηριότητες του DIG είναι η ανάπτυξη ενός καθιερωμένου (standard) τρόπου διασύνδεσης με συστήματα ΠΛ.

2.1 Γενικά

Στην αρχική του έκδοση το πρότυπο βασίζεται σε 3 βασικές υποθέσεις :

- Το πρότυπο δεν αναγνωρίζει πολλαπλές συνδέσεις από πελάτες .Υλοποιήσεις πολυ-νηματικές ,μπορεί να αναπτυχθούν, αλλά δεν υπάρχει καμία εγγύηση για την κατάσταση της βάσης γνώσης, όταν πολλοί πελάτες προσπαθούν ταυτόχρονα να ανανεώσουν ή να κάνουν ερωτήσεις επάνω σε αυτήν.

- Οι συνδέσεις στην μηχανή συλλογιστικής είναι πρακτικά stateless. Οι πελάτες δεν διαφέρουν για τον reasoner ο οποίος δεν κρατάει κανένα αρχείο επικοινωνίας και δεν εκτελεί κανένα έλεγχο concistency για το ποιος πελάτης προσθέτει ή κάνει αιτήσεις για πληροφορία. Συνεπώς ο πελάτης δεν μπορεί να γνωρίζει εάν ο reasoner έχει λάβει κάποια πρόσθετη σημασιολογία, όπως για παράδειγμα νέα αξιώματα, από την τελευταία του επικοινωνία.

- Δεν υπάρχει δυνατότητα ρητού αιτήματος από τον πελάτη προς τον reasoner για την ταξινόμηση classification των εννοιών. Ο reasoner αποφασίζει όποτε κρίνει αυτός απαραίτητο να δημιουργήσει την ταξινόμηση των εννοιών, για παράδειγμα μετά από κάθε αίτημα ενημέρωσης της βάσης γνώσης (TELL) ή όταν υπάρχει μια πτώση στην κίνηση των αιτημάτων.

Το πρότυπο ουσιαστικά αποτελείται από ένα XML schema [11] που περιγράφει :

- τις έννοιες μιας ΠΛ,

- τις λειτουργίες ενημέρωσης της βάσης γνώσης «TELL»,

- τις λειτουργίες ανάκτησης γνώσης από αυτήν «ASK»,

- τις απαντήσεις της μηχανής συλλογιστικής στα παραπάνω αιτήματα,

- τις λειτουργίες διαχείρισης των βάσεων γνώσης,

- την ταυτοποίηση και την παρουσίαση της διαθέσιμης εκφραστικότητας και συλλογιστικής ικανότητας στην απομακρυσμένη εφαρμογή.

[2] http://dl.kr.org/dig

Πρέπει να σημειωθεί ότι το πρότυπο δεν έχει σκοπό να ορίσει ένα σύστημα ΒΔ για βάσεις γνώσης (knowledge bases). Περιγράφει απλά ένα πρωτόκολλο για την σύνδεση εφαρμογών με τις υπηρεσίες συλλογιστικής ενός reasoner, γι αυτό υπάρχουν οι παραπάνω περιορισμοί, καθώς και η απουσία της δυνατότητας αφαίρεσης σημασιολογίας. Το XML schema όπως περιγράφεται σε αυτό το κεφάλαιο είναι διαθέσιμο στην ιστοσελίδα :

http://dl-web.man.ac.uk/dig/2003/02/dig.xsd

2.2 Πρωτόκολλο

Το DIG χρησιμοποιεί σαν υπόβαθρο το πρωτόκολλο επικοινωνίας HTTP[4], δανειζόμενο πολλά από άλλες πρωτοβουλίες όπως το SOAP[3] και το XML-RPC[4] που επίσης έχουν δημιουργήσει πρωτόκολλα μεταφοράς μηνυμάτων, με τη χρήση του XML προτύπου πάνω από το HTTP.

Η χρήση του HTTP επιτρέπει στους προγραμματιστές εφαρμογών και συστημάτων ΠΛ να χρησιμοποιούν διαθέσιμες βιβλιοθήκες ανάπτυξης. Το πρωτόκολλο επιθυμείται να είναι όσο τον δυνατό πιο ελαφρύ, αυτός είναι ο κύριος λόγος που δεν έχει επιλεχθεί το SOAP, εξάλλου ο σκοπός δεν είναι η μεταφορά αντικειμένων μέσα από το interface αλλά απλά μηνυμάτων κειμένου (strings). Η χρήση του πρωτοκόλλου HTTP ως υπόβαθρο δεν είναι δεσμευτική και στο μέλλον κάποιο άλλο πιο πλούσιο πρωτόκολλο μπορεί να επιλεχθεί, ωστόσο προς το παρόν η λειτουργικότητα που παρέχει είναι ικανοποιητική.

Αίτηση - Request Οι πελάτες επικοινωνούν με τον εξυπηρετητή (server) με τη χρήση HTTP POST αιτήσεων. Το κύριο σώμα μιας αίτησης πρέπει να είναι ένα μήνυμα κωδικοποιημένο σε XML μορφή που μεταφέρει το αντίστοιχο DIG αίτημα, όπως περιγράφεται στο XML σχήμα και θα παρουσιαστεί παρακάτω. Το Content-Type είναι text/xml και το Content-Length πρέπει να είναι σωστά ορισμένο. Ο server θα αναγνωρίσει από το βασικό στοιχείο (root element) του σώματος του μηνύματος, το είδος του αιτήματος, δηλαδή αν πρόκειται για tell , ask, αίτημα αναγνώρισης των δυνατοτήτων του reasoner ή διαχείρισης μιας ΒΓ.

Απάντηση - Response Εάν δεν προκύψει κάποιο λάθος, ο server απαντάει με το 200 OK. Όπως στις αιτήσεις το Content-Type είναι text/xml και το Content-Length πρέπει να είναι σωστά ορισμένο. Το σώμα της απάντησης είναι ένα XML κωδικοποιημένο μήνυμα που μεταφέρει την DIG απάντηση, όπως θα περιγραφεί στη συνέχεια.

Επίμονες Συνδέσεις - Persistent Connections Το πρωτόκολλο HTTP επιτρέπει την χρήση persistent connections, επιτρέποντας έτσι τις σειριοποίηση των αιτήσεων. Με αυτό τον τρόπο είναι δυνατή η χρήση μίας μόνο TCP σύνδεσης για πολλαπλές αιτήσεις.

[3]http://w3.org/TR/SOAP
[4]http://www.xmlrpc.com

2.3 Αναγνώριση δυνατοτήτων Reasoner Identification

Ιδανικά, θα περιμέναμε από κάθε reasoner που υλοποιεί το DIG Interface να υποστηρίζει το σύνολο της γλώσσας περιγραφής εννοιών και την tell/ask λειτουργικότητα. Στην πραγματικότητα αυτό είναι απίθανο να συμβεί, καθώς κάποιοι reasoners μπορεί να μην υποστηρίζουν ένα μέρος της εκφραστικότητας, όπως για παράδειγμα το στέρεο χώρο (concrete domains).

Ο reasoner λοιπόν, μαζί με τις γενικές πληροφορίες αναγνώρισης του, οφείλει να παρέχει λεπτομέρειες για την γλώσσα περιγραφής που υποστηρίζει. Με αυτό τον τρόπο οι εφαρμογές - πελάτες μπορούν να αναγνωρίζουν και να επιλέξουν τον reasoner που θα χρησιμοποιήσουν σύμφωνα με την εκφραστική δυνατότητα και τις υπηρεσίες συλλογιστικής που προσφέρονται.

Στην τωρινή έκδοση του DIG η ικανότητα πληροφόρησης για το είδος των παρεχόμενων υπηρεσιών βρίσκεται σε βασικό στάδιο. Ουσιαστικά αποτελείται από μια λίστα με τους τελεστές περιγραφής (κατασκευαστές) εννοιών, τους Tell ισχυρισμούς και τις Ask ερωτήσεις που υποστηρίζονται από τον reasoner. Στο μέλλον αυτό πρόκειται να επεκταθεί με την παρουσίαση των περιορισμών που υφίστανται στην χρήση των διαθέσιμων υπηρεσιών.

Ένα αίτημα αναγνώρισης προς τον reasoner περιέχει ένα μόνο $< getIdentifier >$ στοιχείο, όπως παρουσιάζεται στο Σχήμα 2.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<getidentifier
    xmlns="http://dl.kr.org/dig/2003/02/lang"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
    http://dl-web.man.ac.uk/dig/2003/02/dig.xsd">
<getIdentifier/>
```

Σχήμα 2.1: *Αίτηση Αναγνώρισης Ταυτότητας προς Απομακρυσμένη Μηχανή Συλλογιστικής*

Η απάντηση σε ένα Identifier αίτημα αποτελείται επίσης από ένα μόνο $< identifier >$ στοιχείο, το οποίο περιέχει :

- Το όνομα και την έκδοση του reasoner σε μια συμβολοσειρά .
- Ένα $< supports >$ στοιχείο που περιγράφει την γλώσσα περιγραφής εννοιών και τις υπηρεσίες του reasoner.

Στο Σχήμα 2.2 παρουσιάζεται μια ενδεικτική απάντηση σε ένα αίτημα αναγνώρισης προς ενός reasoner, ο οποίος υποστηρίζει ένα μικρό σύνολο εκφραστικότητας. Συγκεκριμένα υποστηρίζονται οι πρωτογενείς έννοιες, η σύζευξη, η διάζευξη, η άρνηση, ο υπαρξιακός περιορισμός, τα αξιώματα υπαγωγής εννοιών και ένα μόνο ερώτημα, αυτό της ικανοποιησιμότητας.

```
<?xml version="1.0" encoding="UTF-8"?>
<identifier
  xmlns="http://dl.kr.org/dig/2003/02/lang"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
  http://dl-web.man.ac.uk/dig/2003/02/dig.xsd"
  name="FaCT" version="13.0.4" message="FaCT reasoner ↩
     running on
potato.cs.man.ac.uk (130.88.196.149)">
        <supports>
                <language>
                        <and/>
                        <or/>
                        <not/>
                        <some/>
                </language>
                <tell>
                        <impliesc/>
                </tell>
                <ask>
                        <satisfiable/>
                </ask>
        </supports>
</identifier>
```

Σχήμα 2.2: *Παρουσίαση Στοιχείων Ταυτότητας και Συλλογιστικών Δυνατοτήτων από ένα reasoner*

2.4 Διαχείριση Βάσης Γνώσης Knowledge Base management

Ένας DIG reasoner μπορεί να διαχειρίζεται πολλαπλές Βάσεις Γνώσης (ΒΓ). Η ταυτοποίηση κάθε βάσης γνώσης γίνεται με τη χρήση των URIs[5]. Σε κάθε αίτημα δημιουργίας μίας νέας βάσης γνώσης, ο reasoner, εάν αυτό είναι επιτυχές, επιστρέφει ένα URI, με το οποίο ο πελάτης μπορεί να απευθύνει τα TELL και ASK αιτήματα του.

Το URI της Βάσης Γνώσης είναι μοναδικό και έτσι έχει ισχύ μόνο για τον συγκεκριμένο reasoner. Η χρήση των URI επίσης μετριάζει κάποιους περιορισμούς σχετικά με την υποστήριξη πολλαπλών πελατών. Εάν ένας πελάτης δεν κοινοποιήσει το URI μίας βάσης γνώσης, μπορεί να είναι σίγουρος ότι είναι ο μοναδικός που αλληλεπιδρά με αυτή τη ΒΓ. Διαφορετικοί πελάτες μπορούν να διαχειρίζονται την ίδια ΒΓ εάν απλά μοιραστούν μεταξύ τους το αντίστοιχο URI, ωστόσο στη συνέχεια είναι ευθύνη των πελατών να οργανώσουν και συντονίσουν τις εργασίες τους στη ΒΓ.

Υποστηρίζονται δύο αιτήματα διαχείρισης βάσης γνώσης:

1. Αίτημα για δημιουργία καινούριας βάσης γνώσης (Σχήμα 2.3). Εάν αυτό είναι

[5]Uniform Resource Identifier

επιτυχές, η απάντηση θα είναι το URI που ανέθεσε ο reasoner στην ΒΓ (Σχήμα 2.4).

2. Αίτημα για αποδέσμευση μίας βάσης γνώσης. Στην περίπτωση αυτή ο πελάτης δίνει το URI της ΒΓ που θέλει να αποδεσμεύσει (Σχήματα 2.5 και 2.6).

Στο μέλλον πιο πλούσιοι μηχανισμοί ταυτοποίησης θα είναι απαραίτητοι, όσο η χρήση των συστημάτων ΠΛ γίνεται πιο διαδεδομένη.

```
<?xml version="1.0" encoding="UTF-8"?>
<newKB xmlns="http://dl.kr.org/dig/2003/02/lang"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
http://dl-web.man.ac.uk/dig/2003/02/dig.xsd" />
```

Σχήμα 2.3: *Αίτημα για Δημιουργίας Νέας Βάσης Γνώσης*

```
<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://dl.kr.org/dig/2003/02/lang"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
    http://dl-web.man.ac.uk/dig/2003/02/dig.xsd">
    <kb uri="urn:dig:fire:kb1"/>
</response>
```

Σχήμα 2.4: *Απάντηση στην αίτηση δημιουργίας νέας βάσης γνώσης*

2.5 Γλώσσα περιγραφής εννοιών Concept Language

Η γλώσσα περιγραφής εννοιών του DIG είναι βασισμένη στην περιγραφική λογική $SHOIQD_n^-$, η οποία υποστηρίζει τους κατασκευαστές εννοιών ($\sqcap, \sqcup, \neg$), τον καθολικό

```
<?xml version="1.0" encoding="UTF-8"?>
<releaseKB xmlns="http://dl.kr.org/dig/2003/02/lang"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
http://dl-web.man.ac.uk/dig/2003/02/dig.xsd"
uri="urn:dig:fire:kb2" />
```

Σχήμα 2.5: *Αίτηση Αποδέσμευσης Βάσης Γνώσης*

```
<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://dl.kr.org/dig/2003/02/lang"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
    http://dl-web.man.ac.uk/dig/2003/02/dig.xsd">
    <ok/>
</response>
```

Σχήμα 2.6: *Απάντηση στην Αίτηση Αποδέσμευσης Βάσης Γνώσης*

(∀) και τον περιορισμένο υπαρξιακό περιορισμό (∃), ιεραρχία ρόλων, αντίστροφους ρόλους, περιορισμούς πληθυκότητας, τον κατασκευαστή one-of (O) και το στέρεο χώρο (concrete domains). Η $SHOIQD_n^-$ έχει επιλεγεί γιατί η ευρεία εκφραστική της δυνατότητα επιτρέπει την συλλογιστική επάνω στην γλώσσα DAML+OIL και την OWL-DL, προτυποποιημένη γλώσσα οντολογίας για τον Σημασιολογικό Ιστό.

Η τωρινή έκδοση 1.1 του DIG παρέχει περιορισμένη υποστήριξη για τον στέρεο χώρο. Συγκεκριμένα υποστηρίζονται οι ακέραιοι και οι συμβολοσειρές, οι τελεστές εννοιών για το ελάχιστο, το μέγιστο, την ισότητα και το εύρος τιμής. Δεν υποστηρίζονται επίσης οι γραμμικές ανισότητες. Το εύρος τιμής μπορεί να εισαχθεί σαν χαρακτηριστικό (attribute) μέσα από έναν ισχυρισμό όπως οι rangeint και rangestring.

Οι κατασκευαστές εννοιών της γλώσσας στην οποία βασίζεται το DIG, φαίνονται στον Πίνακα 2.1.

2.6 Σύνταξη Tell

Ένα Tell αίτημα περιέχει στο κύριο σώμα του ένα tells στοιχείο (xml element), το οποίο με τη σειρά του αποτελείται από μικρότερες tells δηλώσεις σε μικρότερα elements. Τα αιτήματα Tell είναι monotonic, δηλαδή η πληροφορία που εισάγεται στην βάση γνώσης , δεν μπορεί να διορθωθεί ή να σβηστεί . Η μόνη επιλογή είναι η αποδέσμευση της βάσης γνώσης και η δημιουργία νέας από τη αρχή. Η σειρά των tell στοιχείων (elements) μέσα στο σώμα του κυρίως αιτήματος δεν έχει σημασία. Κάθε Tell αίτημα γίνεται με αναφορά σε μία συγκεκριμένη βάση γνώσης συμπεριλαμβάνοντας το αντίστοιχο URI σαν χαρακτηριστικό attribute του. Στο Σχήμα 2.7 παρουσιάζεται ένα Tell αίτημα ,όπου ορίζονται οι έννοιες vehicle, person και η έννοια driver, ως person who drives vehicle.

Η απάντηση της μηχανής συλλογιστικής σε ένα Tell αίτημα μιας εφαρμογής - πελάτη είναι ένα XML μήνυμα απάντησης (response) που περιέχει ένα OK στοιχείο, εάν όλες οι δηλώσεις του Tell έχουν εισαχθεί στη βάση γνώσης επιτυχώς. Εάν αυτό δεν συνέβη, τότε θα περιέχει ένα error στοιχείο, μαζί με ένα προαιρετικό κωδικό λάθους, ή και μία λεπτομερή δικαιολόγηση. Επιπροσθέτως το OK μήνυμα μπορεί να περιέχει ένα σύνολο από ειδοποιήσεις (warnings) για τα tells στοιχεία που έγιναν δεκτά, όπως για παράδειγμα στο Σχήμα 2.8. Το σύνολο της Tell εκφραστικότητας που υποστηρίζεται από το σχήμα του DIG φαίνεται στον Πίνακα 2.2.

Primitive Concepts	<top/> <bottom/> <catom name="CN"/>
Boolean Concepts	<and>E1...En</and> <or>E1...En</or> <not>E1</not>
Property Restrictions	<some>R E </some> <all>R E</all> <atmost num="n" >R E </atmost> <atleast num="n" >R E </atleast> <iset>I1...In</iset>
Concrete Domain Expressions	<defined>A</defined> <stringmin val="s">A</stringmin> <stringmax val="s">A</stringmax> <stringequals val="s">A</stringequals> <stringrange min="s" max="t">A</stringrange> <intmin val="i">A</intmin> <intmax val="i">A</intmax> <intequals val="i">A</intequals> <intrange min="i" max="j">A</intrange>
Role Expressions	<ratom name="CN"/> <feature name="CN"/> <inverse>R</inverse> <attribute name="CN"/> <chain>F1..Fn</chain>
Individuals	<individual name="CN"/>

Πίνακας 2.1: *Γλώσσα Περιγραφής Εννοιών του DIG*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tells
  xmlns="http://dl.kr.org/dig/2003/02/lang"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
  http://dl-web.man.ac.uk/dig/2003/02/dig.xsd"
  uri="urn:uuid:abcdefgh-1234-1234-12345689ab">
    <defconcept name="driver"/>
    <equalc>
        <catom name="driver"/>
        <and>
            <catom name="person"/>
            <some>
                <ratom name="drives"/>
                <catom name="vehicle"/>
            </some>
        </and>
    </equalc>
    <defconcept name="person"/>
    <defconcept name="vehicle"/>
    <defrole name="drives"/>
</tells>
```

Σχήμα 2.7: *Ενδεικτικό Tells Αίτημα προς τον Reasoner*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<response
  xmlns="http://dl.kr.org/dig/2003/02/lang"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
  http://dl-web.man.ac.uk/dig/2003/02/dig.xsd">
    <ok>
      <warning message="Undeclared Name" code="99">Class ↩
         Cat used
       but not declared</warning>
    </ok>
</response>
```

Σχήμα 2.8: *Δείγμα Απάντησης του Reasoner σε ένα Tells Αίτημα*

Primitive Concept Introduction	<defconcept name="C N"/> <defrole name="C N"/> <deffeature name="C N"/> <defattribute name="C N"/> <defindividual name="C N"/>
Concept Axioms	<impliesc>C1 C2</impliesc> <equalc>C1 C2</equalc> <disjoint>C1...Cn</disjoint>
Role Axioms	<impliesr>R1 R2</impliesr> <equalrr>R1 R2</equalr> <domain>R E</domain> <range>R E</range> <rangeint>R</rangeint> <rangestring>R</rangestring> <transitive>R</transitive> <functional>R</functional>
Individual Axioms	<instanceof>I C</instanceof> <related>I1 R I2</related> <value>I A V</value>

Πίνακας 2.2: *Περιγραφή της Σύνταξης Tell Αιτήματος*

2.7 Σύνταξη Ask

Ένα Ask αίτημα ορίζεται σε ένα γενικό asks xml element, το οποίο περιέχει ερωτήματα προς την μηχανή συλλογιστικής σε μικρότερα ask στοιχεία (elements). Κάθε ask element περιέχει ένα στοιχείο ταυτοποίησης (attribute id), με το οποίο αναγνωρίζεται μοναδικά στα πλαίσια του γενικότερου Ask αιτήματος. Με αυτόν το τρόπο πολλαπλά ερωτήματα μπορούν να εισάγονται στο σώμα ενός μόνο Ask αιτήματος, επιτρέποντας έτσι στην μηχανή συλλογιστικής να βελτιστοποιεί την σειρά με την οποία θα επεξεργαστεί τα ερωτήματα που δέχεται· έτσι βελτιστοποιείται ο χρόνο δημιουργίας της απάντησης. Επιπλέον κάθε asks element περιέχει ένα ακόμη χαρακτηριστικό (attribute) με το URI της βάσης γνώσης στην οποία αναφέρεται. Το σύνολο των επιτρεπόμενων ερωτήσεων φαίνεται στον Πίνακα 2.3. Στο Σχήμα 2.9 παρουσιάζεται ένα παράδειγμα αιτήματος Ask, που αποτελείται από τρία ερωτήματα. Το πρώτο ερώτημα αναφέρεται στην ικανοποιησιμότητα της έννοιας «Vehicle», το δεύτερο στις έννοιες που υπάγονται στην έννοια «person drives vehicle», δηλαδή όλες τις έννοιες που υπάγονται στην έννοια «driver» και το τρίτο που ζητάει όλες τις έννοιες των οποίων στιγμιότυπο αποτελεί το αντικείμενο «John Smith».

2.8 Σύνταξη Απάντησης

Το XML σχήμα του DIG περιγράφει την μορφή της απάντησης του server στα ASK αιτήματα που έχει δεχτεί. Η απάντηση συνίσταται από ένα κυρίως στοιχειό απαντήσεων (responses element), το οποίο με τη σειρά του περιέχει ένα αριθμό από απαντήσεις , μία για κάθε ερώτημα του αντίστοιχου ASK αιτήματος, σε μικρότερα elements. Κάθε συγκεκριμένο element απάντησης φέρει το χαρακτηριστικό ταυτοποίησης (attribute id)

```
<?xml version="1.0"?>
<asks
  xmlns="http://dl.kr.org/dig/2003/02/lang">
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang"
  http://dl-web.man.ac.uk/dig/2003/02/dig.xsd"
  uri="urn:uuid:abcdefgh-1234-1234-12345689ab">
  <satisfiable id="q1">
    <catom name="Vehicle"/>
  </satisfiable>
  <descendants id="q2">
    <and>
      <catom name="person"/>
      <some>
        <ratom name="drives"/>
        <catom name="vehicle"/>
      </some>
    </and>
  </descendants>
  <types id="q3">
    <individual name="John Smith"></individual>
  </types>
</asks>
```

Σχήμα 2.9: *Δείγμα ενος Αιτήματος Ερωτήσεων Asks προς τον Reasoner*

Primitive Concept Retrieval	<allConceptNames/> <allRoleNames/> <allIndividuals/>
Satisfiability	<satisfiable>C</satisfiable> <subsumes>C1 C2</subsumes> <disjoint>C1 C2</disjoint>
Concept Hierarchy	<parents>C</parents> <children>C</children> <ancestors>C</ancestors> <descendants>C</descendants> <equivalents>C</equivalents>
Role Hierarchy	<rparents>R</rparents> <rchildren>R</rchildren> <rancestors>R</rancestors> <rdescendants>R</rdescendants>
Individual Queries	<instances>C</instances> <types>I</types> <instance>I C</instance> <roleFillers>I R</roleFillers> <relatedIndividuals>R</relatedIndividuals> <toldValues>I A </toldValues>

Πίνακας 2.3: *Περιγραφή της Σύνταξης ενός Ask Αιτήματος*

της ερώτησης στην οποία αντιστοιχεί. Στον Πίνακα 2.4 φαίνονται οι μορφές σύνταξης της απάντησης και στον Πίνακα 2.5 οι πιθανοί κωδικοί λάθους.

Οι απαντήσεις σε ερωτήσεις (queries), όπως για παράδειγμα η αναζήτηση όλων των εννοιών στις οποίες υπάγεται μια έννοια, επιστρέφουν ένα σύνολο (set) που αποτελείται από μικρότερα σύνολα (set) συνώνυμων εννοιών. Στο Σχήμα 2.10 φαίνεται ένα παράδειγμα απάντησης σε ένα αίτημα τριών ερωτημάτων. Η απάντηση στο πρώτο είναι τύπου boolean, ενώ στο δεύτερο αποτελείται από μια συλλογή εννοιών, που συμπεριλαμβάνει δύο συνώνυμες έννοιες.

2.9 Κατανομή Υπηρεσιών Interface Granularity

Ένα σημείο στο οποίο αξίζει να αναφερθούμε είναι η λεπτομέρεια, που επιτυγχάνει η πρότυπη διασύνδεση DIG interface στον καταμερισμό των υπηρεσιών που προσφέρει. Έντονη κριτική είχε λάβει, στο παρελθόν, το CORBA-Fact interface, καθώς για τη δημιουργία μίας ιεραρχίας εννοιών -ένα ερώτημα που γίνεται πολύ συχνά στα συστήματα ΠΛ - η εφαρμογή- πελάτης αναγκαζόταν να πραγματοποιήσει πολλές αιτήσεις στον server. Αυτό είχε ως άμεση συνέπεια την πρόκληση υπερβολικού φόρτου συλλογιστικής εργασίας στον server.

Το πρόβλημα αυτό εξομαλύνεται με την υποστήριξη πολλαπλών ερωτημάτων σε μία μόνο σύνδεση. Με την λεπτομερή κατανομή των παρεχόμενων υπηρεσιών, μέσα από την διασύνδεση DIG η εφαρμογή- πελάτης αρκεί να κάνει μόνο δύο αιτήματα για τη δημιουργία της ιεραρχίας των εννοιών classification hierarchy. Για παράδειγμα με τη χρήση ενός αιτήματος για τις έννοιες που υπάγονται σε μία έννοια $< descendants >$

Error	<error/>
Boolean	<true/> <false/>
Concept Set	<conceptSet> <synonyms>S11...S1N</synonyms> <synonyms>SN1...SNN</synonyms> </conceptSet>
Role Set	<roleSet> <synonyms>R11...R1N</synonyms> <synonyms>RN1...RNN</synonyms> </roleSet>
Individual Set	<individualSet>I11...I1N</individualSet>
Individual Pair Set	<individualPairSet> <individualPair>I11...I1N</individualPair> <individualPair>IN1...INN</individualPair> </individualPairSet>
Values	<sval>s</sval> <ival>i</ival>

Πίνακας 2.4: *Σύνταξη Απάντησης (Response)*

```
<?xml version="1.0"?>
<responses
  xmlns="http://dl.kr.org/dig/2003/02/lang">
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://dl.kr.org/dig/2003/02/lang
  http://dl-web.man.ac.uk/dig/2003/02/dig.xsd">
  <error code="99" id="q3" message="No Such Individual">↩
    The individual
named does not exist in the knowledge base.</error>
  <true id="q1"/>
  <conceptset id="q2">
    <synonyms>
      <catom name="bus driver"/>
      <catom name="passenger service vehicle operator"/>
    </synonyms>
    <synonyms>
      <catom name="car driver"/>
    </synonyms>
  </conceptset>
</responses>
```

Σχήμα 2.10: *Δείγμα Απάντησης σε Asks*

Class	**Code**	**Explanation**
General Errors	100	General Unspecified Error
	101	Unknown Request
	102	Malformed Request (XML error)
	103	Unsupported Operation
KB Errors	201	Cannot create new knowledge bases
	202	Malformed KB URIs
	203	Unknown or stale KB URI
	204	KB Release Error
	205	Missing URI
Tell Errors	300	General Tell Error
	301	Unsupported Tell Operation
	302	Unknown Tell Operation
Ask Errors	400	General Ask Error
	401	Unsupported Ask Operation
	402	Unknown Ask Operation

Πίνακας 2.5: *Κωδικοί Λάθους*

ή ενός αιτήματος για όλες τις διαθέσιμες έννοιες $< allConceptNames >$, ακολουθούμενο από ένα ακόμη αίτημα καθορισμού των των εννοιών που υπάγονται σε αυτές. Η πληροφορία αυτή είναι αρκετή για τη δημιουργία της ιεραρχίας από την εφαρμογή, χωρίς να χρειάζεται περαιτέρω επικοινωνία με τον reasoner.

Κεφάλαιο 3

Επέκταση της διασύνδεσης DIG για ασαφείς περιγραφικές λογικές

Η επιτυχία με την οποία η διασύνδεση DIG παρέχει τη δυνατότητα χρήσης εναλλακτικών μηχανών συλλογιστικής στις εφαρμογές συστημάτων ΠΛ, οδήγησε στην ευρεία αποδοχή της από την κοινότητα των ΠΛ. Παράλληλα το συνεχώς αυξανόμενο ενδιαφέρον που παρουσιάζει η ανάπτυξη των συστημάτων ασαφών περιγραφικών λογικών καθιστά εμφανή την ανάγκη επέκτασης της διασύνδεσης, για την υποστήριξη των ασαφών ΠΛ.

3.1 Επέκταση της Σύνταξης

Ο ορισμός της σύνταξης μιας ασαφούς ΠΛ,όπως είδαμε στο 1ο κεφάλαιο δεν παρουσιάζει πολλές διαφορές από την σύνταξη της αντίστοιχης κλασσικής ΠΛ. Συγκεκριμένα, για μια ασαφή ΠΛ αρχικά ορίζουμε ένα αλφάβητο από ατομικές ασαφείς έννοιες (C), ατομικούς ασαφείς ρόλους (R) και άτομα (I). Η σύνταξη των ασαφών εννοιών και ρόλων, ορίζεται με τον ίδιο ακριβώς τρόπο που ορίζεται στην αντίστοιχη κλασσική ΠΛ. Κάτι τέτοιο δεν ισχύει και για την σημασιολογία τους. Οι ασαφείς επεκτάσεις που επιβάλουμε στην κλασσική ΠΛ αναφέρονται στο βαθμό συμμετοχής ενός ατόμου σε μια κλάση, δηλαδή στο επίπεδο των ισχυρισμών, και όχι στο επίπεδο των κλάσεων και των ρόλων.

Η διασύνδεση DIG θέλει να λειτουργεί σαν ενδιάμεσο στρώμα μεταφοράς της περιγραφής των εννοιών από την εφαρμογή στον reasoner και αντίστροφα. Η υποκείμενη σημασιολογία απλά μεταφέρεται παθητικά και δεν αποτελεί πεδίο επεξεργασίας από την διασύνδεση. Αυτό που ενδιαφέρει είναι η σημασιολογία να μπορεί να εκφραστεί από τη διαθέσιμη περιφραστική ικανότητα των εννοιών. Το DIG διαθέτει ευρεία εκφραστική ικανότητα, καθώς βασίζεται στην πολύ εκφραστική ΠΛ $SHOIQD_n^-$. Συνεπώς δεν απαιτείται ουδεμία μετατροπή στον τρόπο σύνταξης των εννοιών και των ρόλων για την υποστήριξη των αντίστοιχων ασαφών εννοιών και ρόλων.

Οι τροποποιήσεις, που απαιτούνται, λοιπόν για την υποστήριξη των ασαφών ΠΛ, τοποθετούνται στο πλαίσιο του σώματος ισχυρισμών. Με την επέκταση της εκφραστικότητας της διασύνδεσης DIG για την περιγραφή των ισχυρισμών ασαφών εννοιών, η διασύνδεση θα είναι απολύτως ικανή να περιγράψει τις σημασιολογικές που της αναθέτουμε, είτε είναι ασαφείς είτε κλασσικές.

3.2 Σαφείς Ατομικοί Ισχυρισμοί στο DIG

Η διασύνδεση DIG όπως φαίνεται στον Πίνακα 3.2 υποστηρίζει τρία είδη ατομικών ισχυρισμών, με ιδιαίτερο ενδιαφέρον θα ασχοληθούμε με τους δύο πρώτους .

Ατομικός Ισχυρισμός	Περιγραφή
<instanceof>I C</instanceof>	το άτομο I είναι στιγμιότυπο της έννοιας C
<related>I1 R I2</related>	το άτομο I1 σχετίζεται με το άτομο I2 μέσω του ρόλου R
<value>I A V</value>	το άτομο I παρουσιάζει το attribute A με διακριτή τιμή V

Πίνακας 3.1: *Ατομικοί Ισχυρισμοί στο DIG*

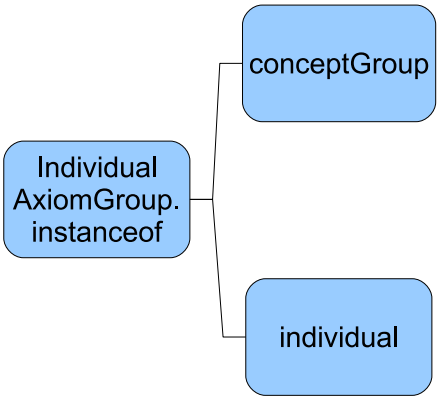
3.2.1 Ατομικός Ισχυρισμός Εννοιών

Στο Σχήμα 3.1 φαίνεται ένα παράδειγμα ατομικού ισχυρισμού εννοιών (instanceof), όπως συντάσσεται σύμφωνα με το DIG σχήμα, και δηλώνει ότι το άτομο eve αποτελεί στιγμιότυπο της έννοιας mother.

```
<instanceof>
    <individual name="eve"/>
    <catom name="mother"/>
<instanceof>
```

Σχήμα 3.1: *Παράδειγμα Ισχυρισμού Εννοιών (Instanceof) στο DIG*

Ένας ισχυρισμός εννοιών σύμφωνα με το XML DIG σχήμα περιγράφεται από ένα στοιχείο (xml element) με όνομα instanceof. Το στοιχείο αυτό διαθέτει δύο υποστοιχεία (sub-elements) όπου ορίζονται το άτομο (individual) ως στιγμιότυπο μίας κλάσης. Οι έννοιες (concepts) περιγράφονται από το σχήμα ως ένα ConceptGroup.Η περιγραφή του ασαφούς ισχυρισμού φαίνεται στα Σχήματα 3.2 και 3.3.



Σχήμα 3.2: *Σχηματική Περιγραφή Ατομικού Ισχυρισμού Εννοιών instanceof*

```
<xs:element name="instanceof">
        xs:complexType>
        <xs:sequence>
            <xs:element ref="individual"/>
            <xs:group ref="conceptGroup"/>
        </xs:sequence>
        /xs:complexType>
</xs:element>
```

Σχήμα 3.3: *XML Περιγραφή Ατομικού Ισχυρισμού Εννοιών instanceof*

3.2.2 Ατομικός Ισχυρισμός Ρόλων

Στο σχήμα 3.4 φαίνεται αντίστοιχα ένα παράδειγμα ατομικού ισχυρισμού ρόλων related που δηλώνει ότι το άτομο betty συνδέεται με το άτομο eve, μέσω του ρόλου has-child.

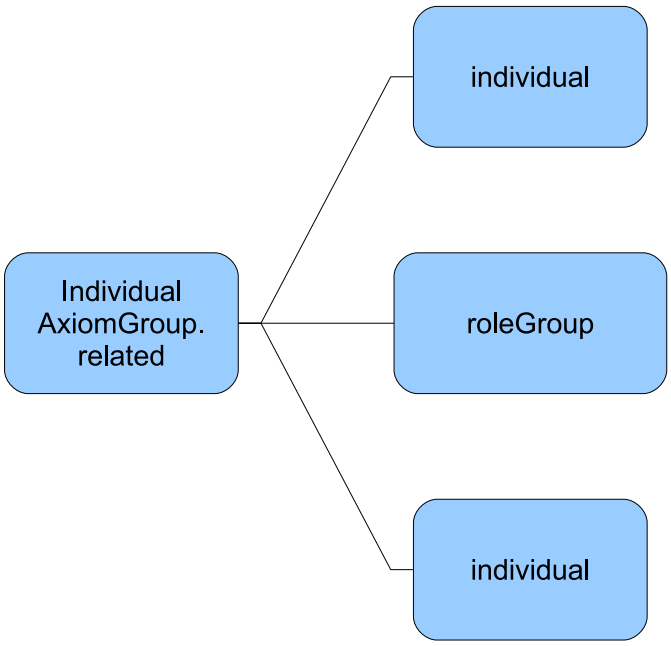
```
<related>
    <individual name="betty"/>
    <ratom name="has-child"/>
    <individual name="eve"/>
</related>
```

Σχήμα 3.4: *Παράδειγμα Ισχυρισμού Ρόλων (related) στο DIG*

'Ενας ισχυρισμός ρόλων σύμφωνα με το DIG XML σχήμα περιγράφεται από ένα στοιχείο (xml element) related. Το στοιχείο αυτό αποτελείται από δύο άτομα (individual), σε αντίστοιχα στοιχεία- παιδιά (xml sub-elements), και έναν ρόλο σε ένα RoleGroup XML υπο-στοιχεία με τον οποίο συνδέονται. Οι αντίστοιχες περιγραφές φαίνονται στα σχήματα 3.5 και 3.6. Τα ConceptGroup και RoleGroup περιγράφουν στο XML σχήμα τις έννοιες και τους ρόλους που υποστηρίζει το DIG αντίστοιχα.

```
<xs:element name="related">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="individual"/>
            <xs:group ref="roleGroup"/>
            <xs:element ref="individual"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Σχήμα 3.5: *XML Περιγραφή Ατομικού Ισχυρισμού Εννοιών (related)*



Σχήμα 3.6: *Σχηματική Περιγραφή Ατομικού Ισχυρισμού Εννοιών (related)*

3.3 Ασαφείς Ατομικοί Ισχυρισμοί

Ένα ασαφές σώμα ισχυρισμών (*Assertional Box ABox*) είναι ένα πεπερασμένο σύνολο από ασαφή αξιώματα ατόμων τα οποία αποκαλούνται ασαφής ισχυρισμοί. Ένας ασαφής ισχυρισμός (fuzzy assertion) είναι μια δήλωση της μορφής :

- $(a : C) \bowtie n$, όπου $\bowtie = (\geq, >, \leq, <), a \in I$ που δηλώνει ότι το άτομο a αποτελεί στιγμιότυπο της κλάσης C με δείκτη αβεβαιότητας μεγαλύτερο (ή μικρότερο)- ίσο από n.

- $((a, b) : R) \bowtie n$, με $a, b \in \mathbf{I}$, που δηλώνει ότι τα άτομα a,b συνδέονται με τον ρόλο R με δείκτη αβεβαιότητας μεγαλύτερο (ή μικρότερο)- ίσο από n.

3.3.1 Επέκταση Ατομικών Ισχυρισμών του DIG

Η επέκταση της εκφραστικής ικανότητας του DIG interface για την υποστήριξη των ασαφών ισχυρισμών, προϋποθέτει πρωτίστως την προσθήκη των συμβόλων ανισότητας και ενός αριθμού - δείκτη αβεβαιότητας , στην εκφραστική ικανότητα του XML σχήματος και ειδικότερα στην περιγραφή των ατομικών ισχυρισμών. Αρχικά λοιπόν επεκτείνουμε το XML σχήμα με τις αντίστοιχες περιγραφές fuzzysymbols και degrees για τα σύμβολα ανισότητας και τον δείκτη αβεβαιότητας. Πιο αναλυτικά ορίζουμε δύο νέους σύνθετους τύπους, όπως φαίνονται στο σχήμα 3.7, που αποτελούνται από ένα χαρακτηριστικό attribute με όνομα fuzzysymbol και degree αντίστοιχα και είναι τύπου string.

3.3.2 Ασαφής Ατομικός Ισχυρισμός Εννοιών

Στη συνέχεια ορίζουμε την περιγραφή ενός ασαφούς ατομικού ισχυρισμού εννοιών στο DIG σχήμα. Επεκτείνουμε την περιγραφή του instanceof στοιχείου (element) με την προσθήκη ενός συμβόλου fuzzysymbol και του βαθμού αβεβαιότητας degree

```
<xs:complexType name="fuzzysymbols">
    <xs:attribute name="fuzzysymbol" type="xs:string" use="↩
        "required" />
</xs:complexType>

<xs:complexType name="degrees">
    <xs:attribute name="degree" type="xs:string" use="↩
        required" />
</xs:complexType>
```

Σχήμα 3.7: *Ορισμός Συμβόλου Ανισότητας και Δείκτη αβεβαιότητας στο XML Σχήμα*

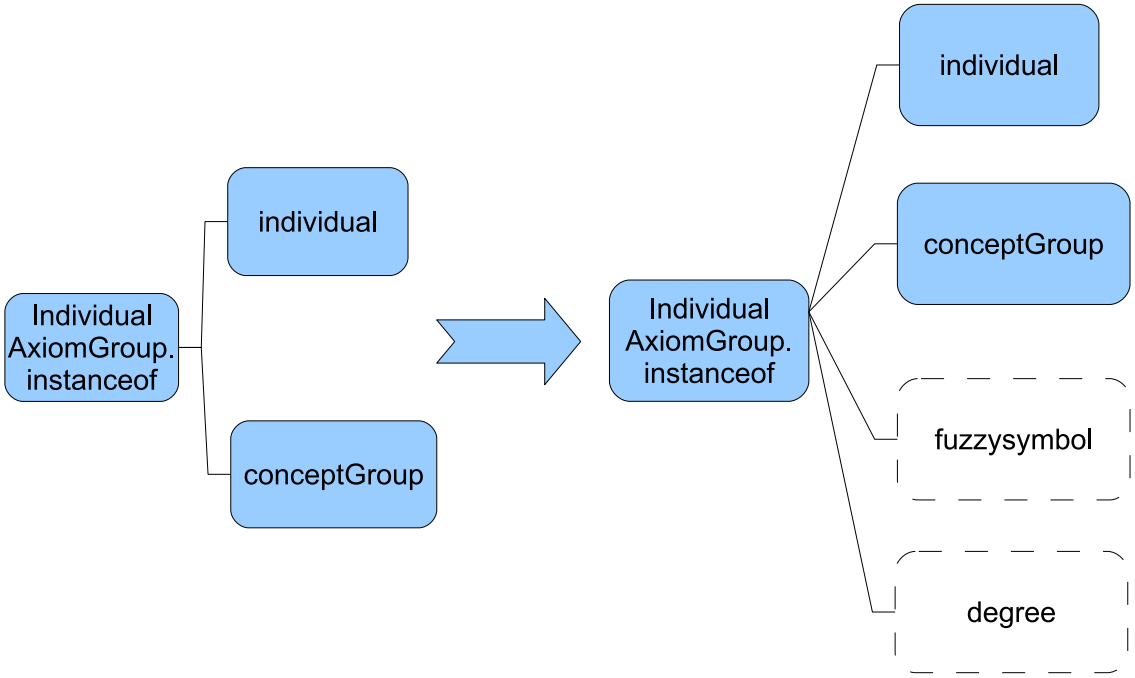
ως προαιρετικά στοιχεία του. Η επέκταση του ισχυρισμού στο XML σχήμα φαίνεται εποπτικά στο Σχήμα 3.8 και αναλυτικότερα στο Σχήμα 3.9. Στο Σχήμα 3.10 φαίνεται ένα παράδειγμα ασαφούς ισχυρισμού εννοιών (instanceof).

3.3.3 Ασαφής Ατομικός Ισχυρισμός Ρόλων

Ακόμη ορίζουμε την περιγραφή ενός ισχυρισμού ρόλων στο DIG σχήμα. Επεκτείνουμε την περιγραφή του related στοιχείου με την προσθήκη του συμβόλου fuzzysymbol και του βαθμού αβεβαιότητας degree ως προαιρετικά στοιχεία του .Η επέκταση του ισχυρισμού στο XML σχήμα φαίνεται εποπτικά στο Σχήμα 3.11 και αναλυτικότερα στο Σχήμα 3.12. Στο Σχήμα 3.13 φαίνεται ένα παράδειγμα ασαφούς ισχυρισμού ρόλων (related).

3.4 Σαφείς βάσεις γνώσης σε ασαφείς reasoners

Με την επέκταση αυτή το DIG interface υποστηρίζει τη χρήση ασαφών περιγραφικών λογικών. Επίσης μπορεί να υποστηριχτεί η μεταφορά και ο συλλογισμός επάνω σε μια σαφή οντολογία από έναν ασαφή reasoner. Καθώς κάθε σαφής ατομικός ισχυρισμός (χωρίς fuzzysymbol και degree), αναγνωρίζεται από τον ασαφή reasoner ως ισχυρισμός με βαθμό βεβαιότητας ίσο με ένα (degree = 1).



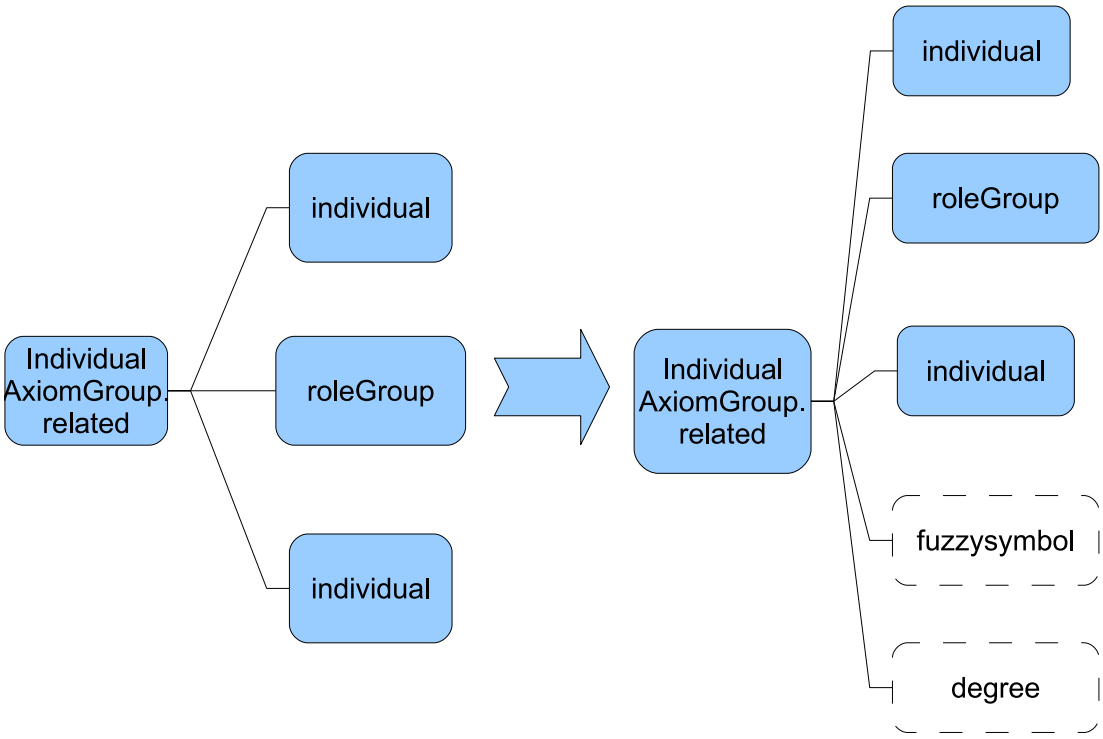
Σχήμα 3.8: *Ασαφής Επέκταση του Ισχυρισμού Εννοιών (instanceof)*

```
<xs:element name="instanceof">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="individual"/>
            <xs:group ref="conceptGroup"/>
            <xs:element name="fuzzysymbol" type="↩
                fuzzysymbols" minOccurs="0"/>
            <xs:element name="degree" type="degrees" ↩
                minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Σχήμα 3.9: *XML Περιγραφή Ασαφούς Ισχυρισμού Εννοιών (instanceof)*

```
<instanceof>
    <individual name="eve"/>
    <catom name="blonde"/>
    <fuzzysymbol fuzzysymbol=">="/>
    <degree degree="0.3"/>
</instanceof>
```

Σχήμα 3.10: *Παράδειγμα Ασαφούς Ισχυρισμού Εννοιών (instanceof) στο DIG*



Σχήμα 3.11: *Ασαφής Επέκταση του Ισχυρισμού Ρόλων (related)*

```
<xs:element name="related">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="individual"/>
            <xs:group ref="roleGroup"/>
            <xs:element ref="individual"/>
            <xs:element name="fuzzysymbol" type="↩
                fuzzysymbols" minOccurs="0"/>
            <xs:element name="degree" type="degrees" ↩
                minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Σχήμα 3.12: *XML Περιγραφή Ασαφούς Ισχυρισμού Ρόλων (related)*

```
<related>
    <individual name="frame"/>
    <ratom name="contains"/>
    <individual name="face"/>
    <fuzzysymbol fuzzysymbol=">="/>
     <degree degree="0.4"/>
</related>
```

Σχήμα 3.13: *Παράδειγμα Ασαφούς Ισχυρισμού Ρόλων (related) στο DIG*

Κεφάλαιο 4

Υλοποίηση

Η υλοποίηση της διεπαφής DIG έγινε στα πλαίσια της επέκτασης της μηχανής ασαφούς συλλογιστικής FiRE[1] που είναι βασισμένη την ΠΛ $f - SHIN$[3].

Το λογισμικό της μηχανής συλλογιστικής FiRE επεκτάθηκε προσφέροντας τις υπηρεσίες της σε απομακρυσμένες εφαρμογές μέσω του πρωτοκόλλου DIG ως server, αλλά και ως εφαρμογή - client σε απομακρυσμένους ασαφείς reasoners που μπορεί στο μέλλον να υποστηρίζουν το DIG.

4.1 Αρχιτεκτονική της υλοποίησης

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java, στην οποία είναι υλοποιημένο επίσης το λογισμικό της μηχανής FiRE.

Η διαχείριση και επεξεργασία των XML στοιχείων, στην υλοποίηση μας βασίζεται στη διασύνδεση DOM[2], που αποτελεί πρότυπο του World Wide Web Concortium. Το XML DOM υποστηρίζεται από το βασικό JAVA API, JAXP[3] για την επεξεργασία XML. Με αυτόν το τρόπο αποφεύγεται η χρήση εξωτερικών βιβλιοθηκών, και η υλοποίηση διατηρείται ανεξάρτητη και ελαφριά. Το DOM interface αναπαριστά ένα αρχείο (XML document), με μια δενδρική δομή· όπου κάθε στοιχείο (element) του XML αρχείου είναι ένας κόμβος του δέντρου.

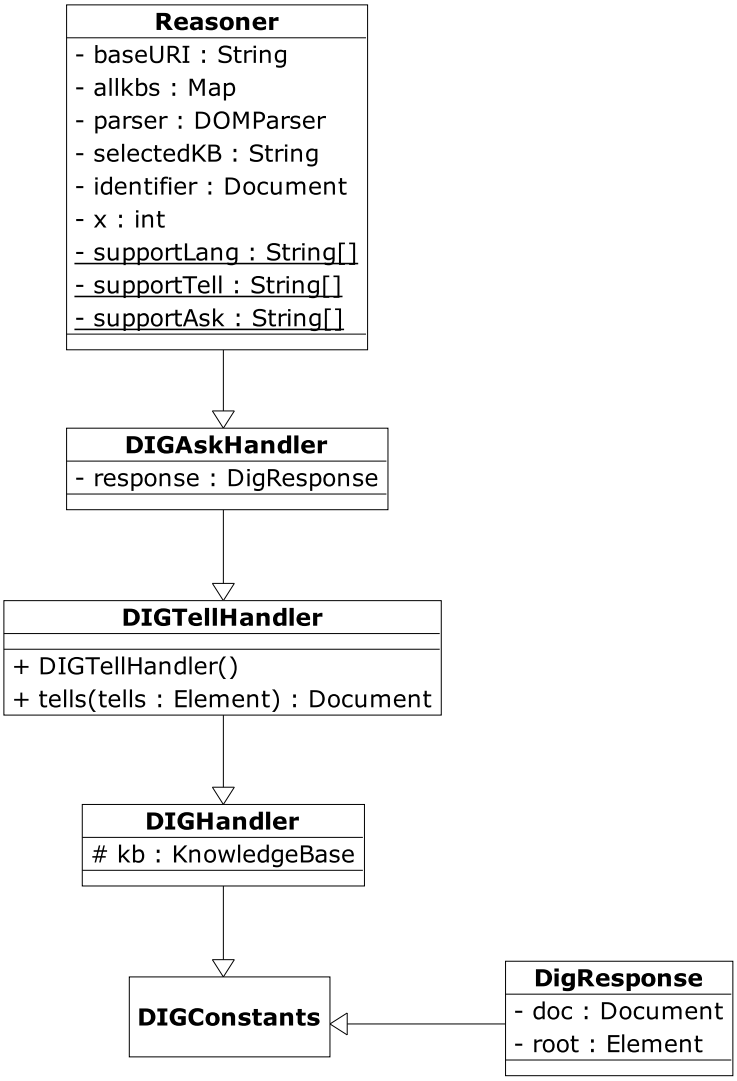
Ο κεντρικός άξονας της υλοποίησης μας αποτελείται από τις βασικές κλάσεις, όπως φαίνονται στην σειρά κληρονομικότητας του Σχήματος 4.1. Η κλάση DIGConstants περιέχει το σύνολο των όρων που χρησιμοποιούνται από τη διασύνδεση DIG με τη μορφή strings· ορίζοντας έτσι ένα λεξικό όρων για χρήση από τις κλάσεις που την κληρονομούν. Η κλάση DIGHandler, άμεση κληρονόμος της DIGConstants περιγράφει τον χειρισμό της γλώσσας αναπαράστασης εννοιών, όπως φαίνεται συγκεντρωτικά στον Πίνακα 2.1. Η κλάση αυτή επεκτείνεται από την κλάση DIGTellHandler, με την περιγραφή του χειρισμού της Tells εκφραστικότητας (Πίνακας 2.2) του DIG. Με τη σειρά της η τελευταία κληρονομείται από την κλάση DIGAskHandler, και επεκτείνεται με τον χειρισμό της σύνταξης Asks (Πίνακας 2.3), αλλά και τη δημιουργία της απάντησης (Reply) (Πίνακας 2.4), μέσω της κλάσης DigResponse.

Επέκταση των παραπάνω κλάσεων είναι η κλάση Reasoner, που αποτελεί την κύρια κλάση της υλοποίησης. Κληρονομεί τις προηγούμενες κλάσεις, διαθέτει δηλαδή την δυνατότητα επεξεργασίας της σύνταξης του DIG στο σύνολο της, χρησιμοποιώντας

[1]http://www.image.ece.ntua.gr/ nsimou/FiRE/

[2]http://www.w3.org/DOM/

[3]https://jaxp.dev.java.net/



Σχήμα 4.1: *UMLΔιάγραμμα Βασικών Κλάσεων*

ως βάση τον DOM parser, και επιπλέον διαχειρίζεται τις βάσεις γνώσεις (knowledge bases) σύμφωνα με τα αιτήματα του χρήστη, απαντώντας κατάλληλα σε αυτόν.

Τελικά ένα αντικείμενο της κλάσης Reasoner περιέχεται σε ένα αντικείμενο μίας κλάσης Server, η οποία υλοποιεί το HTTP interface, όπως αυτό παρέχεται από την βιβλιοθήκη org.mortbay.http και περιγράφει έναν μικρό, ευέλικτο HTTP Server.

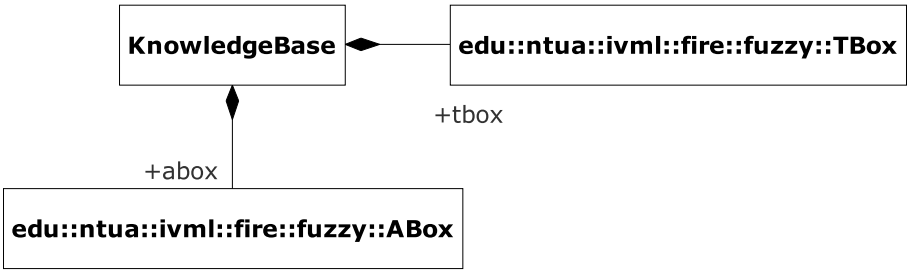
4.2 Περιγραφή βασικών κλάσεων

Στο σημείο αυτό περιγράφεται η γενικότερη μορφή και λειτουργία των βασικότερων κλάσεων της υλοποίησης του DIG interface.Αναλυτικότερη περιγραφή του συνόλου του πηγαίου κώδικα βρίσκεται στο Παράρτημα Α.

4.2.1 KnowledgeBase

Η κλάση KnowledgeBase περιγράφει μία βάση γνώσης και τις απαραίτητες μεθόδους επεξεργασίας της. Όπως φαίνεται στο Σχήμα 4.2 αποτελεί σύνθεση των κλάσεων TBox και ABox ,που ανήκουν στο πακέτο edu.ntua.ivml.fire.fuzzy της μηχανής συλλογιστι-

κήςFiRE. Περιέχει δομές αποθήκευσης και περιγράφει μεθόδους προσθήκης γνώσης με την μορφή εννοιών, ατόμων, ρόλων,αξιωμάτων ή ισχυρισμών στην βάση γνώσης, όπως για παράδειγμα η μέθοδος addconcept (Σχήμα 4.3), που προσθέτει μία νέα έννοια στην Βάση Γνώσης, αλλά και ανάκτησης όπως η getInverse (Σχήμα 4.4) που επιστρέφει τον ανάστροφο εάν υπάρχει ενός ρόλου.



Σχήμα 4.2: *UML Διάγραμμα KnowledgeBase*

```
public void addConcept(String name){
    if (tbox.getConcept(name) == null){
        Concept conc = new Concept(name);
        concepts.add(conc);
    }
}
```

Σχήμα 4.3: *Μέθοδος για την Προσθήκη Νέας Έννοιας σε Βάση Γνώσης*

```
public Role getInverse(Role role){
    Role roleA = tbox.getRole(role.getName());
    if (roleA == null)
        return null;
    Role invroleA  = roleA.getInverse();
    return invroleA != null ? invroleA : null;
}
```

Σχήμα 4.4: *Μέθοδος για την Ανάκτηση Ανάστροφου Ρόλου*

4.2.2 DIGConstants

Η κλάση DIGConstants περιέχει το σύνολο των απαραίτητων όρων για τη δημιουργία ενός XML αρχείου ,με τη μορφή ορισμένων συμβολοσειρών strings. Παρέχει έτσι το λεξικό για χρήση από άλλες κλάσεις που θα διαβάζουν ή θα δημιουργούν XML αρχεία. Ένα μικρό κομμάτι της φαίνεται ενδεικτικά στο Σχήμα 4.5.

```
public static final String EQUALC    = "equalc";
public static final String NAME      = "name";
public static final String SOME      = "some";
public static final String DEGREE    = "degree";
```

Σχήμα 4.5: *Παράδειγμα Δηλώσεων Συμβολοσειράς των Όρων DIG*

4.2.3 DIGHandler

Η κλάση DIGHandler παρέχει την επεξεργασία της βασικής γλώσσας περιγραφής των εννοιών του DIG interface. Περιέχει ως attribute ένα αντικείμενο της κλάσης KnowledgeBase που είναι η Βάση Γνώσης που επεξεργάζεται. Βασίζεται πρωταρχικά στις μεθόδους επεξεργασίας ενός DOM document, για την ανάκτηση πληροφορίας από τους κόμβους ενός XML αρχείου. Οι μέθοδοι αυτοί επεκτείνονται για την ανάκτηση πληροφορίας από περισσότερο σύνθετα στοιχεία (elements) ενός DIG XML αρχείου και την δημιουργία των αντίστοιχων αντικειμένων, στην αναφερόμενη βάσης γνώσης. Όπως για παράδειγμα η μέθοδος parseSome, (Σχήμα 4.6), από ένα XML element περιγραφής ενός μερικού υπαρξιακού περιορισμού (some) δημιουργεί ένα αντικείμενο (RoleConcept) περιγραφής μιας έννοιας υπαρξιακού περιορισμού,στη βάση γνώσης του λογισμικού συλλογιστικής.

```
public  Concept parseSome(Element node ){
     ElementList nodes = getElements(node);
     Role role = makeRole(nodes.item(0));
     Concept concept = makeConcept(nodes.item(1));
     RoleConcept rconcept = new RoleConcept("some",role  , ↩
        concept);
     return rconcept;
        }
```

Σχήμα 4.6: *Ενδεικτική Μέθοδος parseSome του DIGHandler*

4.2.4 DIGTellHandler

Η κλάση DIGTellHandler επεκτείνει την κλάση DIGHandler, περιγράφοντας την επεξεργασία της Tell σύνταξης του DIG interface. Περιέχει μία βασική μέθοδο με το όνομα tells, η οποία διασχίζει το document ενός Tells αρχείου και για κάθε κόμβο που συναντά, ανάλογα με το περιεχόμενο του· όπως το tag στοιχείο του ή τα υπο-στοιχεία (sub-elements) του, δημιουργεί κατάλληλα τα αντίστοιχα αντικείμενα αναπαράστασης εννοιών αξιωμάτων ή ισχυρισμών στην βάση γνώσης, Ένα ενδεικτικό κομμάτι του πηγαίου κώδικα της μεθόδου φαίνεται στο Σχήμα 4.7 όπου διασχίζοντας το XML δέντρο, για κάθε element που συναντά. με τίτλο (tag) DEFCONCEPT η μέθοδος δημιουργεί μια νέα έννοια (concept) στη Βάση Γνώσης με το αντίστοιχο όνομα.

```
public Document tells(Element tells) {
...
    ElementList tellList = getElements(tells);
    for (int i = 0; i < tellList.getLength(); i++) {
        Element tell = tellList.item(i);
        String tag = tell.getTagName();
        try {
            if (tag.equals(DEFCONCEPT)) {
                        kb.addConcept(getName(tell));
            } else if (tag.equals(DEFROLE)) {
                        kb.addRole(getName(tell));

            }
    }       ....
}
```

Σχήμα 4.7: *Κομμάτι Πηγαίου Κώδικα της Μεθόδου tells*

4.2.5 DIGAskHandler

Η κλάση DIGAskHandler επεκτείνει την κλάση DIGTellHandler , περιγράφοντας επιπλέον την επεξεργασία της Ask σύνταξης του DIG interface. Περιέχει μία βασική μέθοδο με το όνομα asks, η οποία διασχίζει το document ενός Asks αρχείου και για κάθε κόμβο (ερώτημα) του, εκτελεί την απαραίτητη συλλογιστική διαδικασία στην αναφερόμενη βάση γνώσης. Επιπλέον περιέχει ένα αντικείμενο της κλάσης DigResponse, όπου περιγράφεται μία απάντηση σε ένα Asks αίτημα. Για κάθε επιμέρους ερώτημα του Asks XML αιτήματος, η μέθοδος καλεί τις αντίστοιχες διαδικασίες συλλογιστικής και συμπληρώνει το σώμα της απάντησης. Τελικά συνθέτει και επιστρέφει το XML της απάντησης Reply για την αποστολή του στην εφαρμογή - πελάτη. Για παράδειγμα όπως φαίνεται στο Σχήμα 4.8, η μέθοδος asks διασχίζοντας το XML δένδρο του αιτήματος, αν συναντήσει ένα XML element με όνομα $ALL\_INDIVIDUALS$ θα καλέσει την αντίστοιχη μέθοδο allIndividuals() για την συγκέντρωση όλων των ατόμων που έχουν οριστεί στη βάση γνώσης και στη συνέχεια θα προσθέσει ένα element στο XML DOM δένδρο της απάντησης με τα ονόματα όλων των ατόμων.

4.2.6 DigResponse

Η κλάση DigResponse περιγράφει την απάντηση (Reply) σε ένα Asks αίτημα του πελάτη. Περιέχει ένα DOM document που αναπαριστά την XML απάντηση. Διαθέτει ακόμη μεθόδους για την δημιουργία των μικρότερων στοιχείων απάντησης (reply elements) σε κάθε ερώτημα, και την σύνθεση τους στο σώμα της κύριας απάντησης. Ένα παράδειγμα τέτοιας μεθόδου που προσθέτει στο XML σώμα της απάντηση ένα set από συνώνυμους ρόλους φαίνεται στο Σχήμα 4.9.

```
public Document asks(Element asks){
    ...
    response = new DigResponse(DIGConstants.RESPONSES);
    ElementList askElements = getElements(asks);
    for (int i = 0;i<askElements.getLength();i++){
        Element ask = askElements.item(i);
        String tag = getTagName(ask);
        Element result = null;
        ...
        if (tag.equals(ALL_CONCEPT_NAMES)){
            result = allConceptNames();
        }
        else if (tag.equals(ALL_ROLE_NAMES)){
            result = allRoleNames();
        }
        else if (tag.equals(ALL_INDIVIDUALS)){

            result = allIndividuals();
        }...
            }
}...
private Element allInvidividuals() {...}
```

Σχήμα 4.8: *Κομμάτι Πηγαίου Κώδικα της Κλάσης DIGAskHandler*

4.2.7 Reasoner

Η κλάση Reasoner (Σχήμα 4.10) αποτελεί το κέντρο της υλοποίησης μας. Κληρονομεί την κλάση DIGAskHandler και επιπλέον περιγράφει την επεξεργασία της σύνταξης του DIG για την αναγνώριση του Reasoner, και για την διαχείριση των βάσεων γνώσεως όπως περιγράφτηκαν στο Κεφάλαιο 2. Τα βασικότερα χαρακτηριστικά της είναι :

- Ένα java.util.Map με όνομα allkbs, όπου περιέχονται όλες οι (KnowledgeBase) βάσεις γνώσεις.
- Το DOM Parser με τον οποίο γίνεται η επεξεργασία κάθε XML αρχείου.

Η κύρια μέθοδος της κλάσης ονομάζεται process ,η οποία διασχίζει το DOM document της εισόδου, και ανάλογα με αυτή, θα καλέσει τις απαραίτητες υπηρεσίες συλλογιστικής, δημιουργώντας τελικά την κατάλληλη απάντηση. Περιληπτικά :

- Εάν η είσοδος είναι ένα newkb ή ένα releasekb, τότε θα εκτελεστούν οι απαραίτητες μέθοδοι διαχείρισης των βάσεων γνώσης και στη συνέχεια θα δημιουργηθεί το xml αρχείο της κατάλληλης απάντησης.
- Εάν η είσοδος είναι ένα tells xml αρχείο, τότε θα εκτελεστεί η ομώνυμη μέθοδος όπως περιγράφεται πιο πάνω.
- Εάν η είσοδος είναι ένα asks xml αρχείο, τότε θα εκτελεστεί η ομώνυμη μέθοδος και η δημιουργία της αντίστοιχης απάντησης.

```
public void addRSynonms(Element parent, Role r) {
    Element synonms = addElement(DIGConstants.SYNONYMS, ↩
       parent);
    Element element = addElement("ratom", synonms);
    element.setAttribute(DIGConstants.NAME, r.getName());
}
public Element addRoleSet(Collection roles) {
    Element set = addElement(DIGConstants.ROLE_SET);
    for (Iterator i = roles.iterator(); i.hasNext();) {
        Role r = (Role) i.next();
        addRSynonms(set, r);
    }
return set;
}
```

Σχήμα 4.9: *Μέθοδος της Κλάσης DigResponse*

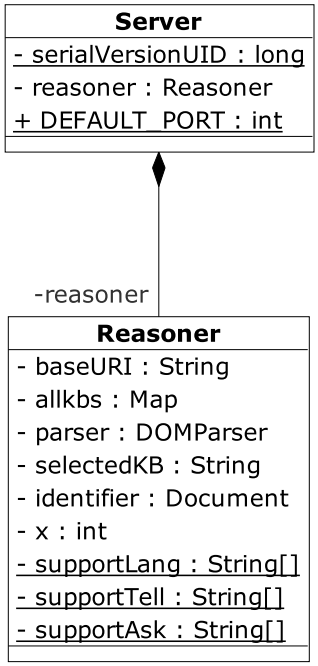
Η κλάση διαθέτει επίσης δύο μεθόδους άξιες αναφοράς.

1. Την μέθοδο kbtofire, η οποία δέχεται σαν είσοδο μία βάση γνώσης (KnowledgeBase) και επιστρέφει την περιγραφή της σε LISP τύπου σύνταξη που υποστηρίζει το λογισμικό FiRE.

2. Την μέθοδο kbtoTell, που από μία βάση γνώσης ,εξάγει ένα Tells XML για χρήση των υπηρεσιών μιας εναλλακτικής απομακρυσμένης μηχανής συλλογιστικής.

4.2.8 Server

Η κλάση Server περιέχει ένα αντικείμενο της κλάσης Reasoner. Υλοποιεί το interface org.mortbay.http.HttpHandler, όπως αυτό παρέχεται από την βιβλιοθήκη org.mortbay η οποία περιγράφει έναν μικρό, ευέλικτο HTTP Server. Ο HTTP Server μεταφέρει τα XML μηνύματα που δέχεται στον Reasoner και αντίστροφα. Η κλάση δέχεται ως όρισμα τον αριθμό της πόρτας (port), στην οποία επιθυμούμε να εκτελείται ο HTTP Server.



Σχήμα 4.10: *UML Διάγραμμα Server - Reasoner*

Κεφάλαιο 5

Εξαγωγή

Συνοψίζοντας, στην εργασία αυτή μελετήθηκε εκτεταμένα το πρότυπο διασύνδεσης με μηχανές συλλογιστικής DIG. Παρουσιάστηκαν οι υποθέσεις στις οποίες βασίζεται το πρότυπο, το πρωτόκολλο επικοινωνίας που προτείνεται, η γλώσσα περιγραφής εννοιών, καθώς και η σύνταξη των αιτημάτων αναγνώρισης, διαχείρισης, ανάπτυξης και συλλογισμού σε μία βάση γνώσης.

Με ιδιαίτερη προσοχή μελετήσαμε τις συντακτικές δομές της γλώσσας περιγραφής των αξιωμάτων ισχυρισμών που υποστηρίζεται από το πρότυπο. Στη συνέχεια επεκτείναμε την ικανότητα περιγραφής αξιωμάτων ισχυρισμών με τη δυνατότητα περιγραφής ασαφών ισχυρισμών· έτσι όπως αυτοί ορίζονται και χρησιμοποιούνται από τις Ασαφείς Περιγραφικές Λογικές. Ορίσαμε έτσι την δική μας επέκταση στο XML Σχήμα που προτείνει το πρότυπο. Με αυτόν το τρόπο προτείνεται στην κοινότητα των ερευνητών και προγραμματιστών συστημάτων ΠΛ ένας κοινός τρόπος διασύνδεσης με μηχανές συλλογιστικής για ασαφείς περιγραφικές λογικές.

Ακολούθως παρουσιάσαμε τις εργασίες μας για την υλοποίηση της διασύνδεσης στα πλαίσια της επέκτασης του λογισμικού ασαφούς συλλογιστικής FiRE. Το λογισμικό επεκτάθηκε με επιτυχία, αποκτώντας τη δυνατότητα εξυπηρέτησης απομακρυσμένων εφαρμογών, που μπορεί να έχουν υλοποιηθεί σε διαφορετική γλώσσα προγραμματισμού και διαθέτουν τη δυνατότητα διασύνδεσης μέσω του DIG πρωτοκόλλου. Έτσι μπορεί να χρησιμοποιηθεί σαν απομακρυσμένη μηχανή συλλογιστικής από λογισμικά σχεδίασης οντολογιών όπως είναι τα OilEd[8] και Protege[1].

Επιπλέον το λογισμικό FiRE επεκτάθηκε με τη δυνατότητα διασύνδεσης με άλλες μηχανές συλλογιστικής, ως εφαρμογή (client). Με αυτόν το τρόπο διευκολύνεται η ερευνητική εργασία με τη δυνατότητα χρήσης και πειραματισμού με εναλλακτικές μηχανές συλλογιστικής.

Κατά την ανάπτυξη της υλοποίησης της διασύνδεσης DIG συναντήσαμε κάποιες ατέλειες στο πρότυπο όπως ορίζεται στην έκδοση 1.1. Μια παρατήρηση αφορά στον διαφορετικό ορισμό των εννοιών $< top >$ και $< bottom >$ από τις υπόλοιπες , οι οποίες ορίζονται με ένα στοιχείο (element catom) που περιέχει το όνομα της έννοιας. Έτσι δημιουργείται μια ασυνέχεια στον τρόπο διαχείρισης των εννοιών προγραμματιστικά, που περιπλέκει για παράδειγμα την δημιουργία ενός συνόλου από έννοιες (Concept Set) για την δημιουργία της απάντησης σε ένα αίτημα .

Το πρότυπο υιοθετεί την υπόθεση του μοναδικού ονόματος (unique name assumption). Αυτό σημαίνει ότι αν δύο στοιχεία της οντολογίας έχουν διαφορετικά ονόματα τότε απαραίτητα αναφέρονται σε διαφορετικά αντικείμενα. Πολλές από τις ΠΛ και τις

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

ασαφείς επεκτάσεις τους, που προορίζονται για συλλογιστικές διαδικασίες στον Σημασιολογικό Ιστό όπως είναι οι OWLκαι DAML έχουν μηχανισμούς που επιτρέπουν την ρητή δήλωση ότι δύο άτομα είναι ίδια, για παράδειγμα owl:sameAs. Αυτή η ισότητα ατόμων δεν μπορεί να εκφραστεί με την διασύνδεση DIG σε αντίθεση με την ισότητα ατόμων και ρόλων που υποστηρίζονται. με τις δηλώσεις equalc και equalr.

Στα ερωτήματα, που μπορεί να γίνουν σε μία μηχανή συλλογιστικής, περιλαμβάνονται τα :

- $< disjoint > C1\ C2 < /disjoint >$ που επιστρέφει μια απάντηση boolean,
- $< equivalent > C1 < /equivalent >$ που επιστρέφει ένα σύνολο από έννοιες ConceptSet.

Για πληρότητα θα έπρεπε να υποστηρίζονται και τα ερωτήματα :

- $< disjoint > C1 < /disjoint >$ που επιστρέφει ένα σύνολο από έννοιες ConceptSet,
- $< equivalent > C1\ C2 < /equivalent >$ που επιστρέφει μια απάντηση boolean.

Η διασύνδεση DIG στην τωρινή αρχική της έκδοση δεν προσφέρει αυτό που θα ονομάζαμε μια πλήρη υπηρεσία συλλογιστικής, ωστόσο καταφέρνει να απομονώνει την ανάπτυξη εφαρμογών ενός συστήματος ΠΛ από το ίδιο το σύστημα, την γλώσσα ανάπτυξης του και την τοποθεσία του. Επιτρέπει έτσι την ανάπτυξη «plug and play» εφαρμογών όπου εναλλακτικοί reasoners μπορούν να εισάγονται και να εξυπηρετούν την εφαρμογή μας.

Η πρότυπη διασύνδεση DIG στην τωρινή έκδοση της 1.1 συνιστά μια πολύ καλή αρχή προς την κατεύθυνση της περιγραφής ενός κοινού τρόπου διασύνδεσης για τα συστήματα Περιγραφικών Λογικών. Η ύπαρξη και υποστήριξη ενός τέτοιου προτύπου είναι πολύ σημαντική για την κοινότητα του σημασιολογικού ιστού. Η επόμενη έκδοση του προτύπου DIG 2.0 βρίσκεται ήδη υπό ανάπτυξη. Οι ατέλειες που παρατηρήσαμε έχουν ήδη διορθωθεί. Στην επόμενη του έκδοση το πρότυπο θα βασίζεται στην περιγραφική λογική $SROIQD_n^-$ υποστηρίζοντας πλήρως την προ- τυποποιημένη γλώσσα OWL 1.1 για τον σημασιολογικό ιστό. Επιπλέον θα προτείνει καλά ορισμένους τρόπους για την επέκταση του βασικού interface. Συγκεκριμένα μία επέκταση του DIG 2.0 προτύπου θα βασίζεται σε ένα XML Σχήμα που θα ορίζει τη σύνταξη και τα μηνύματα της επέκτασης. Κατά αυτό τον τρόπο η επέκταση μας στο πρότυπο της διασύνδεσης DIG για τη διαχείριση των ασαφών περιγραφικών λογικών, είναι ήδη συμβατή με την αναμενόμενη επόμενη έκδοση του.

Παράρτημα Α΄

Παράρτημα **Javadoc** Περιγραφή

Package <edu.ntua.ivml.Dig.Fire >

Package Contents *Page*

Classes

Classes

Class **DIGAskHandler**

Handles DIG ASKS elements , in order to process them with the Fire Reasoner and a create a DIG reply accordingly

Declaration

```
public class DIGAskHandler
extends DIGTellHandler
```

Constructors

- *DIGAskHandler*
  `public` **DIGAskHandler**`( )`
  - **Usage**
    - Sole constructor

Methods

- *allConceptNames*
  `public Element` **allConceptNames**`( )`
  - **Usage**
    - Returns all kb's Concepts
  - **Returns** - a dom element

- *asks*
  `public Document` **asks**`( org.w3c.dom.Element` **asks** `)`
  - **Usage**
    - Method that parses an ASKS DIG element , questions the Fire knowledge base and creates a DIG reply accordingly
  - **Parameters**
    - `asks` - Dom Element containing an ASKS DIG ELEMENT
  - **Returns** - dom document of DIG Response

Class **DIGConstants**

DIG Vocabulary String CLass

Declaration

public class DIGConstants
extends java.lang.Object

Fields

- public static final String TELLS

  –

- public static final String ASKS

  –

- public static final String DEFCONCEPT

  –

- public static final String NAME

  –

- public static final String FUZZYSYMBOL

  –

- public static final String DEGREE

  –

- public static final String DEFROLE

  –

- public static final String DEFATTRIBUTE

  –

- public static final String DEFINDIVIDUAL

  –

- public static final String DEFFEATURE

  –

- public static final String EQUALC

  –

- public static final String IMPLIESC

  –

- public static final String DISJOINT

–

- public static final String IMPLIESR
  –
- public static final String EQUALR
  –
- public static final String FUNCTIONAL
  –
- public static final String TRANSITIVE
  –
- public static final String DOMAIN
  –
- public static final String RANGE
  –
- public static final String RANGEINT
  –
- public static final String RANGESTRING
  –
- public static final String INSTANCEOF
  –
- public static final String RELATED
  –
- public static final String VALUE
  –
- public static final String TOP
  –
- public static final String BOTTOM
  –
- public static final String CATOM
  –
- public static final String AND
  –

- public static final String OR
  –
- public static final String NOT
  –
- public static final String SOME
  –
- public static final String ALL
  –
- public static final String ATMOST
  –
- public static final String ATLEAST
  –
- public static final String ISET
  –
- public static final String DEFINED
  –
- public static final String STRINGMIN
  –
- public static final String STRINGMAX
  –
- public static final String STRINGEQUALS
  –
- public static final String STRINGRANGE
  –
- public static final String INTMIN
  –
- public static final String INTMAX
  –
- public static final String INTEQUALS
  –
- public static final String INTRANGE

–

- public static final String RATOM

  –

- public static final String FEATURE

  –

- public static final String INVERSE

  –

- public static final String ATTRIBUTE

  –

- public static final String CHAIN

  –

- public static final String INDIVIDUAL

  –

- public static final String NUM

  –

- public static final String IVAL

  –

- public static final String SVAL

  –

- public static final String SCHEMA

  –

- public static final String NAMESPACE

  –

- public static final String XSI

  –

- public static final String RESPONSES

  –

- public static final String RESPONSE

  –

- public static final String OK

  –

- public static final String KB

  –

- public static final String URI

  –

- public static final String ERROR

  –

- public static final String CODE

  –

- public static final String MESSAGE

  –

- public static final String ID

  –

- public static final String ALL_CONCEPT_NAMES

  –

- public static final String ALL_ROLE_NAMES

  –

- public static final String ALL_INDIVIDUALS

  –

- public static final String SATISFIABLE

  –

- public static final String SUBSUMES

  –

- public static final String PARENTS

  –

- public static final String CHILDREN

  –

- public static final String ANCESTORS

  –

- public static final String DESCENDANTS

  –

- public static final String EQUIVALENTS

–

- public static final String RPARENTS

  –

- public static final String RCHILDREN

  –

- public static final String RANCESTORS

  –

- public static final String RDESCENDANTS

  –

- public static final String INSTANCES

  –

- public static final String TYPES

  –

- public static final String INSTANCE

  –

- public static final String ROLE_FILLERS

  –

- public static final String RELATED_INDIVIDUALS

  –

- public static final String TOLD_VALUES

  –

- public static final String CONCEPT_SET

  –

- public static final String SYNONYMS

  –

- public static final String ROLE_SET

  –

- public static final String INDIVIDUAL_SET

  –

- public static final String INDIVIDUAL_PAIR_SET

  –

- public static final String INDIVIDUAL_PAIR
  –
- public static final String TRUE
  –
- public static final String FALSE
  –
- public static final String WARNING
  –
- public static final String GET_IDENTIFIER
  –
- public static final String NEWKB
  –
- public static final String CLEARKB
  –
- public static final String RELEASEKB
  –
- public static final String IDENTIFIER
  –
- public static final String VERSION
  –
- public static final String SUPPORTS
  –
- public static final String LANGUAGE
  –
- public static final String ASK
  –
- public static final String TELL
  –

Constructors

- *DIGConstants*
  `public` **DIGConstants**`( )`

Class **DIGErrors**

Dig Error codes String Base

Declaration

public class DIGErrors
extends java.lang.Object

Fields

- public static int GENERAL_UNSPECIFIED_ERROR

  –

- public static int UNKNOWN_REQUEST

  –

- public static int MALFORMED_REQUEST

  –

- public static int UNSUPPORTED_OPERATION

  –

- public static int CANNOT_CREATE_NEW_KNOWLEDGE

  –

- public static int MALFORMED_KB_URI

  –

- public static int UNKNOWN_OR_STALE_KB_URI

  –

- public static int KB_RELEASE_ERROR

  –

- public static int MISSING_URI

  –

- public static int GENERAL_TELL_ERROR

  –

- public static int UNSUPPORTED_TELL_OPERATION

  –

- public static int UNKNOWN_TELL_OPERATION
  - –
- public static int GENERAL_ASK_ERROR
  - –
- public static int UNSUPPORTED_ASK_OPERATION
  - –
- public static int UNKNOWN_ASK_OPERATION
  - –
- public static String codes
  - –

Constructors

- *DIGErrors*
  `public` **DIGErrors**`( )`

Class **DIGHandler**

Handles all DIG language concept encoding terms

Declaration

public class DIGHandler
extends DIGConstants

Constructors

- *DIGHandler*
  `public` **DIGHandler**`( )`
  - **Usage**
    - Sole constructor

- *checkConcept*
  `public boolean` **checkConcept**`( org.w3c.dom.Element` **c** `)`
  - **Usage**
    - Checks if a Dig element representing a complex concept reffers to atomic concepts of the knowledge base
  - **Parameters**
    - `c` - dom element
  - **Returns** - a boolean value

- *getAttributeValue*
  `public static String` **getAttributeValue**`( org.w3c.dom.Element` **node**`, java.lang.String` **name** `)`
  - **Usage**
    - Returns the value of an element attribute
  - **Parameters**
    - `node` - dom element
    - `name` - string , name of attribute
  - **Returns** - string value of attribute

- *getDegree*
  `public static Double` **getDegree**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Returns the value of a Degree attribute
  - **Parameters**
    - `node` - dom element
  - **Returns** - double, value of attribute degree

- *getElement*
  `public static Element` **getElement**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Returns child element of element
  - **Parameters**
    - `node` - dom element
  - **Returns** - dom element or null

- *getElementArray*
  `public static Element` **getElementArray**`( org.w3c.dom.Element` **node** `)`
  - **Usage**

    - Returns an array containing all the child elements of an element
  - **Parameters**
    - `node` - dom element
  - **Returns** - elementlist of element's child nodes

- *getElements*
  `public static ElementList` **getElements**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Returns a list of all child elements of an element
  - **Parameters**
    - `node` - dom element
  - **Returns** - an elementlist

- *getElements*
  `public static ElementList` **getElements**`( org.w3c.dom.Element` **node**`, java.lang.String` **tagName** `)`
  - **Usage**
    - Returns a list of all elements of an element with a specific tag
  - **Parameters**
    - `node` - dom element
    - `tagName` - string tag name of element
  - **Returns** - an elementlist

- *getFuzzySymbol*
  `public static String` **getFuzzySymbol**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Returns the value of a FuzzySymbol attribute
  - **Parameters**
    - `node` - dom element
  - **Returns** - string , of attribute fuzzysymbol

- *getId*
  `public static String` **getId**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Returns the value of an ID attribute
  - **Parameters**
    - `node` - dom element
  - **Returns** - value of attribute ID

- *getInverse*
  `public Role` **getInverse**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Returns the role inside an INVERSE element
  - **Parameters**
    - `node` - dom element
  - **Returns** - Role object or null

- *getName*
  `public static String` **getName**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Returns the value of a NAME attribute
  - **Parameters**
    - `node` - dom element
  - **Returns** - string , of attribute NAME

- *getNum*
  `public static int` **getNum**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Returns the integer value of a NUM atrribute
  - **Parameters**
    - `node` - dom element
  - **Returns** - integer , value of attribute NUM

- *getTagName*
  `public static String` **getTagName**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Returns the Tag name of an xml element
  - **Parameters**
    - `node` - dom element
  - **Returns** - tag name ,string

- *getURI*
  `public static String` **getURI**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Returns the URI value of an xml element
  - **Parameters**
    - `node` -
  - **Returns** - dom element

- *makeConcept*
  `public Concept` **makeConcept**`( org.w3c.dom.Element` **c** `)`
  - **Usage**
    - Returns a Concept defined inside a Dig Element
  - **Parameters**
    - `c` - dom Dom element
  - **Returns** - a Concept object

- *makeRole*
  `public Role` **makeRole**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Creates a Role object out of a dig element representation of role
  - **Parameters**
    - `node` - dom element
  - **Returns** - a Role object

- *parseAll*
  `public Concept` **parseAll**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Parses an ,"All" role concept describing, Dig element
  - **Parameters**
    - `node` - dom element
  - **Returns** - a RoleConcept object

- *parseAtleast*
  `public Concept` **parseAtleast**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Parses ,an "Atleast" number restriction concept, describing Dig element
  - **Parameters**
    - `node` - dom element
  - **Returns** - a RoleConcept object

- *parseAtmost*
  `public Concept` **parseAtmost**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Parses ,an "Atmost" number restriction concept, describing Dig element
  - **Parameters**
    - `node` - dom element
  - **Returns** - a RoleConcept onject

- *parseSome*
  `public Concept` **parseSome**`( org.w3c.dom.Element` **node** `)`
  - **Usage**
    - Parses ,a "Some" role concept, describing Dig element
  - **Parameters**
    - `node` - dom element
  - **Returns** - a RoleConcept object

- *serialize*
  `public static String` **serialize**`( org.w3c.dom.Document` **doc** `)`
  - **Usage**
    - Supporting method to serialize an xml
  - **Parameters**
    - `doc` - Dom document
  - **Returns** - string representing xml file

- *serialize*
  `public static String` **serialize**`( org.w3c.dom.Element` **el** `)`
  - **Usage**
    - Serializes an xml dom document
  - **Parameters**
    - `el` - dom element
  - **Returns** - string xml representetion

- *setKB*
  `public void` **setKB**`( KnowledgeBase` **kb** `)`
  - **Usage**
    - Sets knowledge base
  - **Parameters**
    - `kb` - KnowledgeBase object

Class **DigResponse**

DigResponse is the reply xml element's DOM document , to the client requests provides methods to parse replies into DIG XML format

Declaration

public class DigResponse
extends DIGConstants

Constructors

- *DigResponse*
  `public` **DigResponse**`( java.lang.String` **rootTag** `)`
  - **Usage**
    - DigResponse constructor
  - **Parameters**
    - `rootTag` - root element tag name

- *DigResponse*
  `public` **DigResponse**`( java.lang.String` **rootTag**`, java.lang.String` **uri** `)`
  - **Usage**
    - DIG Tell Constructor
  - **Parameters**
    - `rootTag` - root element tag name
    - `uri` - knowledge base uri string represantation

Methods

- *addBoolean*
  `public Element` **addBoolean**`( boolean` **b** `)`
  - **Usage**
    - Adds a boolean element to response
  - **Parameters**
    - `b` - boolean variable
  - **Returns** - dom element

- *addConceptSet*
  `public Element` **addConceptSet**`( java.util.Collection` **concepts** `)`
  - **Usage**
    - Adds a set of concepts to Response
  - **Parameters**
    - `concepts` - a collection of concept objects
  - **Returns** - a dom element

- *addconcepttodig*
  `public void` **addconcepttodig**`( edu.ntua.ivml.fire.fuzzy.Concept` **c**`, org.w3c.dom.Element` **root** `)`
  - **Usage**

    * Recursive method to add a concept(simple , complex , role or nrconcept) representing element , to a Dig Tells xml file.
  - **Parameters**
    * `c` - Concept object
    * `root` - element's parent

- *addCSynonms*
  `public void` **addCSynonms**`( org.w3c.dom.Element` **parent**`, edu.ntua.ivml.fire.fuz`
  c `)`
  - **Usage**
    * Adds a set of Concept Synonyms to the Response
  - **Parameters**
    * `parent` - element's parent element
    * `c` - a concept object

- *addElement*
  `protected Element` **addElement**`( java.lang.String` **tag** `)`
  - **Usage**
    * Adds new element named tag under root element
  - **Parameters**
    * `tag` - element's tag name
  - **Returns** - a dom element

- *addElement*
  `protected Element` **addElement**`( java.lang.String` **tag**`, org.w3c.dom.Element`
  parent `)`
  - **Usage**
    * Adds new element named tag under parent element
  - **Parameters**
    * `tag` - element's tag name
    * `parent` - element's parent element
  - **Returns** - a dom element

- *addError*
  `public Element` **addError**`( int` **code**`, java.lang.String` **details** `)`
  - **Usage**
    * Helper method to create an error element containing an error-code
  - **Parameters**
    * `code` - error code
    * `details` - error details
  - **Returns** - dom element

- *addIndividualPairSet*
  `public Element` **addIndividualPairSet**`( java.util.Collection` **fuzzypairs** `)`
  - **Usage**
    - Adds pairs of individuals to Response
  - **Parameters**
    - `fuzzypairs` - a collection of fuzzyrelation objects
  - **Returns** - a dom element

- *addIndividualSet*
  `public Element` **addIndividualSet**`( java.util.List` **individuals** `)`
  - **Usage**
    - Adds a set of individuals to Response
  - **Parameters**
    - `individuals` - a list containing individuals names
  - **Returns** - a dom element

- *addRoleSet*
  `public Element` **addRoleSet**`( java.util.Collection` **roles** `)`
  - **Usage**
    - Adds a set of roles to Response
  - **Parameters**
    - `roles` - a collection of role objects
  - **Returns** - a dom element

- *addRSynonms*
  `public void` **addRSynonms**`( org.w3c.dom.Element` **parent**`, edu.ntua.ivml.fire.fuz`
  r `)`
  - **Usage**
    - Adds a set of Role Synonyms to the Response
  - **Parameters**
    - `parent` - element's parent element
    - `r` - a role object

- *createERRORResponse*
  `public static Document` **createERRORResponse**`( int` **code**`, java.lang.String`
  details `)`
  - **Usage**
    - Returns a Response containing an Error element and the specific error-code
  - **Parameters**

        * `code` - error code
        * `details` - code details
    - **Returns** - response dom document

- *createKBResponse*
  `public static Document` **createKBResponse**`( java.lang.String` **uri** `)`
    - **Usage**
        * Returns a Response containing the URI of the knowledge base created
    - **Parameters**
        * `uri` - knowledge base uri string represantation
    - **Returns** - dom document

- *createOkResponse*
  `public static Document` **createOkResponse**`( )`
    - **Usage**
        * Returns a Response containing a simple OK element
    - **Returns** - response's dom document

- *getDocument*
  `public Document` **getDocument**`( )`
    - **Usage**
        * Helper method ,returns Reply's xml doc basic document
    - **Returns** - a DOM Document

Class **DIGTellHandler**

Handles DIG Tells Elements in order to parse a knowledge base from DIG representation to Fire 's knowledge base structures

Declaration

public class DIGTellHandler
extends DIGHandler

Constructors

- *DIGTellHandler*
  `public` **DIGTellHandler**`( )`
    - **Usage**
        * Sole Constructor

Methods

- *tells*
  `public Document` **tells**`( org.w3c.dom.Element` **tells** `)`
  - **Usage**
    - Method that parses a Tells DIG element , creates a Fire knowledge base and a DIG reply accordingly
  - **Parameters**
    - `tells` - the dom element containing the tells
  - **Returns** - Dom document representing Dig Response

Class **ElementList**

List of Element Nodes Helper class to access DOM Element Node Lists

Declaration

public class ElementList
extends java.lang.Object

Constructors

- *ElementList*
  `public` **ElementList**`( org.w3c.dom.NodeList` **nodelist** `)`
- *ElementList*
  `public` **ElementList**`( org.w3c.dom.NodeList` **nodeList**`, int` **limit** `)`

Methods

- *getLength*
  `public int` **getLength**`( )`
- *getNodeArray*
  `public Element` **getNodeArray**`( )`
- *isEmpty*
  `public boolean` **isEmpty**`( )`
- *item*
  `public Element` **item**`( int` **index** `)`
- *toString*
  `public String` **toString**`( )`

Class **KnowledgeBase**

The knowledge base consists of a Tbox and an Abox object and provides methods to add and extract knowledge

Declaration

public class KnowledgeBase
extends java.lang.Object

Fields

- public ABox abox
  –
- public TBox tbox
  –

Constructors

- *KnowledgeBase*
  `public` **KnowledgeBase**`( )`
  - **Usage**
    * Sole Constructor

Methods

- *addAssertion*
  `public void` **addAssertion**`( edu.ntua.ivml.fire.fuzzy.FuzzyAssertion` **fa** `)`
  - **Usage**
    * Adds a new assertion to knowledge base
  - **Parameters**
    * `fa` - FuzzyAssertion

- *addAxiom*
  `public void` **addAxiom**`( edu.ntua.ivml.fire.fuzzy.Axiom` **axiom** `)`
  - **Usage**
    * Adds an Axiom to knowledge base

- **Parameters**
  - `axiom` - Axiom object

- *addConcept*
  `public void` **addConcept**`( edu.ntua.ivml.fire.fuzzy.Concept` **concept** `)`
  - **Usage**
    - Adds a concept to knowledge base
  - **Parameters**
    - `concept` - Concept object

- *addConcept*
  `public void` **addConcept**`( java.lang.String` **name** `)`
  - **Usage**
    - Adds a Concept to knowledge base named name
  - **Parameters**
    - `name` - Concept name

- *addIndividual*
  `public void` **addIndividual**`( java.lang.String` **name** `)`
  - **Usage**
    - Adds an Individual to knowledge base
  - **Parameters**
    - `name` - Individual name

- *addRole*
  `public void` **addRole**`( edu.ntua.ivml.fire.fuzzy.Role` **role** `)`
  - **Usage**
    - Adds a Role to knowledge base
  - **Parameters**
    - `role` - Role object

- *addRole*
  `public void` **addRole**`( java.lang.String` **name** `)`
  - **Usage**
    - Adds a role to knowledge base named name
  - **Parameters**
    - `name` - Role names

- *checkEqualcConcept*
  `public Boolean` **checkEqualcConcept**`( edu.ntua.ivml.fire.fuzzy.Concept` **c** `)`

- *clear*
  `public void` **clear**`( )`
  - **Usage**
    - Clears the abox and the tbox knowledge

- *concExists*
  `public Boolean` **concExists**`( edu.ntua.ivml.fire.fuzzy.Concept` **concept** `)`
  - **Usage**
    - Check if concept exists in knowledge base
  - **Parameters**
    - `concept` - a Concept object
  - **Returns** - a boolean value

- *Fuzzyentailment*
  `public boolean` **Fuzzyentailment**`( edu.ntua.ivml.fire.fuzzy.Concept` **c**`, java.lang.String` **fs**`, java.lang.Double` **deg** `)`

- *getAbox*
  `public String` **getAbox**`( )`
  - **Usage**
    - Returns Abox
  - **Returns** - string represantation of abox

- *getAncestors*
  `public List` **getAncestors**`( edu.ntua.ivml.fire.fuzzy.Concept` **c** `)`

- *getChildren*
  `public List` **getChildren**`( edu.ntua.ivml.fire.fuzzy.Concept` **c** `)`

- *getChildren*
  `public List` **getChildren**`( edu.ntua.ivml.fire.fuzzy.Role` **r** `)`

- *getDescendants*
  `public List` **getDescendants**`( edu.ntua.ivml.fire.fuzzy.Concept` **c** `)`

- *getDescendants*
  `public List` **getDescendants**`( edu.ntua.ivml.fire.fuzzy.Role` **r** `)`

- *getequals*
  `public List` **getequals**`( edu.ntua.ivml.fire.fuzzy.Concept` **c** `)`

- *getInstances*
  `public List` **getInstances**`( edu.ntua.ivml.fire.fuzzy.Concept` **c**`, java.lang.String` **fs**`, java.lang.Double` **deg** `)`

- *getInverse*
  `public Role` **getInverse**`( edu.ntua.ivml.fire.fuzzy.Role` **role** `)`
  - **Usage**
    - Returns the inverse role of a role if exists
  - **Parameters**
    - `role` - Role object
  - **Returns** - a Role object

- *getParents*
  `public List` **getParents**`( edu.ntua.ivml.fire.fuzzy.Concept` **c** `)`

- *getParents*
  `public List` **getParents**`( edu.ntua.ivml.fire.fuzzy.Role` **r** `)`

- *getRancestors*
  `public List` **getRancestors**`( edu.ntua.ivml.fire.fuzzy.Role` **r** `)`

- *getRelatedIndiviuals*
  `public List` **getRelatedIndiviuals**`( edu.ntua.ivml.fire.fuzzy.Role` **r**`,`
  `java.lang.String` **fs**`, java.lang.Double` **deg** `)`

- *getroleFillers*
  `public List` **getroleFillers**`( edu.ntua.ivml.fire.fuzzy.Role` **r**`, java.lang.String`
  ind`, java.lang.String` **fs**`, java.lang.Double` **deg** `)`

- *getTbox*
  `public String` **getTbox**`( )`
  - **Usage**
    - Returns Tbox
  - **Returns** - string represantation of tbox

- *getTypes*
  `public List` **getTypes**`( java.lang.String` **ind** `)`

- *isDisjoint*
  `public boolean` **isDisjoint**`( edu.ntua.ivml.fire.fuzzy.Concept` **c**`, edu.ntua.ivml.f`
  d `)`

- *isSubClassOf*
  `public boolean` **isSubClassOf**`( edu.ntua.ivml.fire.fuzzy.Concept` **c**`,`
  `edu.ntua.ivml.fire.fuzzy.Concept` **d** `)`

- *isType*
  `public boolean` **isType**`( java.lang.String` **ind**`, edu.ntua.ivml.fire.fuzzy.Concept`
  c`, java.lang.String` **fs**`, java.lang.Double` **deg** `)`

- *rolExists*
  `public Boolean` **rolExists**`( edu.ntua.ivml.fire.fuzzy.Role` **role** `)`

- **Usage**
  - Checks if role exists in knowledge base
- **Parameters**
  - `role` - Role object
- **Returns** - a boolean value

Class **Main**

Main demonstration class

Declaration

public class Main
extends java.lang.Object

Constructors

- *Main*
  `public` **Main**`( )`

Methods

- *main*
  `public static void` **main**`( java.lang.String [] args )`

Class **Reasoner**

Reasoner Class reads and parses XML DIG input ,passes Requests to Fire Reasoner and creates Replies for client .

Declaration

public class Reasoner
extends DIGAskHandler

Constructors

- *Reasoner*
  `public` **Reasoner**`( )`
  - **Usage**
    - Sole Constructor

Methods

- *getIdentifier*
  `public Document` **getIdentifier**`( )`
  - **Usage**
    - Creates the XML reply to a GetIdentifier request returns Reasoner 's generic information and the DIG Language supported
  - **Returns** - a dom document

- *kbtofire*
  `public void` **kbtofire**`( java.lang.String` **uri** `)`
  - **Usage**
    - Prints a Fire format representation of knowledge base
  - **Parameters**
    - `uri` - kb's uri

- *kbtotell*
  `public void` **kbtotell**`( java.lang.String` **uri**`, java.io.PrintWriter` **out** `)`
  - **Usage**
    - Creates a DIG TELLS XML file , that represents the particular knowledge base
  - **Parameters**
    - `uri` - kb's uri
    - `out` - printerstream
  - **Exceptions**
    - `java.io.IOException` -

- *newKB*
  `public String` **newKB**`( )`
  - **Usage**
    - Intermediate class to create a knowledge base
  - **Returns** - a unique uri

- *newKB*
  `public KnowledgeBase` **newKB**`( java.lang.String` **newURI** `)`
  - **Usage**
    - Creates a knowledge base , adds it to the knowledgebase hash table and assings a unique uri to it .
  - **Parameters**
    - `newURI` - a string uri
  - **Returns** - the new KnowledgeBase object

- *process*
  `public Document` **process**`( org.w3c.dom.Document` **cmd** `)`
  - **Usage**
    - Process method traverse the Dom tree of input , serves client requests (newkb,releasekb,tells,asks,getidentifier) and creates replies
  - **Parameters**
    - `cmd` - a Dom Document of the DIG XML input
  - **Returns** - the Dom Document of reply

- *process*
  `public void` **process**`( java.io.InputStream` **in**`, java.io.OutputStream` **out** `)`
  - **Usage**
    - Method to pass DIG XML input and output byte stream to character streams
  - **Parameters**
    - `in` - Inputstream
    - `out` - Outputstream
  - **Exceptions**
    - `org.xml.sax.SAXException` -
    - `java.io.IOException` -

- *process*
  `public void` **process**`( java.io.Reader` **in**`, java.io.Writer` **out** `)`
  - **Usage**
    - Method to process DIG XML input ,will call Dom parser's parse method for input , and initialize reply xml output.
  - **Parameters**
    - `in` - Reader Input
    - `out` - Writer Output
  - **Exceptions**
    - `org.xml.sax.SAXException` -
    - `java.io.IOException` -

- *releaseKB*
  `public boolean` **releaseKB**`( java.lang.String` **uri** `)`
  - **Usage**
    - Unselects and removes a particular knowledge base
  - **Parameters**
    - `uri` - kb's uri
  - **Returns** - boolean value

- *selectKB*
  `public boolean` **selectKB**`( java.lang.String` **uri** `)`
  - **Usage**
    - Selects the Knowledge Base in use according to clients request
  - **Parameters**
    - `uri` - a string identification of knowledge base
  - **Returns** - a boolean value

- *unselectKB*
  `public void` **unselectKB**`( java.lang.String` **uri** `)`
  - **Usage**
    - Unselects knowledge base
  - **Parameters**
    - `uri` - identification of knowledge base

Class **Server**

Declaration

public class Server
extends org.mortbay.http.handler.AbstractHttpHandler
implements org.mortbay.http.HttpHandler

Serializable Fields

- private Reasoner reasoner
  –

Fields

- public static int DEFAULT_PORT

  –

Constructors

- *Server*
  `public` **Server**`( )`

Methods

- *handle*
  `public void` **handle**`( java.lang.String` **pathInContext**`, java.lang.String` **pathParams**`, org.mortbay.http.HttpRequest` **request**`, org.mortbay.http.HttpRespo` **response** `)`
- *main*
  `public static void` **main**`( java.lang.String [] args )`
- *usage*
  `static void` **usage**`( )`

Βιβλιογραφία

[1] Enrico Franconi and Gary Ng. The i.com Tool for Intelligent Conceptual Modelling. In 7th Intl. Workshop on Knowledge Representation meets Databases (KRDB'00,Berlin,Germany,August 2000), 2000.

[2] F. Baader, D. McGuinness, D. Nardi, and P.F. Patel-Schneider. The Description Logic Handbook: Theory, implementation and applications. Cambridge University Press, 2002.

[3] G. Stoilos and G. Stamou and V. Tzouvaras and J.Z. Pan and I. Horrocks. The Fuzzy Description Logic f-SHIN. International Workshop on Uncertainty Reasoning For the Semantic Web (2005).

[4] J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and Berners-Lee T. Hypertext Transfer Protocol HTTP/1.1. http://www.w3.org/Protocols/rfc2616/rfc2616.html, 2000.

[5] M. Minski. A Framework for Representing Knowledge. Behavioral Science, 1981.

[6] M. Navara. Satisfiability in fuzzy logic. Neural Network World , 10(5):845-858, 2000.

[7] R. Quillian. Word Concepts. A Theory and Simulation of some basic capabilities. Behavioral Science, 1967.

[8] Sean Bechhofer,Ian Horrocks,Carole A.Goble and Robert Stevens. Oiled:a reason-able ontology editor for the semantic web. In Proceedings of KI2001, Joint German-Austrian conference on Arti?cial Intelligence, volume 2174 of LNAI,Vienna, 2001.

[9] Sean Bechhofer,Ian Horrocks,Peter F. Patel-Schneider and Sergio Tessaris. A Proposal for a Description Logic Interface. In Lambrix, 1999.

[10] U. Straccia. Reasoning within fuzzy description logics. Journal of Artificial Intelligence Research, 2001.

[11] World Wide Web Concortium. XML. http://www.w3.org/XML/Schema.

[12] Peter F. Patel-Schneider and Bill Swartout.Description logic specification from the KRSS effort,1993.