ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΑΣΥΝΔΕΣΗ ΟΝΤΟΛΟΓΙΚΗΣ ΓΝΩΣΗΣ ΜΕ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΣΕ ΠΕΡΙΒΑΛΛΟΝ ΑΒΕΒΑΙΟΤΗΤΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΔΕΣΠΟΙΝΑΣ Ι. ΜΑΓΚΑ

Επιβλέπων : Γιώργος Στάμου
Λέκτορας Ε.Μ.Π.

Αθήνα, Ιούλιος 2008

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Διασύνδεση οντολογικής γνώσης με βάσεις δεδομένων σε περιβάλλον αβεβαιότητας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΔΕΣΠΟΙΝΑΣ ΜΑΓΚΑ

Επιβλέπων : Γιώργος Στάμου
Λέκτορας Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 18η Ιουλίου 2008.

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

Αθήνα, Ιούλιος 2008

(Υπογραφή)

.....

ΔΕΣΠΟΙΝΑ ΜΑΓΚΑ

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

Περίληψη

Ο σκοπός της διπλωματικής εργασίας ήταν η μελέτη μεθόδων σύνδεσης βάσεων δεδομένων και οντολογικής γνώσης σε εφαρμογές στις οποίες υπεισέρχεται αβεβαιότητα. Στόχος είναι να δίδεται η δυνατότητα στο μηχανικό γνώσης να μετατρέπει τα περιεχόμενα μιας βάσης δεδομένων σε ένα σύνολο ασαφών ισχυρισμών, στο πλαίσιο μιας οντολογικής γνώσης. Με τον τρόπο αυτό αυτοματοποιείται η διαδικασία αναπαράστασης των περιεχομένων της βάσης δεδομένων σε ασαφείς ισχυρισμούς.

Πιο συγκεκριμένα, στο πλαίσιο της παρούσας διπλωματικής εργασίας υλοποιήθηκε ένα σύστημα το οποίο επιτρέπει στο χρήστη αρχικά να αντιστοιχίσει τα ιδιοχαρακτηριστικά και τους πίνακες μιας βάσης δεδομένων με τις κλάσεις και ιδιότητες μιας οντολογίας. Στη διασύνδεση αυτή βασίζεται το σύστημα για την παραγωγή των ασαφών ισχυρισμών. Ο χρήστης ορίζει ένα σύνολο από συναρτήσεις συμμετοχής, προκειμένου να καθορίσει με ακρίβεια τον τρόπο με τον οποίο θα εισαχθεί η ασάφεια στους ισχυρισμούς. Μετά την είσοδο των παραπάνω πληροφοριών το σύστημα αναλαμβάνει να διατυπώσει την πληροφορία που περιέχεται στη βάση δεδομένων σε μορφή ισχυρισμών της ορολογίας που περιγράφεται στην οντολογία. Οι ισχυρισμοί αυτοί που παράγονται παρουσιάζονται στο χρήστη μέσα από τη διαπροσωπεία του συστήματος. Επιπλέον αποθηκεύονται σε αρχείο στο οποίο έχει πρόσβαση ο χρήστης.

Λέξεις Κλειδιά: Αναπαράσταση γνώσης, Βάσεις Δεδομένων, οντολογίες, ασαφή σύνολα, ασαφείς γλώσσες περιγραφικής λογικής, fuzzy OWL, συναρτήσεις συμμετοχής, σώμα ορολογιών, σώμα ισχυρισμών.

Abstract

The main objective of this diploma thesis was the study of methods, which connect databases and ontological knowledge under the presence of uncertainty. The aim is to give to the knowledge engineer the possibility to convert database content into a set of fuzzy assertions, as a part of an ontological knowledge. This way, the process of representing databases' content via fuzzy assertions is being automatized.

A system was developed for the purposes of the present diploma thesis. This system, initially, permits the user to correspond the attributes and tables of a database to the classes and properties of a TBox. This mapping is used by the system during the creation of the assertions (ABox). The user defines a set of membership functions in order to determine with accuracy how fuzziness will be inserted into the assertions. After the above interaction between the user and the system, the system expresses the information of the database content using assertions, which are based on the terminologies found in the ontology. The produced fuzzy assertions are presented to the user via system's interface. Moreover, the user can access a file where the produced assertions are stored.

Key words: Knowledge representation, databases, ontologies, fuzzy sets, fuzzy description logics languages, fuzzy OWL, membership functions, TBox, ABox.

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο εργαστήριο Ψηφιακής Επεξεργασίας Σημάτων και Εικόνας του Εθνικού Μετσοβίου Πολυτεχνείου. Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα υπεύθυνο της διπλωματικής μου εργασίας Γιώργο Στάμου για την καθοριστική καθοδήγηση του, τα μέλη του εργαστηρίου Γιώργο Στοΐλο και Νίκο Σίμου για τις καίριες συμβουλές τους καθώς επίσης και το διπλωματούχο Ηλεκτρολόγο Μηχανικό και Μηχανικό Υπολογιστών Μάριο Μεϊμάρη. Τέλος τους γονείς μου Γιάννη και Κατερίνα και το θείο μου Γιώργο για την έμπρακτη υποστήριξη που μου παρείχαν στο σύνολο και ιδιαίτερα στο τελευταίο εξάμηνο των σπουδών μου.

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Αναπαράσταση γνώσης και συλλογιστική

Στη σύγχρονη εποχή η πληθώρα των πληροφοριών που διακινούνται καθιστά επιτακτική την ανάγκη ανάλυσης και κατανόησης τους, ώστε να μας είναι χρήσιμες. Εξαιτίας της αυξημένης πολυπλοκότητας που απαιτεί η διαχείριση ενός τόσο μεγάλου όγκου πληροφοριών απαιτούνται μέθοδοι και τεχνικές που προσεγγίζουν το πρόβλημα σημασιολογικά. Έτσι τις τελευταίες δεκαετίες, η τεχνητή νοημοσύνη, που γεννήθηκε στο χώρο της επιστήμης των υπολογιστών, έχει αναδείξει έναν νέο κλάδο, αυτόν της αναπαράστασης γνώσης και συλλογιστικής. Ένας ορισμός που μπορεί να δοθεί για την τεχνητή νοημοσύνη, είναι ότι πραγματεύεται τη μελέτη της ευφυούς συμπεριφοράς όταν αυτή επιτυγχάνεται με υπολογιστικά μέσα. Βασιζόμενοι σε αυτό τον ορισμό, μπορούμε να οριοθετήσουμε την αναπαράσταση γνώσης και συλλογιστική, ως τον τομέα εκείνο της τεχνητής νοημοσύνης που ασχολείται με το πώς ένας (ηλεκτρονικός) πράκτορας διατυπώνει και χρησιμοποιεί τη γνώση που διαθέτει για τη λήψη αποφάσεων.

Ο κλάδος της αναπαράστασης γνώσης και συλλογιστικής επικεντρώνεται σε 3 βασικές έννοιες: τη γνώση, την αναπαράσταση της και τη συλλογιστική. Η γνώση, αποτελεί μια αρκετά αφηρημένη έννοια, η οποία είναι δύσκολο να οριστεί. Ωστόσο μία προσεγγιση είναι ότι αποτελεί ένα είδος συσχέτισης ανάμεσα σε ένα υποκείμενο, κάποιον που *γνωρίζει,* και μία πρόταση, η οποία αποτελεί το περιεχόμενο της γνώσης. Ένα σχετικό παράδειγμα αποτελεί η φράση «Ο Κώστας γνωρίζει ότι ο Νίκος θα συμμετέχει στη συναυλία». Η αναπαράσταση

πάλι, πολυ αδρομερώς, μπορεί να περιγραφεί ως μία αντιστοίχιση ανάμεσα σε δύο κόσμους: ο ένας κόσμος αντιπροσωπεύει κομμάτια του δεύτερου, όμως με περισσότερο αναγνωρίσιμο και παραστατικό τρόπο. Π.χ. το σύμβολο € χρησιμοποιείται για να εκφράσει την έννοια της κοινής ευρωπαϊκής χρηματικής μονάδας. Τέλος, αναφερόμενοι στη συλλογιστική, αυτή σχετίζεται με την επεξεργασία συμβόλων τα οποία αντιπροσωπεύουν ένα σύνολο από προτάσεις, προκειμένουν να επιτύχουν την παραγωγή νέων συμβόλων. Η λογική επαγωγή αποτελεί τη ραχοκοκκαλιά των περισσότερων αλγορίθμων που προέρχονται από αυτόν τον τομέα.

Βασική εφαρμογή του παραπάνω κλάδου είναι η ανάπτυξη *συστημάτων βασισμένων σε γνώση (knowledge-based systems)*. Κύριο χαρακτηριστικό αυτων των συστημάτων είναι η παρουσία μιας *βάσης γνώσης,* δηλαδή μιας συλλογής δομών με συμβολικό χαρακτήρα που είναι αποθηκευμένα στο σύστημα και καθοδηγούν τη διαδικασία συλλογιστικής.

1.2 Αντικείμενο διπλωματικής

Άμεση συνέπεια της ραγδαίας ανάπτυξης που έχει γνωρίσει ο κλάδος αυτός τα τελευταία χρόνια είναι η εμφάνιση πολλών προβλημάτων και η ανάγκη διερεύνησης και επίλυσης τους. Καθώς πρόκειται για κλάδο ο οποίος έχει αρχίσει να εφαρμόζεται ευρέως τα τελευταία μόνο χρόνια, η αναπαράσταση των πληροφοριών προηγουμένως γινόταν με τη χρήση διαφορετικών τεχνολογιών. Έτσι, άμεσο λογικό επακόλουθο είναι η ανάγκη γεφύρωσης των διαφορετικών αυτών τρόπων αναπαράστασης της γνώσης με αυτόματο και συστηματικό τρόπο.

Όπως θα περιγραφεί και με περισσότερη λεπτομέρεια στο κομμάτι του θεωρητικού υποβάθρου ένας από τους πλέον δημοφιλείς και σύγχρονους τρόπους αναπαράστασης της γνώσης που καταλαμβάνει όλο και περισσότερο έδαφος, είναι οι *οντολογίες*. Στο παρελθόν ο κατεξοχήν τρόπος κωδικοποίησης και αποθήκευσης των πληροφοριών που εξακολουθεί να υφίσταται σε πολύ μεγάλο βαθμό και σήμερα είναι οι σχεσιακές βάσεις δεδομένων. Πρόκειται για δύο διαφορετικές τεχνολογίες, οι οποίες έχουν όχι μόνο ανταγωνιστικό αλλά και συμπληρωματικό χαρακτήρα. Άρκετα συχνά αναφέρονται στις ίδιες πληροφορίες, αλλά από διαφορετική οπτική γωνία γι'αυτό και ανακύπτει άμεσα η ανάγκη ανάπτυξης συστημάτων που θα ενοποιούν τους δύο διαφορετικούς τρόπους αναπαράστασης. Τα συστήματα θα λαμβάνουν υπόψη τους τη διαφορετικότητα των αναπαραστάσεων και θα εκτελούν τις αντίστοιχες μετατροπές. Επιπρόσθετα, η δυσκολία περιγραφής με απόλυτη μαθηματική ακρίβεια του κόσμου που μας περιβάλλει έχει καταστήσει τα ασαφή σύνολα ως έναν από τους πλέον εύστοχους τρόπους περιγραφής του υλικού και κοινωνικού μας περίγυρου σε γλώσσα κατανοητή από τους υπολογιστές..

Αρκετά συχνά ο μηχανικός γνώσης έρχεται σε επαφή με περιπτώσεις όπου οι πληροφορίες είναι αποθηκευμένες σε μία σχεσιακή βάση δεδομένων. Ας υποθέσουμε ότι αναφερόμαστε στο σύνολο των πολιτών μιας χώρας και μας ενδιαφέρει η οικονομική τους κατάσταση. Ο τρόπος με τον οποίο είναι συγκροτημένες οι πληροφορίες στη σχεσιακή βάση δεδομένων επιτρέπει στο μηχανικό γνώσης να διαθέτει για κάθε πολίτη μία μόνο αριθμητική τιμή, αυτή του ετήσιου εισοδήματος του. Μία οντολογία για το ίδιο θέμα περιλαμβάνει ισχυρισμούς οι οποίοι συνδέουν κάθε πολίτη της χώρας μοναδικά με μία έννοια π.χ. εάν είναι οικονομικά αδύνατος, εύπορος ή ανήκει στα μεσαία οικονομικά στρώματα. Ωστόσο ο μηχανικός γνώσης δε διαθέτει κάποιο άμεσο και αυστηρό τρόπο προκειμένου να απεικονίσει πληροφορίες που αναφέρονται σε ένα συγκεκριμένο πολίτη από τη σχεσιακή βάση δεδομένων στην οντολογία. Επιπρόσθετα οι ισχυρισμοί μιας οντολογίας δε δύνανται να περιγράψουν με αρκετά ρεαλιστικό τρόπο τον πραγματικό κόσμο. Η καθημερινότητα μας υποδεικνύει ότι ένα άτομο μπορεί να ανήκει σε διαφορετικές κλάσεις και μάλιστα σε διαφορετικό βαθμό σε κάθε μία από αυτές. Όταν έχουμε να κάνουμε με αριθμητικές τιμές που βρίσκονται στο μεταίχμιο είναι αδύνατο να κατατάξουμε σε μία συγκεκριμένη κατηγορία ένα άτομο. Για παράδειγμα ένας πολίτης με εισόδημα 20,000€ μπορεί να θεωρηθεί ότι ανήκει είτε στην κλάση του μεσαίου εισοδήματος είτε σε αυτή των οικονομικά αδυνάτων και μάλιστα σε *διαφορετικό βαθμό* σε κάθεμια από αυτές. Οι ασαφείς ισχυρισμοί μιας οντολογίας επιλύουν αυτό το πρόβλημα. Ωστόσο παραμένει το ζήτημα απεικόνισης της πληροφορίας από τη βάση δεδομένων στην οντολογία. Η παρούσα διπλωματική εργασία επιχειρεί να προσεγγίσει το παραπάνω πρόβλημα. Αντικειμενικός σκοπός είναι η διασύνδεση των πληροφοριών που μια βάση δεδομένων διαθέτει με τη γνώση μιας οντολογίας υπό το πρίσμα της αβεβαιότητας και η παραγωγή οντολογικής γνώσης που βασίζεται σε αυτή την ενοποίηση. Στην κατεύθυνση αυτή προτείνουμε ένα φορμαλιστικό τρόπο περιγραφής αυτής της ενοποίησης και παρουσιάζουμε ένα σύστημα που υλοποιεί αυτή τη διαδικασία.

1.3 Οργάνωση κειμένου

Η διπλωματική εργασία οργανώνεται με τον ακόλουθο τρόπο:

- Το κεφάλαιο 2 συνοψίζει τα θεωρητικά θεμέλια πανω στα οποία στηρίχτηκε η διερεύνηση και η ανάπτυξη του θέματος της παρούσας διπλωματικής εργασίας .
- Το κεφάλαιο 3 παρουσιάζει με επίσημο τρόπο τη μεθοδολογία που προτείνουμε για την διασύνδεση των δύο τρόπων αναπαράστασης γνώσης με τη χρήση αβεβαιότητας.
- Το κεφάλαιο 4 περιγράφει την αρχιτεκτονική του συστήματος, τα υποσυστήματα που το αποτελούν, πώς αυτά συνεργάζονται μεταξύ τους, πώς επιμερίζονται οι

διαφορετικές λειτουργίες, οι βιβλιοθήκες που χρησιμοποιήθηκαν κατά την ανάπτυξη του συστήματος καθώς και οι πλατφόρμες και τα προγραμματιστικά εργαλεία με τα οποία έγινε αυτή.

- Το κεφάλαιο 5 επιδεικνύει τη συμπεριφορά του συστήματος με τέτοιον τρόπο ώστε να περιγράφεται με ακρίβεια και παραστατικότητα στο μέλλοντα χρήστη-μηχανικό γνώσης πώς θα αλληλεπιδράσει με αυτό.
- Το κεφάλαιο 6 προσφέρει μια λεπτομερή και διεξοδική περιγραφή του λογισμικού του συστήματος και συγκεκριμένα των επιμέρους κλάσεων που το αποτελούν, εφόσον αναπτύχθηκε με χρήση της αντικειμονστρεφούς γλώσσας προγραμματισμού java.
- Το κεφάλαιο 7 συνοψίζει την εργασία εκθέτοντας τα συμπεράσματα που αποκομίσαμε καθώς και πιθανές μελλοντικές επεκτάσεις.
- Το κεφάλαιο 8 παρουσιάζει το σύνολο της βιβλιογραφίας που χρησιμοποιήθηκε για την εκπονηση της παρούσας διπλωματικής εργασίας.

2

Θεωρητικό υπόβαθρο

2.1 Το σημασιολογικό διαδίκτυο

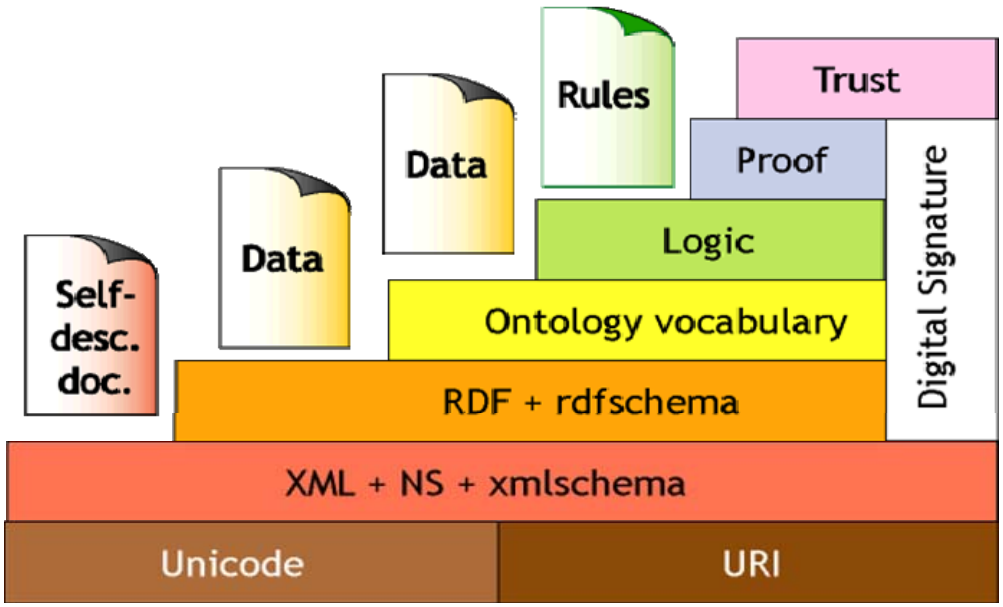
Η μετεξέλιξη του World Wide Web στο Semantic Web αποτελεί ένα τεχνολογικό στόχο που πρώτη φορά τέθηκε ξεκάθαρα το 2001 από τον Tim Berners-Lee. Πρόκειται ουσιαστικά για την αναδιάρθρωση της λειτουργικότητας του διαδικτύου, μετατρέποντας το από έναν ωκεανό ασύνδετων πληροφοριών σε μία σημασιολογικά ταξινομημένη δεξαμενή γνώσης. Πρακτικός σκοπός αυτής της προσπάθειας είναι η βελτίωση της καθημερινότητας των ανθρώπων αναθέτοντας σε προγράμματα λογισμικού τη διεκπεραίωση εργασιών με πιο αποτελεσματικό, γρήγορο και ξεκούραστο τρόπο.

Το σημασιολογικό διαδίκτυο αποσκοπεί στο να εξάγει σταδιακά τις πληροφορίες που βρίσκονται σήμερα διασκορπισμένες στο διαδίκτυο και να τις οργανώσει με τέτοιο τρόπο έτσι ώστε να είναι επεξεργάσιμες από τις μηχανές που θα αναλάβουν αυτόν τον ρόλο. Με τον τρόπο αυτό θα αποτελέσουν τμήμα ενός γενικότερου οικοδομήματος γνώσης στο οποίο η πρόσβαση θα γίνεται με ομοιόμορφο και ενιαίο τρόπο. Άμεση συνέπεια αυτής της ομοιομορφίας θα είναι η διαφάνεια μεταξύ υπηρεσιών διαφορετικού χαρακτήρα και η κατ' επέκταση συνεργασία μεταξύ τους χωρίς την ανάγκη ανθρώπινης παρέμβασης για συντονισμό των επιμέρους καθηκόντων. Έτσι οι μηχανές δε θα επεξεργάζονται πλέον μηχανικά τα έγγραφα και τα δεδομένα αλλά θα τα «κατανοούν» ταυτοποιώντας τα με συγκεκριμένη σημασία και περιεχόμενο.

Παρακάτω δίδεται ένα παράδειγμα, δανεισμένο από τη [20], το οποίο σκιαγραφεί τις μελλοντικές δυνατότητες του σημασιολογικιού διαδικτύου με άμεσο και παραστατικό τρόπο. Ας υποθέσουμε ότι μία ηλικιωμένη κυρία έχει μόλις επισκεφθεί το γιατρό της ο οποίος της έχει συστήσει να ακολουθήσει ένα ειδικό πρόγραμμα φυσιοθεραπείας, δύο φορές την εβδομάδα. Ο γιος της, Πέτρος, και η κόρη της, Λουκία, θα αναλάβουν τη μετακίνηση της μητέρας τους και γι' αυτό το λόγο πρέπει πρώτα να συνεννοηθούν. Εκμεταλλευόμενοι τις ευκολίες που το Σημασιολογικό Διαδίκτυο τους προσφέρει θέτουν το εξής αίτημα σε ειδικά προγράμματα τα οποία ονομάζονται πράκτορες λογισμικού:να βρεθεί ιατρικό κέντρο το οποίο να προσφέρει τις συγκεκριμένες υπηρεσίες, σε ακτίνα συγκεκριμένης χιλιομετρικής απόστασης από το σπίτι της ασθενούς και το οποίο να συμβαδίζει με το γεμάτο πρόγραμμα της Λουκίας και του Πέτρου όπως αυτό έχει διαμορφωθεί μέχρι στιγμής. Επιπλέον το εν λόγω ιατρικό κέντρο θα πρέπει να διαθέτει έγκυρη πιστοποίηση για την ποιότητα των υπηρεσιών του καθώς και να είναι συμβατό με την ασφαλιστική εταιρεία της μητέρας τους. Μέσα σε λίγα λεπτά εμφανίζεται ένα προτεινόμενο πλάνο που με προσοχή εξετάζουν ο Πέτρος και η Λουκία. Σύμφωνα με αυτό το πλάνο όμως ο Πέτρος θα πρέπει να διασχίσει ένα κεντρικό δρόμο της πόλης σε ώρα αιχμής, προοπτική όχι επιθυμητή γι' αυτόν. Προκειμένου να το αποφύγει θέτει εκ νέου το αίτημα στον πράκτορα με αυστηρότερα κριτήρια τοποθεσίας και χρόνου, επιτρέποντας του όμως να είναι περισσότερο ελαστικός σε άλλες παραχωρήσεις. Όντως μετά από λίγη ωρά εμφανίζεται ένα αναδιαμορφωμένο πλάνο το οποίο ικανοποιεί και τους δύο από άποψη τοποθεσίας και χρόνου, αλλά δεν είναι συμβατό με την ασφαλιστική εταιρεία και επιπλέον υποχρεώνει τον Πέτρο να αναβάλει δύο από τις *λιγότερο σημαντικές* συναντήσεις του. Ωστόσο αποτελεί μία προτιμώτερη επιλογή την οποία και αποφασίζουν να υιοθετήσουν τα δύο αδέλφια. Σίγουρα πρόκειται για ένα σενάριο το οποίο συγκρινόμενο με τη σημερινή πραγματικότητα εξοικονομεί σημαντική ποσότητα χρόνου (εάν π.χ. έπρεπε να αναζητήσουν μια λίστα με τα ιατρικά κέντρα και να τηλεφωνήσουν σε κάθε ένα από αυτά) και ακόμη διασφαλίζει ότι ερευνήθηκαν όλα τα δυνατά σενάρια για την εύρεση της βέλτιστης λύσης.

Κομβικό ρόλο στη διάδοση του σημασιολογικού διαδικτύου διαδραματίζουν τα προγράμματα εκείνα που ονομάζουμε πράκτορες λογισμικού. Στους πράκτορες λογισμικού ανατίθεται, όπως και στην καθημερινή ζωή, διεκπεραίωση εργασιών που απαιτούν ειδική διαδικασία αναζήτησης της γνώσης. Μία επιπρόσθετη εφαρμογή του Σημασιολογικού Διαδικτύου είναι η συμβολή που αναμένεται να έχει στη καθιέρωση μίας νέας γενιάς μηχανών αναζήτησης. Οι εν λόγω μηχανές αναζήτησης θα βασίζουν τη λειτουργία τους όχι στην καθιερωμένη λεξικογραφική συσχέτιση λέξεων-κλειδιών, αλλά σε μία αναζήτηση προσανατολισμένη στη σημασιολογία η οποία εξ'ορισμού θα εξαλείφει πιθανές δισημίες και αποτυχημένες ανακλήσεις εγγράφων.

Η παρακάτω εικόνα, παρμένη από μία παρουσίαση του Tim Berners-Lee το 2000 ([1]) αναπαριστά την αρχιτεκτονική του εγχειρήματος του σημασιολογικού διαδικτύου. Τεχνολογίες που σχετίζονται με την αναπαράσταση της γνώσης και τη συλλογιστική απαρτίζουν κάθε στρώμα. Βασικό χαρακτηριστικό κάθε επιπέδου είναι η άμεση εξάρτηση του από το αμέσως κατώτερο επίπεδο στο οποίο βασίζεται και το οποίο επεκτείνει.



Σχήμα 1. Αρχιτεκτονική του σημασιολογικού διαδικτύου.

Στο κατώτερο επίπεδο βρίσκεται το διεθνώς καθιερωμένο πρότυπο κωδικοποίησης χαρακτήρων Unicode το οποίο καλύπτει όλες τις γλώσσες του πλανήτη με γραπτή υπόσταση. Επιπλεόν στο επίπεδο αυτό συναντάμε το URI, που αποτελεί ένα σύστημα διευθυνσιοδότησης και ταυτοποίησης όλων των διαδικτυακών (και μη) πόρων. Πρόκειται για ένα σύστημα που επιδέχεται επεκτάσεις με τέτοιο τρόπο ώστε να καθορίζει μία ένα προς ένα αντιστοίχιση μεταξύ οποιωνδήποτε αντικειμένων του πραγματικού κόσμου (είτε πρόκειται για ηλεκτρονικά έγγραφα είτε πρόκειται για υλικά αντικείμενα) και μοναδικά καθορισμένων συμβολοσειρών.

Το επόμενο επίπεδο σχετίζεται με την XML, μία ευρέως χρησιμοποιούμενη γλώσσα για την αναπαράσταση δεδομένων στο διαδίκτυο. Το βασικό της πλεονέκτημα είναι η δυνατότητα που δίνει στο χρήστη να ορίζει το δικό του λεξιλόγιο, χωρίς να υπάρχει κάποιο παγκόσμιο δεσμευτικό πρότυπο.

Προχωρώντας προς τα πάνω συναντάμε την RDF και rdfschema δύο γλώσσες οι οποίες αποτελούν επέκταση της XML. Η RDF αναπαριστά ισχυρισμούς της μορφής υποκείμενο-ιδιότητα-αντικείμενο, ενώ η rdfschema είναι μία εμπλουτισμένη εξέλιξη της RDF που

προσθέτει περαιτέρω δυνατότητες στην αναπαράσταση των ισχυρισμών όπως για παράδειγμα ιεραρχία κλάσεων, ιεραρχία ιδιοτήτων ή περιορισμούς στο πεδίο ορισμού και στο πεδίο τιμών μιας ιδιότητας.

Κατόπιν συναντάμε τις οντολογίες στις οποίες θα αναφερθούμε εκτενώς αργότερα και στη συνέχεια μεταβαίνουμε στο λογικό επίπεδο το οποίο επιτρέπει τη λογική συσχέτιση μεταξύ διαφορετικών διαδικτυακών πόρων καθώς και τη χρησιμοποίηση κανόνων οι οποίοι απεικονίζουν τη σημασιολογική συγγένεια των δεδομένων.

Ακολουθεί το επίπεδο των αποδείξεων που επιτρέπει την εξαγωγή συμπερασμάτων χρησιμοποιώντας τα υπάρχοντα δεδομένα. Επιπλέον προσφέρει στο χρήστη τη δυνατότητα επισκόπησης των λογικών βημάτων που μεσολάβησαν μεταξύ των συμπερασμάτων, αυξάνοντας την εμπιστοσύνη του χρήστη απέναντι στο σύστημα. Δεδομένου του αυξημένου βαθμού εχεμύθειας που απαιτούν οι κάθε λογής εφαρμογές του διαδικτύου η παρουσία των ψηφιακών υπογραφών είναι έντονη στα 4 τελευταία επίπεδα. Εγγυάται ασφάλεια στις συνάλλαγες (οικονομικές ή άλλου είδους) και πιστοποίηση για την ταυτότητα των συμμετεχόντων.

Τέλος το τελευταίο επίπεδο αναφέρεται στον έλεγχο της αξιοπιστίας ο οποίος είναι απαραίτητος προκειμένου να εμπιστευτούν και να χρησιμοποιήσουν οι χρήστες τις υπηρεσίες που το Σημασιολογικό Διαδίκτυο προσφέρει. Πρόκειται ουσιαστικά για ηλεκτρονικά έγγραφα που έχουν προκύψει ως αποτέλεσμα κάποιων συγκεκριμένων ενεργειών τις οποίες και πιστοποιούν. Έτσι εάν προκύψει θέμα εγκυρότητας θα είναι εύκολο χρησιμοποιώντας τα έγγραφα και ακολουθώντας μία σειρά υπολογιστικών βημάτων βασισμένα σε αυτά να επαληθευθεί η ισχύς των στοιχείων που περιέχουν.

Όπως ήδη αναφέραμε ο βασικός τρόπος αναπαράστασης της πληροφορίας στα πλαίσια του Σημασιολογικού Διαδικτύου είναι οι *οντολογίες,* που αποτελούν κεντρικό θέμα της επόμενης ενότητας.

2.2 Οντολογίες

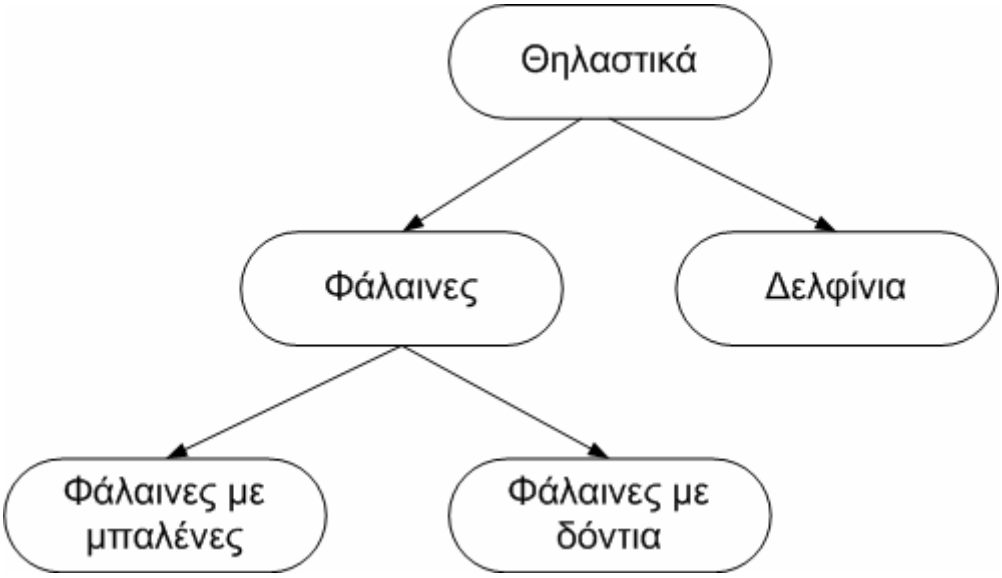
Η οντολογία για πρώτη φορά εμφανίστηκε ως ένας κλάδος της φιλοσοφίας και συγκεκριμένα της μεταφυσικής, ο οποίος εξετάζει την ύπαρξη των πραγμάτων, εντάσσοντας τα σε κατηγορίες και περιγράφοντας τις σχέσεις που αναπτύσσονται μεταξύ τους. Στη συνέχεια υιοθετήθηκε σαν ορολογία από την επιστήμη των υπολογιστών και συγκεκριμένα από τους μηχανικούς γνώσης ως ένας συγκεκριμένος τρόπος αναπαράστασης της γνώσης. Η ευέλικτη φύση τους καθώς και η αυστηρή μαθηματική τους περιγραφή έχουν καταστήσει τις οντολογίες αναπόσπαστο κομμάτι της τεχνολογίας του σημασιολογικού διαδικτύου. Ένας ορισμός διατυπωμένος από τον Tom Gruber είναι ότι μια οντολογία ουσιαστικά συνιστά «μια

περιγραφή εννοιών και συσχετίσεων όπως την αντιλαμβάνεται ένας πράκτορας (λογισμικού) ή μία κοινότητα από πράκτορες».Συνεχίζοντας πρόκειται «για ένα σύνολο από ορισμούς που προέρχονται από ένα αυστηρά διατυπωμένο λεξιλόγιο».

Βασικός σκοπός χρήσης των οντολογιών είναι η δυνατότητα που μας προσφέρουν για συλλογιστική. Εφόσον αυτό που τις αποτελεί είναι μία αυστηρά οργανωμένη γνώση, είναι λογικό μία τέτοια δομή να είναι η πλέον κατάλληλη είσοδος για έναν αλγόριθμο που εκτελεί ελέγχους συνέπειας και απαντά σε λογικά ερωτήματα.

Ας προχωρήσουμε στα κύρια συστατικά που απαρτίζουν μια οντολογία. Αρχικά έχουμε τα άτομα τα οποία αντιστοιχούν σε συγκεκριμένα υπαρκτά αντικείμενα, είτε υλικά είτε αφηρημένα. Τέτοια παραδείγματα μπορεί να είναι άνθρωποι, σπίτια, περιοδικά, τραγούδια, καλλιτεχνικά ρεύματα ή μία συναυλία. Ωστόσο δεν είναι υποχρεωτικό για μία οντολογία να περιλαμβάνει και άτομα. Δεδομένου όμως ότι ένας από τους κύριους λόγους ύπαρξης της οντολογίας, είναι η ικανότητα της να συνάγει συμπεράσματα μέσα από λογικές διεργασίες για τα άτομα (όπως π.χ. να τα ταξινομεί), είναι μάλλον χρήσιμο να συγκαταλέγεται ένας αριθμός ατόμων στα συστατικά της.

Οι κλάσεις αποτελούν μία από τις δύο κυριότερες έννοιες που συναντάμε σε μια οντολογία. Πρόκειται για μία συλλογή ομοειδών αντικειμένων που παρουσιάζουν κοινά γνωρίσματα και χαρακτηριστικά έτσι ώστε να θεωρείται ότι ανήκουν στην ίδια κατηγορία. Ένα από τα κύρια χαρακτηριστικά που εμφανίζουν οι κλάσεις είναι η ιδιότητα του εγκλεισμού. Πρόκειται ουσιαστικά για ένα σχήμα κληρονομικότητας κατά το οποίο ένα αντικείμενο ανήκει σε περισσότερες από μία κλάσεις με το σύνολο γνωρισμάτων της μίας κλάσης να αποτελεί υποσύνολο γνωρισμάτων της άλλης κλάσης. Για παράδειγμα εάν έχουμε δύο κλάσεις Α και Β τότε εάν η κλάση Β περιλαμβάνει όλα τα άτομα της κλάσης Α, προσθέτοντας τους περαιτέρω χαρακτηριστικά και εξειδικεύοντας τα, λέμε ότι η κλάση Α εγκλείει την κλάση Β ή αλλιώς ότι η κλάση Α είναι υπερκλάση της Β και η Β υποκλάση της Α. Σε ορισμένες περιπτώσεις είναι επιτρεπτό το φαινόμενο της πολλαπλής κληρονομικότητας κατά το οποίο μία κλάση δύναται να εγκλείεται σε περισσότερες από μία κλάσεις. Με τον τρόπο αυτό δύναται να δημιουργηθούν αρκετά περίπλοκες αλλά και εξαιρετικά εκφραστικές ταξονομίες, οι οποίες παρουσιάζουν με ευστοχία πολλά από τα στοιχεία του κόσμου μας. Ένα απλο παράδειγμα, που δίνεται και σχεδιαγραμματικά είναι το παρακάτω:



Σχήμα 2. Παράδειγμα ιεραρχίας κλάσεων

Ξεκινάμε με την κλάση των θηλαστικών. Τα μέλη αυτής της κλάσης διαθέτουν ένα σύνολο από καλά ορισμένα χαρακτηριστικά όπως ότι πρόκειται για ζώα, με συγκεκριμένο τρόπο αναπαραγωγής κλπ. Στη συνέχεια τα μέλη της κλάσης των φαλαινών είναι μεν θηλαστικά ωστόσο διαθέτουν και επιπλέον χαρακτηριστικά που σχετίζονται με την ανατομία τους κλπ. Αυτή η σχέση κληρονομικότητας συνεχίζεται για ένα ακόμη επίπεδο κατά το οποίο η πατρική κλάση είναι αυτή των φαλαινών που εξειδικεύεται σε δύο επιμέρους υποκλάσεις: τις φάλαινες με δόντια και τις φάλαινες με μπαλένες. Τέτοιου είδους ταξονομίες μπορούν να διατυπωθούν για όλες τις κατηγορίες αντικειμένων που αποτελούν μέρος του περιβάλλοντος μας, ανεξάρτητα από το αν αυτά έχουν υλική υπόσταση ή όχι.

Η δεύτερη από τις δύο κυριότερες έννοιες των οντολογιών είναι αυτή των συσχετίσεων μεταξύ δύο διαφορετικών ή ίδιων κλάσεων. Μία τέτοια περίπτωση που ήδη έχουμε συναντήσει είναι π.χ. για δύο κλάσεις Α και Β «η κλάση Α είναι υπερκλάση της κλάσης Β» ή «η κλάση Β είναι υποκλάση της κλάσης Α». Γενικότερα η σύνδεση μεταξύ δύο οποιωνδήποτε αντικειμένων γίνεται χρησιμοποιώντας σχέσεις, οι οποίες ουσιαστικά περιγράφουν την τοποθέτηση ενός αντικειμένου O_1 στον κόσμο ως προς ένα άλλο αντικείμενο O_2. Στο παραπάνω παράδειγμα ένα παράδειγμα σχέσης θα μπορούσε να είναι η «ζει σε» η οποία καθορίζει τον τόπο διαμονής κάποιου από τα παραπάνω θηλαστικά. Πρόκειται για μία σχέση η οποία συνδέει ένα αντικείμενο της κλάσης *θηλαστικά* με ένα αντικείμενο της κλάσης *χώρα*. Ανάλογα, είναι δυνατόν να εμφανίζονται σχέσεις μεταξύ αντικειμένων της ίδιας κλάσης, όπως «έχει μεγάλο βαθμό γενετικής ομοιότητας» για την κλάση των θηλαστικών και «συνορεύει με» για την κλάση των χωρών. Οι σχέσεις

προσφέρουν στις οντολογίες τη δυνατότητα να ορίσουν μοντέλα πολυ πιο πλούσια σημασιολογικά.

Οι οντολογίες επομένως εξασφαλίζουν το απαραίτητο θεωρητικό υπόβαθρο προκειμένου να δημιουργηθούν εργαλεία λογισμικού που θα προσφέρουν μία σειρά από υπηρεσίες συλλογιστικής. Έτσι ελέγχοντας το ιεραρχικό σχήμα μιας οντολογίας θα έχουμε τη δυνατότητα να αποφανθούμε κατά πόσο ένα αντικείμενο ανήκει ή όχι σε μία κλάση. Επιπλέον θα μπορεί να γίνει εντοπισμός ενδεχομένων ασυνεπειών που περιέχει η οντολογία οι οποίες οφείλονται σε αντιφατικούς περιορισμούς που έχουν τεθεί.

Πέρα από τη θεωρητική υπόσταση τους, οι οντολογίες έχουν αναλάβει ένα πολύ σημαντικό πρακτικό ρόλο στο χώρο του Σημασιολογικού Διαδικτύου. Αυτός είναι η μοντελοποίηση με δοδημένο και οργανωμένο τρόπο ενός τομέα γνώσης έτσι ώστε να υπάρχει η δυνατότητα περιγραφής αντικειμένων σε μεγάλο βαθμό λεπτομέρειας και σαφήνειας. Ωστόσο το πιο σημαντικό είναι η ύπαρξη ενός κοινού λεξιλογίου που χρησιμοποιείται από όλα τα μέλη της εκάστοτε κοινότητας αποτρέποντας έτσι τις νοηματικές αμφισημίες και παρεξηγήσεις. Έχουν ήδη δημιουργηθεί από τους ειδήμονες κάθε κλάδου όπως π.χ. βιολογία, ανατομία, βιοϊατρική, πολιτιστική κληρονομιά κ.α. μία σειρά από οντολογίες οι οποίες εξυπηρετούν τις ανάγκες κάθε τομέα. Απώτερος σκοπός είναι κάποια στιγμή στο μέλλον να ενοποιηθούν οι επιμέρους αυτές οντολογίες συνθέτοντας μία ενιάια οντολογία που θα περιγράφει τον κόσμο σε όλες του τις εκφάνσεις.

Εξαιτίας της πληθώρας των δυνατοτήτων και των πλεονεκτημάτων που μια οντολογία προσφέρει, έχει καθιερωθεί ως ο πλέον κατάλληλος τρόπος αναπαράστασης γνώσης για το Σημασιολογικό Διαδίκτυο. Η επόμενη άμεση ανάγκη που προέκυψε ήταν η δημιουργία και προτυποποίηση μιας γλώσσας η οποία περιγράφει τις οντολογίες με τυπικό τρόπο, έτσι ώστε να είναι εφικτή η επεξεργασία των δεδομένων απευθείας από τις μηχανές. Όντως το 2004 η OWL (Ontology Web Language), προτυποποίηθηκε από το Web Ontology Working Group του World Wide Web Consortium. Πριν συνεχίσουμε με μία περιγραφή της παραπάνω γλώσσας θα ασχοληθούμε με το μαθηματικό της υπόβαθρο, που είναι οι γλώσσες περιγραφικής λογικής.

2.3 Γλώσσες Περιγραφικής Λογικής

Οι γλώσσες περιγραφικής λογικής συνιστούν γλώσσες αναπαράστασης της γνώσης των οποίων τα κύρια χαρακτηριστικά είναι ο αυστηρός μαθηματικός φορμαλισμός, η απλότητα και η κομψότητα. Αποτελούν υποσύνολο του πρωτοβάθμιου κατηγορηματικού λογισμού ο οποίος διαθέτει μεν πλούσια εκφραστικότητα αλλά επίσης και ανεπιθύμητη μη

αποφασισιμότητα. Σκοπός των γλωσσών αυτών είναι η αποθήκευση μιας ποσότητας γνώσης με τυπικό τρόπο έτσι ώστε αυτή όχι μόνο να γίνεται αντιληπτή από τον υπολογιστή αλλά και να χρησιμέυει ως βάση για τη συναγωγή περαιτέρω συμπερασμάτων.

Όπως σε κάθε γλώσσα, έτσι και στις γλώσσες περιγραφικής λογικής η περιγραφή εστιάζεται στο αλφάβητο, το συντακτικό και τη σημασιολογία. Το αλφάβητο, δηλαδή τα δομικά στοιχεία που σχηματίζουν τη γλώσσα, περιλαμβάνει τις *ατομικές έννοιες (atomic concepts),* τους *ατομικούς ρόλους (atomic roles)* ή *σχέσεις (relations)* και τα *άτομα.* Δεν είναι αυστηρά καθορισμένο από την αρχή αλλά ορίζεται απο το χρήστη με βάση τις ανάγκες του και τους σκοπούς της περιγραφής του. Ένα άτομο αντιστοιχεί σε μία οντότητα του οποίου την ύπαρξη θέλουμε να εισάγουμε στη βάση γνώσης χωρίς απαραίτητα να έχει υλική υπόσταση. Η αντιστοίχιση με άτομα του πραγματικού κόσμου θα γίνει στη συνέχεια με την προσθήκη της ερμηνείας, η οποία αφορά στο σημασιολογικό κομμάτι της γλώσσας. Παραδείγματα ατόμων μπορεί να είναι ο Γιώργος, η Μαρία, η (διεύθυνση) Πανεπιστημίου47 και το (αυτοκίνητο) IZA2848. Μία ατομική έννοια ουσιαστικά είναι, με όρους πρωτοβάθμιου κατηγορηματικού λογισμού, ένα κατηγόρημα πρώτου βαθμού ή αλλιώς η απόδοση μιας ιδιότητας σε ένα άτομο. Παραδείγματα ατομικών εννοιών είναι τα Άνθρωπος, Διεύθυνση και Αυτοκίνητο. Ένας ατομικός ρόλος είναι αντίστοιχα ένα κατηγόρημα δευτέρου βαθμού η αλλιώς μία συσχέτιση μεταξύ δύο ατόμων. Ένα παράδειγμα ατομικού ρόλου είναι η σχέση εχειΦιλο. Κατα σύμβαση τα άτομα και οι ατομικές έννοιες ξεκινούν με κεφαλαίο γράμμα ενώ οι ατομικοί ρόλοι με μικρό.

Στη συνέχεια θα προχωρήσουμε στην παρουσίαση του συντακτικού της γλώσσας. Στο σημείο αυτό να επισημάνουμε ότι οι περιγραφικές λογικές δεν είναι μία ενιαία γλώσσα αλλά μία οικογένεια γλωσσών με μεγάλη ποικιλία εκφραστικότητας και πολυπλοκότητας. Όσο πιο εκφραστική είναι μία γλώσσα, δηλαδή όσο περισσότερες δυνατότητες μας προσφέρει να αποτυπώσουμε με ευστοχία τη γνώση, τόσο πιο πλούσιο είναι το συντακτικό της. Θα ξεκινήσουμε με μία απλή γλώσσα την $\mathcal{AL}$ (attributive language) , η οποία περιλαμβάνει τους εξής κατασκευαστές $\{\neg, \sqcap, \forall, \exists\}$. Ακολουθεί ο τυπικός όρισμός του συντακτικού της γλώσσας.

Έστω A μία ατομική έννοια, R ένας ατομικός ρόλος και C, D δύο περιγραφές εννοιών. Οι *περιγραφές εννοιών* για τη γλώσσα $\mathcal{AL}$ ορίζονται από την παρακάτω αφηρημένη σύνταξή:

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

Στον παραπάνω επαγωγικό ορισμό οι έννοιες $\top$ και $\bot$ αντιστοιχούν στην καθολική και κενή έννοια (universal concept, bottom concept) ενώ οι έννοιες $\forall R.C$ και $\exists R.\top$ στον

καθολικό περιορισμό (universal restriction) και περιορισμένο υπαρξιακό περιορισμό (limited existential restriction) αντίστοιχα. Τέλος να επισημάνουμε ότι στην $\mathcal{AL}$ η άρνηση συναντάται μόνο μπροστά από ατομικές έννοιες.

Στη συνέχεια δίνεται η σημασιολογία που αντιστοιχεί στο παραπάνω συντακτικό. Πρόκειται για μία *ερμηνεία* η οποία προσδίδει νόημα στις έννοιες και τους ρόλους που έχουν οριστεί. Η ερμηνεία $\mathcal{I}$ αποτελείται από ένα ζεύγος $(\Delta^{\mathcal{I}},\cdot^{\mathcal{I}})$ όπου $\Delta^{\mathcal{I}}$ είναι ένα μή κενό σύνολο που αποτελείται από τα *αντικείμενα* του κόσμου στον οποίο αναφερόμαστε και ονομάζεται *χώρος ερμηνείας* ενώ το $\cdot^{\mathcal{I}}$ ονομάζεται *συνάρτηση ερμηνείας* και αντιστοιχεί κάθε ατομική έννοια A σε ένα υποσύνολο $A^{\mathcal{I}}$ του $\Delta^{\mathcal{I}}$ και κάθε ατομικό ρόλο R σε ένα υποσύνολο $R^{\mathcal{I}}$ του $\Delta^{\mathcal{I}}\times\Delta^{\mathcal{I}}$.

Μέχρι στιγμής έχουμε ορίσει μέσω της συνάρτησης ερμηνείας τη σημασιολογία μόνο των ατομικών εννοιών και ρόλων. Παρακάτω δίνεται η επέκταση της συνάρτησης ερμηνείας για περιγραφές εννοιών:

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

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

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

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

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

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

Η παραπάνω γλώσσα δεν παρέχει ιδιαίτερα πλούσια εκφραστικότητα. Για το λόγο αυτό εισάγονται μία σειρά από κατασκευαστές οι οποίοι επεκτείνουν τις δυνατότητες τις γλώσσας. Μία τέτοια επέκταση είναι η άρνηση σε περίπλοκες έννοιες. Δηλαδή, για μία σύνθετη έννοια C, η σύνταξη είναι ¬C, ενώ η σημασιολογία $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$. Στο σημείο αυτό να προσθέσουμε ότι ο εμπλουτισμός μιας γλώσσας με έναν επιπλέον κατασκευαστή αντικατοπτρίζεται στην ονομασία της γλώσσας. Έτσι μετά την προσθήκη της παραπάνω άρνησης η γλώσσα ονομάζεται πλέον $\mathcal{ALC}$.

Το επόμενο βήμα είναι να εισάγουμε στη γλώσσα τα αξιώματα μεταβατικών ρόλων. Ένας ρόλος είναι μεταβατικός εάν για R(a,b) και R(b,c) ισχύει επιπλέον R(a,c). Η σύνταξη τους είναι της μορφής **Tr**(R), όπου R είναι ένας ρόλος. Η σημασιολογία τους πάλι είναι: αν

$\{(a,b)^{\mathcal{I}}, (b,c)^{\mathcal{I}}\} \subseteq R^{\mathcal{I}}$ τότε $(a^{\mathcal{I}},c^{\mathcal{I}}) \in R^{\mathcal{I}}$. Μία γλώσσα που διαθέτει αξιώματα μεταβατικών ρόλων περιλαμβάνει στην ονομασία της ένα R^+ στη θέση του δείκτη, δηλαδή η παραπάνω γλώσσα γίνεται $\mathcal{ALC}_{R^+}$. Για να αποφύγουμε τις μακροσκελείς ονομασίες έχει καθιερωθεί οποιαδήποτε γλώσσα περιγραφικής λογικής περιλαμβάνει τους κατασκευαστές της $\mathcal{ALC}_{R^+}$ να συμβολίζεται με το $\mathcal{S}$.

Οι γλώσσες περιγραφικής λογικής επιπλέον μας προσφέρουν τη δυνατότητα να ορίσουμε ένα σύνολο από *αξιώματα ορολογίας*. Τα αξιώματα ορολογίας έχουν είτε τη μορφή $C \sqsubseteq D$, είτε τη μορφή $C \equiv D$ και ονομάζονται αξιώματα υπαγωγής και ισοδυναμίας αντίστοιχα. Τα αξιώματα υπαγωγής είναι ιδιαίτερα σημαντικά καθώς μας επιτρέπουν τη δημιουργία ιεραρχιών. Διασθητικά ένα αξίωμα υπαγωγής συνεπάγεται ότι η έννοια D είναι πιο γενική από την έννοια C, δηλαδή ότι η έννοια D εγκλείει την έννοια C. Όσον αφορά στη σημασιολογία λέμε ότι μία ερμηνεία $\mathcal{I}$ ικανοποιεί ένα αξίωμα υπαγωγής $C \sqsubseteq D$ εάν ισχύει $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Σχετικά με τα αξιώματα ισοδυναμίας αυτά μας προσφέρουν τη δυνατότητα να ορίζουμε καινούριες σύνθετες έννοιες με το όνομα που επιθυμούμε χρησιμοποιώντας ατομικές έννοιες ή έννοιες που έχουν ήδη οριστεί και κατασκευαστές. Δηλαδή ένα αξίωμα ισοδυναμίας ταυτίζει τις δύο έννοιες, δηλώνοντας ότι περιγράφουν το ίδιο πράγμα. Σημασιολογικά, μια ερμηνεία $\mathcal{I}$ ικανοποιεί ένα αξίωμα ισοδυναμίας $C \equiv D$ εάν ισχύει $C^{\mathcal{I}}=D^{\mathcal{I}}$. Ένα σύνολο από αξιώματα υπαγωγής και αξιώματα ισοδυναμίας συνιστά ένα *σώμα ορολογίας (Terminological Box-TBox)*, που συμβολίζεται με το γράμμα $\mathcal{T}$. Μία ερμηνεία $\mathcal{I}$ ικανοποιεί ένα σώμα ορολογίας $\mathcal{T}$ εάν ικανοποιεί όλα τα αξιώματα υπαγωγής και ισοδυναμίας που αυτή περιέχει. Σε αυτήν την περίπτωση λέμε ότι η ερμηνεία $\mathcal{I}$ αποτελεί ένα μοντέλο του $\mathcal{T}$. Δηλαδή μία ορολογία θέτει ένα σύνολο από προϋποθέσεις που απαράιτητα πρέπει να ικανοποιούν τα μοντέλα που θα οριστούν.

Εκτός από αξιώματα ορολογίας στις γλώσσες περιγραφικής λογικής, ορίζουμε και *ισχυρισμούς*. Έχοντας ορίσει ένα σύνολο από έννοιες, άτομα και ρόλους, οι ισχυρισμοί είναι είτε έννοιες που συνδέονται με άτομα είτε ρόλοι που συνδέονται με ζεύγη ατόμων. Όσον αφορά το συντακτικό τους αυτό είναι της μορφής a:C ή C(a) για τις έννοιες και (a,b):R ή R(a,b) για τους ρόλους. Σημασιολογικά ένας ισχυρισμός έννοιας συνεπάγεται ότι $a^{\mathcal{I}} \in C^{\mathcal{I}}$ ενώ ένας ισχυριμός ρόλου ότι $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. Ένα σύνολο από ισχυρισμούς συγκροτεί το *σώμα ισχυρισμών (Assertional Box-ABox)* και συμβολίζεται με $\mathcal{A}$. Μια ερμηνεία $\mathcal{I}$ ικανοποιεί ένα

σώμα ισχυρισμών $\mathcal{A}$ εάν ικανοποιεί όλους τους ισχυρισμούς που περιέχονται σε αυτό. Σε αυτήν την περίπτωση η ερμηνεία $\mathcal{I}$ αποτελεί μοντέλο του $\mathcal{A}$. Ένα σώμα ορολογίας $\mathcal{T}$ μαζί με ένα σώμα ισχυρισμών $\mathcal{A}$ συγκροτούν μία *βάση γνώσης (knowledge base)* $\mathcal{K}=(\mathcal{T},\mathcal{A})$. Μία ερμηνεία $\mathcal{I}$ λέμε ότι ικανοποιεί ένα σώμα ισχυρισμών $\mathcal{A}$ με βάση ένα σώμα ορολογίας $\mathcal{T}$ εάν είναι ταυτόχρονα μοντέλο του $\mathcal{A}$ αλλά και του $\mathcal{T}$.

Στη συνέχεια διευρύνουμε τη γλώσσα με τα αξιώματα υπαγωγής ρόλων. Διαισθητικά τα αξιώματα υπαγωγής ρόλων λειτουργούν όπως τα αξιώματα υπαγωγής εννοιών, δηλαδή υπάρχει κάποιος πιο γενικός ρόλος ο οποίος εγκλείεται από έναν άλλο ρόλο περισσότερο ειδικό. Συντακτικά, αυτό εκφράζεται όπως και στις έννοιες π.χ. για τους ρόλους R_1 και R_2, $R_1 \sqsubseteq R_2$. Η ερμηνεία που αντιστοιχεί σε αυτόν τον συμβολισμό είναι $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$, όπου $R_1^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ και $R_2^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Η ύπαρξη αξιωμάτων υπαγωγής ρόλων αντικατοπτρίζεται στην προσθήκη του γράμματος $\mathcal{H}$ στην ονομασία της γλώσσας. Ένα παράδειγμα υπαγωγής ρόλων είναι μεταξύ των ρόλων έχειΠατέρα $\sqsubseteq$ έχειΠρόγονο.

Μία άλλη προσθήκη που μπορεί να γίνει στη γλώσσα μας είναι αυτή των ονοματικών εννοιών (nominal concepts). Μία ονοματική έννοια επιτρέπει σε ένα συγκεκριμένο άτομο να χρησιμοποιηθεί για την περιγραφή μίας έννοιας και να ταυτισθεί με αυτήν. Έτσι από τα άτομα a,b,c μπορούν να ληφθούν οι έννοιες {a},{b},{c}. Αναφορικά με την ερμηνεία των ονοματικών εννοιών, ο χώρος ερμηνείας $\Delta^{\mathcal{I}}$ που αντιστοιχεί στη συγκεκριμένη έννοια αποτελείται μόνο από το αντικείμενο $a^{\mathcal{I}}$, δηλαδή είναι το μονοσύνολο $\{a\}^{\mathcal{I}}$. Το αντίστοιχο γράμμα που χρησιμοποιείται για την ονομασία της γλώσσας είναι το $\mathcal{O}$. Για παράδειγμα συνδυάζοντας τον κατασκευαστή της ένωσης (του οποίου η σημασιολογία είναι $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$) και αυτόν της ονοματικής έννοιας έχουμε τη δυνατότητα να διατυπώσουμε την περίπλοκη έννοια ΜέρεςΕβδομάδας={Δευτέρα} $\sqcup$ {Τρίτη} $\sqcup$ {Τετάρτη} $\sqcup$ {Πέμπτη} $\sqcup$ {Παρακευή} $\sqcup$ {Σάββατο} $\sqcup$ {Κυριακή}.

Μία περαιτέρω ιδιότητα που χρησιμοποιείται σε διάφορες γλώσσες περιγραφικής λογικής είναι αυτός της αντιστροφής ρόλων. Στην περίπτωση αυτή εάν ένα άτομο a συνδέεται με ένα άτομο b μέσω ενός ρόλου R τότε το άτομο b συνδέεται με το άτομο a μέσω του αντίστροφου ρόλου R, ο οποίος συμβολίζεται με R^-. Η ερμηνεία που αποδίδεται στους αντίστροφους

ρόλους είναι ότι (a,b) $\in R^{\mathcal{I}}$ αν και μόνο αν (a,b) $\in (R^{-})^{\mathcal{I}}$. Μία γλώσσα που διαθέτει αντίστροφους ρόλους περιλαμβάνει το $\mathcal{I}$ στο όνομα της. Επιπλέον να σημειώσουμε ότι το $R \sqsubseteq S$ συνεπάγεται το $R^{-} \sqsubseteq S^{-}$. Ακόμη να επισημάνουμε ότι οι διάφορες ιδιότητες που εμφανίζονται μεταξύ των ρόλων (π.χ. ιεραρχίες που βασίζονται σε αξιώματα υπαγωγής) οργανώνονται κατ'αντιστοιχία με τα TBox και τα ABox σε RBox (σώμα ρόλων-Role Box) . Το RBox συμβολίζεται με το γράμμα $\mathcal{R}$.

Στο σημείο αυτό να προσθέσουμε έναν κατασκευαστή που χρησιμοποιείται συχνά στις περιγραφικές λογικές και συμβάλλει στην αύξηση της εκφραστικότητας τους, τον κατασκευαστή περιορισμού πληθικότητας (number restriction). Ο εν λόγω κατασκευαστής θέτει περιορισμούς (άνω ή κάτω φράγματα) ως προς τον αριθμό των αντικειμένων με τα οποία μπορεί να συνδέεται ένα άλλο αντικείμενο μέσω ενός συγκεκριμένου ρόλου. Πρόκειται ουσιαστικά για δύο κατασκευαστές έννοιας, τον *το-πολύ* που συμβολίζεται $\leq$nR και αντίστοιχα τον κατασκευαστή *το-λιγότερο* $\geq$nR όπου n είναι φυσικός αριθμός και R ρόλος. Η τυπική σημασιολογία των παραπάνω δύο κατασκευαστών ακολουθεί

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

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

Μία γλώσσα περιγραφικής λογικής με τον παραπάνω κατασκευαστή περιλαμβάνει στο όνομα της το $\mathcal{N}$. Να σημειώσουμε ότι στην περίπτωση που το n λαμβάνει μόνο την τιμή 1, πρόκειται για μία ειδική περίπτωση του παραπάνω κατασκευαστή που ονομάζεται συναρτησιακός περιορισμός πληθικότητας και αντιστοιχεί στο γράμμα $\mathcal{F}$.

Τέλος οι περιγραφικές λογικές προσφέρουν και δυνατότητες για *τύπους δεδομένων*. Πέρα από τις κλασικές εφαρμογές των τύπων δεδομένων όπως π.χ. στις γλώσσες προγραμματισμού έχουν αναπτυχθεί και άλλες προσεγγίσεις, όπως π.χ. η type system που συμβολίζεται με το γράμμα **D.** Ένα type system **D** ορίζεται από ένα ζευγάρι ($\Delta_{\mathbf{P}}$,$\Phi_{\mathbf{P}}$) όπου $\Phi_{\mathbf{P}}$ είναι το σύνολο των ονομάτων των τύπων δεδομένων και $\Delta_{\mathbf{P}}$ είναι ο χώρος ερμηνείας όλων αυτών των ονομάτων. Αυτός ο χώρος ερμηνείας είναι ξένος από το χώρο ερμηνείας των εννοιών, των ατόμων και των ρόλων και σε αντίθεση με τις έννοιες, τους ρόλους και τα άτομα που μπορεί να έχουν πολλές ερμηνείες ένα όνομα τύπου δεδομένων μπορεί να έχει μόνο μία ερμηνεία. Ως παράδειγμα αναφέρουμε το string που ανήκει στο $\Phi_{\mathbf{P}}$ με το αντίστοιχο σύνολο όλων των δυνατών συμβολοσειρών που ανήκει στο $\Delta_{\mathbf{P}}$**.** Ως παράδειγμα έννοιας η οποία ορίζεται με τη

βοήθεια τύπων δεδομένων είναι η ∃ έχειΕισόδημα.$>_{50,000}$, και αναφέρεται στο σύνολο των ανθρώπων των οποίων το εισόδημα ξεπερνά τις 50,000.

Χρησιμοποιώντας το πλήθος των κατασκευαστών που αναφέρθηκαν παραπάνω ουσιαστικός σκοπός των γλωσσών περιγραφικής λογικής είναι η εξαγωγή νέας γνώσης από την ήδη υπάρχουσα γνώση, χρησιμοποιώντας αλγορίθμους συλλογιστικής. Οι αλγόριθμοι που συνήθως χρησιμοποιούνται ανήκουν στην οικογένεια των αλγορίθμων tableaux. Δε θα αναφερθούμε σε λεπτομέρειες εξαιτίας της πολυπλοκότητας που παρουσιάζει η κατανόηση τους. Μπορούμε μόνο να σκιαγραφήσουμε κάποια σημεία τους. Μία έννοια C λέμε ότι είναι ικανοποιήσιμη με βάση ένα TBox εάν υπάρχει μοντέλο $\mathcal{I}$ του TBox τέτοιο ώστε $C^{\mathcal{I}} \neq \varnothing$. Εάν θέλουμε να εξετάσουμε κατά πόσο μία έννοια είναι ικανοποιήσιμη εξετάζουμε την ικανοποιησιμότητα της άρνησης της. Στη συνέχεια ανάγουμε το πρόβλημα στο πρόβλημα της συνέπειας τους ABox και το αποτέλεσμα εξαρτάται από το εάν θα εμφανιστούν λογικές συγκρούσεις σε αυτό. Πλήθος μεθόδων, τεχνικών και βελτιστοποιήσεων έχουν αναπτυχθεί στα πλαίσια των παραπάνω αλγορίθμων. Όσον αφορά την υπολογιστική τους πολυπλοκότητα υπάρχει μεγάλη ποικιλία η οποία κατά βάση εξαρτάται από τα χαρακτηριστικά που υλοποιεί η εκάστοτε γλώσσα περιγραφικής λογικής, δηλαδή την ποικιλία και το πλήθος των κατασκευαστών που υποστηρίζει. Κανόνα αποτελεί το γεγονός ότι το τίμημα που πληρώνουμε για μία εκφραστική γλώσσα είναι η αυξημένη και σε ορισμένες περιπτώσεις απαγορευτική πολυπλοκότητα των αλγορίθμων συλλογιστικής της. Γι'αυτό το λόγο η χρυσή τομή που πρέπει να βρεθεί μεταξύ των εκφραστικών δυνατοτήτων της γλώσσας και των υπολογιστικών απαιτήσεων που αυτή πρέπει να ικανοποιεί είναι προϊόν ενδελεχών εξετάσεων και πειραματισμών.

Στο σημείο αυτό να τονίσουμε ότι στις περιγραφικές λογικές ισχύει η *υπόθεση ανοικτού κόσμου*, δηλαδή για δηλώσεις που δεν περιλαμβάνονται ρητά ούτε αποδεικνύονται από τη βάση γνώσης δεν υποθέτουμε την άρνηση τους. Αυτό έρχεται σε αντίθεση με την *υπόθεση κλειστού κόσμου* που έχει υιοθετηθεί σε sql συστήματα ή στη γλώσσα λογικού προγραμματισμού prolog.

Παραπάνω δόθηκε μία σύντομη εισαγωγή σχετικά με το συντακτικό και τη σημασιολογία των γλωσσών ΠΛ καθώς επίσης και των εκφραστικών δυνατοτήτων που παρουσιάζουν μέσω των διαφόρων κατασκευαστών και αξιωμάτων. Με τον τρόπο αυτό δόθηκε μια γενική εικόνα της γλώσσας $\mathcal{SHOIN}$**(D),** η οποία αποτελεί το μαθηματικό υπόβαθρο της προτυποποιημένης από τη W3C γλώσσας OWL DL, υπογλώσσα της γλώσσας OWL. Η γλώσσα OWL, όπως έχει ήδη αναφερθεί αποτελεί γλώσσα περιγραφής οντολογιών και περιγράφεται με περισσότερες λεπτομέρειες στη συνέχεια.

2.4 Η γλώσσα OWL

Η γλώσσα OWL αποτελεί κορύφωση μιας παγκόσμιας ερευνητικής προσπάθειας για καθιέρωση μιας γλώσσας αναπαράστασης γνώσης η οποία μπορεί να χρησιμοποιηθεί με συνέπεια στο πλαίσιο του Σημασιολογικού Διαδικτύου. Προτυποποιήθηκε επίσημα το Φεβρουάριο του 2004 από τη W3C. Τα αρχικά αντιστοιχούν στη Web Ontology Language ωστόσο προτιμήθηκε, για λόγους ευφωνίας η ονομασία OWL αντί για WOL. Όπως ανέφερε χαρακτηριστικά και ο Guus Schreiber, που ήταν ένας από τους πρωτεργάτες του Web Ontology Working Group, που προτυποποίησε την OWL, «Why not be inconsistent in at least one aspect of a language which is all about consistency?».

Πρόκειται για μία γλώσσα της οποίας τα θεμέλια έχουν τεθεί από τις περιγραφικές λογικές που περιγράφηκαν σε προηγούμενη ενότητα. Με τον τρόπο αυτό υπάρχει η δυνατότητα να χρησιμοποιηθούν υπαρκτοί και υλοποιημένοι αλγόριθμοι για τη συλλογιστική της γλώσσας, εξοικονομώντας έτσι σημαντικούς πόρους. Η εν λόγω γλώσσα περιγραφής οντολογιών έχει εμπλουτιστεί όσον αφορά την εκφραστικότητα της σε σύγκριση με τις περιγραφικές λογικές με επιπλέον δυνατότητες. Αναφερόμενοι στην διατυπωμένη από τον Tim Berners-Lee αρχιτεκτονική του Σημασιολογικού Διαδικτύου, η OWL αντιστοιχεί στο λογικό επίπεδο, το οποίο επιτρέπει την επεξεργασία δεδομένων κατώτερου επιπέδου τα οποία πλέον είναι φορτισμένα σημασιολογικά.

Οι τρεις διαφορετικές μορφές στις οποίες εμφανίζεται η γλώσσα OWL είναι οι εξής:

- **OWL Lite.** Προκύπτει από τη γλώσσα περιγραφικής λογικής $\mathcal{SHIF}$(D) και προσφέρει στοιχειώδεις εκφραστικές δυνατότητες σε συνδυασμό όμως με γρήγορες υπολογιστικές επιδόσεις.
- **OWL DL.** Βασίζεται στη γλώσσα περιγραφικής λογικής $\mathcal{SHOIN}$(**D**), διευρύνοντας σημαντικά την OWL Lite από άποψη εκφραστικότητας. Το υπολογιστικό κόστος που συνοδεύει την εκφραστική επέκταση είναι σημαντικό, ωστόσο πρόκειται για την πιο ευρέως διαδεδομένη γλώσσα περιγραφής οντολογιών.
- **OWL Full.** Πρόκειται για την πλέον εμπλουτισμένη εκφραστικά εκδοχή της γλώσσας της οποίας η OWL DL είναι υποσύνολο. Περιλαμβάνει χαρακτηριστικά της γλώσσας rdf-schema, παραχωρώντας έτσι και περιθώρια μετα-μοντελοποίησης. Οι ανεξέλεγκτες όμως αυτές εκφραστικές δυνατότητες την καθιστούν *μη-αποφασίσιμη* και κατ'επέκταση μη χρησιμοποιήσιμη σε εφαρμογές.

Δεδομένου ότι η OWL βρίσκεται στο λογικό επίπεδο της αρχιτεκτονικής του Σημασιολογικού Διαδικτύου αποτελεί επέκταση των εργαλείων XML, RDF και RDF-schema. Επειδή η περιγραφή των σωμάτων ορολογίας και ισχυρισμών σε XML είναι αρκετά μακροσκελής έχει καθιερωθεί μία αφηρημένη σύνταξη η οποία επιτρέπει την παρουσίαση της γνώσης με περισσότερο εποπτικό τρόπο.

Όσον αφορά το αλφάβητο της OWL αυτό περιλαμβάνει τις *κλάσεις(classes), τις ιδιότητες (properties) και τα άτομα(individuals)*. Η αντιστοιχία, σε όρους περιγραφικής λογικής, είναι με τις έννοιες, τους ρόλους και τα άτομα. Ξεκινάμε την παρουσίαση του συντακτικού της OWL με ένα σύνολο από συμβολισμούς. Για τις κλάσεις C_1, C_2,…, C_n, το intersectionOf (C_1, C_2,…, C_n) παριστάνει την τομή τους. Για τις ίδιες κλάσεις το unionOf (C_1, C_2,…, C_n) χρησιμοποιείται για την αναπαράσταση της ένωσης τους. Επιπλέον με το complementOf(C) συμβολίζουμε το συμπλήρωμα μιας κλάσης, που περικλείει όλα τα αντικείμενα που δεν ανήκουν σε αυτή.

Η OWL παρέχει επιπλεόν αφηρημένη σύνταξη για τη διατύπωση υπαρξιακών, πληθικών και καθολικών περιορισμών. Η λέξη κλειδί που χρησιμοποιείται για τον περιορισμό ιδιότητας είναι restriction. Για τον καθολικό και τον υπαρξιακό περιορισμό αντίστοιχα χρησιμοποιούνται οι όροι allValuesFrom και someValuesFrom. Ας υποθέσουμε ότι θέλουμε να αναπαραστήσουμε την κλάση των αντικειμένων που έχουν τουλάχιστον έναν φίλο που είναι γιατρός. Συμβολίζεται με restriction(hasFriend someValuesFrom(Doctor)). Ανάλογα για τα αντικείμενα των οποίων όλοι οι φίλοι (εάν και εφόσον έχουν) είναι γυναίκες θα είναι restriction(hasFriend allValuesFrom(Woman)). Ανατρέχοντας στις περιγραφικές λογικές εμφανίζονται και οι περιορισμοί πληθικότητας, για την απεικόνιση των οποίων χρησιμοποιούμε τις λέξεις minCardinality, maxCardinality και cardinality. Έτσι για την κλάση των αντικειμένων που μιλάνε τουλάχιστον δύο γλώσσες έχουμε το restriction(speaksLanguage minCardinality(2)), γι'αυτούς που μιλάνε το πολύ 3 restriction(speaksLanguage maxCardinality(3)) ενώ γι'αυτούς που μιλάνε ακριβώς 4 restriction(speaksLanguage cardinality(4)).

Μια επιπλέον δυνατότητα που μας προσφέρεται είναι αυτή του ορισμού κλάσεων με απαρίθμηση των ατόμων που ανήκουν σε αυτές. Η αντίστοιχη σύνταξη είναι oneOf (o_1, o_2,…, o_n) για τα άτομα o_1, o_2,…, o_n. Ένα παράδειγμα θα μπορούσε να είναι για την κλάση των ηπείρων , oneOf(Europe, Asia, Africa, America, Australia, Antartica).

Όσον αφορά στα αξιώματα κλάσεων, μπορούμε να εκφράσουμε με τη βοήθεια αφηρημένης σύνταξης, δύο περιπτώσεις σχέσεων υπαγωγής. Η μία είναι η περίπτωση κατά την οποία μία κλάση C_1 είναι υποκλάση μίας κλάσης C_2, με subClassOf(C_1,C_2). Η άλλη είναι η περίπτωση που μία κλάση A είναι υποκλάση της τομής ενός συνόλου κλάσεων C_1, C_2,…, C_n και εκφράζεται ως Class(A partial C_1, C_2,…, C_n). Παρόμοια στα αξιώματα ισοδυναμίας κλάσεων

μία κλάση A που είναι ισοδύναμη με την τομή ενός συνόλου κλάσεων C_1, C_2,…, C_n γράφεται ως Class(A complete C_1, C_2,…, C_n). Σε περίπτωση που μόνο ένα σύνολο συγκεκριμένων ατόμων συγκροτούν μία κλάση ο ορισμός του αξιώματος μπορεί να γίνει με απαρίθμηση ως εξής: EnumeratedClass(A complete o_1, o_2,…, o_n), όπου o_1, o_2,…, o_n αντιστοιχούν σε άτομα.

Μία ενδιαφέρουσα εκφραστική δυνατότητα που δε συναντάμε στις περιγραφικές λογικές είναι να ορίσουμε ότι δύο κλάσεις είναι ξένες μεταξύ τους, με τη συνολοθεωρητική έννοια του όρου. Δηλαδή η αντίστοιχη ερμηνεία για τις ξένες κλάσεις A και B θα είναι $A^{\mathcal{I}} \cap B^{\mathcal{I}} = \varnothing$.

Έτσι για παράδειγμα, για να εκφράσουμε ότι οι κλάσεις Married και Single είναι ξένες μεταξύ τους γράφουμε στην OWL DisjointClasses(Married,Single). Επιπλέον όπως και στις γλώσσες περιγραφικής λογικής έτσι και στην OWL έχουμε δύο ξεχωριστές κλάσεις, τις owl:Thing και owl:Nothing που αντιστοιχούν στις έννοιες $\top$ και $\bot$ αντίστοιχα.

Στην OWL έχουμε δύο κατηγορίες από ιδιότητες, τις *ιδιότητες αντικειμένων* και τις ιδιότητες *τύπων δεδομένων*. Οι ιδιότητες αντικειμένων συσχετίζουν ένα αντικείμενο με ένα δεύτερο αντικείμενο, όπως για παράδειγμα η ιδιότητα έχειΠατέρα. Οι ιδιότητες τύπων δεδομένων πάλι συσχετίζουν ένα αντικείμενο με έναν τύπο δεδομένων όπως για παράδειγμα η ιδιότητα hasAge. Δήλωνουμε ότι πρόκειται για ιδιότητα τύπου δεδομένων με το αξίωμα DatatypeProperty(hasAge).

Όσον αφορά τα αξιώματα ιδιοτήτων η υπαγωγή ιδιοτήτων γράφεται SubPropertyOf(R S). Μπορούμε να ορίσουμε ότι μία ιδιότητα αντικειμένου R είναι υποϊδιότητα μιας λίστας ιδιοτήτων R_1, R_2,…, R_n με τη σύνταξη ObjectProperty(R super(R_1) …. super(R_n)). Επιπλέον για την ιδιότητα R που είναι αντίστροφη της ιδιότητας S γράφουμε ObjectProperty(R inverseOf(S)). Ακόμη μπορούμε να εκφράσουμε ισοδυναμία ιδιοτήτων γράφοντας EquivalentProperties(R S). Όσον αφορά τις ιδιότητες μπορούμε να εισάγουμε στη βάση γνώσης μας πληροφορίες σχετικά με το πεδίο ορισμού και το σύνολο τιμών στα οποία υπόκειται μία ιδιότητα. Έτσι για μία ιδιότητα αντικειμένων της οποία το πεδίο ορισμού προέρχεται από την κλάση A γράφεται ObjectProperty(R domain(A)). Αντίστοιχα για το σύνολο τιμών έχουμε ObjectProperty(R range(A)). Εάν αναφερόμαστε σε ιδιότητα τύπου δεδομένων γράφουμε αντίστοιχα DataProperty(T range(d)), όπου d πλέον είναι ένας τύπος δεδομένων.

Παράλληλα μία ιδιότητα R μπορεί να είναι συμμετρική κάτι που στην OWL γράφεται ως ObjectProperty(R Symmetric). Πρόκειται για κάτι που δεν έχουμε συναντήσει στις περιγραφικές λογικές και η ερμηνεία που αποδίδεται είναι ότι για τη συμμετρική ιδιότητα R, $\forall$ a,b με R(a,b) συνεπάγεται ότι R(b,a). Ακόμη μία ιδιότητα R μπορεί να είναι μεταβατική κάτι που συμβολίζεται με ObjectProperty(R Transitive). Μία ιδιότητα R ορίζεται ως συναρτησιακή γράφοντας ObjectProperty(R Functional). Η ερμηνεία αυτής της περίπτωσης

σχετίζεται με τον αριθμό των αντικειμένων με τα οποία συνδέεται ένα αντικείμενο μέσω του ρόλου R και τα οποία είναι το πολύ 1. Παρόμοια μπορεί να οριστεί για ιδιότητες τύπου δεδομένων το DatatypeProperty(R Functional). Εκτός από τα παραπάνω υπάρχει και ο χαρακτηρισμός μιας ιδιότητας ως αντίστροφη συναρτησιακή με ObjectProperty(R InverseFunctional). Σε αυτήν την περίπτωση το πολύ ένα αντικείμενο μπορεί να συνδεθεί με άλλα μέσω του ρόλου R.

Περνώντας στην περίπτωση των ατόμων γράφοντας Individual(o type(C_1) … type(C_n) value(R_1,o_1) … value(R_n,o_n)) ορίζουμε ότι το άτομο ο ανήκει στις κλάσεις C_1,…,C_n και επιπλέον συμμετέχει στις ιδιότητες R_1,…,R_n με τα άτομα o_1,…,o_n αντίστοιχα. Επιπλέον έχουμε τη δυνατότητα να ορίσουμε για τα άτομα o_1,…,o_n ότι ταυτίζονται γράφοντας SameIndividual(o_1,…,o_n) και ότι είναι διαφορετικά γράφοντας DifferentIndividuals(o_1,…,o_n).

Μετά την επίσημη προτυποποίηση της γλώσσας OWL από την W3C εμφανίστηκε η ανάγκη εμπλουτισμού της ήδη καθιερωμένης γλώσσας OWL με επιπλέον χαρακτηριστικά και διευρυμένες λειτουργικότητες. Η ανάγκη αυτή αποτυπώθηκε σε μία νέα έκδοση της OWL, την OWL 2.0 (ή άλλιώς OWL 1.1) η οποία τώρα βρίσκεται στο στάδιο της προτυποποίησης. Η OWL 2.0 επεκτείνει την W3C OWL Web Ontology Language με ένα μικρό σύνολο από χρήσιμα χαρακτηριστικά τα οποία όλο και πιο συχνά ζητούνται από τους χρήστες. Γι'αυτά τα χαρακτηριστικά υπάρχουν ήδη αποδοτικοί αλγόριθμοι συλλογιστικής. Επιπλέον είναι ενσωματωμένα σε πολλά από τα εργαλεία λογισμικού που αναπτύσονται για την OWL. Πιο συγκεκριμένα αυτά τα χαρακτηριστικά περιλαμβάνουν επιπλέον συντακτικούς συμβολισμούς, επιπρόσθετους κατασκευαστές (προσοντούχους περιορισμούς πληθικότητας), υποστήριξη για επεκτεταμένους τύπους δεδομένων, απλή μετα-μοντελοποίηση και τέλος επεκτεταμένα annotations. Ειδικά η προσθήκη των annotations άπτεται άμεσα της παρούσας εργασίας, καθώς το υλοποιημένο σύστημα το οποίο πραγματεύεται χρησιμοποιεί τα εν λόγω annotations με τρόπο που θα εξηγηθεί σε επόμενη παράγραφο της εργασίας.

2.5 Θεωρία ασαφών συνόλων

Στη συνέχεια, καθώς όπως αναφέραμε σκοπός της παρούσας διπλωματικής εργασίας είναι η ενοποίηση οντολογιών και βάσεων δεδομένων σε περιβάλλον αβεβαιότητας, θα αναφερθούμε σε κάποια κομμάτια της ασαφούς θεωρίας που σχετίζονται άμεσα με την αναπαράσταση ασαφών δεδομένων, όπως στα ασαφή σύνολα, τους ασαφείς αριθμούς και τις γλωσσικές μεταβλητές.

2.5.1 Ασαφή σύνολα

Η παραδοσιακή επιστήμη ξεκινώντας από την αριστοτέλεια λογική και φτάνοντας μέχρι την θεμελίωση των λογισμών από το Leibniz και τον Newton ανέκαθεν ήταν υπέρμαχος της ακρίβειας, της αυστηρότητας και της σαφούς διατύπωσης των ιδεών. Όποτε η ασάφεια ή η αβεβαιότητα παρεισέφρυαν η ανάγκη για διόρθωση και επανατοποθέτηση έκανε την εμφάνιση της. Έτσι όποια σκέψη δε διέθετε την απαραίτητη ευκρίνεια καταδικαζόταν σε αφανισμό από το επιστημονικό στερέωμα. Μόλις στα τέλη του 19^{ου} αιώνα, άρχισε να αλλάζει ο τρόπος θεώρησης των πραγμάτων όταν η Νευτώνεια μηχανική φάνηκε για πρώτη φορά ανεπαρκής να περιγράψει τις εξαιρετικά πολύπλοκες αντιδράσεις μεταξύ των δομικών στοιχείων της ύλης, σε μοριακό επίπεδο. Τη θέση της κατέλαβαν στατιστικές μέθοδοι, κατά τις οποίες αντί να εξετάζονται τα μικροσκοπικά σωματίδια κάθε ένα ξεχωριστά, λαμβάνονταν μέσοι όροι μακροσκοπικών μεταβλητών τους. Πράγματι μια τέτοια αφαιρετική και καινοτόμος προσέγγιση παρουσίασε αξιοσημείωτη επιτυχία στην περιγραφή των μορίων των αερίων και των φαινομένων που συνδέονται με αυτά.

Έτσι στην επιστημονική κοινότητα άρχισε να επικρατεί η άποψη ότι τα προβλήματα διακρίνονται σε δύο κατηγορίες με εκ διαμέτρου αντίθετη φύση. Από τη μία έχουμε τα προβλήματα με λίγες και καλά ορισμένες μεταβλητές που σχετίζονται με προβλέψιμο τρόπο μεταξύ τους, ενώ από την άλλη υπάρχουν τα προβλήματα με πολύ μεγάλο αριθμό μεταβλητών και εξαιρετικά υψηλό βαθμό τυχαιότητας. Ο Warren Weaver σε μία διάσημη δημοσίευση του, το 1948, χαρακτηρίζει τα πρώτα προβλήματα *οργανωμένης απλότητας* ενώ τα δεύτερα *ανοργάνωτης πολυπλοκότητας*. Οι μέθοδοι που χρησιμοποιούνται για την επιτυχημένη επίλυση των παραπάνω προβλημάτων είναι συμπληρωματικές, δηλαδή όποτε ευστοχεί η μία αποτυχαίνει η άλλη και αντίστροφα. Δυστυχώς όμως τα προβλήματα συστημάτων τα οποία καθημερινά παρουσιάζονται στις διάφορες εφαρμογές, δεν ανήκουν κατηγορηματικά σε καμία από τις δύο κατηγορίες αλλά μάλλον κάπου στο ενδιάμεσο.

Μία άλλη πεποίθηση που αναπτύχθηκε στο χώρο της επιστήμης, ιδιαίτερα μετά το Β' Παγκόσμιο Πόλεμο και την αλματώδη ανάπτυξη των υπολογιστών ήταν ότι επειδή η τεχνολογική πρόοδος εξασφάλιζε την επίλυση όλο και πιο υπολογιστικά απαιτητικών προβλημάτων, κάποια στιγμή στο μακρινό μέλλον θα ήταν εφικτή η επίλυση προβλημάτων οποιασδήποτε υπολογιστικής πολυπλοκότητας. Επρόκειτο για μία μη ρεαλιστική αντιμετώπιση η οποία άρχισε να εξαλείφεται στις αρχές της δεκαετίας του 1960. Ήταν τότε που οι επιστήμονες άρχισαν να συνειδητοποιούν τα φυσικά όρια τόσο των τεχνολογικών επιτευγμάτων όσο και των ανθρώπινων δυνατοτήτων. Πράγματι ο Bremerman το 1962 ([22]) έθεσε το μέγιστο όριο υπολογιστικής ικανότητας που μπορεί να τεθεί ποτέ στη διάθεση του ανθρώπου. Στηριγμένος σε εκτιμήσεις της κβαντικής θεωρίας, συμπέρανε ότι οποιοδήποτε υπολογιστικό σύστημα, τεχνητό ή ζωντανό, δε δύναται να επεξεργαστεί περισσότερα από

2×10^{47} bits ανά δευτερόλεπτο και ανά γραμμάριο της μάζας του. Πολλαπλασιάζοντας την ηλικία της γης και τη μάζα της κατέληξε σε ένα νούμερο (περίπου 10^{93} bits), το οποίο όταν οι υπολογιστικές απαιτήσεις ενός προβλήματος ξεπερνούν, καθιστούν το πρόβλημα *υπερυπολογιστικό(transcomputational)*.

Τι γίνεται επομένως με τα υπολογιστικά προβλήματα των οποίων οι υπέρμετρες υπολογιστικές αξιώσεις τα θέτουν πάνω από το παραπάνω όριο; Παραιτούμαστε και αποσυρόμαστε από το ενδεχόμενο επίλυσης τους; Οι επιστήμονες προτίμησαν να υιοθετήσουν μια νέα αντιμετώπιση εισάγοντας στα προβλήματα νησίδες αβεβαιότητας και μετριάζοντας έτσι το υπολογιστικό τους κόστος. Στην πραγματικότητα επιχείρησαν να βρουν μία σχέση που συνδέει τα τρία θεμελιώδη χαρακτηριστικά κάθε μοντέλου: την *πολυπλοκότητα,* την *αξιοπιστία* και την *αβεβαιότητα.* Παρά το γεγονός ότι δε διατυπώθηκε κάποια ακριβής συσχέτιση η γενική εντύπωση που αποκόμισαν είναι ότι η αύξηση της αβεβαιότητας τείνει να μειώσει την πολυπλοκότητα και να αυξήσει την αξιοπιστία του εξεταζόμενου μοντέλου.

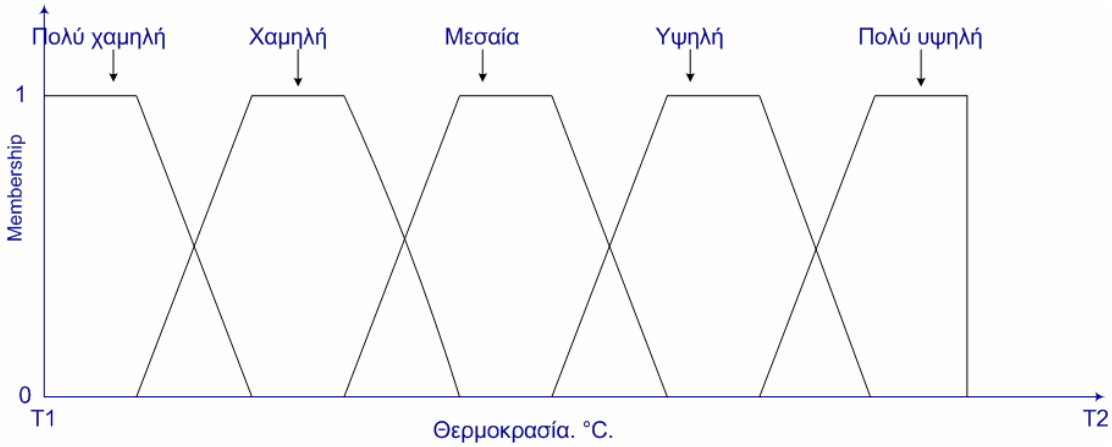
Οι παραπάνω διαπιστώσεις συνοδεύτηκαν από την ανάδυση μιας σειρά θεωριών αβεβαιότητας, χωρίς απαραίτητα πιθανοτικό χαρακτήρα όπως ήταν συνηθισμένο εώς τότε. Κομβικό ρόλο στην παραπάνω εξέλιξη διαδραμάτισε η δημοσίευση του Zadeh το 1965 ([48]) πάνω στα ασαφή σύνολα. Τα ασαφή σύνολα, σε αντίθεση με τα παραδοσιακά σύνολα της κλασικής συνολοθεωρίας, διακρίνονται από το γεγονός ότι δε διαθέτουν αυστηρά όρια με αποτέλεσμα τα αντικείμενα του κόσμου να εντάσσονται ή όχι στο εσωτερικό τους κατά ένα *βαθμό* και όχι απόλυτα. Ο βαθμός αυτός κυμαίνεται στο κλειστό διάστημα [0,1] με το 0 να καταδεικνύει ότι ένα αντικείμενο δεν ανήκει σε καμία περίπτωση στο εξεταζόμενο σύνολο και το 1 να σημαίνει ότι αποτελεί σε κάθε περίπτωση μέλος του συνόλου. Ωστόσο περισσότερο ενδιαφέρον παρουσιάζουν οι ενδιάμεσες περιπτώσεις. Για παράδειγμα σε ένα υποθετικό σχολείο 100 μαθητών πότε λέμε ότι ένας μαθητής είναι καλός; Μία απάντηση είναι όταν συγκαταλέγεται στους πρώτους 10. Αυτό επομένως σημαίνει ότι ο $11^{ος}$ μαθητής δεν εμπίπτει στον χαρακτηρισμό και αυτόματα χαρακτηρίζεται ως κακός τη στιγμή που η διαφορά με τον αντίστοιχο καλό $10^{ο}$ μαθητή μπορεί να είναι απειροελάχιστη; Πρόκειται για μια μάλλον αποτυχημένη μοντελοποίηση. Ωστόσο με τη χρήση ενός ασαφούς συνόλου για την έννοια «καλός μαθητής» μπορούμε να διατυπώσουμε μία ταξινόμηση σύμφωνα με την οποία ο $1^{ος}$ μαθητής είναι καλός σε βαθμό 1, ο $5^{ος}$ σε βαθμό 0.9 , ο $11^{ος}$ σε βαθμό 0.8, ο $50^{ος}$ σε βαθμό 0.1 κ.ο.κ.

Κάθε ασαφές σύνολο διαθέτει μία συνάρτηση συμμετοχής η οποία ακριβώς αποδίδει σε κάθε αντικείμενο μία τιμή που είναι αντιπροσωπευτική του κατά πόσο αυτό αποτελεί μέλος του ασαφούς συνόλου. Εάν με Χ συμβολίσουμε το καθολικό σύνολο τότε για τη συνάρτηση συμμετοχής που συμβολίζουμε με Α θα ισχύει:

$$A : X \rightarrow [0,1]$$

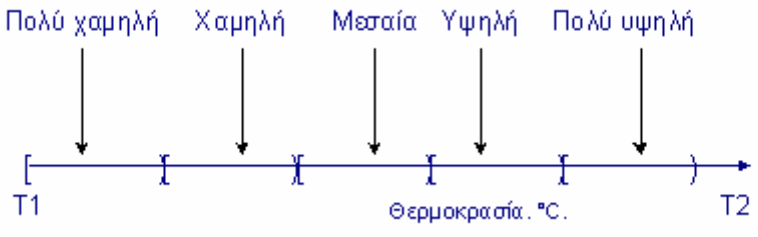
Μία κλασική περίπτωση χρήσης συνάρτηση συμμετοχής είναι όταν θέλουμε να καθορίσουμε κατά πόσο ένα αντικείμενο σχετίζεται με μία έννοια. Π.χ. κατά πόσο ένας άνθρωπος είναι πλούσιος, λεπτός, μελαχρινός ή κατά πόσο ο καιρός είναι ζεστός. Ωστόσο όλα αυτά εξαρτώνται από το περιβάλλον στα πλαίσια του οποίου διατυπώνεται μία συνάρτηση συμμετοχής με αποτέλεσμα να έχουμε διαφορετικές συναρτήσεις συμμετοχής για διαφορετικές περιστάσεις. Π.χ. σε μία μεσογειακή χώρα ο χαρακτηρισμός ζεστός για τον καιρό μπορεί να αποδίδεται διαφορετικά από ότι π.χ. σε μία σκανδιναβική χώρα.

Ένας αριθμός από ασαφή σύνολα συχνά αντιστοιχεί σε γλωσσικούς χαρακτηρισμούς που εκφράζουν ποσότητα και αναφέρονται σε μία έννοια. Μία τέτοια περίπτωση είναι για παράδειγμα η πολύ χαμηλή, χαμηλή, μέση, υψηλή και πολύ υψηλή θερμοκρασία. Το παρακάτω διάγραμμα δείχνει τις συναρτήσεις συμμετοχής που έχουν οριστεί για κάθε κατάσταση της μεταβλητής θερμοκρασία.



Σχήμα 3.

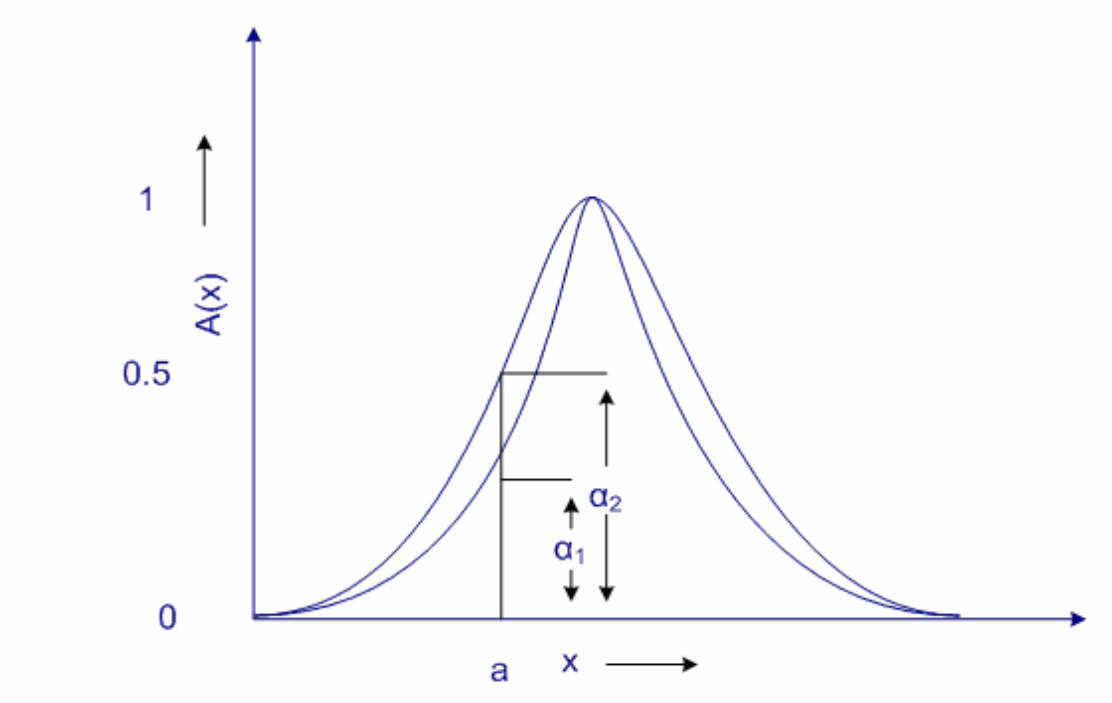
Στον οριζόντιο άξονα δίνεται η τιμή της θερμοκρασίας και στον κάθετο η τιμή της εκάστοτε συνάρτησης συμμετοχής. Παρατηρούμε το τραπεζοειδές σχήμα που λαμβάνουν οι εκάστοτε συναρτήσεις συμμετοχής. Το τραπεζοειδές σχήμα μαζί με το τριγωνικό αποτελούν τις πιο συνηθισμένες μορφές MF. Στη συνέχεια δίνεται και μία αναπαράσταση των χαρακτηρισμών υπό την αυστηρή συνολοθεωρητική σκοπιά.



Σχήμα 4.

Αυτό που παρατηρούμε είναι πόσο πιο φτωχή από άποψη εκφραστικότητας και ακρίβειας είναι η δεύτερη προσέγγιση. Πράγματι με τη βοήθεια των ασαφών συνόλων αυτό που επιτυγχάνουμε είναι πιο ομαλή και ρεαλιστική μετάβαση μεταξύ των ποσοτικών χαρακτηρισμών της θερμοκρασίας. Επιπλέον παρατηρούμε ότι η αβεβαιότητα λαμβάνει τη μέγιστη τιμή της στα όρια, πράγμα λογικό αν αναλογιστεί κανείς ότι αυτό ακριβώς είναι το σημείο για το οποίο μπορεί δεν μπορεί κανείς να αποκριθεί σε ποια από τις δύο καταστάσεις βρισκόμαστε. Έτσι οι περιπτώσεις κατά τις οποίες η μέτρηση δίνει μία οριακή τιμή μεταξύ δύο καταστάσεων καλύπτονται με περισσότερη ακρίβεια από τα ασαφή σύνολα. Αυτό ακριβώς είναι και το παράδοξο των ασαφών συνόλων, ότι παρά την εγγενή τους αβεβαιότητα παρέχουν περισσότερη ακρίβεια και ρεαλισμό στις εκτιμήσεις τους. Ωστόσο απαραίτητη προϋπόθεση για να λειτουργήσουν όπως επιθυμούμε τα ασαφή σύνολα είναι η ικανότητα του χρήστη να ορίζει κατάλληλες συναρτήσεις συμμετοχής.

Στο σημείο αυτό να προσθέσουμε ότι εκτός από τα συνηθισμένα ασαφή σύνολα υπάρχουν και τα ασαφή σύνολα διαστήματος τιμών στα οποία αντί να αποδίδεται ένας πραγματικός αριθμός σε κάθε στοιχείο του καθολικού συνόλου αποδίδεται ένα διάστημα τιμών. Με τον τρόπο αυτό εισάγουμε αβεβαιότητα στην τιμή της καθολικής συνάρτησης κάθε στοιχείου του συνόλου αναπαριστώντας ακόμα πιο ρεαλιστικά την πραγματικότητα εάν και εφόσον αυτό είναι αναγκαίο. Παρακάτω δίδεται ένα σχήμα που αναπαριστά ένα ασαφές σύνολο αυτού του τύπου.



Σχήμα 5.Παράδειγμα ενός ασαφούς συνόλου διαστημάτων τιμών.

Προχωράμε σε μία σειρά από ορισμούς για τα ασαφή σύνολα. *A-cut* ενός ασαφούς συνόλου Α (με συνάρτηση συμμετοχής Α) ονομάζουμε ένα κλασικό σύνολο το οποίο δεδομένου ότι α $\in$ [0,1] συμβολίζεται με ${}^{\alpha}A$ και ορίζεται ως ${}^{\alpha}A = \{ x \mid A(x) \geq \alpha\}$. Το *δυνατό α-cut* ενός ασαφούς συνόλου συμβολίζεται με ${}^{\alpha+}A$ και ορίζεται ως ${}^{\alpha+}A = \{ x \mid A(x) > \alpha\}$. *Στήριγμα* ενός ασαφούς συνόλου Α ονομάζεται το κλασικό εκείνο σύνολο το οποίο περιλαμβάνει όλα εκείνα τα στοιχεία του καθολικού συνόλου Χ τα οποία λαμβάνουν μη μηδενική τιμή στη συνάρτηση συμμετοχής Α. Επιπλέον *πυρήνα* ενός ασαφούς συνόλου Α ονομάζουμε το σύνολο εκείνο στο οποίο ανήκουν όλα τα στοιχεία του καθολικού συνόλου Χ των οποίων η τιμή της συνάρτησης συμμετοχής Α είναι ίση με 1. Μια εναλλακτική διατύπωση του πυρήνα ενός ασαφούς συνόλου Α είναι το 1-cut του Α. *Ύψος* ενός ασαφούς συνόλου Α ονομάζουμε τη μέγιστη τιμή που λαμβάνει η χαρακτηριστική συνάρτηση Α και το συμβολίζουμε με h(A). Εάν ισχύει h(A)=1 τότε το ασαφές σύνολο ονομάζεται *κανονικό* ενώ εάν έχουμε h(A)<1 τότε το ασαφές σύνολο ονομάζεται *υποκανονικό*.

Όπως στα κλασικά σύνολα ορίζουμε μία σειρά από πράξεις μεταξύ συνόλων, το ίδιο συμβαίνει και με τα ασαφή σύνολα. Εξαιτίας όμως της ιδιαίτερης φύσης τους η πράξη του συμπληρώματος, της ένωσης και της τομής ορίζεται με διαφορετικό τρόπο. Η πράξη του *standard συμπληρώματος* ενός ασαφούς συνόλου Α ορίζεται από την εξίσωση:

$$\overline{A}(x) = 1 - A(x)$$

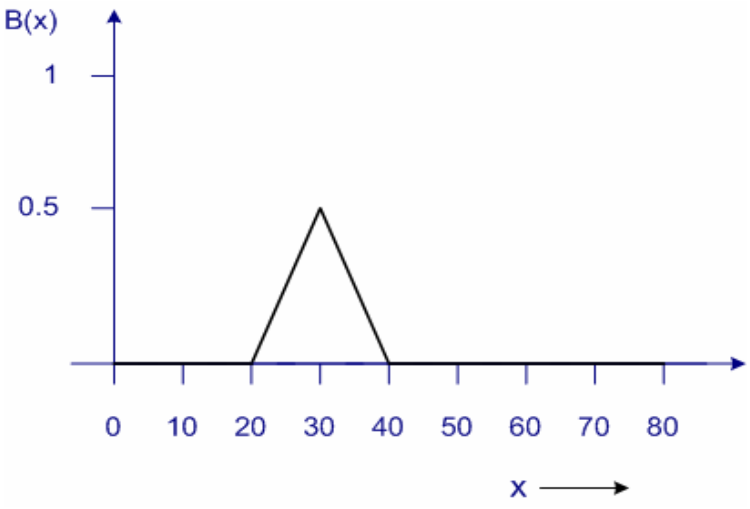
Σημεία του καθολικού συνόλου Α για το οποίο ισχύει $\overline{A}(x) = A(x)$ ονομάζονται σημεία ισορροπίας του ασαφούς συνόλου Α.

Οι δύο παρακάτω εξισώσεις ορίζουν τις πράξεις της *standard ένωσης* και *standard τομής* αντίστοιχα για δύο ασαφή σύνολα Α και Β.

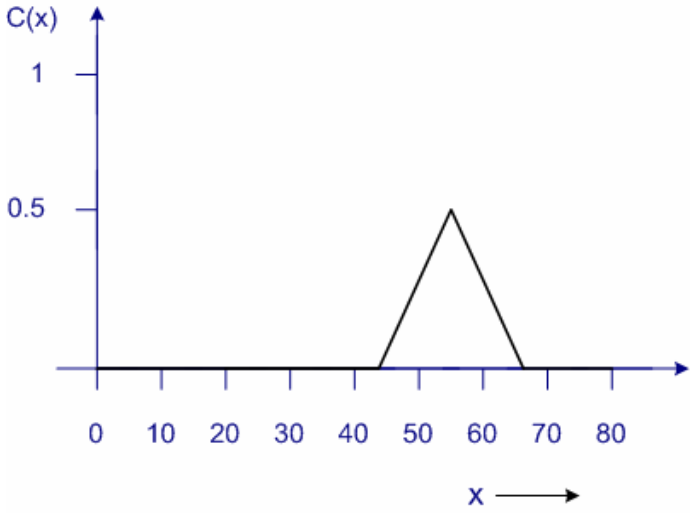
$$(A \cup B)(x) = \max[A(x), B(x)]$$

$$(A \cap B)(x) = \min[A(x), B(x)]$$

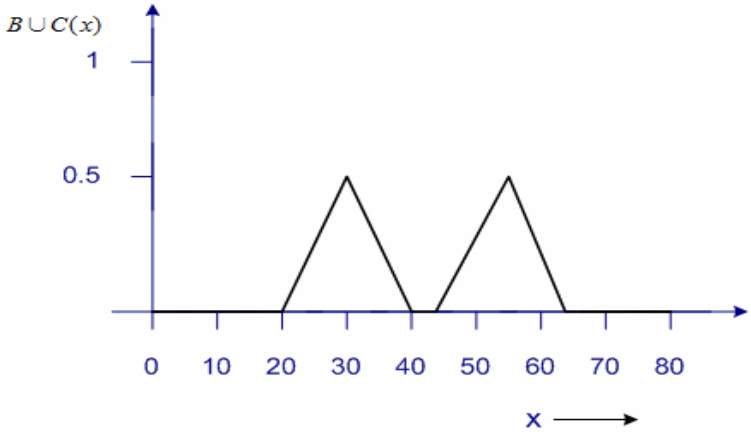
Τα παρακάτω 4 σχήματα περιγράφουν με περισσότερο εποπτικό τρόπο τις παραπάνω βασικές πράξεις που εμφανίζονται μεταξύ των συνόλων.



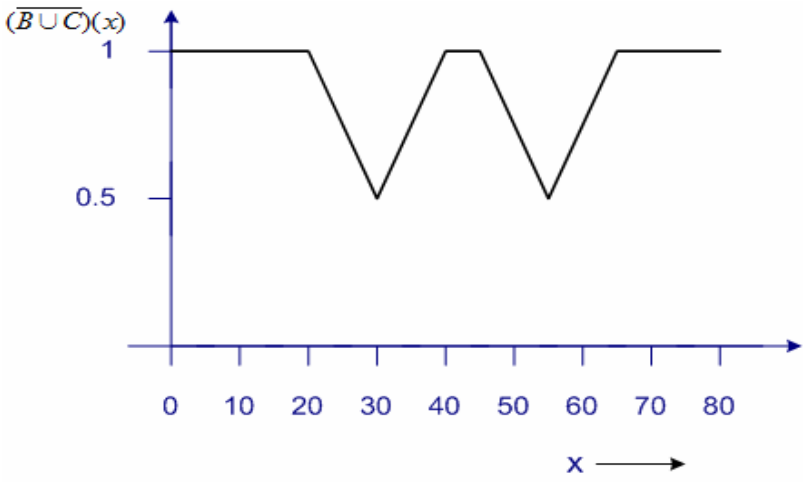
Σχήμα 6. Ασαφές σύνολο B



Σχήμα 7. Ασαφές σύνολο C



Σχήμα 8. Ασαφής ένωση των B και C



Σχήμα 9. Ασαφές συμπλήρωμα της ασαφούς ένωσης των B και C

Οι παραπάνω standard πράξεις ένωσης, τομής και συμπληρώματος είναι αυτές που χρησιμοποιούνται τις περισσότερες φορές στα ασαφή σύνολα. Ωστόσο γι'αυτό ακριβώς ονομάζονται standard, επειδή υπάρχουν περιθώρια για χρήση και άλλων τεχνικών για τις πράξεις ένωσης, τομής και συμπληρώματος. Εντούτοις, οι εν λόγω τεχνικές δεν επιλέγονται αυθαίρετα αλλά πρέπει να ικανοποιούν ένα σύνολο από αξιώματα ([31]). Όσον αφορά το συμβολισμό είναι συγκεκριμένα τα ονόματα που χρησιμοποιούνται για κάθε πράξη ανεξάρτητα από τον αλγόριθμο που κρύβεται πίσω από αυτές. Έτσι με c(A(x)) συμβολίζουμε το συμπλήρωμα ενός ασαφούς συνόλου, το οποίο με τη σειρά του πρόκειται για μία συνάρτηση c: [0,1] → [0,1]. Με t(A(x),B(x)) συμβολίζεται η πράξη της τομής μεταξύ δύο ασαφών συνόλων, όπου t:[0,1]×[0,1] → [0,1], ενώ οι ασαφείς τομές είναι γνωστές και ως t-norms. Αντίστοιχα για την ασαφή ένωση χρησιμοποιούμε το u(A(x),B(x)), με u: [0,1]×[0,1] → [0,1], ενώ χρησιμοποιείται και ο όρος t-conorms γι'αυτές.

Επιπλέον όσον αφορά στα *ασαφή υποσύνολα*, αυτά ορίζονται με βάση τη συνάρτηση συμμετοχής. Συγκεκριμένα, δεδομένων δύο ασαφών συνόλων A και B λέμε ότι το A είναι υποσύνολο του B και γράφουμε $A \subseteq B$ αν και μόνο αν $A(x) \leq B(x)$, για κάθε x που ανήκει στο καθολικό σύνολο X.

2.5.2 Ασαφείς αριθμοί και γλωσσικές μεταβλητές

Διαισθητικά, ένας *ασαφής αριθμός* δεν είναι τίποτα παραπάνω από ένα ασαφές σύνολο με πεδίο ορισμού το σύνολο των πραγματικών αριθμών, που αντιπροσωπεύει έναν πραγματικό

αριθμό. Λέγοντας αντιπροσωπεύει έναν πραγματικό αριθμό, εννοούμε ότι η συνάρτηση συμμετοχής λαμβάνει τιμές κοντά στο 1 όσο προσεγγίζουμε τον αριθμό αυτό (είτε από αριστερά είτε από δεξιά) ενώ οι τιμές της μειώνονται καθώς απομακρυνόμαστε από αυτόν. Πιο συγκεκριμένα, ένα ασαφές σύνολο Α, με Α: $\Re \rightarrow [0,1]$ για να χαρακτηριστεί ασαφής αριθμός θα πρέπει να διαθέτει τις παρακάτω ιδιότητες:

1. Το Α θα πρέπει να είναι κανονικό ασαφές σύνολο.
2. Το ${}^{\alpha}A$ θα πρέπει να είναι ένα κλειστό διάστημα για κάθε $\alpha \in (0,1]$.
3. Το στήριγμα του Α, δηλαδή το ${}^{0+}A$, θα πρέπει να έχει όρια.

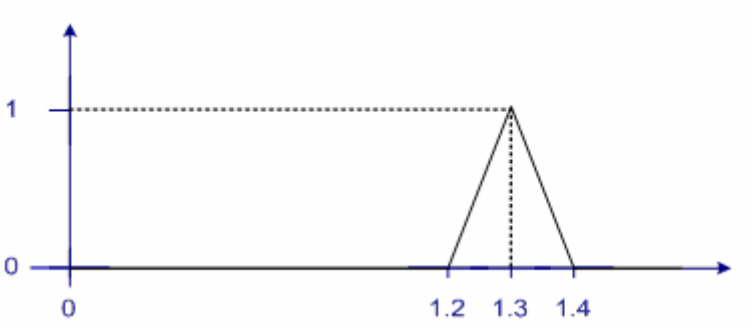
Η πρώτη ιδιότητα συμβαδίζει με την αντίληψη που έχουμε για τους ασαφείς αριθμούς, σύμφωνα με την οποία στο πεδίο ορισμού του ασαφούς αριθμού θα υπάρχει τουλάχιστον ένας αριθμός για τον οποίο η συνάρτηση συμμετοχής θα είναι 1. Οι άλλες δύο ιδιότητες σχετίζονται με την ικανότητα μας να ορίσουμε αριθμητικές πράξεις μεταξύ ασαφών αριθμών χρησιμοποιώντας ήδη ανεπτυγμένες τεχνικές που προέρχονται από την κλασική ανάλυση. Παρακάτω δίνεται ένα θεώρημα το οποίο περιγράφει τους ασαφείς αριθμούς και συγκεκριμένα τη συνάρτηση συμμετοχής τους με περισσότερο αυστηρό και τυπικό τρόπο.

Θεώρημα. Έστω ένα ασαφές σύνολο Α, με Α: $\Re \rightarrow [0,1]$. Τότε το Α είναι ασαφής αριθμός εάν και μόνο εάν υπάρχει ένα κλειστό διάστημ $[a,b] \neq \varnothing$ τέτοιο ώστε

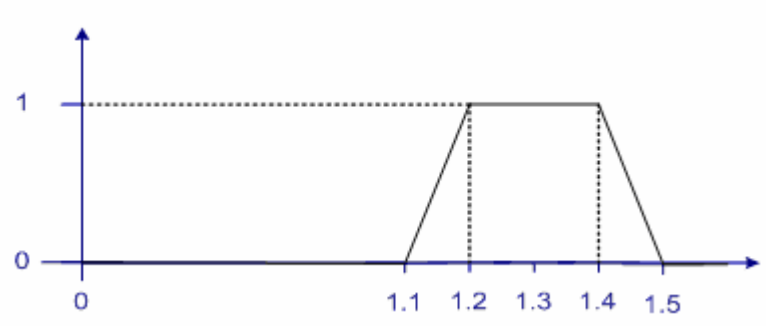
$$A(x) = \begin{cases} 1 & \text{για } x \in [a,b] \\ l(x) & \text{για } x \in (-\infty, a) \\ r(x) & \text{για } x \in (b, \infty) \end{cases}$$

όπου l είναι μία συνάρτηση από το $(-\infty,\alpha)$ στο $[0,1]$, μονότονα αύξουσα, συνεχής από τα δεξιά και τέτοια ώστε $l(x)=0$ για $x \in (-\infty,\omega_1)$, r είναι μία συνάρτηση από το (b,∞) στο $[0,1]$, μονότονα φθίνουσα, συνεχής από τα αριστερά και τέτοια ώστε $r(x)=0$ για $x \in (\omega_2, \infty)$.

Τυπικές συναρτήσεις συμμετοχής που χρησιμοποιούνται για τους ασαφείς αριθμούς είναι οι τριγωνικές και οι τραπεζοειδείς ενώ αρκετά δημοφιλείς είναι και οι γκαουσιανές Παρακάτω δίνονται δύο ασαφείς αριθμοί με διαφορετικές συναρτήσεις συμμετοχής, μία τριγωνική και μία τραπεζοειδή.



Σχήμα 10. Ασαφής αριθμός με τριγωνική συνάρτηση συμμετοχής



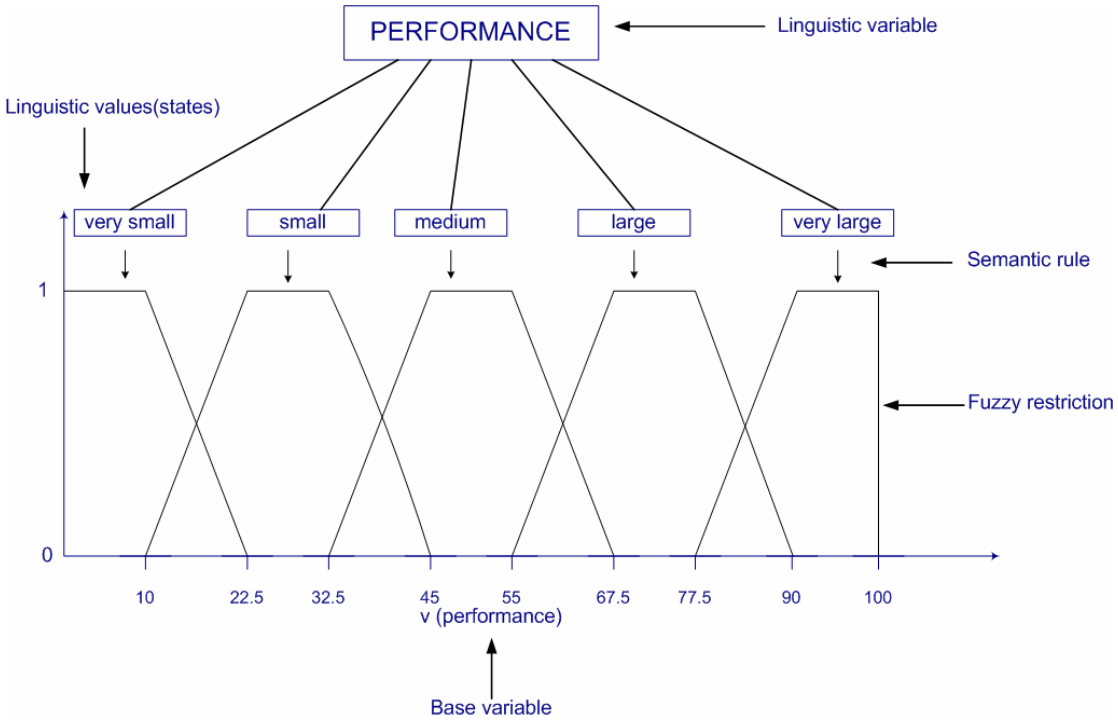
Σχήμα 11. Ασαφής αριθμός με τραπεζοειδή συνάρτηση συμμετοχής

Βασιζόμενοι στους ασαφείς αριθμούς μπορούμε να ορίσουμε τις γλωσσικές μεταβλητές, μία μαθηματική κατασκευή που συναντάται αρκετά συχνά στις εφαρμογές. Μία γλωσσική μεταβλητή διαθέτει μία *μεταβλητή βάσης,* οι τιμές της οποίας είναι πραγματικοί αριθμοί σ'ένα προκαθορισμένο εύρος. Μία μεταβλητή βάσης πρόκειται ουσιαστικά για ένα μέγεθος του οποίου οι τιμές κυμαίνονται και το οποίο μπορεί να έχει είτε φυσική υπόσταση (π.χ. θερμοκρασία) είτε να πρόκειται για κάτι μετρήσιμο (π.χ. αξιοπιστία). Στα πλαίσια μιας γλωσσικής μεταβλητής χρησιμοποιούνται διαβαθμίσεις (π.χ. πολύ λίγο, λίγο, μέτρια, πολύ πάρα πολύ) οι οποίες αντιστοιχούν σε ασαφείς αριθμούς.

Με πιο τυπικό τρόπο, μία γλωσσική μεταβλητή αποτελείται από μία πεντάδα (υ, T, X, g, m), όπου υ είναι το όνομα της μεταβλητής, T είναι το σύνολο των γλωσσικών όρων που αναφέρονται στη μεταβλητή υ και των οποίων η σημασία καλύπτει όλο το εύρος του καθολικού συνόλου X, g είναι ένας γραμματικός κανόνας για την παραγωγή γλωσσικών όρων και m είναι ένας σημασιολογικός κανόνας ο οποίος αναθέτει σε κάθε γλωσσικό όρο $t \in T$ μία σημασία m(t) η οποία με τη σειρά της αντιστοιχεί σε ένα ασαφές σύνολο στο X.

Στο παρακάτω σχήμα δίνεται ένα διάγραμμα με τα ασαφή σύνολα που αντιστοιχούν στη γλωσσική μεταβλητή της απόδοσης. Παρατηρούμε έτσι το ρόλο που διαδραματίζουν σε μία γλωσσική μεταβλητή η μεταβλητή βάσης (performance), οι γλωσσικές τιμές (very small,

small, medium, large , very large) και οι σημασιολογικοί κανόνες που αντιστοιχίζουν κάθε γλωσσική τιμή στο ανάλογο ασαφές σύνολο.



Σχήμα 12.Γλωσσική μεταβλητή για την απόδοση

2.6 Ασαφείς οντολογίες

Στην προσπάθεια τους οι επιστήμονες να περιγράψουν με όσο το δυνατό πιο εκφραστικό και άμεσο τρόπο τη γνώση, και είδικα τη γνώση που αφορά καθημερινές εφαρμογές, εισήγαγαν ασάφεια στη χρήση των οντολογιών. Προϊόν αυτής της προσπάθειας είναι η γλώσσα περιγραφικής λογικής f-$\mathcal{SHOIN}$ και η αντίστοιχη γλώσσα περιγραφής οντολογιών f-OWL.

2.6.1 Η περιγραφική λογική f-$\mathcal{SHOIN}$

Όπως έγινε αντιληπτό και στην ενότητα των ασαφών συνόλων η χρήση των συναρτήσεων συμμετοχής τούς προσθέτει ρεαλισμό και ακρίβεια στην αναπαράσταση της γνώσης. Δεδομένου ότι οι περιγραφικές λογικές αποτελούν έναν από τους πιο δημοφιλείς τρόπους περιγραφής της γνώσης, ο συνδυασμός των δύο παραπάνω δεν μπορεί παρά να έχει θετικά αποτελέσματα στην αύξηση της αμεσότητας και της αξιοπιστίας ενός συστήματος γνώσης. Εάν λάβει κανείς υπόψη του μάλιστα το σκοπό για τον οποίο προορίζονται οι περιγραφικές λογικές, δηλαδή το Σημασιολογικό Διαδίκτυο, η ανάγκη για αναπαράσταση της ασάφειας με

έναν τυπικό τρόπο αυξάνεται σημαντικά. Το Σημασιολογικό Διαδίκτυο σκοπό έχει τη διεκπεραίωση ανθρώπινων υποχρεώσεων με αυτόματο τρόπο, και δεδομένου ότι κάθε πλευρά της καθημερινής μας ζωής εμπεριέχει υψηλό βαθμό αβεβαιότητας ως προς την περιγραφή της, μία ασαφής γλώσσα περιγραφικής λογικής καθιστά αυτήν την περιγραφή πιο εύκολη και εκφραστική.

Η γλώσσα περιγραφικής λογικής η οποία έχει επεκταθεί με ασάφεια είναι η $\mathcal{SHOIN}$ και είναι γνωστή ως f-$\mathcal{SHOIN}$. Το αλφάβητο της ακολουθώντας τις παραδοσιακές περιγραφικές λογικές, αποτελείται από ονόματα ατόμων (**I**), ρόλων (**R**) και εννοιών (**C**). Ο παρακάτω ορισμός περιγράφει με αυστηρό τρόπο τους f-$\mathcal{SHOIN}$ ρόλους και τις f-$\mathcal{SHOIN}$ έννοιες.

Ορισμός. Έστω RN ∈ **R** ένα όνομα ρόλου και R ένας f-$\mathcal{SHOIN}$ ρόλος. Ένας f-$\mathcal{SHOIN}$ ρόλος ορίζεται από την αφηρημένη σύνταξη: R ::= RN| R^-. Η αντίστροφη σχέση στους ρόλους είναι συμμετρική και για να αποφεύγονται ρόλοι του τύπου R^-, ορίζουμε μία συνάρτηση Inv(R) := RN^-, εάν R=RN και Inv(R) := RN, εάν R=RN^-. Το σύνολο των f-$\mathcal{SHOIN}$ εννοιών είναι το μικρότερο σύνολο τέτοιο ώστε:

1. κάθε όνομα έννοιας CN ∈ **C** είναι μία f-$\mathcal{SHOIN}$ έννοια.
2. εάν o ∈ **I** τότε {o} είναι μία f-$\mathcal{SHOIN}$ έννοια.
3. εάν C και D είναι f-$\mathcal{SHOIN}$ έννοιες, R είναι ένας f-$\mathcal{SHOIN}$ ρόλος, S ένας απλός f-$\mathcal{SHOIN}$ ρόλος (δηλαδή όχι μεταβατικός και χωρίς μεταβατικούς υπορόλους) και p ∈ ℵ, τότε τα (C⊔D), (C⊓D), (¬C), (∀R.C), (∃R.C), (≤pS) και (≥pS) είναι επίσης f-$\mathcal{SHOIN}$ έννοιες.

Ένα ασαφές σώμα ορολογιών (fuzzy TBox) είναι ένα πεπερασμένο σύνολο από αξιώματα ασαφών εννοιών. Τα αξιώματα ασαφών εννοιών διακρίνονται με τη σειρά τους σε *ασαφείς υπαγωγές* (με τη μορφή A⊑C) και *ασαφείς ισοδυναμίες* (με τη μορφή A≡C). Ακόμη, ένα ασαφές σώμα ρόλων (fuzzy RBox) είναι ένα πεπερασμένο σύνολο από αξιώματα ασαφών ρόλων. Τα αξιώματα ασαφών ρόλων είναι της μορφής Trans(RN) (όπου RN είναι το όνομα ενός ρόλου) και ονομάζονται *αξιώματα ασαφών μεταβατικών ρόλων* καθώς και της μορφής R⊑S και ονομάζονται *αξιώματα υπαγωγής ασαφών ρόλων*. Τέλος ένα ασαφές σώμα ισχυρισμών (fuzzy ABox) περιλαμβάνει ένα πεπερασμένο σύνολο από ασαφείς ισχυρισμούς. Ένας ασαφής ισχυρισμός παίρνει μία από τις μορφές $\langle a : C \rhd\lhd n \rangle$, $\langle (a,b) : R \rhd\lhd n \rangle$ και

α⇌b, όπου $\rhd\lhd \in \{<,\leq,>,\geq\}$ και $a,b \in \mathbf{I}$. Μία ασαφής βάση γνώσης αποτελείται από την τριάδα $\langle \mathcal{T},\mathcal{R},\mathcal{A}\rangle$ της οποίας τα μέλη αντιστοιχούν στο fuzzy TBox, fuzzy RBox και fuzzy ABox αντίστοιχα.

Όσον αφορά τη σημασιολογία αυτό το οποίο ουσιαστικά συμβαίνει πλέον στην f-$\mathcal{SHOIN}$ είναι ότι ένα αντικείμενο (ή ζεύγος από αντικείμενα) ανήκει σε μία έννοια (ή σε ένα ρόλο) κατά ένα βαθμό ο οποίος κυμαίνεται από 0 εώς 1. Παρακάτω δίδεται η πλήρης σημασιολογία:

$$\{o\}^{\mathcal{I}}(\alpha)= 1 \text{ εάν } \alpha \in \{o^{\mathcal{I}}\}$$

$$\bot^{\mathcal{I}}(\alpha)= 0$$

$$\top^{\mathcal{I}}(\alpha)= 1$$

$$(C\sqcap D)^{\mathcal{I}}(\alpha)=t(C^{\mathcal{I}}(\alpha), D^{\mathcal{I}}(\alpha))$$

$$(C\sqcup D)^{\mathcal{I}}(\alpha)=u(C^{\mathcal{I}}(\alpha), D^{\mathcal{I}}(\alpha))$$

$$(\neg C)^{\mathcal{I}}(\alpha)=c(C^{\mathcal{I}}(\alpha))$$

$$(\exists R.C)^{\mathcal{I}}(\alpha)=\sup_{b \in \Delta^{\mathcal{I}}} t(R^{\mathcal{I}}(\alpha,b), C^{\mathcal{I}}(b))$$

$$(\forall R.C)^{\mathcal{I}}(\alpha)=\inf_{b \in \Delta^{\mathcal{I}}} \mathcal{J}(R^{\mathcal{I}}(\alpha,b), C^{\mathcal{I}}(b))$$

$$(\geq nR)^{\mathcal{I}}(\alpha)= \sup_{b_1,\ldots, b_n \in \Delta^{\mathcal{I}}} t_{i=1}^{n} R^{\mathcal{I}}(\alpha,b_i)$$

$$(\leq nR)^{\mathcal{I}}(\alpha)= \inf_{b_1,\ldots, b_{n+1} \in \Delta^{\mathcal{I}}} u_{i=1}^{n+1} c(R^{\mathcal{I}}(\alpha,b_i))$$

Στην παραπάνω σημασιολογία το c αντιστοιχεί σε ένα ασαφές συμπλήρωμα, το t σε μία ασαφή τομή, το u σε μία ασαφή ένωση και το $\mathcal{J}$ σε μία ασαφή συνεπαγωγή. Επιπλέον από τον ορισμό της παραπάνω σημασιολογίας παρατηρούμε ότι έννοιες της μορφής $\{o\}$, όπου $o \in \mathbf{I}$ (nominals), δεν έχουν ασαφοποιηθεί. Πρόκειται για μία λογική εξέλιξη καθώς συνήθως οι ονοματικές έννοιες δε φέρουν ασαφή και αβέβαιη πληροφορία. Μία άλλη παρατήρηση είναι ότι στις ακραίες περιπτώσεις που ο βαθμός ασάφειας είναι 0 είτε 1 και το σύμβολο $\rhd\lhd$ $\leq$ ή $\geq$ αντίστοιχα, η ασαφής ερμηνεία συμπίπτει με την κλασική ερμηνεία.

Μία f-$\mathcal{SHOIN}$ έννοια είναι ικανοποιήσιμη αν και μόνο αν υπάρχει μία ασαφή ερμηνεία $\mathcal{I}$ για την οποία υπάρχει κάποιο $\alpha \in \Delta^{\mathcal{I}}$ τέτοιο ώστε $C^{\mathcal{I}}(\alpha)=n$, με $n \in (0,1]$. Μία ασαφής ερμηνεία $\mathcal{I}$ ικανοποιεί ένα ασαφές TBox $\mathcal{T}$ ανν $\forall\ \alpha \in \Delta^{\mathcal{I}}$, $A^{\mathcal{I}}(\alpha) \leq C^{\mathcal{I}}(\alpha)$ για κάθε $A \sqsubseteq C \in \mathcal{T}$ και $A^{\mathcal{I}}(\alpha)=C^{\mathcal{I}}(\alpha)$ για κάθε $A \equiv C \in \mathcal{T}$. Μία ασαφής ερμηνεία $\mathcal{I}$ ικανοποιεί ένα ασαφές RBox $\mathcal{R}$ ανν $\forall\ \alpha,c \in \Delta^{\mathcal{I}}$, $R^{\mathcal{I}}(\alpha,c) \geq \sup_{b \in \Delta^{\mathcal{I}}} \{t(R^{\mathcal{I}}(\alpha,b), R^{\mathcal{I}}(b,c))\}$ για κάθε $trans(R) \in \mathcal{R}$, $\forall \langle \alpha,b \rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ $R^{\mathcal{I}}(\alpha,b) \leq S^{\mathcal{I}}(\alpha,b)$ για κάθε $R \sqsubseteq S \in \mathcal{R}$ και $\forall \langle \alpha,b \rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ $(R^{-})^{\mathcal{I}}(b,\alpha) \leq R^{\mathcal{I}}(\alpha,b)$ για κάθε $R \in \mathcal{R}$. Επίσης δεδομένης μιας ασαφούς ερμηνείας $\mathcal{I}$, η $\mathcal{I}$ ικανοποιεί το $\langle \alpha : C \geq n \rangle$ εάν $C^{\mathcal{I}}(\alpha^{\mathcal{I}}) \geq n$, το $\langle (\alpha,b) : R \geq n \rangle$ εάν $R^{\mathcal{I}}(\alpha^{\mathcal{I}}, b^{\mathcal{I}}) \geq n$ και τέλος το $\alpha \doteq b$ εάν $\alpha^{\mathcal{I}} \neq b^{\mathcal{I}}$. Ανάλογα ορίζεται η ικανοποιησιμότητα για τα $\leq, <, >, \geq$ ενώ μία ασαφής ερμηνεία $\mathcal{I}$ ικανοποιεί ένα ασαφές ABox εάν ικανοποιεί όλους τους ισχυρισμούς που περιέχονται σε αυτό. Σε αυτή την περίπτωση λέμε ότι η $\mathcal{I}$ αποτελεί ένα μοντέλο του $\mathcal{A}$. Εάν το $\mathcal{A}$ έχει τουλάχιστον ένα μοντέλο τότε λέμε ότι είναι *συνεπές*. Τέλος, μία ασαφής βάση γνώσης Σ είναι ικανοποιήσιμη αν και μόνο αν υπάρχει μία ασαφής ερμηνεία $\mathcal{I}$ η οποία ικανοποιεί όλα τα αξιώματα που η Σ περιέχει. Επιπλέον η Σ, συνεπάγεται έναν ισχυρισμό $\langle \varphi \rhd \lhd n \rangle$ (γράφεται $\Sigma \models \langle \varphi \rhd \lhd n \rangle$) αν και μόνο αν κάθε μοντέλο του Σ επίσης ικανοποιεί τον ασαφή ισχυρισμό.

Όσον αφορά τον αλγόριθμο συλλογιστικής που θα εφαρμοστεί στην παραπάνω περίπτωση ανήκει και πάλι στην οικογένεια των αλγορίθμων tableaux. Τροποποιώντας ήδη υπάρχοντες αλγορίθμους έχει διατυπωθεί αλγόριθμος ο οποίος αποφασίζει για τη συνέπεια ενός ABox σε f_{KD}-$\mathcal{SHIN}$. Η f_{KD}-$\mathcal{SHIN}$ γλώσσα προέρχεται από τη f-$\mathcal{SHIN}$ γλώσσα στην οποία όμως έχουν καθοριστεί ποιες θα είναι οι ασαφείς συνολοθεωρητικές πράξεις. Συγκεκριμένα για το ασαφές συμπλήρωμα έχουμε $c(a)=1-a$, για την ασαφή τομή $t(a,b)=\min(a,b)$, για την ασαφή ένωση $u(a,b)=\max(a,b)$ και για την ασαφή συνεπαγωγή $\mathcal{J}(a,b)=\max(1-a,b)$.

2.6.2 Η γλώσσα αναπαράστασης ασαφούς λογικής f-OWL

Η σύνταξη της f-OWL είναι ίδια με τη σύνταξη της OWL DL, με τη μόνη διαφορά να εντοπίζεται στο κομμάτι των ατόμων. Πράγματι οι ισχυρισμοί που εμπεριέχονται στο ABox είναι οι μόνοι στους οποίους υπεισέρχεται η ασάφεια γι'αυτό και η μορφή τους είναι

διαφορετική από αυτή που συναντάμε στην OWL DL. Παρακάτω δίδεται αυτή η μορφή σε αφηρημένη σύνταξη:

individual ::= 'Individual(' [individualID] {annotation}
{'type'(type ')' [membership]} {value [membership] } ')'
membership ::= ineqType degree
ineqType ::= '>=' | '>' | '<=' | '<'
degree ::= 'degree(' real-number-between-0-and-1-inclusive ')'

Συνήθως ένα ερώτημα που απευθύνουμε σε μία βάση γνώσης έχει τη μορφή $\exists x_1 \ldots x_n(q_1 \wedge \ldots \wedge q_n)$, όπου $q_1 \ldots q_n$ είναι όροι εννοιών ή ρόλων. Στις κλασικές γλώσσες περιγραφικής λογικής η τύχη ενός τέτοιου ερωτήματος που δεν ικανοποιεί επακριβώς τους περιορισμούς που έχουν τεθεί, είναι να επιστρέψει ένα κενό αποτέλεσμα. Αντίθετα, στις ασαφείς γλώσσες περιγραφικής λογικής παρουσιάζεται το ενδιαφέρον ενδεχόμενο να επιστραφεί ένα σύνολο από στιγμιότυπα τα οποία βρίσκονται εντός του εύρους που η ασάφεια έχει θέσει και μάλιστα με φθίνουσα σειρά, ξεκινώντας από το πιο κατάλληλο για τους περιορισμούς του ερωτήματος. Έτσι εμφανίζονται στον χρήστη οι πιο σχετικές πληροφορίες αλλά και οι λιγότερο σχετικές σε χαμηλότερη θέση στην κατάταξη, δίανοίγοντας έτσι στις περιγραφικές λογικές τη δυνατότητα να χρησιμοποιηθούν επιτυχημένα σε συστήματα ανάκτησης πληροφοριών.

Αφηρημένη σύνταξη	Σημασιολογία
(Class A partial $C_1 \ldots C_n$)	$A^{\mathcal{I}}(\alpha) \leq t(C_1^{\mathcal{I}}(\alpha), \ldots, C_n^{\mathcal{I}}(\alpha))$
(Class A complete $C_1 \ldots C_n$)	$A^{\mathcal{I}}(\alpha) = t(C_1^{\mathcal{I}}(\alpha), \ldots, C_n^{\mathcal{I}}(\alpha))$
(Enumerated Class A $o_1 \ldots o_n$)	$A^{\mathcal{I}}(\alpha) = 1$ εάν $\alpha \in \{ o_1^{\mathcal{I}}, \ldots, o_n^{\mathcal{I}}\}$ $A^{\mathcal{I}}(\alpha) = 0$ αλλιώς
(SubClassOf C_1 C_2)	$C_1^{\mathcal{I}}(\alpha) \leq C_2^{\mathcal{I}}(\alpha)$
(EquivalentClasses $C_1 \ldots C_n$)	$C_1^{\mathcal{I}}(\alpha) = \ldots = C_n^{\mathcal{I}}(\alpha)$

(DisjointClasses $C_1 \ldots C_n$)	$t(C_i^{\mathcal{I}}(\alpha),\ldots, C_j^{\mathcal{I}}(\alpha))= 0\ 1\leq i<j\leq n$
(SubPropertyOf R_1 R_2)	$R_1^{\mathcal{I}}(\alpha,b) \leq R_2^{\mathcal{I}}(\alpha,b)$
(EquivalentProperties $R_1 \ldots R_n$)	$R_1^{\mathcal{I}}(\alpha,b) =\ldots=R_n^{\mathcal{I}}(\alpha,b)$
ObjectProperty (R super(R_1). . .super(R_n)	$R^{\mathcal{I}}(\alpha,b) \leq R_i^{\mathcal{I}}(\alpha,b)$
Domain(C_1). . . domain(C_k)	$R^{\mathcal{I}}(\alpha,b) \leq C_i^{\mathcal{I}}(\alpha)$
range(C_1). . . range(C_h)	$R^{\mathcal{I}}(\alpha,b) \leq C_i^{\mathcal{I}}(b)$
[Symmetric]	$R^{\mathcal{I}}(\alpha,b) =(R^-)^{\mathcal{I}}(\alpha,b)$
[Functional]	$\forall\alpha \in \Delta^{\mathcal{I}}\ \inf_{b_1,\ b_2 \in \Delta^{\mathcal{I}}} u(R(\alpha,b_1), R(\alpha,b_2))\geq 1$
[InverseFunctional]	$\forall\alpha \in \Delta^{\mathcal{I}}\ \inf_{b_1,\ b_2 \in \Delta^{\mathcal{I}}} u(R^-(\alpha,b_1), R^-(\alpha,b_2))\geq 1$
[Transitive])	$\sup_{b \in \Delta^{\mathcal{I}}} t(R^{\mathcal{I}}(\alpha,b), R^{\mathcal{I}}(b,c))\leq R^{\mathcal{I}}(a,c)$
Individual(o type(C_1)[$\rhd\lhd$degree(m_1)]… type(C_n) [$\rhd\lhd$degree(m_n)]	$C_i^{\mathcal{I}}(\ o^{\mathcal{I}}) \rhd\lhd m_i,\ m_i \in [0,1],\ 1\leq i\leq n$
value(R_1,o_1)[$\rhd\lhd$degree(k_1)]… value(R_l,o_l) [$\rhd\lhd$degree(k_l)])	$R_i^{\mathcal{I}}(\ o^{\mathcal{I}},o^{\mathcal{I}}_i) \rhd\lhd k_i,\ k_i \in [0,1],\ 1\leq i\leq l$
SameIndividual(o_1 ... o_n)	$o^{\mathcal{I}}_1=\ldots= o^{\mathcal{I}}_n$
DifferentIndividual(o_1... o_n)	$o^{\mathcal{I}}_i\neq o^{\mathcal{I}}_j,\ 1\leq i<j\leq n$

Πίνακας 1.Σημασιολογία της f-OWL

Στο σημείο αυτό να επισημάνουμε, ότι εάν για έναν ισχυρισμό της f-OWL δεν υπάρχει βαθμός, υποννοείται το 1.

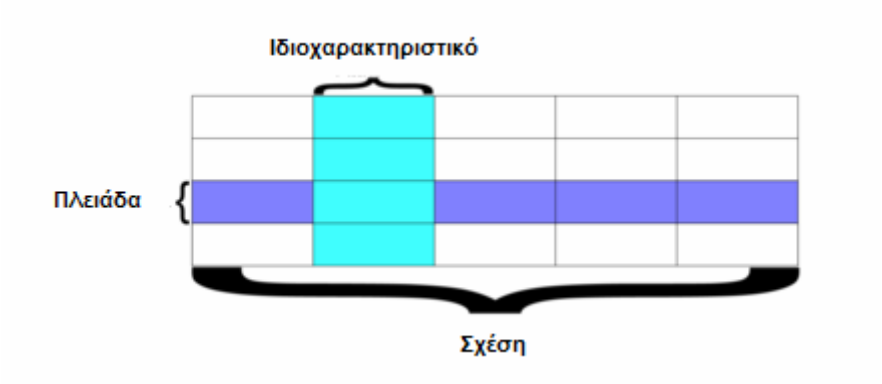
2.7 Σχεσιακές βάσεις δεδομένων

Τέλος, δεδομένου ότι το σύστημα σκοπό έχει να διασυνδέσει δεδομένα που αντλούνται από σχεσιακές βάσεις δεδομένων με οντολογίες, θα αναφέρουμε ορισμένα στοιχειώδη πράγματα από τη θεωρία των σχεσιακών βάσεων δεδομένων.

2.7.1 Εφαρμογές και ορισμοί

Οι σχεσιακές βάσεις δεδομένων αποτελούν έναν από τους πλέον δημοφιλείς τρόπους αναπαράστασης των δεδομένων, που βρίσκονται σε μορφή πινάκων. Το ευρός των εφαρμογών είναι τεράστιο και ποικίλλει· κομβικά συστήματα της παγκόσμιας οικονομίας, συστήματα αποθήκευσης δεδομένων εταιρειών κάθε είδους και μεγέθους, αναπαράσταση προσωπικών δεδομένων αποτελούν μερικά μόνο από τα πεδία των εφαρμογών.

Σύμφωνα με την καθιερωμένη ορολογία, μία σχεσιακή βάση δεδομένων συνίσταται από ένα σύνολο από *σχέσεις (relations)* οι οποίες αντιστοιχούν στους πίνακες δεδομένων. Εξετάζοντας μία σχέση, σε οριζόντιο επίπεδο αποτελείται από *πλειάδες (tuples)* ενώ σε κάθετο από *ιδιοχαρακτηριστικά (attributes)*. Παρακάτω δίδεται η σχηματική αναπαράσταση των παραπάνω εννοιών.



Σχήμα 13. Μία σχέση, το δομικό στοιχείο κάθε σχεσιακής βάσης δεδομένων

Πρωτεύον κλειδί μιας σχέσης σε μια βάση δεδομένων ονομάζουμε το attribute εκείνο το οποίο έχει διαφορετική τιμή για κάθε πλειάδα της σχέσης. Με τον τρόπο αυτό όλες οι πλειάδες της σχέσης είναι ανά δύο διαφορετικές μεταξύ τους, μία ιδιότητα ιδιαίτερα χρήσιμη στις πρακτικές εφαρμογές όπου θέλουμε να υπάρχει ένα προς ένα αντιστοιχία μεταξύ μιας πλειάδας και της αντίστοιχης ερμηνείας της στον πραγματικό κόσμο. Σε ορισμένες περιπτώσεις είναι απαραίτητη η χρήση δύο ή περισσότερων attributes προκειμένου να διαφοροποιηθούν οι πλειάδες μεταξύ τους, οπότε έχουμε ένα *σύνθετο κλειδί*. Τέλος, ξένο

κλειδί είναι το attribute εκείνο που ενώ υπάρχει σε μία σχέση και δεν αποτελεί πρωτεύον κλειδί γι'αυτή, είναι το πρωτεύον κλειδί μιας δεύτερης σχέσης με την οποία η πρώτη συνδέεται νοητά.

2.7.2 Σχεσιακή άλγεβρα

Οι σχεσιακές βάσεις δεδομένων στηρίζονται σε ένα στέρεο και εξαιρετικά ισχυρό μαθηματικό υπόβαθρο, τη *σχεσιακή άλγεβρα* η οποία έχει τις βάσεις τις στον πρωτοβάθμιο κατηγορηματικό λογισμό. Ασχολείται με ένα σύνολο από σχέσεις, το οποίο είναι κλειστό κάτω από τους τελεστές που επιδρούν σε αυτές, δηλαδή των οποίων τα αποτελέσματα είναι επίσης σχέσεις. Παρακάτω δίδεται ο ορισμός δύο εκ των τελεστών που χρησιμοποιούνται ιδιαίτερα συχνά στο χώρο της σχεσιακής άλγεβρας.

Μία *προβολή* είναι ένας εναδικός τελεστής ο οποίος γράφεται ως $\pi_{a_1,...,a_n}(R)$, όπου $a_1,...,a_n$ είναι ένα σύνολο από ονόματα ιδιοχαρακτηριστικών της σχέσης R. Το αποτέλεσμα της προβολής είναι το σύνολο εκείνο το οποίο αποκτάται από τις πλειάδες της R, λαμβάνοντας όμως μόνο τα ιδιοχαρακτηριστικά $a_1,...,a_n$.

Μία *γενικευμένη επιλογή* είναι ένας εναδικός τελεστής ο οποίος γράφεται ως $\sigma_{\varphi}(R)$, όπου φ είναι μία προτασιακή φόρμουλα που αποτελείται από *άτομα* (ορίζονται παρακάτω) τα οποία συνδέονται μεταξύ τους με τους τελεστές $\wedge$ (and), $\vee$ (or), $\neg$(not). Το αποτέλεσμα περιλαμβάνει όλες εκείνες τις πλειάδες της σχέσης R για τις οποίες η φ ισχύει. Τα άτομα είναι κατασκευές της μορφής $\alpha\theta b$ ή $\alpha\theta v$ όπου α και b είναι attributes, θ είναι ένας από τους δυαδικούς τελεστές σύγκρισης $\{<,\leq,=,>,\geq\}$ και v είναι μία σταθερά.

3

Προτεινόμενη μεθοδολογία

3.1 Αντιστοίχιση απόδοσης νοήματος

3.1.1 Διατύπωση του προβλήματος

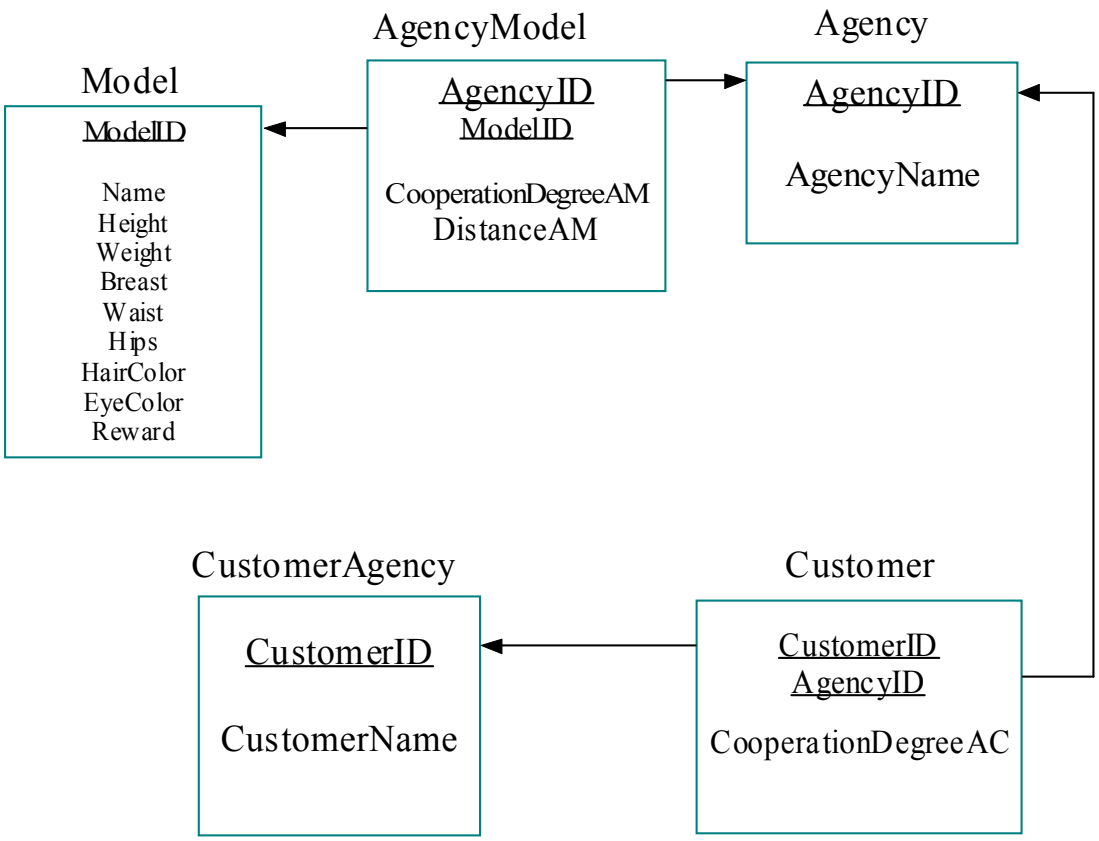
Στις παραπάνω παραγράφους έχουμε ήδη παρουσιάσει τους επίσημους και μαθηματικά τεκμηριωμένους τρόπους που διαθέτει η επιστημονική κοινότητα για την αναπαράσταση της ασαφούς οντολογικής γνώσης καθώς επίσης και των σχεσιακών βάσεων δεδομένων. Όσον αφορά στις οντολογίες μπορούμε με τη βοήθεια ενός συνόλου ασαφών ισχυρισμών που υπακούν σε συγκεκριμένη σύνταξη να διατυπώσουμε με ακρίβεια και πληρότητα γεγονότα που προέρχονται από τον κόσμο. Οι ασαφείς περιγραφικές λογικές οριοθετούν τους κανόνες αυτής της σύνταξης.Οι περιορισμοί που τίθενται ως προς τη σύνταξη καθιστούν τους ισχυρισμούς αυτούς επεξεργάσιμους από μηχανές και κατ'επέκταση κατάλληλους για είσοδο σε αλγορίθμους συλλογιστικής. Οι εν λόγω αλγόριθμοι θα παράγουν εκ νέου ισχυρισμούς οι οποίοι είναι διατυπωμένοι με όμοιο τυπικό τρόπο, συντελώντας έτσι στην αυτοματοποίηση της διαδικασίας. Από την άλλη πλευρά πάλι, οι τρόποι αναπαράστασης των σχεσιακών βάσεων δεδομένων είναι καθιερωμένοι εδώ και δεκαετίες μέσω της σχεσιακής άλγεβρας, ένας συμβολισμός που έχει αποδειχθεί ιδιαίτερα αποδοτικός και αποτελεσματικός ως προς τη λειτουργία συστημάτων διαχείρισης βάσεων δεδομένων.

Αντικειμενικός σκοπός της παρούσας διπλωματικής εργασίας είναι η διασύνδεση οντολογικής γνώσης με βάσεις δεδομένων, υπό την παρουσία πάντα της αβεβαιότητας. Η

διασύνδεση αυτή εκφράζεται με ένα σύνολο από κανόνες. Πρόκειται για κανόνες οι οποίοι αναλαμβάνουν να απομονώσουν τμήματα απο το περιεχόμενο μιας βάσης δεδομένων και σε συνδυασμό με συναρτήσεις συμμετοχής- διατυπωμένες σύμφωνα με τη θεωρία των ασαφών συνόλων- καθώς επίσης και μη ασαφείς ορολογίες από οντολογίες, να παράγουν ένα σύνολο απο ασαφείς ισχυρισμούς ως τμήμα μιας οντολογικής γνώσης. Ωστόσο οι κανόνες αυτοί δεν μπορούν να εκφράζονται περιφραστικά π.χ. με φυσική γλώσσα καθώς πρέπει να είναι επεξεργάσιμα από προγράμματα λογισμικού που εμπλέκονται σε πρακτικές εφαρμογές. Οφείλουν επομένως να διατυπωθούν με εναν τυπικό μαθηματικό τρόπο, του οποίου η σημασιολογία να είναι αυστηρά ορισμένη και απαλλαγμένη από αμφισημίες. Η σημασιολογία αυτή συνδέεται άμεσα με τη λειτουργικότητα που εξυπηρετούν οι κανόνες αυτοί, την παραγωγή δηλαδή των ασαφών ισχυρισμών που αποτελεί και το λόγο υπαρξης τους.

Στο σημείο αυτό θα εισάγουμε ένα παράδειγμα το οποίο θα μας ακολουθήσει στο υπόλοιπο παρόν κεφάλαιο και θα καταστήσει περισσότερο σαφείς και αντιληπτους τους συμβολισμούς και τις ερμηνείες που έχουμε υιοθετήσει γι'αυτούς. Ας υποθέσουμε ότι διαθέτουμε μία βάση δεδομένων που ανήκει σε μία επιχείρηση πρακτορείων μοντέλων, ένα παράδειγμα δανεισμένο από την [37]. Η βάση δεδομένων διαθέτει πληροφορίες σχετικά με κάθε ένα από τα μοντέλα ξεχωριστά και τα φυσικά χαρακτηριστικά τους, καθώς επίσης και τις επιχειρήσεις με τις οποίες αυτά έχουν συνεργαστεί. Επιπλέον περιέχει πληροφορίες σχετικά με συνεργασίες που έχουν γίνει είτε μεταξύ μοντέλων και πρακτορείων είτε μεταξύ πρακτορείων και πελατών με ποιοτικά χαρακτηριστικά γι'αυτές (π.χ. βαθμός συνεργασίας). Το σχήμα 14 δίνει το πλήρες σχήμα της βάσης. Όσον αφορά τις ορολογίες της οντολογικής γνώσης που θα χρησιμοποιηθούν κατά τη σύνθεση των κανόνων δίνονται από τον πίνακα 2. Πρόκειται για ατομικές έννοιες και ρόλους οι οποίοι σχετίζονται διασθητικά με κάποιο ιδιοχαρακτηριστικό της βάσης δεδομένων (π.χ. το reward της ΒΔ με τα Cheap, MediumReward, Expensive της οντολογίας). Σκοπός του παρόντος κεφαλαίου είναι να δώσει μια αυστηρή και τυπική υπόσταση σε αυτή τη διασθητική σχέση.

Η ονομασία που επιλέξαμε να δώσουμε στις δομές που θα περιγράψουμε είναι Αντιστοιχία Απόδοσης Νοήματος. Θεωρούμε ότι πρόκεται για μία επιλογή που σκιαγραφεί με ευκρίνεια το σκοπό τον οποίο ο κανόνας εξυπηρετεί. Ένας τέτοιος κανόνας ουσιαστικά γεφυρώνει τα επιμέρους δομικά στοιχεία που προέρχονται από βάσεις δεδομένων και οντολογίες με μία ερμηνεία, τους προσδίδει δηλαδή ένα νόημα. Το νόημα αυτό σχετίζεται άμεσα με τους ασαφείς ισχυρισμούς στους οποίους θα καταλήξουμε και τους αλγόριθμους οι οποίοι ακολουθούνται για την παραγωγή τους.



Σχήμα 14. Σχήμα της βάσης δεδομένων του παραδείγματος.

Atomic concepts	Atomic roles
Cheap	hasModelName
MediumReward	badModelAgencyCooperation
Expensive	goodModelAgencyCooperation
Sexy	excellentModelAgencyCooperation
Elegant	badCustomerAgencyCooperation
Fit	goodCustomerAgencyCooperation
Thin	excellentCustomerAgencyCooperation
MediumWeight	smallModelAgencyDistance
Fat	mediumModelAgencyDistance
Short	longModelAgencyDistance
MediumHeight	isCooperativeWith
Tall	hasAge

Πίνακας 2. Ατομικές έννοιες και ατομικοί ρόλοι που θα χρησιμοποιηθούν στο παράδειγμα.

3.1.2 Ορισμός Αντιστοίχισης Απόδοσης Νοήματος

Παρακάτω κατά την περιγραφή της τυπικής μορφής μιας Αντιστοίχισης Απόδοσης Νοήματος χρησιμοποιούνται ένα σύνολο από συμβολισμοί τους οποίους θα εισάγουμε εδώ. Κατά τη διατύπωση των κανόνων θεωρούμε ότι διαθέτουμε οντολογική γνώση που περιλαμβάνει ένα σύνολο από ατομικές έννοιες και ατομικούς ρόλους-την Οντολογία- καθώς επίσης και μία βάση δεδομένων με ένα σύνολο από σχέσεις – την DB. Η DB έχει *n* σχέσεις συνολικά, ενώ κάθε σχέση της DB έχει l_i ιδιοχαρακτηριστικά, όπου με Rel_i συμβολίζουμε την *i*-οστή σχέση της βάσης. Η Οντολογία διαθέτει m_{AC} ατομικές έννοιες και m_{AR} ατομικούς ρόλους. Συνοψίζοντας:

$n \rightarrow$ ο αριθμός των σχέσεων της DB

$l_i \rightarrow$ ο αριθμός των attributes κάθε σχέσης Rel_i

$m_{AC} \rightarrow$ ο αριθμός των ατομικών εννοιών της Οντολογίας

$m_{AR} \rightarrow$ ο αριθμός των ατομικών ρόλων της Οντολογίας

Έστω λοιπόν ότι διαθέτουμε μία βάση δεδομένων DB η οποία αποτελείται από τις σχέσεις $Rel_1,...,Rel_n$. Κάθε Rel_i, $1 \le i \le n$, αποτελείται από l_i attributes, $a_{i1},...,a_{il_i}$, $1 \le i \le n$. Επίσης όπως προαναφέραμε διαθέτουμε μία οντολογία που διαθέτει τις m_{AC} ατομικές έννοιες $Concept_1,...,Concept_{m_{AC}}$ και τους m_{AR} ατομικούς ρόλους $Role_1,...,Role_{m_{AR}}$. Ορίζουμε ως *αντιστοίχιση απόδοσης νοήματος* την τετράδα $\langle State, OntologyEntity, DatabaseEntity, A \rangle$ στην οποία:

- $State \in \{Fuzzy, Crisp\}$. Μία Αντιστοίχιση Απόδοσης Νοήματος για την οποία ισχύει $State = Fuzzy$ παράγει *ασαφείς* ισχυρισμούς, ενώ για $State = Crisp$ ο κανόνας παράγει *μη ασαφείς* ισχυρισμούς.
- $OntologyEntity \in \{Class, ObjectProperty, DataProperty\}$. Για κάθε μία από τις τρεις αυτές περιπτώσεις έχουμε:
  - *Class* παριστάνει μία κλάση η οποία ορίζεται από το ζεύγος $< Concept_i, key >$, όπου $i \in m_{AC}$ και $key \in \pi_{a_{ij}}(Rel_i)$.
  - *ObjectProperty* παριστάνει μια ιδιότητα αντικειμένου η οποία ορίζεται από την τριάδα $< Role_i, domain, range >$ όπου $i \in m_{AR}$, $domain \in \pi_{a_{ij}}(Rel_i)$ και $range \in \pi_{a_{ik}}(Rel_i)$.

- *DataProperty* παριστάνει μια ιδιότητα τύπου δεδομένων η οποία ορίζεται από το ζεύγος $< Role_i, subject >$ όπου $i \in m_{AR}$ και $subject \in \pi_{a_{ij}}(Rel_i)$.

- $DatabaseEntity \in \{\pi_{a_{il}}(Rel_i), Rel_i\}$.Το $\pi_{a_{il}}(Rel_i)$ αποτελεί μία παράσταση σχεσιακής άλγεβρας η οποία καταδεικνύει το attribute a_{il} της σχέσης Rel_i της DB. Το Rel_i υποδεικνύει την αντίστοιχη σχέση της DB.
- Το $A \in \{MF, 1\}$, όπου MF, παριστάνει μία συνάρτηση συμμετοχής η οποία με τη σειρά της ορίζεται από ένα ζεύγος $\left\langle f(x), \vec{p} \right\rangle$ στην οποία το $f(x)$ αποτελεί τη συνάρτηση $f : (-\infty, +\infty) \to [0,1]$, ενώ $\vec{p}$ είναι το διάνυσμα των παραμέτρων με βάση τις οποίες καθορίζεται επακριβώς το σχήμα της συνάρτησης.

Το πρώτο δομικό στοιχείο της Αντιστοίχισης Αποδοσης Νοήματος σχετίζεται με το είδος των ισχυρισμών που αυτό θα παράγει. Οι Αντιστοιχίσεις μπορούν να μας οδηγήσουν είτε σε ασαφείς είτε σε μη ασαφείς ισχυρισμούς ανάλογα με την τιμή που αυτό το πεδίο θα πάρει. Αναφερόμενοι στο παράδειγμα που περιγράφηκε στην εισαγωγή ένας ισχυρισμός που θα αποδίδει ένα συγκεκριμένο όνομα σε ένα μοντέλο π.χ. (*modelID43,Miranda):hasModelName* δεν έχει νόημα να εμπεριέχει ασάφεια εφόσον ένα όνομα συνδέεται με έναν άνθρωπο απόλυτα και όχι σε μεγαλύτερο ή μικρότερο βαθμό. Αντίθετα ένας ισχυρισμός της μορφής *modelID32:MediumWeight* είναι θεμιτό να περιέχει αβεβαιότητα εφόσον είναι μάλλον ασαφή παρά ευδιάκριτα τα όρια του συνόλου που περιλαμβάνουν ανθρώπους μεσαίου βάρους.
Το δεύτερο δομικό στοιχείο της Αντιστοίχισης Απόδοσης Νοήματος εκφράζει μία πρώτη συσχέτιση μεταξύ της οντολογίας και της βάσης δεδομένων. Διακρίνουμε τρεις διαφορετικές περιπτώσεις ανάλογα με το αν αναφερόμαστε σε κλάση, ιδιότητα αντικειμένων ή ιδιότητα τύπων δεδομένων.

- Για την περίπτωση της κλάσης συνδέουμε μία ατομική έννοια της Οντολογίας με ένα ιδιοχαρακτηριστικό ενός πίνακα της DB. Η σύνδεση αυτή επιτρέπει να γίνει το πρώτο βήμα για την παραγωγή των ισχυρισμών που θα ακολουθήσει καθώς συνδέει ένα συγκεκριμένο σύνολο από άτομα τα οποία θα προέλθουν από τη βάση δεδομένων με μία ατομική έννοια προερχόμενη από την οντολογία. Π.χ. μπορεί να διατυπώνει ισχυρισμούς για ένα σύνολο από μοντέλα και το βάρος τους, δηλαδή *modelID432:Thin*.Το αν αυτός ο ισχυρισμός θα ενισχυθεί με ασάφεια εξαρτάται από άλλο σημείο της Αντιστοίχισης.
- Για την περίπτωση της ιδιότητας αντικειμένων συνδέουμε έναν ατομικό ρόλο της Οντολογίας με ένα ζεύγος ιδιοχαρακτηριστικών ενός πίνακα της DB. Η σύνδεση αυτή επιτρέπει να γίνει το πρώτο βήμα για την παραγωγή των ισχυρισμών που θα ακολουθήσει καθώς συσχετίζει ένα συγκεκριμένο σύνολο ζευγών από άτομα τα οποία θα προέλθουν από τη βάση δεδομένων με έναν ατομικό ρόλο προερχόμενο από

την οντολογία. Π.χ. μπορεί να θέλουμε μέσω ενός ισχυρισμού να εφκράσουμε τη συνεργασία μεταξύ ενός μοντέλου και ενός πρακτορείου, δηλαδή (*modelID789,agencyID34):BadModelAgencyCooperation.*.Το αν αυτός ο ισχυρισμός θα ενισχυθεί με ασάφεια εξαρτάται από άλλο σημείο της Αντιστοίχισης.

- Τέλος για την περίπτωση της ιδιότητας τύπου δεδομένων αυτή συνδέει έναν ατομικό ρόλο της οντολογίας με ένα σύνολο από άτομα και κάποια τιμή ξεχωριστή για το καθένα από αυτά που αντιστοιχεί σε κάποιο ιδιοχαρακτηριστικό της βάσης δεδομένων. Με τον τρόπο αυτό θα παραχθούν ισχυρισμοί που συνδέουν άτομα με κάποιες ιδιότητες, οι οποίες όμως εκφράζονται με τη βοήθεια ενός τύπου δεδομένων. Το παράδειγμα της ηλικίας και του ισχυρισμού π.χ. (*modelID83,42):hasAge* εμπίπτει σε αυτή την περίπτωση. Όπως θα φανεί και στη συνέχεια δεν έχει νόημα να εισάγουμε ασάφεια σε έναν τέτοιο ισχυρισμό καθώς εκφράζει κάτι απόλυτο που δεν είναι αλήθεια κατά ένα βαθμό αλλά καθολικά.

Το τρίτο δομικό στοιχείο της Αντιστοίχισης λαμβάνει πληροφορία από τη DB και συγκεκριμένα αναφέρεται είτε σε ένα ιδιοχαρακτηριστικό κάποιας σχέσης της είτε σε κάποια σχέση. Στην περίπτωση του ιδιοχαρακτηριστικού πρόκειται είτε για την αριθμητική τιμή ενός πεδίου με τη βοήθεια του οποίου θα εισαχθεί η ασάφεια στους ασαφείς ισχυρισμούς που θα παραχθούν, είτε για την τιμή ενός πεδίου με συγκεκριμένο τύπο δεδομένων που θα χρησιμέψει κατά την παραγωγή ενός μη ασαφούς ισχυρισμού. Δύο αντίστοιχα παραδείγματα είναι οι ισχυρισμοί *modelID789:Thin≤0.4* και (*modelID3,Giorgio):hasModelName.* Το *0.4* έχει προκύψει ως αποτέλεσμα υπολογισμού βασιζόμενου στην τιμή weight για τον πίνακα Model ενώ το *Giorgio* από την τιμή του πεδίου Name του ίδιου πίνακα που έχει πεδίο τιμών String. Όσον αφορά στην περίπτωση της σχέσης βρισκόμαστε στην κατάσταση εκείνη στην οποία η ατομική έννοια είναι ταυτόχρονα το όνομα της σχέσης, δηλαδή *modelID90:Model.*

Τέλος το τέταρτο δομικό στοιχείο της ακολουθίας σχετίζεται με την εισαγωγή της ασάφειας κατά την παραγωγή των ισχυρισμών. Η τιμή 1 αντιστοιχεί στην περίπτωση κατά την οποία παράγουμε μη ασαφείς ισχυρισμούς. Εναλλακτικά μπορεί να θεωρήσει κάποιος την παραγωγή ασαφών ισχυρισμών με τιμή συνάρτησης συμμετοχής 1, οπότε καταλήγουμε στην περίπτωση όπου ακραίες τιμές ασαφών συνόλων ταυτίζονται με τα κλασικά σύνολα. Η τιμή MF αντιστοιχεί στην περίπτωση κατά την οποία παράγουμε ασαφείς ισχυρισμούς και η συνάρτηση αυτή χρησιμεύε ως εργαλείο υπολογισμού των βαθμών της εκάστοτε συνάρτησης συμμετοχής. Αναφερόμενοι στο παράδειγμα ισχυρισμού *modelID789:Thin≤0.4* που χρησιμοποιήσαμε πριν η τιμή 0.4 έχει προκύψει ως αποτέλεσμα μιας συνάρτησης συμμετοχής στην οποία έχει δοθεί ως είσοδος μια τιμή που αντιστοιχεί σε κιλά, π.χ. 80. Ο ορισμός τώρα αυτής της συνάρτησης συμμετοχής απαιτεί δύο πράγματα: τον αναλυτικό τύπο

της συνάρησης, δηλαδή το *f(x)* και μία σειρά από παραμέτρους οι οποίες εμφανίζονται στον αναλυτικό τύπο δηλαδή το $\vec{p}$.

3.1.3 Περιορισμοί Αντιστοίχισης Απόδοσης Νοήματος

Κατά το σχηματισμό των Αντιστοιχίσεων Απόδοσης Νοήματος είναι απαραίτητο να τηρούνται μια σειρά από περιορισμοί οι οποίοι εγγυώνται τη λογική συνέπεια των παραγόμενων ισχυρισμών και περιγράφονται παρακάτω.

Εάν $State = Crisp$ θα πρέπει:

- *OntologyEntity=DataProperty* ∨ *OntologyEntity=Class* ∨ *OntologyEntity=ObjectProperty*
- A=1

Η παραγωγή των μη ασαφών ισχυρισμών μπορεί να γίνει για οποιαδήποτε από τις 3 περιπτώσεις της *OntologyEntity*. Ωστόσο επιβάλλει το Α να λαμβάνει την τιμή 1 εφόσον οι μη ασαφείς ισχυρισμοί δεν απαιτούν την ύπαρξη συνάρτησης συμμετοχής για να δημιουργηθούν και δεν έχει νόημα να περιέχουν περιττές πληροφορίες οι Αντιστοιχίσεις μας.

Εάν *State=Fuzzy* θα πρέπει:

- A ∈ MF.
- *OntologyEntity=Class* ∨ *OntologyEntity=ObjectProperty*.

Αντίθετα, η παραγωγή ασαφών ισχυρισμών μπορεί να γίνει αν και μόνο αν διαθέτουμε μία κατάλληλη συνάρτηση συμμετοχής. Όπως έχουμε ήδη εξηγήσει μέσω των παραδειγμάτων (*modelID3,Giorgio):hasModelName* ή (*modelID83,42):hasAge* οι ισχυρισμοί που εμπλέκουν ιδιότητες τύπων δεδομένων δεν έχει νόημα να είναι ασαφείς, γι'αυτό και δεν είναι δυνατό για ασαφείς ισχυρισμούς η *OntologyEntity* να λαμβάνει την τιμή *DataProperty*.

Συγκεκριμένα για τις παρακάτω περιπτώσεις τίθενται οι επιπλέον παρακάτω περιορισμοί:

Εάν $State = Fuzzy$ και $OntologyEntity = Class$ θα πρέπει:

- $DatabaseEntity \in \{\pi_{a_{il}}(Rel_i)\}$ και το a_{il} της Αντιστοίχισης Απόδοσης Νοήματος να έχει αριθμητικό πεδίο τιμών. Το πεδίο αυτό της DB θα χρησιμέψει ως είσοδος για

τη συνάρτηση συμμετοχής η οποία με τη σειρά της έχει πεδίο ορισμού το σύνολο των πραγματικών αριθμών.

- Το *key* της Αντιστοίχισης Απόδοσης Νοήματος να είναι primary key για το Rel_i. Από αυτό το *key* θα προέλθει η ονομασία των ατόμων κατά την παραγωγή των ισχυρισμών. Πρέπει λοιπόν να εξασφαλίσουμε ότι δε θα παραχθούν άτομα με την ίδια ονομασία, καθώς προκειμένου να δοθεί μια επιτυχημένη ερμηνεία στην οντολογική γνώση που θα παραχθεί θα πρέπει να υπάρχει 1-1 αντιστοιχία με άτομα του πραγματικού κόσμου. Αυτή η 1-1 αντιστοιχία μπορεί να εγγυηθεί μέσα απο τη DB και τα πρωτεύοντα κλειδιά τα οποία διαβεβαιώνουν με τον τρόπο αυτό ότι δε θα παραχθεί άτομο με το ίδιο όνομα.
- Ισχύει $l \neq j$. Με αυτό τον περιορισμό εξασφαλίζουμε ότι το ιδιοχαρακτηριστικό το οποίο αντιστοιχεί στο *key* (δηλαδή αυτό που θα χρησιμοποιηθεί για την παραγωγή των ατόμων) θα είναι διαφορετικό από το ιδιοχαρακτηριστικό του *DatabaseEntity* που θα δοθεί ως είσοδος στη συνάρτηση συμμετοχής.

Εάν $State = Fuzzy$ και $OntologyEntity = ObjectProperty$ θα πρέπει:

- $DatabaseEntity \in \{\pi_{a_{il}}(Rel_i)\}$ και το a_{il} της Αντιστοίχισης Απόδοσης Νοήματος να έχει αριθμητικό πεδίο τιμών. Όπως και παραπάνω το πεδίο αυτό της DB θα χρησιμέψει ως είσοδος για τη συνάρτηση συμμετοχής η οποία με τη σειρά της έχει πεδίο ορισμού το σύνολο των πραγματικών αριθμών.
- Το *domain* και *range* της Αντιστοίχισης Απόδοσης Νοήματος να αποτελούν primary key για το Rel_i. Ανάλογα με την παραπάνω περίπτωση από το ζεύγος *domain* και *range* θα προέλθει η ονομασία του ζεύγους των ατόμων κατά την παραγωγή των ισχυρισμών. Όπως και πριν υπάρχει απαίτηση για 1-1 αντιστοιχία η οποία όμως μόνο μέσω του πρωτεύοντος κλειδιού της DB μπορεί να επιτευχθεί.
- Τα $domain$ και $range$ είναι ξένα κλειδιά δύο άλλων σχέσεων $\mathrm{Re}l_d$ και $\mathrm{Re}l_r$, όπου $d \neq r \neq i$. Εάν λάβουμε ως παράδειγμα τον ασαφή ισχυρισμό *(modelID89,agencyID39): goodCooperationDegree* $\geq$ *0.9* αντιλαμβανόμαστε ότι τα *modelID89* και *agencyID39* θα πρέπει να προέρχονται από ιδιοχαρακτηριστικά τα οποία είναι ξένα κλειδιά σε άλλες, διαφορετικές σχέσεις. Στη συγκεκριμένη περίπτωση πρόκειται για τις σχέσεις Model και Agency.

- Ισχύει $l \neq j \neq k$. Ο περιορισμός αυτός απαιτεί τα ιδιοχαρακτηριστικά από τα οποία προέρχεται το ζεύγος των ατόμων καθώς επίσης και το ιδιοχαρακτηριστικό το οποίο

χρησιμεύει ως είσοδος στη συνάρτηση συμμετοχής να είναι όλα διαφορετικά μεταξύ τους.

Εάν $State = Crisp$ και $OntologyEntity = Class$ θα πρέπει:

- $DatabaseEntity \in \{Rel_i\}$
- $Concept_i = Rel_i$

Αυτή είναι η περίπτωση κατά την οποία παράγονται μη ασαφείς ισχυρισμοί και η ονομασία των ατόμων προέρχεται από το *key*. Το όνομα της ατομικής έννοιας συμπίπτει με το όνομα του πίνακα, κάτι που ικανοποιείται με τη βοήθεια των παραπάνω περιορισμών. Ένα παράδειγμα τέτοιου ισχυρισμου είναι ο *customerID90:Customer*.

Εάν $State = Crisp$ και $OntologyEntity = ObjectProperty$ θα πρέπει:

- $DatabaseEntity \in \{Rel_i\}$
- $Role_i = Rel_i$

Αυτή είναι η περίπτωση κατά την οποία παράγονται μη ασαφείς ισχυρισμοί και η ονομασία των ζευγών των ατόμων προέρχεται από το ζεύγος *domain-range* αντίστοιχα. Το όνομα του ατομικού ρόλου συμπίπτει με το όνομα του πίνακα, κάτι που ικανοποιείται με τη βοήθεια των παραπάνω περιορισμών. Ένα παράδειγμα τέτοιου ισχυρισμου είναι ο *(customerID90,agencyID89):CustomerAgency,* που υποδηλώνει ότι τα δύο αυτά άτομα έχουν συνεργαστεί στο παρελθόν.

Τέλος για να είναι συνεπές το σύνολο των Αντιστοιχίσεων Απόδοσης Νοήματος είναι απαραίτητο να υπάρχει 1-1 αντιστοιχία μεταξύ μιας Αντιστοίχισης Απόδοσης Νοήματος και ενός στοιχείου του συνόλου $\{Concept_1,...,Concept_{m_{AC}}, Role_1,...,Role_{m_{AR}}\}$. Αυτή η 1-1 αντιστοιχία είναι απαραίτητη προκειμένου να εξασφαλιστεί η συνέπεια του παραγόμενου σώματος ισχυρισμών. Εάν για παράδειγμα έχουμε δύο Αντιστοιχίσεις Απόδοσεις Νοήματος για το ίδιο $Concept_i$ (π.χ. το Thin) μπορεί να παραχθούν δύο ασαφείς ισχυρισμοί της μορφής *modelID:Thin*$\geq$*0.6* και *modelID:Thin*$\leq$*0.4* που θα εισάγουν ασυνέπεια στην παραγόμενη γνώση.

3.1.4 Παραγωγή σώματος ισχυρισμών

Επιδίωξη της κατασκευής ενός συνόλου Αντιστοιχίσεων Απόδοσης Νοήματος είναι η παραγωγή ασαφών και μη ασαφών ισχυρισμών. Ωστόσο δεδομένης μιας DB, μιας

Οντολογίας και του συνόλου Αντιστοιχίσεων Απόδοσης Νοήματος δεν εξυπακούεται αυτόματα η δημιουργία των ισχυρισμών αλλά θα πρέπει να ακολουθηθεί ένας συγκεκριμένος αλγόριθμος για την κατασκευή τους. Ο αλγόριθμος αυτός αποτελεί το αντικείμενο της παρούσας παραγράφου.

Πριν περιγράψουμε τις τεχνικές παραγωγής ισχυρισμών που προτείνουμε θα κάνουμε ορισμένες παραδοχές οι οποίες χωρίς βλάβη της γενικότητας περιγράφουν με πιο τυπικό και εύληπτο τρόπο τη διαδικασία παραγωγής των ισχυρισμών:

- Έστω concat(s1,s2) μία συνάρτηση δύο ορισμάτων ($concat : S \times S \rightarrow S$, όπου S το σύνολο όλων των δυνατών strings της αγγλικής γλώσσας) η οποία επιστρέφει την παράθεση των strings s1 και s2.
- Έστω p_i ο αριθμός των tuples της σχέσης Rel_i για μία Αντιστοίχιση Απόδοσης Νοήματος.
- Επίσης να προσθέσουμε ότι κάθε φορά που περικλείουμε ένα τμήμα μιας σχέσης Rel_i σε αγκύλες ακολουθούμενο από ένα δείκτη u (π.χ. $[\pi_{a_{ij}}(Rel_i)]_u$) αναφερόμαστε στη u-οστή γραμμή του τμήματος αυτής της σχέσης. Ο συμβολισμός αυτός υιοθετείται επειδή δεν υπάρχει κάτι αντίστοιχο από το χώρο της σχεσιακής άλγεβρας για να χρησιμοποιηθεί.

ModelID	Name	Height	Weight	Reward
1	Heidi	1.85	67	4000
2	Elle	1.90	78	1500
3	Gisele	1.79	63	20000

Πίνακας 3. Παράδειγμα τμήματος περιεχομένων DB.

ModelID	AgencyID	CooperationDegreeAM	DistanceAM
1	2	0.7	54
2	4	0.1	79
3	6	0.6	22

Πίνακας 4. Παράδειγμα τμήματος περιεχομένων DB.

3.1.4.1 Παραγωγή ABox για Αντιστοίχιση Απόδοσης Νοήματος με State=Crisp και OntologyEntity =DataProperty

Για κάθε Αντιστοίχιση Απόδοσης Νοήματος θα παραχθεί ένα σύνολο από assertions που θα υπακούουν στη σύνταξη f-$\mathcal{SHOIN}$. Συγκεκριμένα για *RoleName=Role*$_i$ θα έχουμε:

$$\langle (individualName, dataValue) : RoleName \rangle_u$$

Στον παραπάνω ισχυρισμό ισχύει $individualName = concat(Rel_i, [subject]_u)$ και $dataValue = [\pi_{a_{ij}}(Rel_i))]_u$.

Η παρούσα περίπτωση παράγει ένα σύνολο από μη ασαφείς ισχυρισμούς που εμπλέκουν ιδιότητες τύπου δεδομένων. Κάθε ισχυρισμός αυτής της μορφής συντίθεται από 3 κάθε φορά στοιχεία: το όνομα του ατόμου, την ιδιότητα τύπου δεδομένων και την τιμή του τύπου δεδομένων. Το όνομα του ατόμου δημιουργείται από την παράθεση του ονόματος του ιδιοχαρακτηριστικού *subject* και την τιμή αυτού του ιδιοχαρακτηριστικού για τη *u*-οστή γραμμή. Το όνομα της ιδιότητας τύπου δεδομένων είναι το ίδιο για όλους τους ισχυρισμούς που παράγονται από αυτή την Αντιστοίχιση και είναι το *Role*$_i$. Τέλος η τιμή του τύπου δεδομένων προκύπτει έπειτα από αναζήτηση στη DB και πρόκειται για την τιμή του πεδίου *DatabaseEntity* της u-οστής γραμμής. Για κάθε Αντιστοίχιση Απόδοσης Νοήματος αυτής της μορφής παράγεται αριθμός μη ασαφών ισχυρισμών ίσος με τον αριθμό των πλειάδων της σχέσης *Rel*$_i$. Π.χ. για την Αντιστοίχιση Απόδοσης Νοήματος $\langle Crisp, \langle hasModelName, Model.ModelID \rangle, Model.Name, 1 \rangle$ και δεδομένου ότι διαθέτουμε τη βάση δεδομένων με περιεχόμενα του πίνακα 3 οι μη ασαφείς ισχυρισμοί που θα παραχθούν θα είναι οι εξής 3: *(ModelID1,Heidi):hasModelName, (ModelID2,Elle):hasModelName, (ModelID3,Gisele):hasModelName.*

3.1.4.2 Παραγωγή ABox για Αντιστοίχιση Απόδοσης Νοήματος με State=Crisp και OntologyEntity =Class

Για κάθε Αντιστοίχιση Απόδοσης Νοήματος θα παραχθεί ένα σύνολο από assertions που θα υπακούουν στη σύνταξη f-$\mathcal{SHOIN}$. Συγκεκριμένα για *ConceptName=Concept*$_i$*=Rel*$_i$ θα έχουμε:

$$\langle individualName : ConceptName \rangle_u$$

Στον παραπάνω ισχυρισμό ισχύει $individualName1 = concat(Rel_i, [key]_u)$.

Η παρούσα περίπτωση παράγει ένα σύνολο από μη ασαφείς ισχυρισμούς που εμπλέκουν κλάσεις. Συντίθενται από δύο στοιχεία, το όνομα του ατόμου και το όνομα της κλάσης. Το όνομα του ατόμου προέρχεται και πάλι από την παράθεση του ονόματος του ιδιοχαρακτηριστικού *key* και της τιμής αυτού του ιδιοχαρακτηριστικού για τη *u*-οστή γραμμή. Το όνομα της κλάσης είναι το ίδιο για όλους τους ισχυρισμούς που παράγονται από αυτή την Αντιστοίχιση και είναι το Concept$_i$. Για κάθε Αντιστοίχιση Απόδοσης Νοήματος αυτής της μορφής παράγεται αριθμός μη ασαφών ισχυρισμών ίσος με τον αριθμό των πλειάδων της σχέσης *Rel*$_i$. Π.χ. για την Αντιστοίχιση Απόδοσης Νοήματος $\langle Crisp, \langle Model, Model.ModelID \rangle, Model, 1 \rangle$ και δεδομένου ότι διαθέτουμε τη βάση δεδομένων με περιεχόμενα του πίνακα 3 οι μη ασαφείς ισχυρισμοί που θα παραχθούν θα είναι οι εξής 3: *ModelID1:Model, ModelID2:Model* και *ModelID3:Model.*

3.1.4.3 Παραγωγή ABox για Αντιστοίχιση Απόδοσης Νοήματος με State=Crisp και OntologyEntity =ObjectProperty

Για κάθε Αντιστοίχιση Απόδοσης Νοήματος θα παραχθεί ένα σύνολο από assertions που θα υπακούουν στη σύνταξη f-$\mathcal{SHOIN}$. Συγκεκριμένα για *RoleName=Role*$_i$*=Rel*$_i$ θα έχουμε:

$$\langle (individualName1, individualName2) : RoleName \rangle_u$$

Στον παραπάνω ισχυρισμό ισχύει $individualName1 = concat(Rel_i, [domain]_u)$ και $individualName2 = concat(Rel_i, [range]_u)$.

Η παρούσα περίπτωση είναι η τελευταία για τους μη ασαφείς ισχυρισμούς και παράγει ένα σύνολο από μη ασαφείς ισχυρισμούς που εμπλέκουν ιδιότητες αντικειμένων. Συντίθενται από τρία στοιχεία, τα ονόματα των 2 ατόμων και το όνομα της ιδιότητας αντικειμένων. Το όνομα του ατόμου για το domain προέρχεται και πάλι από την παράθεση του ονόματος του ιδιοχαρακτηριστικού *domain* και της τιμής αυτού του ιδιοχαρακτηριστικού για τη *u*-οστή γραμμή. Το όνομα για το *range* προκύπτει με αντίστοιχο τρόπο. Το όνομα της κλάσης είναι το ίδιο για όλους τους ισχυρισμούς που παράγονται από αυτή την Αντιστοίχιση και είναι το Role_i. Για κάθε Αντιστοίχιση Απόδοσης Νοήματος αυτής της μορφής παράγεται αριθμός μη ασαφών ισχυρισμών ίσος με τον αριθμό των πλειάδων της σχέσης Rel_i. Π.χ. για την $\langle Crisp, \langle AgencyModel, AgencyModel.ModelID, AgencyModel.AgencyID \rangle, AgencyModel, 1 \rangle$ και δεδομένου ότι διαθέτουμε τη βάση δεδομένων με περιεχόμενα του πίνακα 4 οι μη ασαφείς ισχυρισμοί που θα παραχθούν θα είναι οι εξής 3: (*ModelID1,AgencyID2):AgencyModel,* (*ModelID2,AgencyID4):AgencyModel και* (*ModelID3,AgencyID6):AgencyModel,*

3.1.4.4 Παραγωγή ABox για Αντιστοίχιση Απόδοσης Νοήματος με State=Fuzzy και OntologyEntity =Class

Για κάθε Αντιστοίχιση Απόδοσης Νοήματος θα υπολογιστεί η τιμή της MF για τις p_i γραμμές του πίνακα Rel_i. Για όσες από αυτές το αποτέλεσμα είναι διάφορο (δηλαδή μεγαλύτερο) από το 0 θα παραχθεί ένα ζεύγος ισχυρισμών που θα υπακούει στη σύνταξη f-$\mathcal{SHOIN}$. Συγκεκριμένα για $ConceptName=Concept_i$ θα έχουμε:

$$\langle individualName : ConceptName \geq degree \rangle_u$$
$$\langle individualName : ConceptName \leq degree \rangle_u, 1 \leq u \leq p_i$$

Στον παραπάνω ισχυρισμό ισχύει $individualName = concat(Rel_i, [key]_u)$ και $degree = f([\pi_{a_{ij}}(Rel_i)])_u)$. Πρέπει απαραίτητα να ισχύει $degree > 0$ αλλιώς δεν παράγονται ζεύγη ισχυρισμών.

Η παρούσα περίπτωση αντιστοιχει σε παραγωγή ασαφών ισχυρισμών που εμπλέκουν κλάσεις. Τρία στοιχεία συνεισφέρουν στην κατασκευή κάθε ισχυρισμού: το όνομα του

ατόμου, το όνομα της ατομική έννοιας και ο βαθμός της ασάφειας. Η συμβολοσειρά του ονόματος του ατόμου παράγεται όπως και στους μη ασαφείς ισχυρισμούς. Το όνομα της ατομικής έννοιας δεν είναι άλλο από το $Concept_i$. Ο βαθμός της ασάφειας προκύπτει εάν υπολογίσουμε την τιμή της συνάρτησης *f(x)* με είσοδο την τιμή του ιδιοχαρακτηριστικού της *DatabaseEntity* για τη u-οστή γραμμή. Ο αριθμός των ασαφών ισχυρισμών που παράγονται για κάθε Αντιστοίχιση αυτού του τύπου δεν είναι σταθερός αλλά είναι διπλάσιος του αριθμού πλειάδων για τις οποίες η συνάρτηση συμμετοχής λαμβάνει μη μηδενική τιμή. Έτσι π.χ. για

$\langle Fuzzy, \langle MediumReward, Model.ModelID \rangle, Model.Reward, \langle triangular, (500, 3000, 5000) \rangle \rangle$

, όπου *triangular* θεωρούμε ότι αντιστοιχεί στον αναλυτικό τύπο της τριγωνικής συνάρτησης που διαθέτει 3 παραμέτρους, θα παραχθούν οι εξής 4 ισχυρισμοί: *modelID1:MediumReward* $\geq$ *0.4 , modelID1: MediumReward* $\leq$ *0.4 , modelID2:MediumReward* $\geq$ *0.5 , modelID21: MediumReward* $\leq$ *0.5.*

3.1.4.5 Παραγωγή ABox για Αντιστοίχιση Απόδοσης Νοήματος με State=Fuzzy και OntologyEntity =ObjectProperty

Για κάθε Αντιστοίχιση Απόδοσης Νοήματος θα υπολογιστεί η τιμή της MF για τις p_i γραμμές του πίνακα Rel_i. Για όσες από αυτές το αποτέλεσμα είναι διάφορο (δηλαδή μεγαλύτερο) από το 0 θα παραχθεί ένα ζεύγος ισχυρισμών που θα υπακούει στη σύνταξη f-$\mathcal{SHOIN}$. Συγκεκριμένα για *RoleName=*$Role_i$ θα έχουμε:

$$\langle (individualName1, individualName2) : RoleName \geq degree \rangle_u$$
$$\langle (individualName1, individualName2) : RoleName \leq degree \rangle_u , 1 \leq u \leq p_i$$

Στον παραπάνω ισχυρισμό ισχύουν:

$$individualName1 = concat(Rel_i, [domain]_u)$$
$$individualName2 = concat(Rel_i, [range]_u)$$
$$degree = f([\pi_{a_{ij}}(Rel_i)]_u) \ .$$

Πρέπει απαραίτητα να ισχύει $degree > 0$ αλλιώς δεν παράγονται ισχυρισμοί.

Η παρούσα περίπτωση αντιστοιχει σε παραγωγή ασαφών ισχυρισμών που εμπλέκουν ιδιότητες αντικειμένων. Τρία στοιχεία συμμετέχουν στην κατασκευή κάθε ισχυρισμού: τα ονόματα των δύο ατόμων, το όνομα του ατομικού ρόλου και ο βαθμός της ασάφειας. Η συμβολοσειρά των ονομάτων των ατόμων παράγεται όπως και στους μη ασαφείς ισχυρισμούς. Το όνομα του ατομικού ρόλου δεν είναι άλλο από το $Role_i$. Ο βαθμός της ασάφειας προκύπτει εάν υπολογίσουμε την τιμή της συνάρτησης *f(x)* με είσοδο την τιμή του ιδιοχαρακτηριστικού της *DatabaseEntity* για τη u-οστή γραμμή. Ο αριθμός των ασαφών ισχυρισμών που παράγονται για κάθε Αντιστοίχιση αυτού του τύπου δεν είναι σταθερός αλλά είναι διπλάσιος του αριθμού πλειάδων για τις οποίες η συνάρτηση συμμετοχής λαμβάνει μη μηδενική τιμή. Έτσι π.χ. για

$$\langle Fuzzy, \langle excellentModelAgencyCooperation, Model.ModelID, Agency.AgencyID\rangle, AgencyModel.CooperationDegreeAM, \langle trapezoidal, (0.5, 0.8, 1, 1)\rangle\rangle ,$$

όπου *trapezoidal* θεωρούμε ότι αντιστοιχεί στον αναλυτικό τύπο της τραπεζοειδούς συνάρτησης που διαθέτει 4 παραμέτρους, θα παραχθούν οι εξής 4 ισχυρισμοί: (*modelID1,agencyID2):ExcellentModelAgencyCooperation*$\geq$*0.66,*

(*modelID1,agencyID2):ExcellentModelAgencyCooperation*$\leq$*0.66,*

(*modelID3,agencyID6):ExcellentModelAgencyCooperation*$\geq$*0.33*

και (*modelID3,agencyID6):ExcellentModelAgencyCooperation* $\leq$ *0.33.*

3.1.5 Χρήση OWL Annotations για την αναπαράσταση της ασάφειας των ισχυρισμών

Επειδή η fuzzy OWL δεν αποτελεί ακόμη επίσημο πρότυπο της W3C δεν υπάρχει κάποιος επίσημος συντακτικός φορμαλισμός μέσω του οποίου να εκφράζονται οι συντελεστές ασάφειας για τα διάφορα αξιώματα και τα σύμβολα τους. Για το λόγο αυτό στις διάφορες υλοποίησεις που χρησιμοποιούν fuzzy OWL έχουν χρησιμοποιηθεί κάποιες δομές της OWL, που ονομάζονται OWL Annotations. Τα OWL Annotations αναφέρονται σε OWL αξιώματα. Θα επικεντρωθούμε στις δύο περιπτώσεις αξιωμάτων που μας ενδιαφέρουν δηλαδή τους ισχυρισμούς για object properties και classes. Σύμφωνα με την επίσημη γραμματική της OWL ισχύουν οι ακόλουθοι κανόνες:

ObjectPropertyAssertion := 'PropertyAssertion' '(' { Annotation } ObjectPropertyExpression sourceIndividual targetIndividual ')'

ClassAssertion := 'ClassAssertion' '(' { Annotation } ClassExpression Individual ')'

Προχωρώντας οδηγούμαστε στους κανόνες που έχουν στο αριστερό μέρος τα Annotations. Δηλαδή:

Annotation := AnnotationByConstant

Στη συνέχεια για το παραπάνω έχουμε:

AnnotationByConstant := explicitAnnotationByConstant

Τέλος καταλήγουμε στον κανόνα:

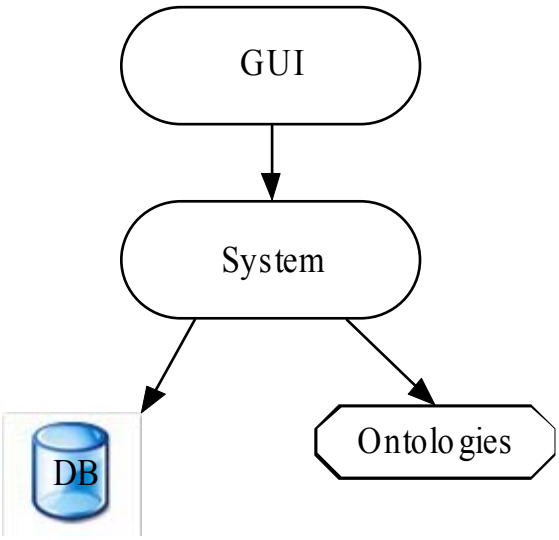
explicitAnnotationByConstant := 'Annotation' '(' AnnotationProperty Constant ')'

Στην περίπτωση μας το AnnotationProperty θα αντιστοιχεί στην ιδιότητα “hasFuzziness” ενώ το Constant θα είναι ένα String το οποίο θα περιλαμβάνει τόσο το σύμβολο ($\leq$ ή $\geq$) όσο και το συντελεστή ασάφειας. Στη λεπτομερή περιγραφή του συστήματος θα δείξουμε πώς τα παραπάνω εφαρμόζονται με τη βοήθεια του OWL API. Το OWL API είναι μια βιβλιοθήκη σε java που έχει κατασκευαστεί για τη διαχείριση οντολογιών από τους προγραμματιστές σε αρκετά υψηλό και αφαιρετικό επίπεδο, χωρίς να τους υποχρεώνει να ασχοληθούν με χαμηλού επιπέδου λεπτομέρειες, όπως π.χ. xml parsing κλπ.

4

Αρχιτεκτονική του συστήματος

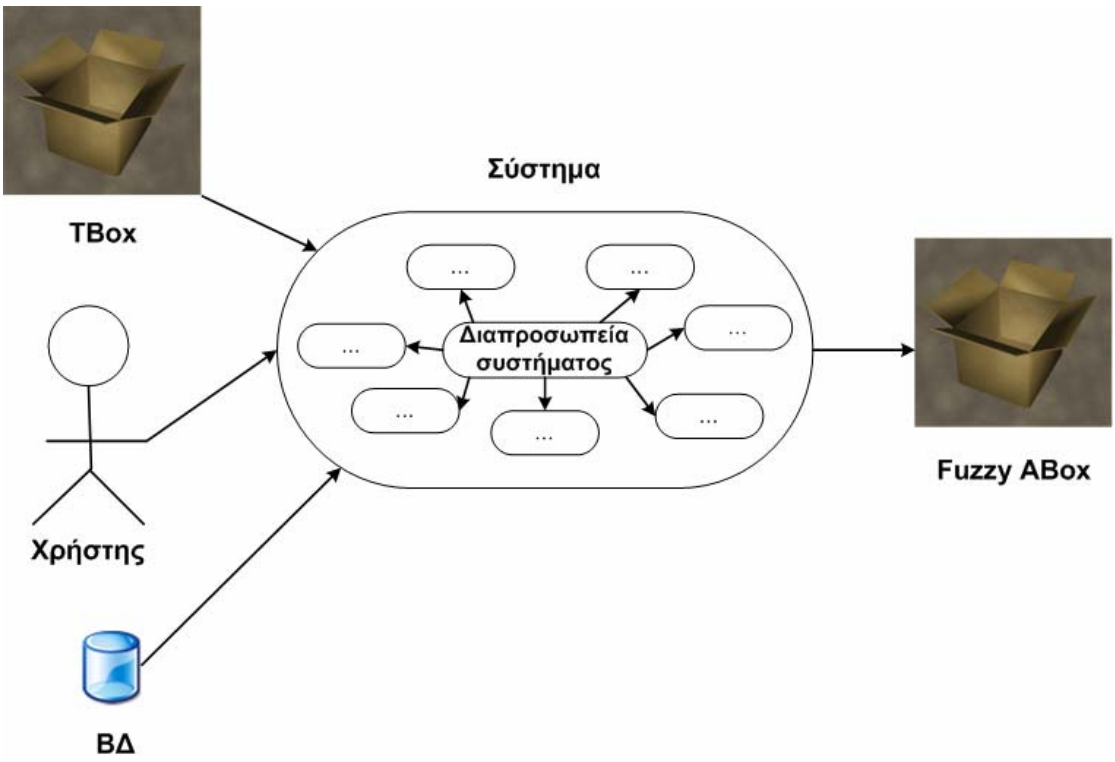
4.1 Γενική περιγραφή του συστήματος



Σχήμα 15. Αλληλεπιδράσεις του συστήματος.

Το σχήμα 15 παριστάνει τη διαστρωμάτωση που σχηματίζεται κατά τη λειτουργία του συστήματος. Το σύστημα επικοινωνεί με μία ΒΔ καθώς και με οντολογίες. Λαμβάνει δεδομένα από αυτές τις πηγές πληροφοριών ενώ παράληλα τροποποιεί κάποιες από αυτές. Το

ποια λειτουργία θα εκτελεστεί κάθε φορά και ποιες παράμετροι θα καθορίσουν την αμφίδρομη αυτή επικοινωνία εξαρτάται από την αλληλεπίδραση με το χρήστη, η οποία γίνεται μέσω του Graphical User Interface που το σύστημα προσφέρει.



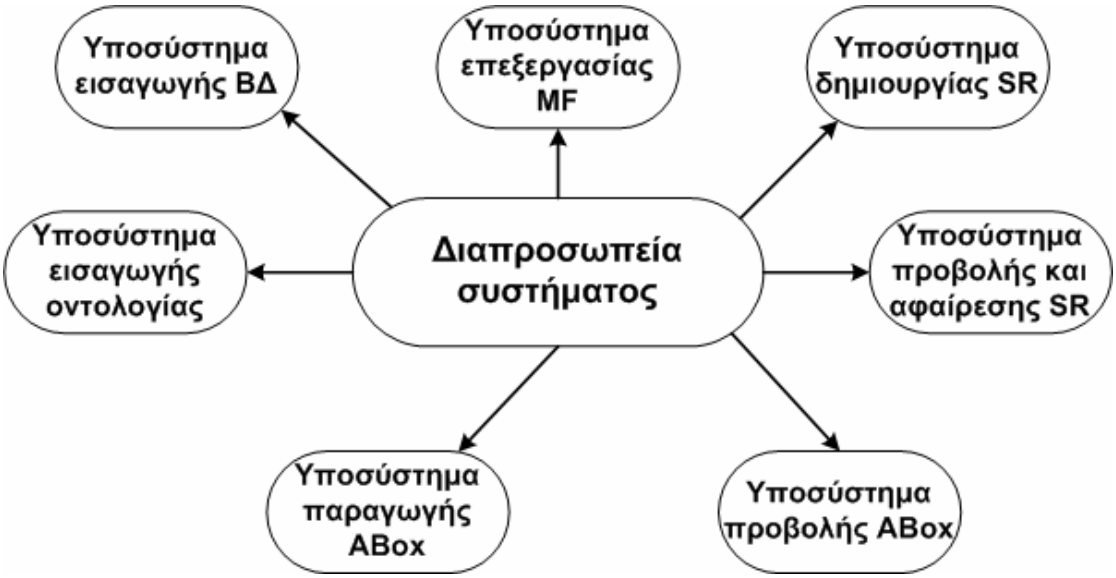
Σχήμα 16. Είσοδοι και έξοδος του συστήματος.

Το σχήμα 16 δείχνει το σύστημα με τις εισόδους που δέχεται και την έξοδο που δίνει. Τα περιεχόμενα της ΒΔ, οι ενέργειες του χρήστη καθώς επίσης και το (crisp) ABox δρουν ως είσοδος ενώ η έξοδος που το σύστημα παράγει είναι το fuzzy ABox.

Το σχήμα 17 δείχνει τα κύρια υποσυστήματα από τα οποία απαρτίζεται το σύστημα. Μία τυπική αλληλεπίδραση συστήματος-χρήστη περιλαμβάνει τα εξής βήματα:

- Εισαγωγής της ΒΔ
- Εισαγωγή της οντολογίας
- Δημιουργία ενός συνόλου MF
- Δημιουργία ενός συνόλου SR
- Προβολή και αφαίρεση των SR
- Παραγωγή του ασαφούς ABox
- Προβολή του ασαφούς ABox

Στην παραπάνω ακολουθία ενεργειών με MF σημειώνουμε τις membership functions (συναρτήσεις συμμετοχής) τις οποίες ο χρήστης εισάγει για την ασαφοποίηση των περιεχομένων της ΒΔ. Με SR σημειώνουμε τους semantic rules (Αντιστοιχίσεις Απόδοσης Νοήματος) τους οποίους περιγράψαμε εκτενώς στο προηγούμενο κεφάλαιο. Οι δύο αυτές συντομογραφίες θα χρησιμοποιηθούν και στη συνέχεια. Αναφορικά με την περιγραφή που δόθηκε στο προηγούμενο κεφάλαιο οι MF του συστήματος αντιστοιχούν στις MF των Αντιστοιχίσεων Απόδοσης Νοήματος. Παρατηρούμε ότι για κάθε βήμα της παραπάνω αλληλεπίδρασης έχει αναπτυχθεί ένα υποσύστημα που το εξυπηρετεί. Στη συνέχεια περιγράφουμε αναλυτικά κάθε ένα από τα επιμέρους υποσυστήματα.



Σχήμα 17. Υποσυστήματα του συστήματος.

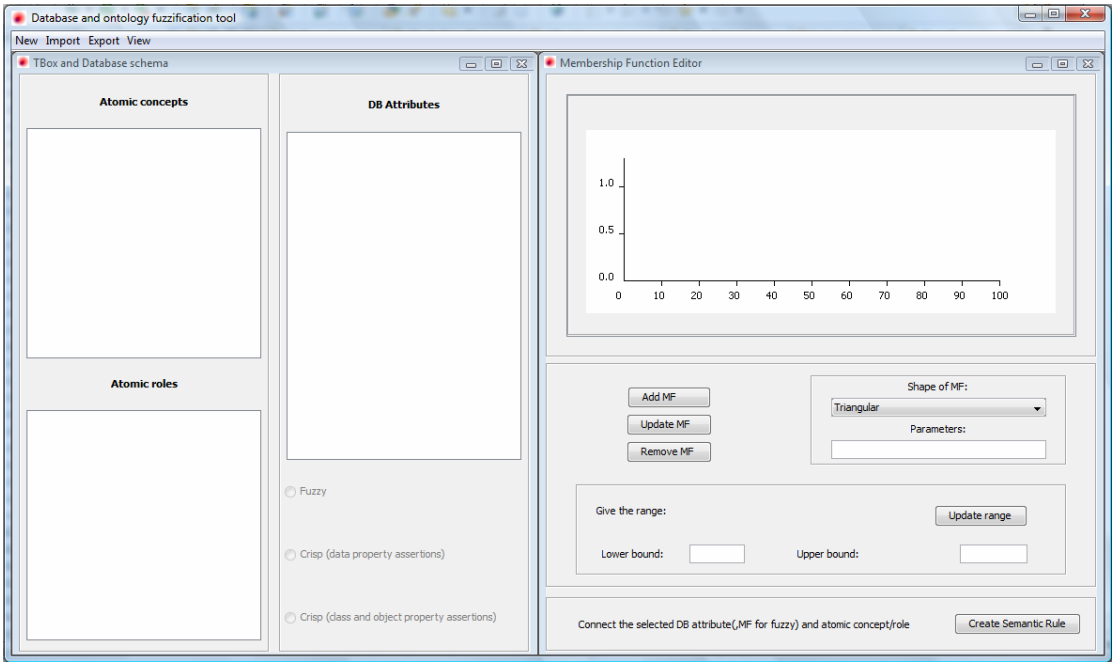
4.2 Επιμέρους υποσυστήματα

4.2.1 Διαπροσωπεία συστήματος

Πρόκειται για το συνδετικό κρίκο μεταξύ του χρήστη και του συστήματος- και κατ'επέκταση του συνόλου των υποσυστημάτων που αυτό περιλαμβάνει. Είναι μία γραφική διαπροσωπεία που επιτρέπει στο χρήστη να αλληλεπιδράσει με το σύστημα και να εκμεταλλευτεί το σύνολο των λειτουργιών του. Διαθέτει μία μπάρα με 4 μενού κάθε ένα από τα οποία σχετίζεται με λειτουργίες των επιμέρους υποσυστημάτων. Επιπλέον χωρίζεται σε δύο κύρια μέρη το αριστερό και το δεξί πλαίσιο. Το αριστερο πλαίσιο σχετίζεται με το υποσύστημα εισαγωγής ΒΔ και το υποσύστημα εισαγωγής οντολογίας ενώ το δεξιό πλαίσιο με το υποσύστημα

επεξεργασίας MF και δημιουργίας SR. Τα υπόλοιπα 3 υποσυστήματα (προβολής και αφαίρεσης SR, παραγωγής ABox και προβολής ABox) διατίθενται στο χρήστη μέσω του μενού.

Η κύρια διαπροσωπεία του συστήματος αναλαμβάνει και το ρόλο συντονιστή καθώς είναι η πρώτη που τίθεται σε λειτουργία κατά την εκκίνηση του συστήματος. Είναι επομένως υπεύθυνη για μία σειρά από αρχικοποιήσεις που είναι απαραίτητες για τη μετέπειτα εύρυθμη λειτουργία του συστήματος. Η πλειοψηφία των ακροατών που αναμένουν για γεγονότα (δηλαδή για ενέργειες του χρήστη αναφορικά με το GUI) είναι εγγεγραμένοι στην κύρια διαπροσωπεία του συστήματος. Επομένως πρόκειται για το κομμάτι εκείνο που θα προβεί στις απαραίτητες πράξεις (εμφάνιση παραθύρων, δημιουργία αντικειμένων, δημιουργία διαχειριστών) κάθε φορά που θα ανιχνεύει μία είσοδο από το χρήστη.



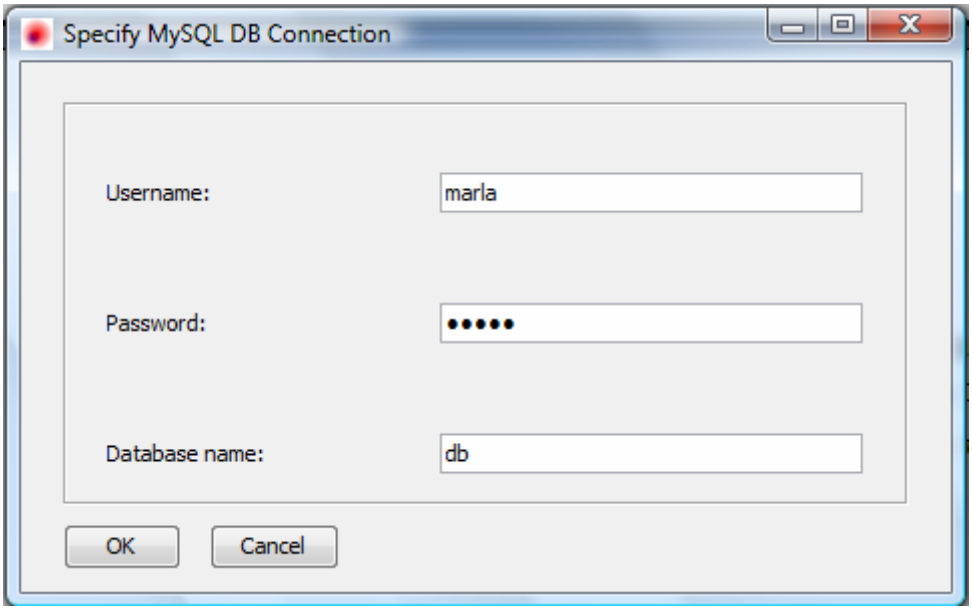
Σχήμα 18. Κύρια διαπροσωπεία του συστήματος.

Εκτός από το γραφικό interface που εμφανίζεται στο σχήμα 18 με την κύρια διαπροσωπεία σχετίζονται και ένα σύνολο παραθύρων που αναδύονται ή προειδοποιητικών-πληροφοριακών μηνυμάτων και καθοδηγούν τη συμπεριφορά του χρήστη. Η κύρια διαπροσωπεί αποτελεί το γενικό ρυθμιστή του συστήματος από τον οποίο απορρέει το σύνολο των λειτουργιών που αυτό υλοποιεί. Διευθύνει δηλαδή τις αλληλεπιδράσεις μεταξύ των επιμέρους υποσυστημάτων, φροντίζοντας ωστόσο να μην επεμβαίνει στην εσωτερική λειτουργία και συγκρότηση τους και να διατηρεί τη συνεκτικότητα στο εσωτερικό τους.

Τέλος να προσθέσουμε ότι το σύστημα έχει αναπτυχθεί με τέτοιο τρόπο ώστε να αντιδρά με κατάλληλο τρόπο σε απρόοπτες καταστάσεις (ασυνεπή είσοδο από το χρήστη κλπ). Ωστόσο επειδή είναι πιθανό το σύστημα να περιέλθει σε ανεπιθύμητη κατάσταση που δεν έχει προβλεφθεί έχει προνοηθεί σε μια τέτοια περίπτωση να εμφανίζεται προειδοποιητικό μήνυμα στο χρήστη με την αιτία του προβλήματος. Μετά την εμφάνιση ενός τέτοιου μηνύματος το σύστημα επιστρέφει στην κατάσταση που βρισκόταν πριν εμφανιστεί το πρόβλημα.

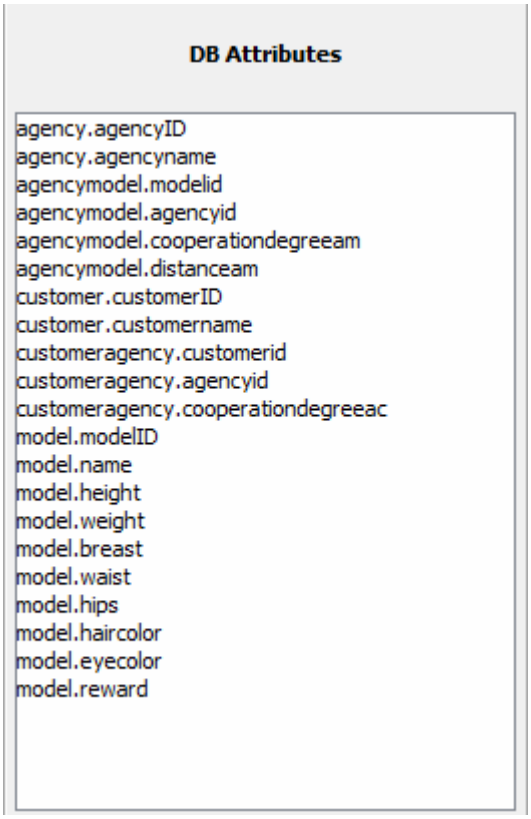
4.2.2 Υποσύστημα εισαγωγής ΒΔ

Το παρόν υποσύστημα είναι υπεύθυνο για την αρχική διασύνδεση του χρήστη με τη ΒΔ και την ανάκτηση του σχήματος της. Όπως φαίνεται και από το σχήμα 19 που αποτελεί το interface του συστήματος, ζητείται από το χρήστη το username και το password, καθώς επίσης και το όνομα της ΒΔ που θα διαβαστεί.



Σχήμα 19. Διαπροσωπεία του υποσυστήματος εισαγωγής ΒΔ.

Εάν ο χρήστης δώσει τα σωστά στοιχεία, πραγματοποιείται η σύνδεση με τη ΒΔ και ο χρήστης βλέπει πλέον το σχήμα της βάσης στη λίστα της κύριας διαπροσωπείας που φέρει την ετικέτα DB Attributes, όπως φαίνεται στο σχήμα 20. Το σύστημα εκτελεί ένα σύνολο από ελέγχους για την αποφυγή απροόπτων, όπως π.χ. λάθος συνδυασμό username/password/database name. Σε μία τέτοια περίπτωση το κατάλληλο προειδοποιητικό μήνυμα θα εμφανιστεί στο χρήστη. Ανάλογο προειδοποιητικό μήνυμα εμφανίζεται και σε περίπτωση που ο χρήστης επιχειρήσει σύνδεση με τον mysql server χωρίς αυτόν να βρίσκεται σε λειτουργία.



Σχήμα 20. Μετά την εισαγωγή της ΒΔ.

Εκτός από την ανάκτηση του σχήματος της ΒΔ το υποσύστημα δημιουργεί ένα διαχειριστή ΒΔ ο οποίος αποθηκεύει τα απαραίτητα στοιχεία για εγκατάσταση σύνδεσης με τη ΒΔ για μελλοντική χρήση. Στη μελλοντική αυτή χρήση θα επανέλθουμε κατά την περιγραφή του υποσυστήματος παραγωγής ABox, όταν θα απαιτηθεί εκ νέου επικοινωνία με τη ΒΔ για να παραχθεί με βάση το περιεχόμενο της το σώμα των ισχυρισμών.

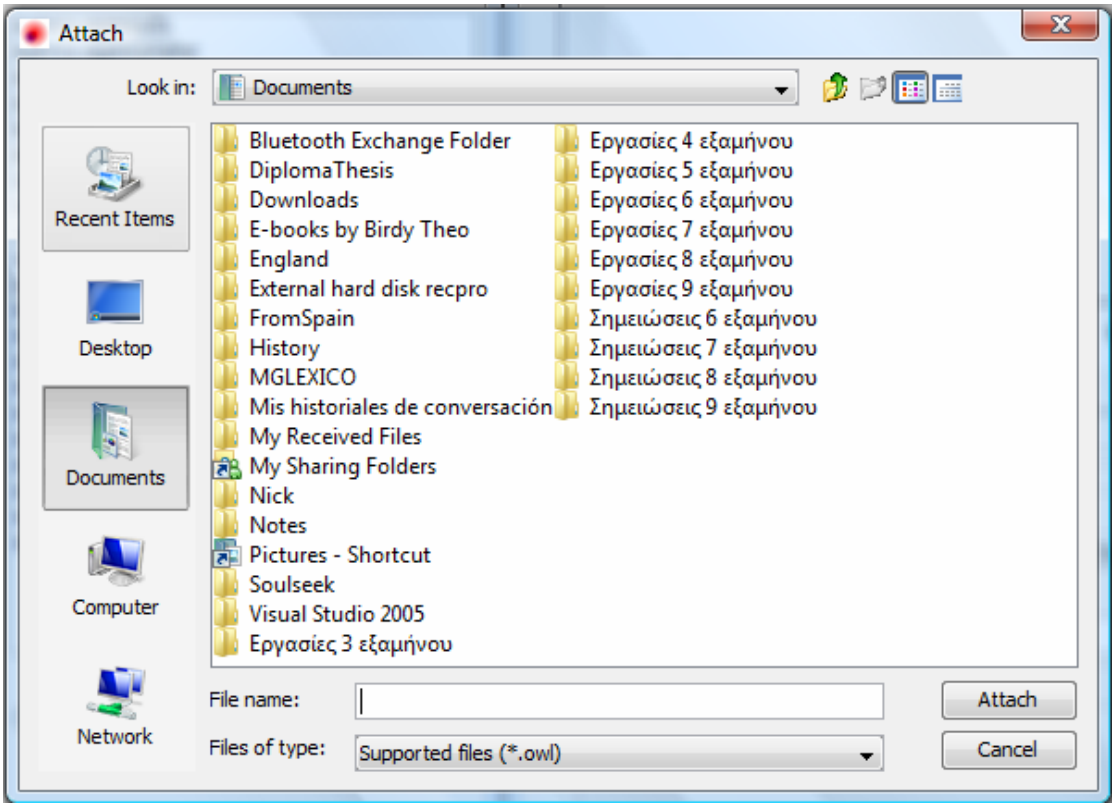
Όσον αφορά στη μορφή με την οποία εμφανίζει το υποσύστημα το σχήμα της βάσης δεδομένων είναι με την εμφάνιση κάθε ιδιοχαρακτηριστικού σε συνδυασμό με τον πίνακα από τον οποίο αυτό προέρχεται. Η μορφή αυτή θα διευκολύνει στη συνέχεια κατά τη δημιουργία των SR, οι οποίοι αντιστοιχουν στις Αντιστοιχίσης Απόδοσης Νοήματος που περιγράψαμε παραπάνω . Συγκεκριμένα διευκολύνει τη δημιουργία μιας Αντιστοίχισης για την οποία το *DatabaseEntity* είναι ιδιοχαρακτηριστικό και όχι σχέση. Εναλλακτικά έχουμε τη δυνατότητα να εμφανίσουμε τις σχέσεις και όχι τα ιδιοχαρακτηριστικά της ΒΔ με έναν τρόπο ο οποίος θα περιγραφεί με περισσότερη λεπτομέρεια κατά την παρουσίαση του υποσυστήματος δημιουργίας SR.

4.2.3 Υποσύστημα εισαγωγής οντολογίας

Το υποσύστημα αυτό απαρτίζεται από δύο βασικά συστατικά: το υποσύστημα εισαγωγής οντολογίας με αρχείο και το υποσύστημα εισαγωγής νέων ατομικών εννοιών και ρόλων.

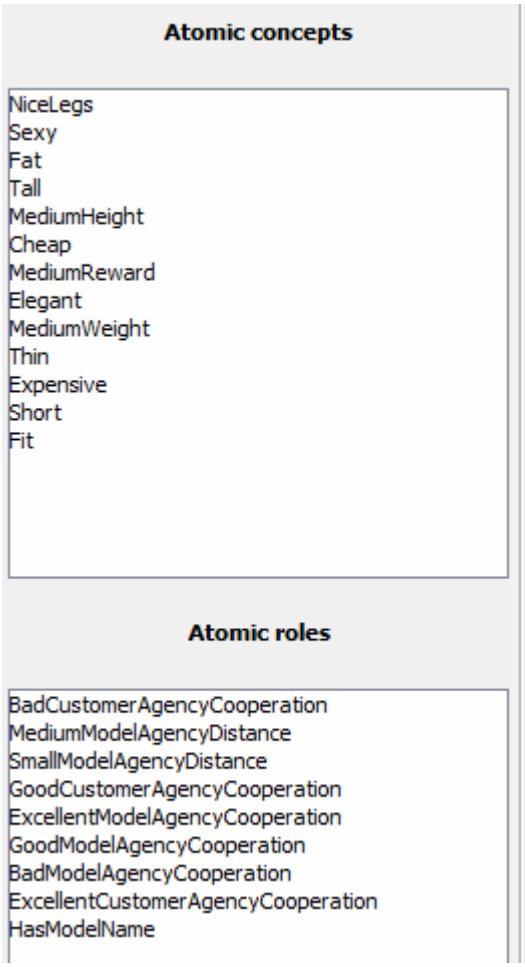
4.2.3.1 Υποσύστημα εισαγωγής οντολογίας με αρχείο

Το παρόν υποσύστημα αναλαμβάνει τη διασύνδεση με ένα .owl αρχείο και την απομόνωση των ατομικών εννοιών και των ατομικών ρόλων που αυτό περιέχει. Το σχήμα 21 δείχνει τον file chooser που επιτελεί αυτή τη λειτουργία. Ο συγκεκριμένος file chooser χρησιμοποιεί φίλτρο ώστε να εμφανίζονται στο χρήστη μόνο φάκελοι και αρχεία .owl της μνήμης του υπολογιστή του.



Σχήμα 21. Διαπροσωπεία υποσυστήματος εισαγωγής οντολογίας με αρχείο.

Μετά την επιλογή του αρχείου το σύστημα δημιουργεί έναν διαχειριστή για την οντολογία που έχει επιλέξει ο χρήστης. Ο διαχειριστής αυτός αναλαμβάνει να εξάγει τις ατομικές έννοιες και τους ατομικούς ρόλους από την οντολογία και να τις εμφανίσει στις αντίστοιχες λίστες της γενικής διαπροσωπείας όπως φαίνεται στο σχήμα 22.

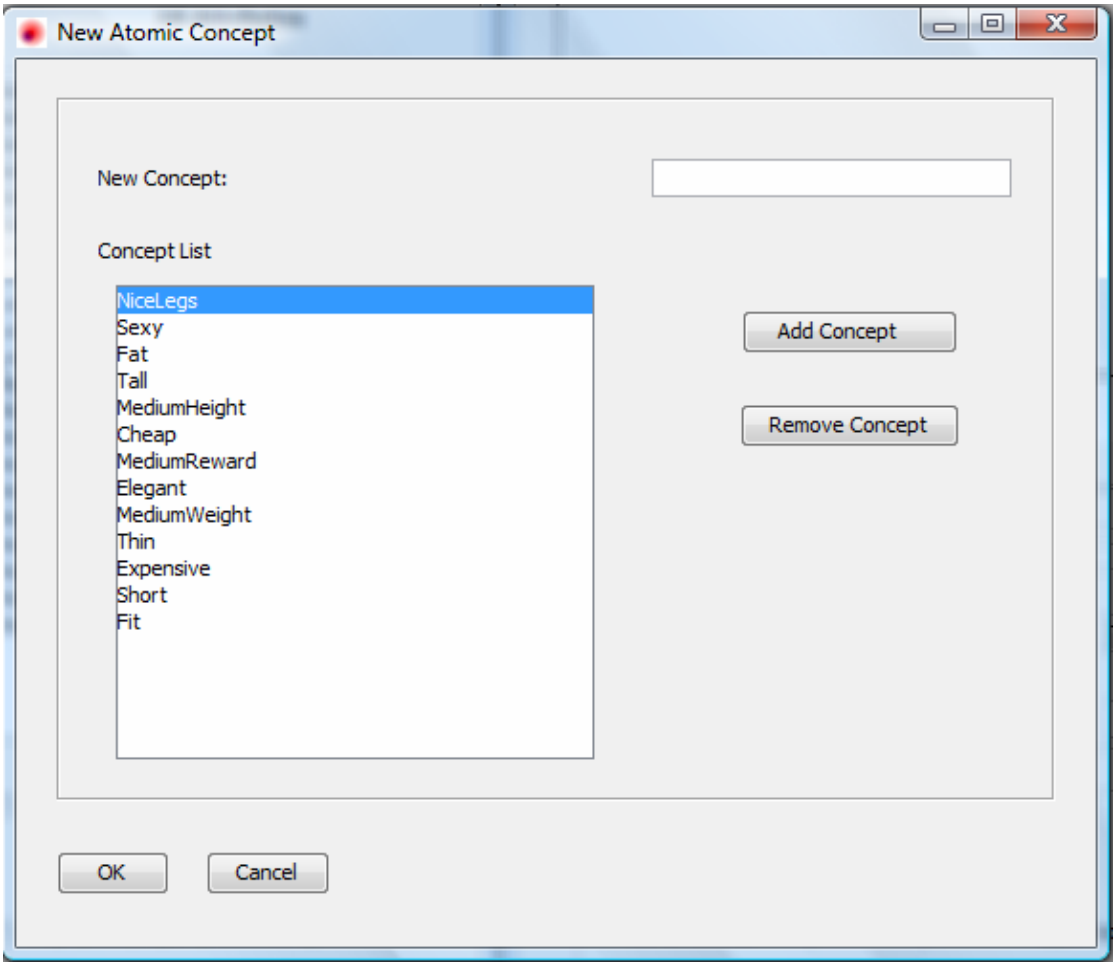


Σχήμα 22. Μετά την εισαγωγή της οντολογίας.

4.2.3.2 Υποσύστημα εισαγωγής νέων ατομικών εννοιών και ρόλων

Το παρόν υποσύστημα προσφέρει στο χρήστη τη δυνατότητα να εισάγει σταδιακά ένα σύνολο από ατομικές έννοιες μέσα από τη διαπροσωπεία του σχήματος 23. Με όμοιο τρόπο εισάγει και ατομικούς ρόλους. Το υποσύστημα πραγματοποιεί έλεγχο έτσι ώστε να μην εισάγονται ατομικές έννοιες ή ρόλοι με το ίδιο όνομα και επιπλέον να μην έχουν κενά στο όνομα τους.

Όσον αφορά τους δύο διαφορετικούς τρόπους εισαγωγής ατομικών εννοιών και ρόλων, στην πρώτη περίπτωση αφαιρούνται όλες οι υπαρχουσες ατομικές έννοιες και ρόλοι και προστίθενται όσοι εμφανίζονται μέσα στο αρχείο .owl. Αντίθετα στη δεύτερη περίπτωση εμφανίζονται οι ήδη υπάρχουσες ατομικές έννοιες και ρόλοι και με βάση αυτή τη λίστα ο χρήστης μπορεί να αφαιρέσει παλιούς ή να προσθέσει καινούριους.

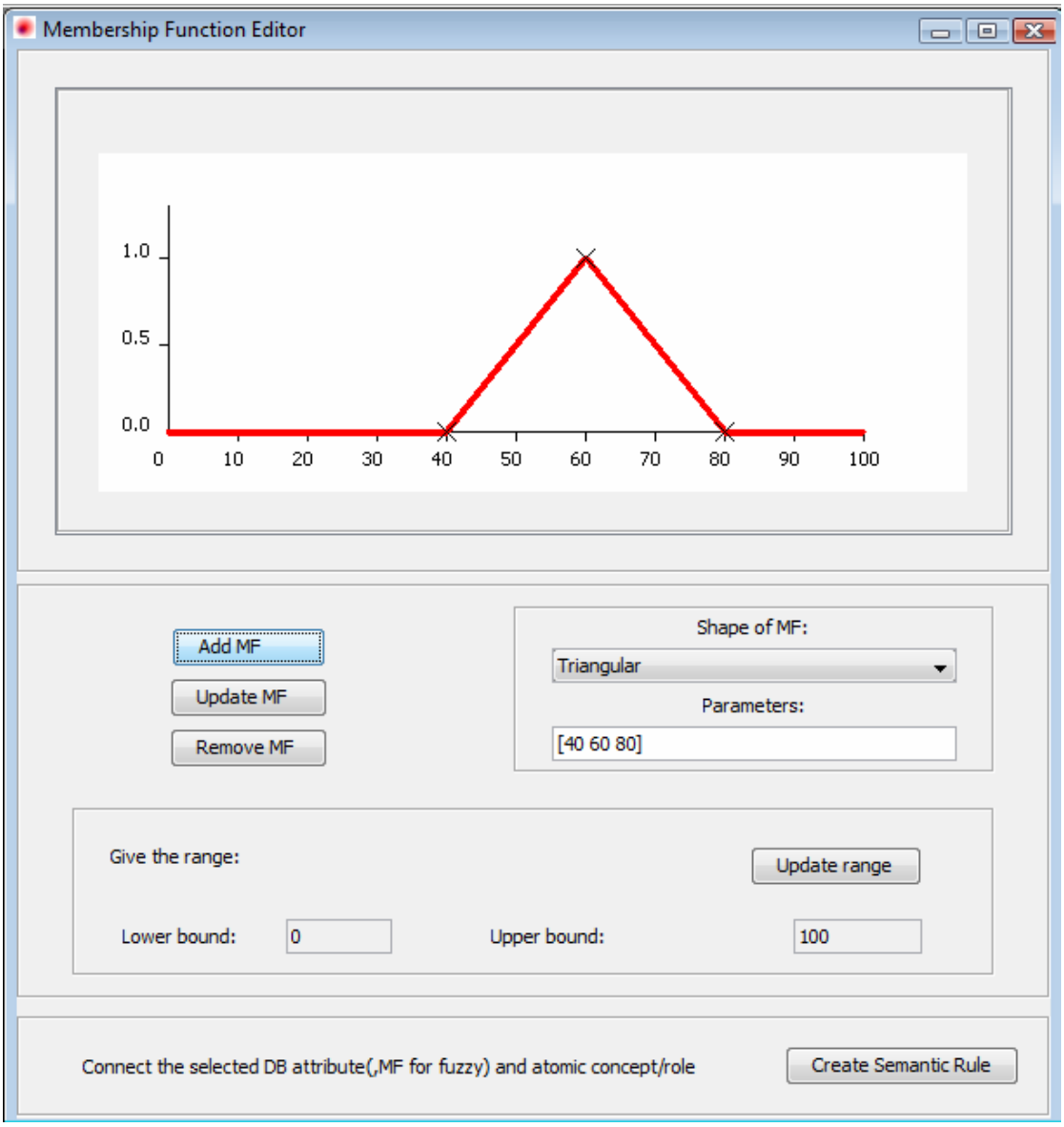


Σχήμα 23. Διαπροσωπεία υποσυστήματος εισαγωγής νέων ατομικών εννοιών.

4.2.4 Υποσύστημα επεξεργασίας MF

Το παρόν υποσύστημα επιτρέπει στο χρήστη να επεξεργαστεί ένα σύνολο από MF, που θα του χρησιμέψουν στη συνέχεια για τη δημιουργία των SR. Το σχήμα 24 δείχνει τη διαπροσωπεία του υποσυστήματος αφού έχει προστεθεί μία τριγωνική συνάρτηση. Το υποσύστημα προσφέρει στο χρήστη τη δυνατότητα εισαγωγής νέας MF και διαγραφής ή τροποποίησης μιας ήδη υπάρχουσας. Η εισαγωγή μιας νέας MF μπορεί να γίνει μόνο σε συνάρτηση με κάποιο ιδιοχαρακτηριστικό της ΒΔ που έχει επιλεγεί από τη λίστα του αριστερού πλαισίου και μόνο όταν το radio button “Fuzzy” είναι επιλεγμένο. Για να τροποποιηθεί ή να διαγραφεί μία MF είναι απαραίτητο πρώτα να επιλεγεί, κάτι που ο χρήστης επιτυγχάνει κάνοντας κλικ με το ποντίκι του πάνω στη γραφική παράσταση της συνάρτησης, οπότε και αυτή αλλάζει από μπλε σε κόκκινο χρώμα. Για το ίδιο ιδιοχαρακτηριστικό ο χρήστης μπορεί να προσθέσει όσες MF θέλει. Κατά την προσθήκη της πρώτης MF για ένα συγκεκριμένο ιδιοχαρακτηριστικό ο χρήστης καθορίζει και τα όρια απεικόνισης της γραφικής παράστασης της MF. Τα όρια αυτά μπορούν να αλλάξουν μέσα

από τη διαπροσωπεία αλλαγής ορίων που το σύστημα διαθέτει. Όσον αφορά την τροποποίηση των MF ο χρήστης αφού διαλέξει μία από αυτές διαθέτει δύο τρόπους να την ενημερώσει: είτε εισάγοντας νέες παραμέτρους στο αντίστοιχο πεδίο κειμένου και πατώντας το αντίστοιχο κουμπί είτε σύροντας και αφήνοντας το ποντίκι του πάνω στα ειδικά σημάδια της γραφικής παράστασης της MF.



Σχήμα 24. Διαπροσωπεία υποσυστήματος επεξεργασίας MF.

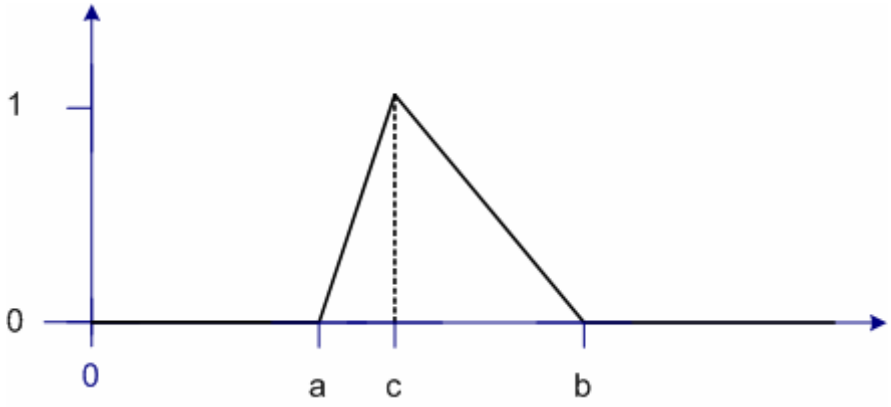
Στο παρόν σύστημα ο χρήστης έχει τη δυνατότητα να χρησιμοποιήσει 4 διαφορετικές MF για την παραγωγή των ασαφών υποθέσεων.

4.2.4.1 Τριγωνική συνάρτηση

Πρόκειται για μία από τις πλέον δημοφιλείς συναρτήσεις συμμετοχής. Εφόσον το σύστημα πραγματεύεται περιπτώσεις που καλύπτουν κανονικά σύνολα η κορυφή του τριγώνου θα έχει πάντοτε τεταγμένη 1. Ο αναλυτικός τύπος της συνάρτησης είναι ο εξής:

$$f_{triangular}(x) = \begin{cases} 1, \text{ εάν } x = a = c = b \\ \dfrac{b-x}{b-a}, \text{ εάν } a = c, a <= x, x <= b \\ \dfrac{x-a}{c-a}, \text{ εάν b} = c, a <= x, x <= c \\ \dfrac{x-a}{c-a}, \text{ εάν } a <= x, x <= c \\ \dfrac{b-x}{b-c}, \text{ εάν } c <= x, x <= b \\ 0 \quad , \text{ διαφορετικά} \end{cases}$$

Στον παραπάνω τύπο οι παράμετροι a,c,b αντιστοιχούν στην αριστερή, μεσαία και δεξιά τετμημένη των τριων γωνιών του τριγώνου, όπως φαίνεται και στο σχήμα 25.



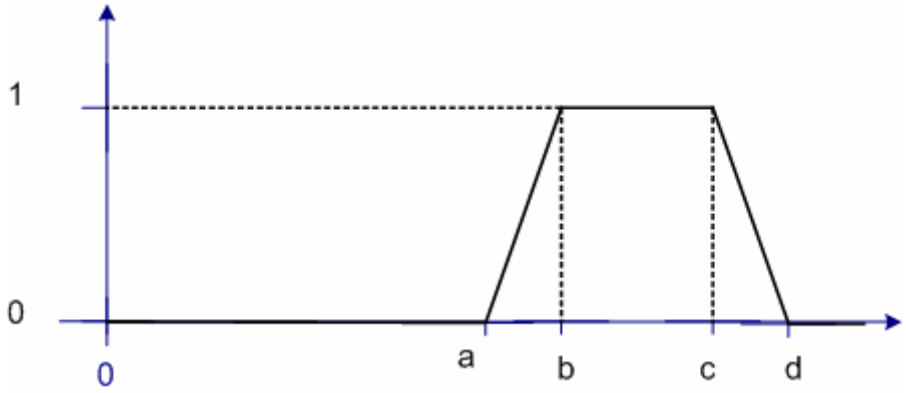
Σχήμα 25. Τριγωνική MF.

4.2.4.2 Τραπεζοειδής συνάρτηση

Η τραπεζοειδής συνάρτηση χρησιμοποιείται συχνά ως συνάρτηση συμμετοχής. Εφόσον το σύστημα πραγματεύεται περιπτώσεις που καλύπτουν κανονικά σύνολα η άνω πλευρά του τραπεζίου θα έχει πάντοτε τεταγμένη 1. Ο αναλυτικός τύπος της συνάρτησης είναι ο εξής:

$$f_{trapezoidal}(x) = \begin{cases} 1, \text{ εάν } x = a = c = b = d \\ f_{triangular}(x), \text{ εάν } b = c \\ \frac{d-x}{d-c}, \text{ εάν } a = b = c \\ \frac{x-a}{b-a}, \text{ εάν } b = c = d \\ \frac{x-a}{b-a}, \text{ εάν } a <= x, x <= b \\ 1, \text{ εάν } b <= x, x <= c \\ \frac{d-x}{d-c}, \text{ εάν } c <= x, x <= d \\ 0 \quad , \text{ διαφορετικά} \end{cases}$$

Στον παραπάνω τύπο οι παράμετροι a,b,c,d αντιστοιχούν στην αριστερή, μεσαία αριστερή μεσαία δεξιά και δεξιά τετμημένη των τεσσάρων γωνιών του τραπεζίου, όπως φαίνεται και στο σχήμα 26.



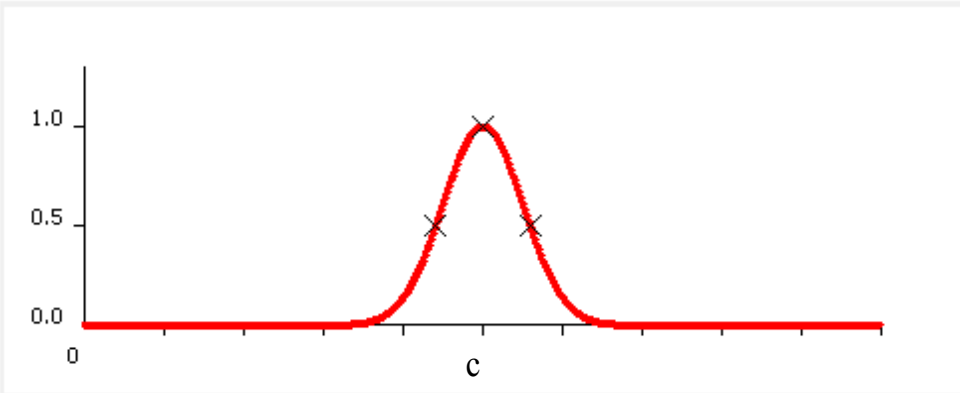
Σχήμα 26. Τραπεζοειδής MF.

4.2.4.3 Γκαουσιανή συνάρτηση

Η γκαουσιανή MF διαφέρει ως προς τις δύο παραπάνω επειδή διαθέτει λεία άκρα τα οποία τείνουν ασυμπτωτικά στο μηδέν και επομένως μπορεί να χρησιμοποιηθεί σε άλλη κατηγορία περιπτώσεων. Ο αναλυτικός τύπος της συνάρτησης είναι ο εξής:

$$f_{gaussian}(x) = e^{\frac{(x-c)^2}{-2\sigma^2}}$$

Η παράμετρος c αντιστοιχεί στην τετμημένη της κορυφής της καμπύλης, ενώ η παράμετρος σ στο «άνοιγμα» της καμπύλης.



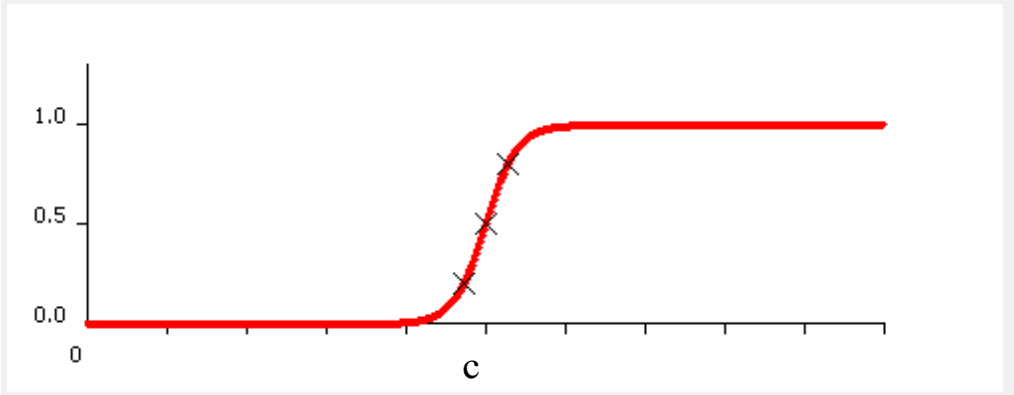
Σχήμα 27. Γκαουσιανή MF.

4.2.4.4 Σιγμοειδής συνάρτηση

Η σιγμοειδής MF διαφέρει ως προς τις παραπάνω επειδή δεν είναι συμμετρική ως προς το σημείο που έχει τεταγμένη 1 αλλά τείνει ασυμπτωτικά σε αυτό. Ο αναλυτικός τύπος της συνάρτησης είναι ο εξής:

$$f_{sigmoid}(x) = \frac{1}{1 + e^{\frac{-a}{x-c}}}$$

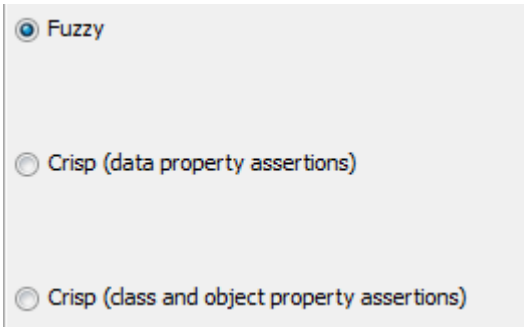
Η παράμετρος c αντιστοιχεί στην τετμημένη της καμπύλης με τεταγμένη 0.5, ενώ η παράμετρος α στο «άνοιγμα» της καμπύλης.



Σχήμα 28. Σιγμοειδής MF.

4.2.5 Υποσύστημα δημιουργίας SR

Το παρόν υποσύστημα είναι υπεύθυνο για τη δημιουργία των SR οι οποίοι όπως έχουμε ήδη αναφέρει αποτελούν υλοποιήσεις των Αντιστοιχίσεων Απόδοσης Νοήματος που περιγράφηκαν στο προηγούμενο κεφάλαιο. Στο προηγούμενο κεφάλαιο διατυπώθηκαν 5 διαφορετικά είδη Αντιστοιχίσεων ανάλογα με τις τιμές που λαμβάνουν τα πεδία State και OntologyEntity. Όπως δείχνει και το σχήμα 29 το σύστημα διαθέτει 3 διαφορετικά radio buttons. Το πρώτο "Fuzzy" αντιστοιχεί σε SR οι οποίοι υλοποιούν Αντιστοιχίσεις με State=Fuzzy. Το δεύτερο "Crisp(data property assertions)" αντιστοιχεί σε SR οι οποίοι υλοποιούν Αντιστοιχίσεις με State=Crisp και OntologyEntity=DataProperty ενώ το τρίτο "Crisp (class and object property assertions)" είναι για State=Crisp και OntologyEntity=Class ή OntologyEntity=ObjectProperty.

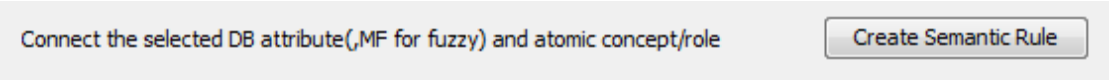


Σχήμα 29. Διαφορετικά radio buttons για δημιουργία διαφορετικών SR.

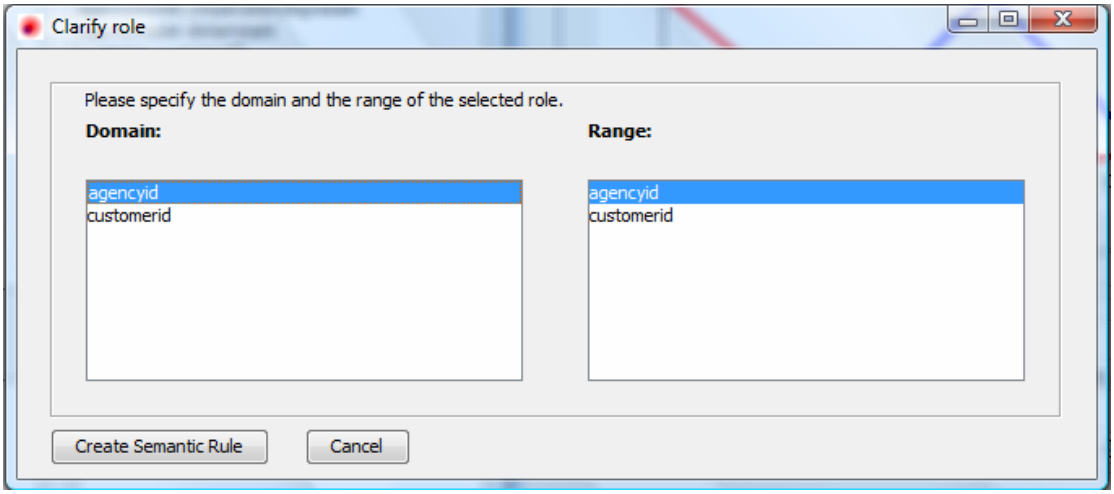
Το σχήμα 30 δείχνει πώς ο χρήστης θα παράξει τον αντίστοιχο SR. Ωστόσο πριν το πάτημα αυτού του κουμπιού ο χρήστης οφείλει να έχει καθορίσει επακριβώς τα δομικά στοιχεία με βάση τα οποία θα συντεθεί η Αντιστοίχιση Απόδοσης Νοήματος. Διακρίνουμε 3 περιπτώσεις που αντιστοιχούν στα 3 διαφορετικά radio buttons:

- "Fuzzy". Ο χρήστης θα πρέπει απαραίτητα να έχει επιλέξει μία ατομική έννοια ή ρόλο, μία MF και ένα ιδιοχαρακτηριστικό. Όσον αφορά το πεδίο key για ατομική έννοια (ή τα πεδία domain και range για ατομκούς ρόλους) της Αντιστοίχισης Απόδοσης Νοήματος διευκρινίζονται από το χρήστη με τη βοήθεια παραθύρων τα οποία αναδύονται κατά το πάτημα του "CreateSemantic Rule". Μια τέτοια περίπτωση δείχνει το σχήμα 31.
- "Crisp(data property assertions)". Ο χρήστης οφείλει να έχει επιλέξει ένα ιδιοχαρακτηριστικό και έναν ατομικό ρόλο. Ζητάται από το χρήστη διευκρίνιση για το πεδίο subject όπως δείχνει και το σχήμα 32.
- "Crisp (class and object property assertions)". Ο χρήστης οφείλει να έχει επιλέξει μόνο μία σχέση από το σχήμα της βάσης. Στη συνέχεια επιλέγει εάν επιθυμεί class ή

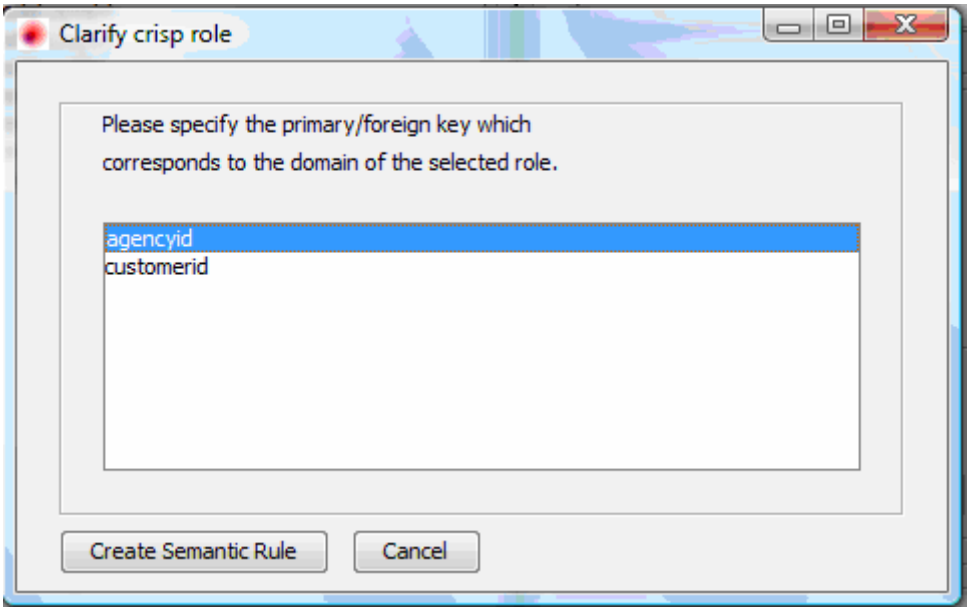
object property και τα αντίστοιχα πεδία (key ή domain/range) που εμφανίζονται στις λίστες (σχήμα 33).



Σχήμα 30. Παραγωγή SR.



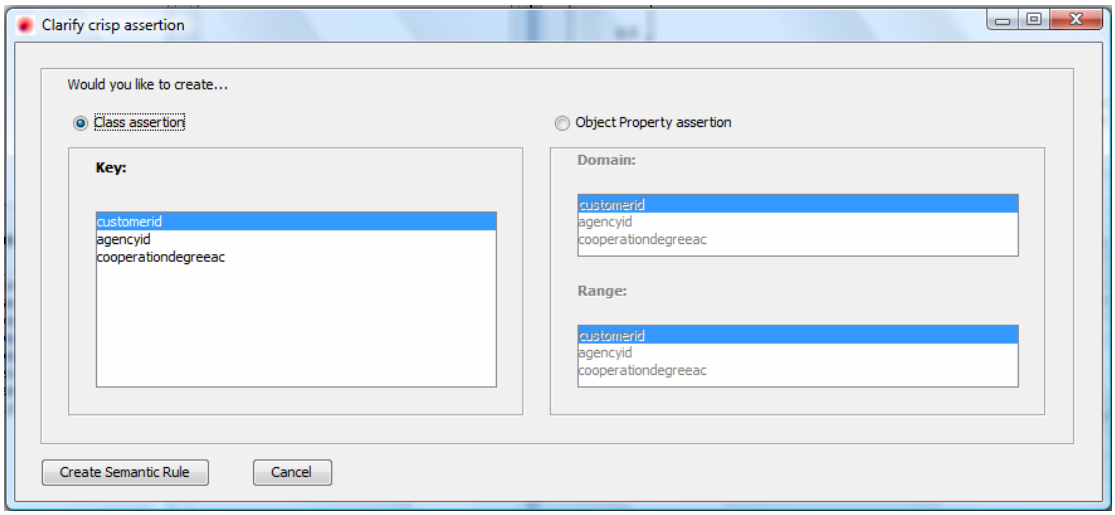
Σχήμα 31. Διευκρίνιση domain και range κατά τη δημιουγία SR.



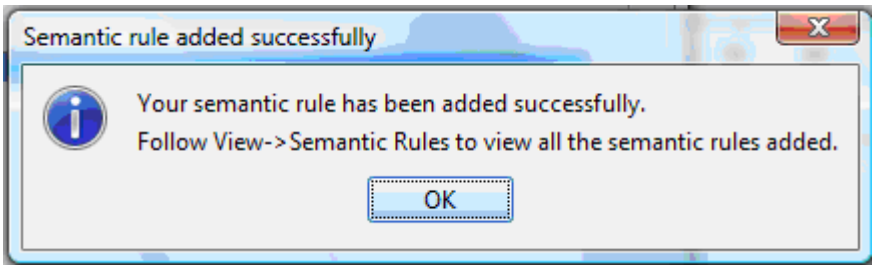
Σχήμα 32. Διευκρίνιση subject κατά τη δημιουργία SR.

Στο σημείο αυτό να τονίσουμε ότι το σύστημα εκτελεί όλους τους ελέγχους έτσι ώστε να ικανοποιούνται οι περιορισμοί που περιγράφηκαν στο προηγούμενο κεφάλαιο. Σε περίπτωση που κάποιος από αυτούς δεν ικανοποιείται εμφανίζεται αντίστοιχο μήνυμα στο χρήστη. Ανάμεσα σε αυτούς συμπεριλαμβάνεται και η 1-1 αντιστοιχία μεταξύ ατομικής

έννοιας/ρόλου και SR. Τέλος, εάν έχει κατασκευαστεί ένας ορθός SR που ικανοποιεί όλους τους περιορισμούς το μήνυμα του σχήματος 34 εμφανίζεται στο χρήστη.



Σχήμα 33. Διευκρίνιση για δημιουργία crisp SR.

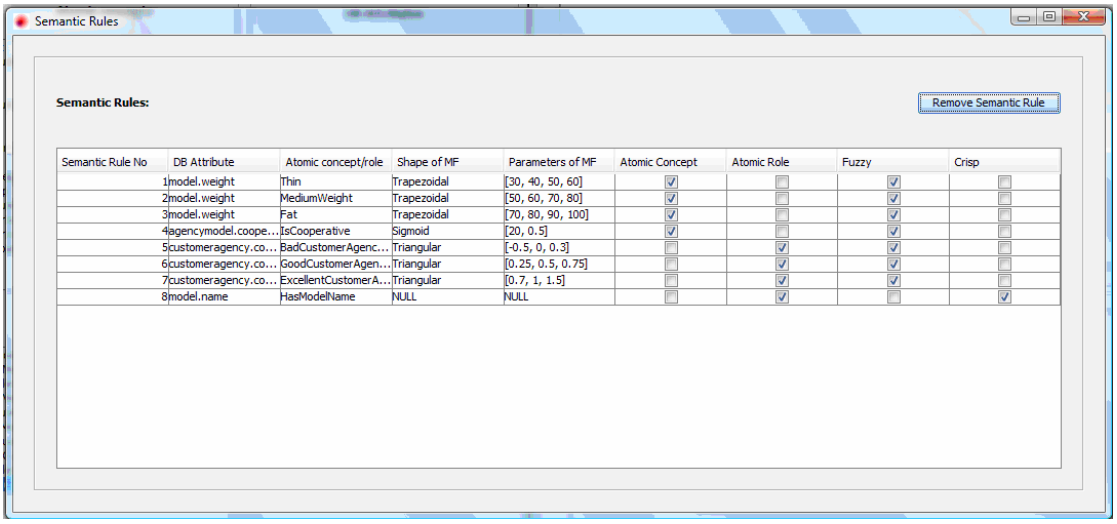


Σχήμα 34. Μήνυμα επιτυχημένης δημιουργίας ενός SR.

4.2.6 Υποσύστημα προβολής και αφαίρεσης SR

Το παρόν υποσύστημα είναι υπεύθυνο για την προβολή με εποπτικό τρόπο στο χρήστη των SR που αυτός έχει δημιουργήσει. Για κάθε SR δίνονται:

- Ο αύξων αριθμός του.
- Η αντίστοιχη DatabaseEntity της Αντιστοίχισης Απόδοσης Νοήματος.
- Η αντίστοιχη ατομική έννοια ή ρόλος.
- Το σχήμα της MF (εάν πρόκειται για fuzzy SR).
- Οι παράμετροι της MF (εάν πρόκειται για fuzzy SR).
- Εάν πρόκειται για SR που εμπλέκει ατομική έννοια ή ρόλο
- Εάν πρόκειται για fuzzy ή crisp SR.



Σχήμα 35. Υποσύστημα προβολής SR.

Σε περίπτωση που ο χρήστης επιθυμεί να αφαιρέσει έναν SR, επειδή π.χ. θέλει να δημιουργήσει άλλον SR για την ίδια ατομική έννοια ή ρόλο μπορεί να το κάνει με τη βοήθεια αυτού του υποσυστήματος.

4.2.7 Υποσύστημα παραγωγής ABox

Το υποσύστημα αυτό διαθέτει έναν διαχειριστή που αναλαμβάνει την παραγωγή του Abox. Ο εν λόγω διαχειριστής εξετάζει κάθε SR (Αντιστοιχιση Απόδοσης Νοήματος) που έχει δημιουργηθεί και ακολουθεί τον αντίστοιχο αλγόριθμο, όπως περιγράφηκε στο προηγούμενο κεφάλαιο. Διακρίνονται 5 σενάρια:

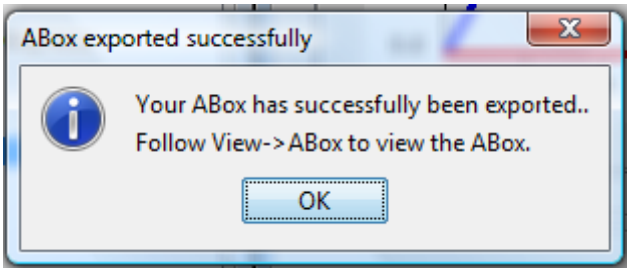
- Fuzzy SR που αναφέρεται σε ατομική έννοια. Αρχικά ανακτώνται από τη βάση δεδομένων τα ζευγάρια πρωτεύον κλειδί – DB Attribute . Για κάθε ένα από αυτά τα ζευγάρια τώρα θα προστεθούν μια σειρά από αξιώματα. Δημιουργείται ένα OWL άτομο του οποίου το όνομα (τμήμα του URI) προέρχεται από το concatenation του ονόματος της στήλης του πρωτεύοντος κλειδιού και της αριθμητικής τιμής του πρωτεύοντος κλειδιού (π.χ. ModelID647). Επίσης, δημιουργείται μία OWL κλάση με όνομα (τμήμα του URI) το όνομα της ατομικής έννοιας. Με βάση αυτά τα δύο δημιουργείται ένα αντικείμενο axiom της κλάσης OWLClassAssertionAxiom. Στη συνέχεια υπολογίζεται η ασάφεια, λαμβάνοντας τη συνάρτηση από τον κανόνα και υπολογίζοντας το y δίνοντας ως x την τιμή της ΒΔ για το DB Attribute. Εάν η ασάφεια δεν είναι 0, προστίθενται συνολικά 3 αξιώματα. Το πρώτο είναι το axiom το οποίο προστίθεται με τη βοήθεια της μεθόδου applyChange. Το δεύτερο και τρίτο απαιτούν κάποια περαιτέρω δομικά στοιχεία. Χρησιμοποιώντας την ασάφεια που υπολογίστηκε από τη MF δημιουργείται ένα αντικείμενο constant της κλάσης OWLConstant και το “leq” ή το “geq” για το

δεύτερο και τρίτο αξίωμα αντίστοιχα. Κατόπιν κατασκευάζεται ένα αντικείμενο fuzzyDegree της κλάσης OWLConstantAnnotation χρησιμοποιώντας το hasFuzzinessURI και το constant. Με τη χρήση του fuzzyDegree αλλά και του axiom κατασκευάζεται το αντικείμενο annAx της κλάσης OWLAxiomAnnotationAxiom. Στην τελευταία κατασκευή το axiom είναι απαραίτητο προκειμένου να διευκρινίζεται σε ποιο αξίωμα αναφέρεται το συγκεκριμένο annotation.Τέλος τα δύο αξιώματα που μόλις κατασκευάστηκαν προστίθενται στην οντολογία με το γνωστό Visitor design pattern.

- Fuzzy SR που αναφέρεται σε ατομικό ρόλο.Αυτό σημαίνει ότι έχουμε να κάνουμε με Object Property. Αρχικά ανακτώνται από τη βάση δεδομένων οι τριπλέτες domain-range – DB Attribute . Για κάθε μία από αυτές τις τριπλέτες τώρα θα προστεθούν μια σειρά από αξιώματα. Δημιουργούνται δύο OWL άτομα των οποίων τα ονόματα (τμήματα του URI) προέρχονται από το concatenation του ονόματος της στήλης που αντιστοιχεί στο domain ή range και της αριθμητικής τιμής αυτής της στήλης (π.χ. AgencyID566, CustomerID89). Επίσης, δημιουργείται μία OWL Object Property με όνομα (τμήμα του URI) το όνομα του ρόλου. Με βάση αυτά τα δύο δημιουργείται ένα αντικείμενο axiom της κλάσης OWLObjectPropertyAssertionAxiom. Στη συνέχεια υπολογίζεται η ασάφεια, λαμβάνοντας τη συνάρτηση από τον κανόνα και υπολογίζοντας το y δίνοντας ως x την τιμή της ΒΔ για το DB Attribute. Εάν η ασάφεια δεν είναι 0, προστίθενται συνολικά 3 αξιώματα. Το πρώτο είναι το axiom το οποίο προστίθεται με τη βοήθεια της μεθόδου applyChange του omanager. Το δεύτερο και τρίτο απαιτούν κάποια περαιτέρω δομικά στοιχεία. Χρησιμοποιώντας την ασάφεια που υπολογίστηκε από τη MF δημιουργείται ένα αντικείμενο constant της κλάσης OWLConstant και το “leq” ή το “geq” για το δεύτερο και τρίτο αξίωμα αντίστοιχα. Κατόπιν κατασκευάζεται ένα αντικείμενο fuzzyDegree της κλάσης OWLConstantAnnotation χρησιμοποιώντας το hasFuzzinessURI και το constant. Με τη χρήση του fuzzyDegree αλλά και του axiom κατασκευάζεται το αντικείμενο annAx της κλάσης OWLAxiomAnnotationAxiom. Στην τελευταία κατασκευή το axiom είναι απαραίτητο προκειμένου να διευκρινίζεται σε ποιο αξίωμα αναφέρεται το συγκεκριμένο annotation.Τέλος τα δύο αξιώματα που μόλις κατασκευάστηκαν προστίθενται στην οντολογία με το γνωστό Visitor design pattern.
- Crisp SR που εμπλέκει data property. Αρχικά ανακτώνται από τη βάση δεδομένων οι τριπλέτες domain-data value– DB Attribute . Για κάθε μία από αυτές τις τριπλέτες τώρα θα προστεθεί ένα αξίωμα. Δημιουργείτα ένα OWL άτομο του οποίου το όνομα (τμήμα του URI) προέρχεται από το concatenation του ονόματος της στήλης που αντιστοιχεί στο domain και της αριθμητικής τιμής αυτής της στήλης (π.χ. ModelID789). Επίσης, δημιουργείται μία OWL Data Property με όνομα (τμήμα του URI) το όνομα του ρόλου. Με βάση αυτά τα δύο δημιουργείται ένα αντικείμενο axiom της κλάσης OWLDataPropertyAssertionAxiom.

- Crisp SR που εμπλέκει class. Αρχικά ανακτώνται από τη βάση δεδομένων οι τιμές της ΒΔ για το ιδιοχαρακτηριστικό key. Για κάθε μία από αυτές τώρα θα προστεθεί ένα αξίωμα. Δημιουργείτα ένα OWL άτομο του οποίου το όνομα (τμήμα του URI) προέρχεται από το concatenation του ονόματος της στήλης που αντιστοιχεί στο key και της τιμής αυτής της στήλης (π.χ. ModelID789). Επίσης, δημιουργείται μία OWL Class με όνομα (τμήμα του URI) το όνομα της σχέσης. Με βάση αυτά δημιουργείται ένα αντικείμενο axiom της κλάσης OWLClassAssertionAxiom.
- Crisp SR που εμπλέκει object property. Αρχικά ανακτώνται από τη βάση δεδομένων οι τιμές της ΒΔ για τα ιδιοχαρακτηριστικά domain και range. Για κάθε ένα από αυτά τα ζεύγη τώρα θα προστεθεί ένα αξίωμα. Δημιουργείται ένα OWL άτομο του οποίου το όνομα (τμήμα του URI) προέρχεται από το concatenation του ονόματος της στήλης που αντιστοιχεί στο domain και της τιμής αυτής της στήλης (π.χ. AgencyID73). Αντίστοιχα δημιουργείται το OWL άτομο για το range. Επίσης, δημιουργείται μία OWL Object Property με όνομα (τμήμα του URI) το όνομα της σχέσης. Με βάση αυτά δημιουργείται ένα αντικείμενο axiom της κλάσης OWLObjectPropertyAssertionAxiom.

Έτσι το σώμα των ισχυρισμών κατασκευάζεται σταδιακά εφαρμόζοντας έναν έναν τους SR στην οντολογία και τη ΒΔ. Ο τρόπος κατασκευής των ασαφών και των μη ασαφών ισχυρισμών υποδεικνύεται από τη μεθοδολογία που περιγράφηκε από το προηγούμενο κεφάλαιο. Η κατασκευή του ABox πρακτικά σημαίνει τη δημιουργία ενός αρχείου με το όνομα MyABox.owl το οποίο τοποθετείται μέσα σε ένα φάκελο tmp, στο path στο οποίο τρέχει η εφαρμογή. Εάν γίνει με επιτυχία η παραγωγή του ABox εμφανίζεται το μήνυμα του σχήματος 36.



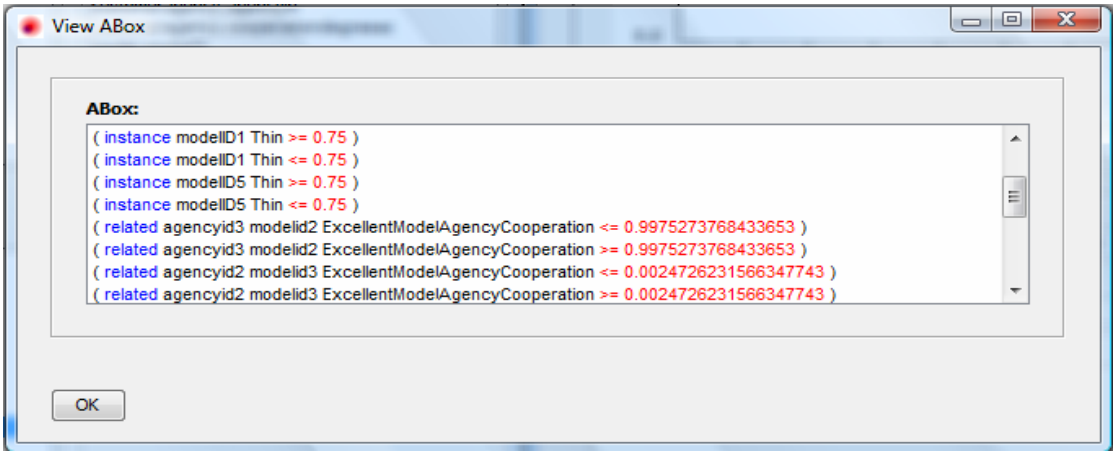
Σχήμα 36. Μήνυμα για επιτυχημένη παραγωγή ABox.

4.2.8 Υποσύστημα προβολής ABox

Το παρόν υποσύστημα παρέχει στο χρήστη οπτική πρόσβαση στο ABox. Για να το επιτύχει αυτό επεξεργάζεται το αρχείο MyBox.owl το οποίο έχει παραχθεί από το προηγούμενο

υποσύστημα και εμφανίζει τα αξιώματα που βρίσκει σε αυτό με τρόπο ευπαρουσίαστο και εύληπτο στο χρήστη(σχήμα 37). Πιο συγκεκριμένα εξετάζει έναν έναν τους ισχυρισμούς και:

- Εάν πρόκειται για ισχυρισμό κλάσης λαμβάνονται όλα τα αξιώματα που σχετίζονται με το συγκεκριμένο αξίωμα και περιλαμβάνουν annotations. Για κάθε ένα από αυτά εισάγεται ένα σύνολο απο strings τα οποία περιλαμβάνουν το όνομα της κλάσης, το όνομα του ατόμου, το σύμβολο του ασαφούς ισχυρισμού καθώς επίσης και το βαθμό αυτού. Τα strings αυτά τυπώνονται κάθε ένα με το κατάλληλο χρώμα. Έτσι η λέξη instance (η οποία δηλώνει ότι πρόκειται για αξίωμα κλάσης) γράφεται με μπλε, το σύμβολο και ο βαθμός της ασάφειας με κόκκινο ενώ όλα τα υπόλοιπα με μαύρο. Εάν δεν υπάρχουν annotations τυπώνεται μόνο το άτομο και η κλάση στην οποία αυτό ανήκει.
- Εάν πρόκειται για ισχυρισμό Object Property λαμβάνονται όλα τα αξιώματα που σχετίζονται με το συγκεκριμένο αξίωμα και περιλαμβάνουν annotations. Για κάθε ένα από αυτά εισάγεται ένα σύνολο απο strings τα οποία περιλαμβάνουν το όνομα της object property, τα ονόματα των ατόμων που εμπλέκονται, το σύμβολο του ασαφούς ισχυρισμού καθώς επίσης και το βαθμό αυτού. Τα strings αυτά τυπώνονται κάθε ένα με το κατάλληλο χρώμα. Έτσι η λέξη related (η οποία δηλώνει ότι πρόκειται για αξίωμα ιδιότητας) γράφεται με μπλε, το σύμβολο και ο βαθμός της ασάφειας με κόκκινο ενώ όλα τα υπόλοιπα με μαύρο. Εάν δεν υπάρχουν annotations τυπώνονται μόνο τα άτομα και η ιδιότητα αντικειμένων που τα συνδέει.
- Εάν τέλος πρόκειται για ισχυρισμό Data Property τυπώνεται το όνομα της data property, το όνομα του ατόμου που αφορά και η τιμή της data property. Τα strings αυτά τυπώνονται κάθε ένα με το κατάλληλο χρώμα. Έτσι η λέξη related (η οποία δηλώνει ότι πρόκειται για αξίωμα ιδιότητας) γράφεται με μπλε, ενώ όλα τα υπόλοιπα με μαύρο.



Σχήμα 37. Υποσύστημα προβολής ABox.

4.3 Χρήση βιβλιοθηκών

Για την ανάπτυξη και τη λειτουργία του συστήματος χρησιμοποιήθηκαν οι εξής δύο βιβλιοθήκες σε μορφή .jar αρχείων:

- mysql-connector-java-5.0.7-bin.jar, για τη σύνδεση και τη μεταφορά δεδομένων με τη mysql βάση
- owlapi-bin.jar, για τη διαχείριση των οντολογιών

Ειδικά για τη δεύτερη βιβλιοθήκη ορισμένες κλάσεις χρησιμοποιήθηκαν τροποποιημένες από τον Γ. Στοΐλο (gstoil@image.ntua.gr), επειδή η επίσημη έκδοση του API που διανέμεται εμφανίζει κάποια προβλήματα στις λειτουργίες των annotations. Συγκεκριμένα οι κλάσεις που χρησιμοποιήθηκαν τροποποιημένες ήταν οι:

- OWLClassAssertionAxiom.java του πακέτου org.semanticweb.owl.model
- OWLDataFactory.java του πακέτου org.semanticweb.owl.model
- OWLObjectPropertyAssertionAxiom.java του πακέτου org.semanticweb.owl.model
- OWLClassAssertionAxiomImpl.java του πακέτου uk.ac.manchester.cs.owl
- OWLDataFactory.java του πακέτου uk.ac.manchester.cs.owl
- OWLObjectPropertyAssertionAxiom.java του πακέτου uk.ac.manchester.cs.owl
- OWLClassAssertionAxiomElementHandler.java του πακέτου org.coode.owl.owlxmlparser
- OWLObjectPropertyAssertionAxiomElementHandler.java του πακέτου org.coode.owl.owlxmlparser

4.4 Πλατφόρμες και προγραμματιστικά εργαλεία

Η ανάπτυξη του συστήματος έγινε σε λειτουργικό σύστημα Windows Vista Home Premium, ενώ ο java editor που χρησιμοποιήθηκε ήταν το Eclipse 3.2.2. Επίσης έγινε χρήση της έκδοσης 1.6.0_10 του jdk.

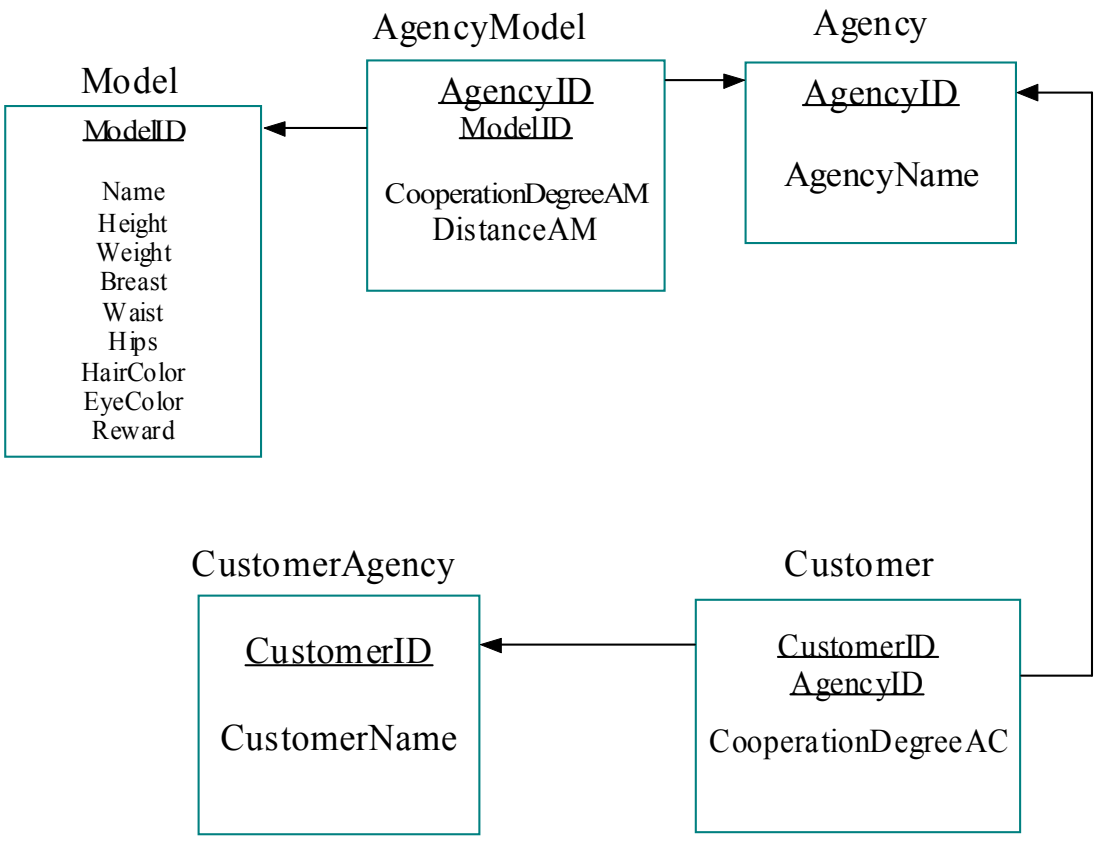
Το σύστημα μετατράπηκε σε αρχείο .jar έτσι ώστε να εξασφαλίζεται η λειτουργία του σε όλους τους υπολογιστές που τρέχουν java εφαρμογές. Το .jar αρχείο περιλαμβάνει το σύνολο των κλάσεων, καθώς επίσης και όλες τις κλάσεις των βοηθητικών βιβλιοθηκών και τέλος τις εικόνες (αρχεία .gif) που χρησιμοποιούνται.

5 *Εγχειρίδιο χρήσης του συστήματος*

Παρακάτω παρουσιάζουμε ένα σύνολο από σενάρια τα οποία δοκιμάστηκαν και τα οποία ευελπιστούμε ότι καλύπτουν το σύνολο των δυνατοτήτων αυτού του λογισμικού καθώς επίσης και των δυνατών καταστάσεων στις οποίες αυτό μπορεί να περιέλθει. Πριν ωστόσο προχωρήσουμε στην αλληλεπίδραση του χρήστη με την κυρίως εφαρμογή θα περιγράψουμε όσα προαπαιτούμενα χρειάζονται για την εύρυθμη λειτουργία του συστήματος. Τα εν λόγω προαπαιτούμενα συνοψίζονται στα εξής 2: την ύπαρξη μίας mysql βάσης δεδομένων στην οποία θα επιδράσει το σύστημα καθώς επίσης και την ύπαρξη ενός αρχείου .owl το οποίο θα περιλαμβάνει εκτός των άλλων ένα σύνολο από ατομικές έννοιες και ένα σύνολο από ατομικές ιδιότητες οι οποίες προορίζονται για ασαφοποίηση.

5.1 Δημιουργία mysql βάσης δεδομένων και πλήρωση της με δεδομένα

Αρχικά δίδεται σχηματικά η περιγραφή της mysql βάσης:



Σχήμα 38. Σχήμα της σχεσιακή βάσης δεδομένων που θα χρησιμοποιηθεί για τον έλεγχο του συστήματος.

Τα πεδία που συγκροτούν τα πρωτεύοντα κλειδιά εμφανίζοντα υπογραμμισμένα ενώ τα βελάκια υποδεικνύουν τις σχέσεις ξένων κλειδιών. Ονομάσαμε τη βάση δεδομένων db και δημιουργήσαμε έναν mysql account με username και password. Στη συνέχεια δίδονται οι εντολές sql που δημιούργησαν την παραπάνω βάση δεδομένων:

```
create database db;

use db;

create table Model
(       ModelID  INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        Name VARCHAR(30),
        Height FLOAT,
```

```
        Weight FLOAT,
        Breast FLOAT,
        Waist  FLOAT,
        Hips   FLOAT,
        HairColor VARCHAR(30),
        EyeColor VARCHAR(30),
        Reward FLOAT          );

create table Agency
(       AgencyID  INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        AgencyName VARCHAR(30)   );

create table Customer
(       CustomerID  INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        CustomerName VARCHAR(30));

create table AgencyModel
(       AgencyID  INT NOT NULL references Agency(AgencyID),
        ModelID          INT NOT NULL references Model(ModelID),
        CooperationDegreeAM FLOAT,
        DistanceAM FLOAT     ,
        primary key (AgencyID,ModelID));

create table CustomerAgency
(       CustomerID  INT NOT NULL references Customer(CustomerID),
        AgencyID    INT NOT NULL references Agency(AgencyID),
        CooperationDegreeAC FLOAT,
        primary key (AgencyID,CustomerID));
```

Με τη βοήθεια εντολών της μορφής:

insert into Model(ModelID, Name, Height, Weight, Breast, Waist, Hips, HairColor, EyeColor, Reward) values (1,'Marina',180,55,85,62,90,'black','black',200);

γέμισε ο πίνακας Model με μία σειρά από πλειάδες. Οι εντολές για τους πίνακες Agency, Customer, AgencyModel, CustomerAgency είναι αντίστοιχα:

insert into Agency(AgencyID, AgencyName) values (1,'Stef');

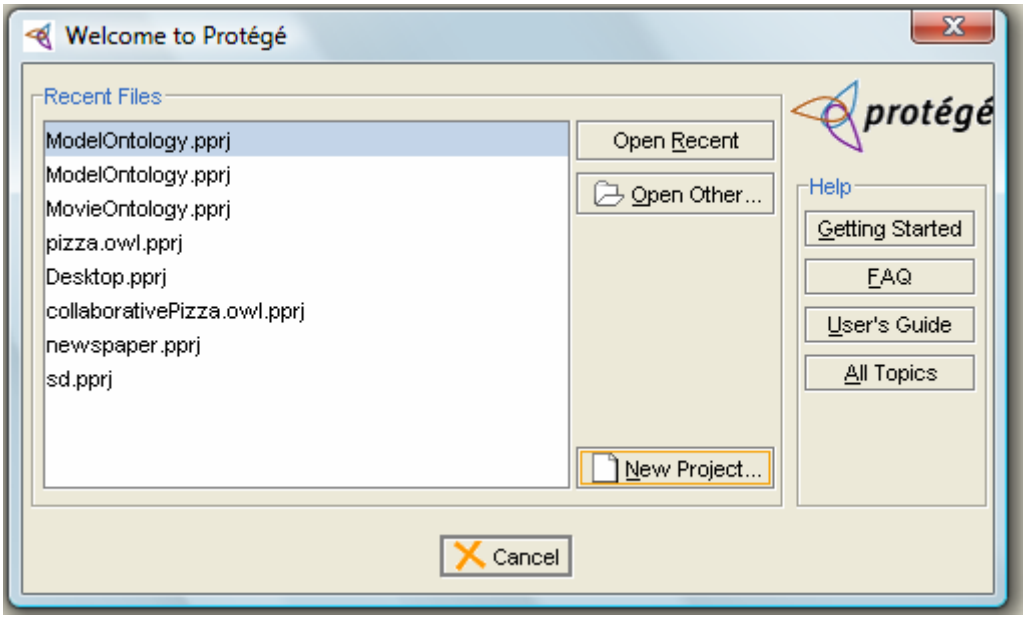
insert into Customer(CustomerID, CustomerName) values (1,'Rilke');

insert into AgencyModel(AgencyID, ModelID, CooperationDegreeAM ,DistanceAM) values (3,1,0.4,100);

insert into CustomerAgency(CustomerID, AgencyID, CooperationDegreeAC) values (4,4,0.42);

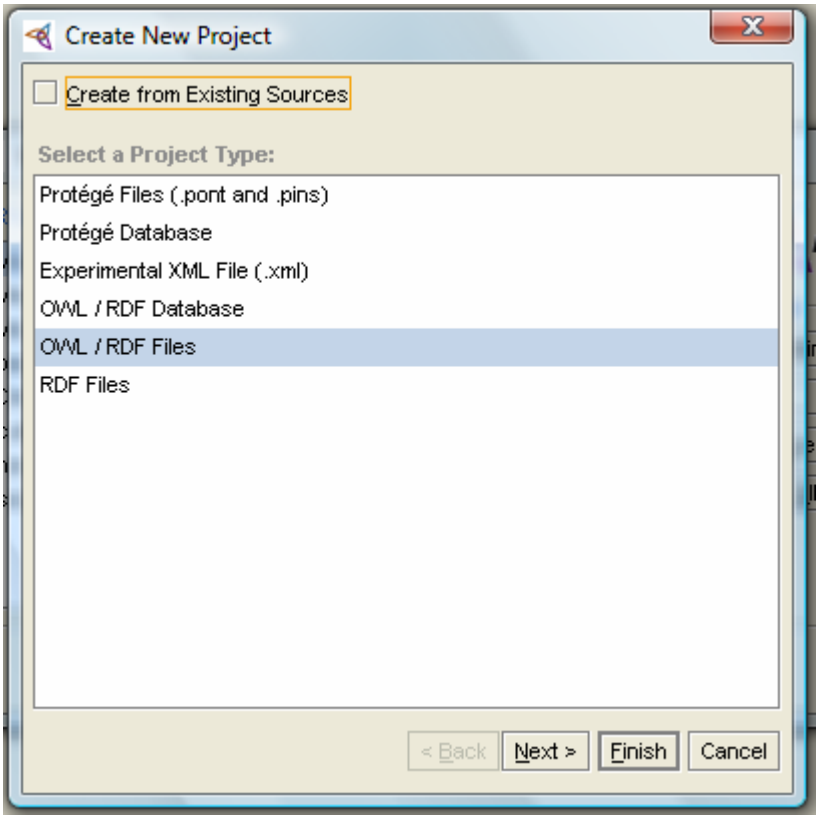
5.2 Δημιουργία οντολογίας

Σκοπός είναι να δημιουργηθεί ένα .owl αρχείο το οποίο θα ορίζει μία οντολογία και κατ'επέκταση ένα σύνολο από έννοιες και ρόλους που θα χρησιμοποιηθούν από το σύστημα μας ως είσοδος. Το παραπάνω αρχείο το δημιούργήσαμε με τη βοήθεια της πλατφόρμας Protégé. Η διαδικασία περιγράφεται παρακάτω. Με την εκκίνηση της εφαρμογής Protégé εμφανίζεται το παρακάτω παράθυρο στο οποίο επιλέγουμε το κουμπί New Project.



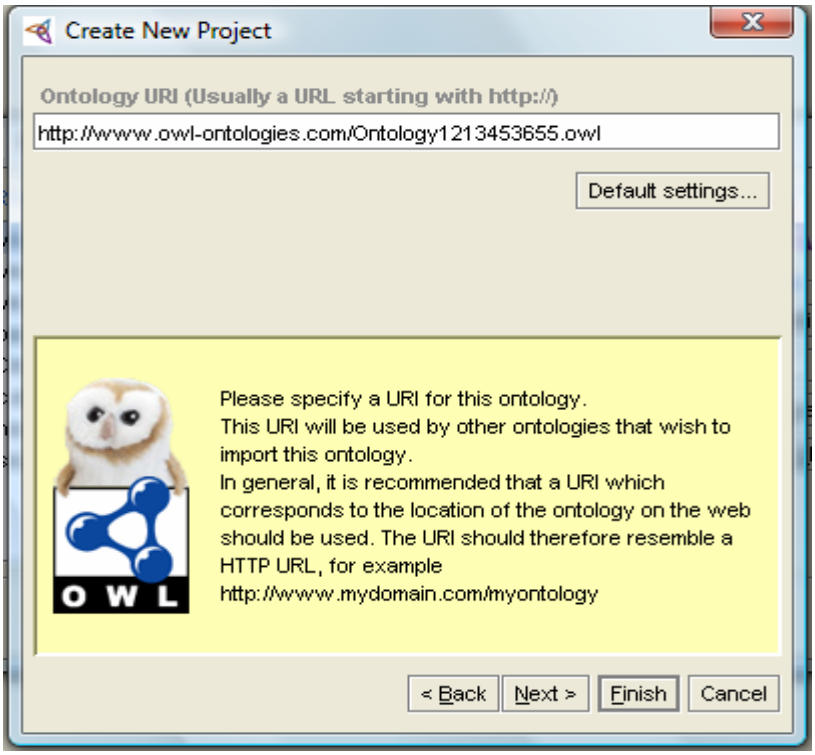
Σχήμα 39. Πρώτο βήμα για τη δημιουργία .owl αρχείου με τη βοήθεια του Protégé.

Κατόπιν εμφανίζεται η παρακάτω φόρμα



Σχήμα 40. Πρώτο βήμα για τη δημιουργία .owl αρχείου με τη βοήθεια του Protégé.

όπου με το πάτημα του Next (και το προεπιλεγμένο Project Type OWL/RDF Files) προχωράμε στην επόμενη φόρμα:

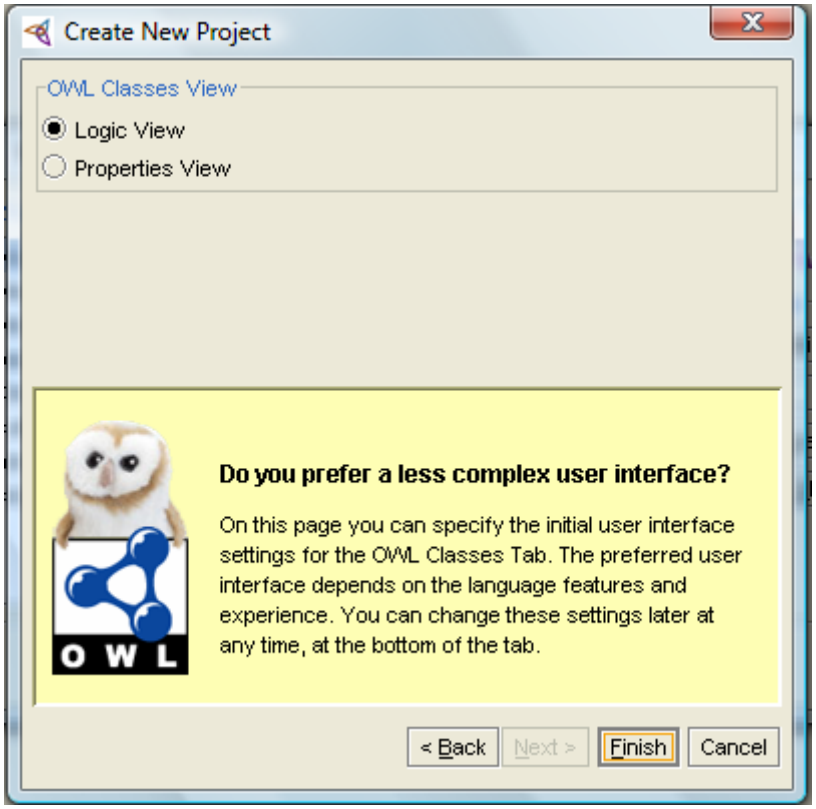


Σχήμα 41. Τρίτο βήμα για τη δημιουργία .owl αρχείου με τη βοήθεια του Protégé.

Αφού καθοριστεί το URI (έχουμε τη δυνατότητα να επεξεργαστούμε την default επιλογή που μας προσφέρει το Protégé) στη συνέχεια προχωρούμε στη φόρμα:

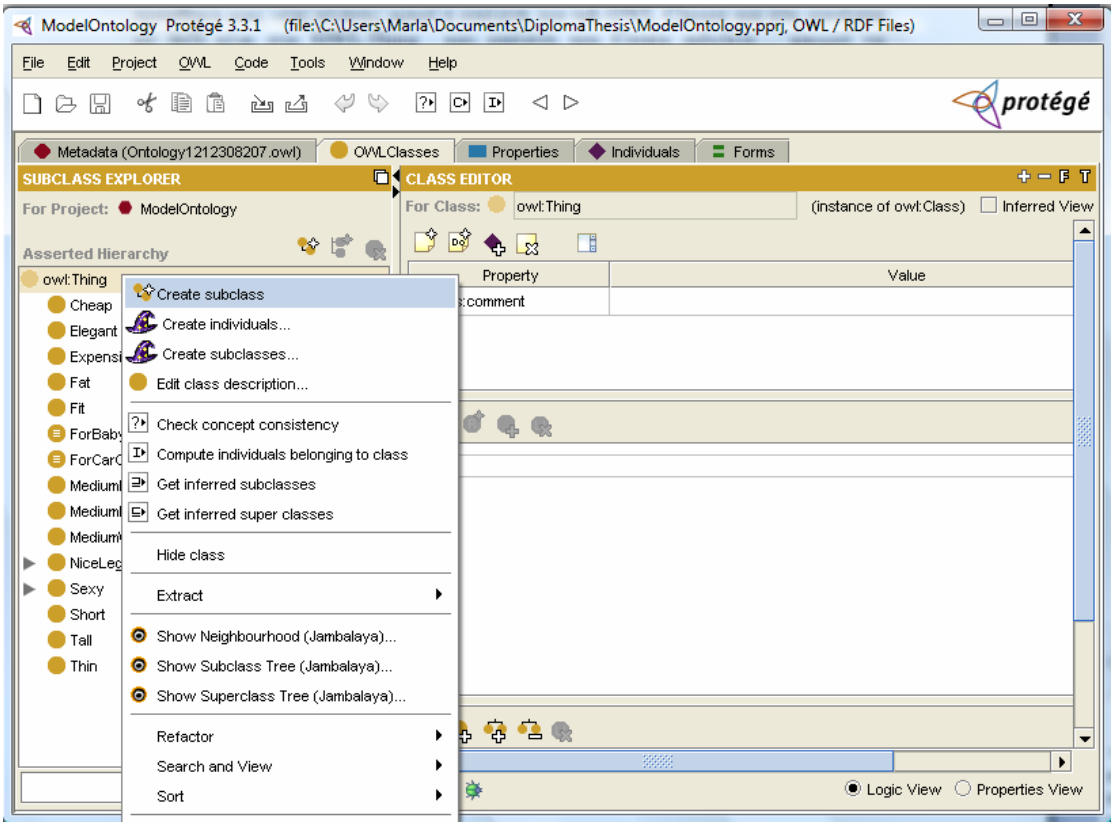
Σχήμα 42. Τέταρτο βήμα για τη δημιουργία .owl αρχείου με τη βοήθεια του Protégé.

και επιλέγουμε το radiobutton OWL DL εφόσον το σύστημα είναι προορισμένο να επεξεργάζεται αρχεία που είναι γραμμένα σε αυτή τη διάλεκτο της OWL.

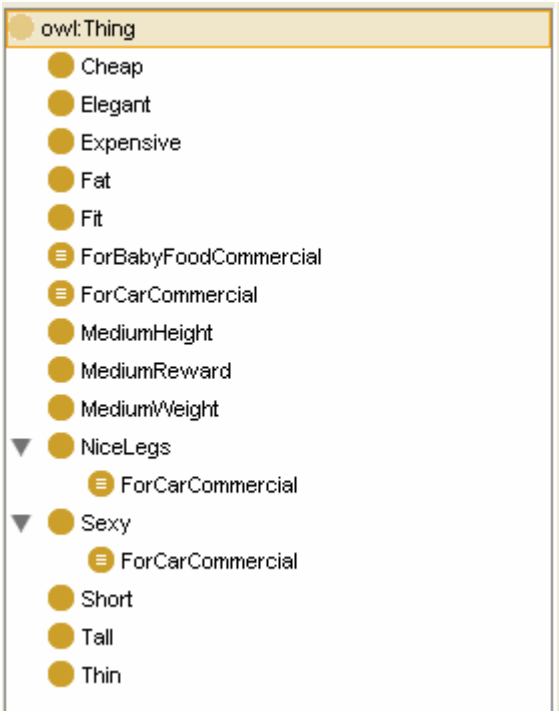


Σχήμα 43. Τελευταίο βήμα για τη δημιουργία .owl αρχείου με τη βοήθεια του Protégé.

Τέλος μεταφερόμαστε στην παραπάνω φόρμα στην οποία με το πάτημα του Finish αρχίζουμε την επεξεργασία της οντολογίας μας. Στη συνέχεια γίνεται η προσθήκη των Classes και (Object ή Data) properties κατά βούληση. Συγκεκριμένα για την προσθήκη μιας νέας κλάσης αρκεί η επιλογή του tab OWL Classes και στη συνέχεια με δεξί κλικ στο OWL:Thing (φαίνεται στην ακόλουθη εικόνα) και επιλογή του Create subclass… μπορεί να δημιουργηθεί μία νέα κλάση.



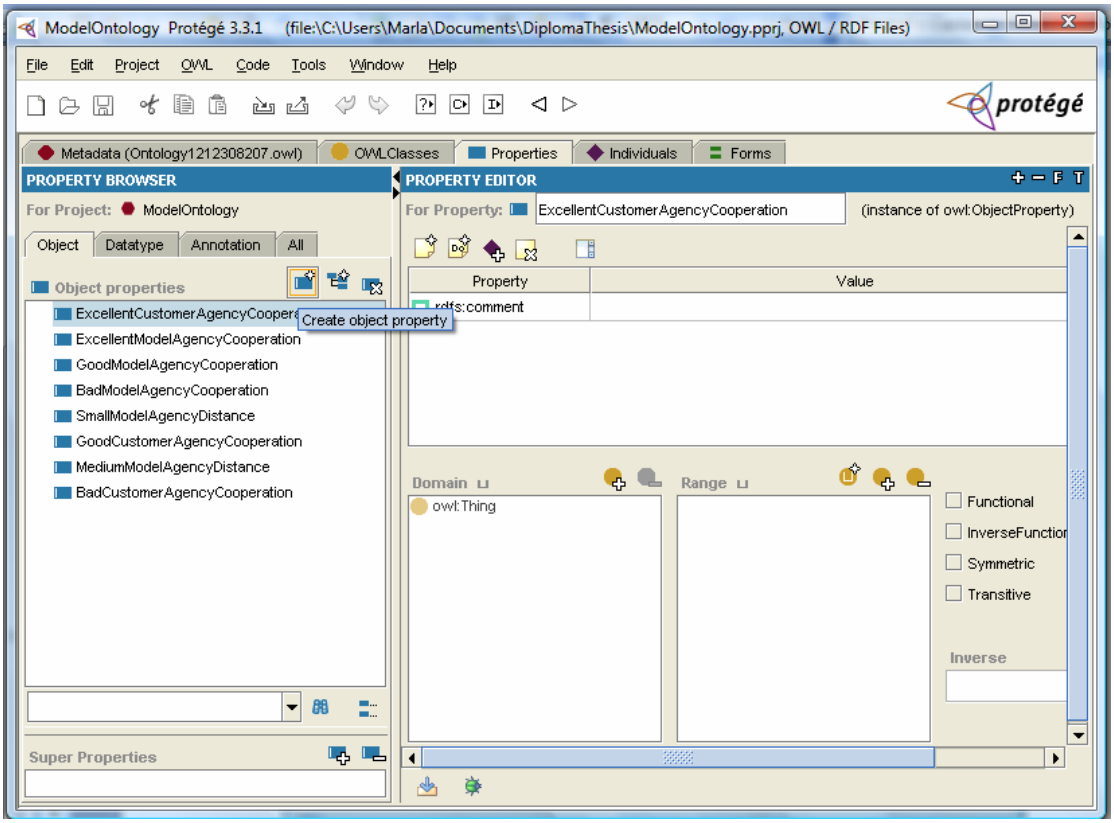
Σχήμα 44. Προσθήκη κλάσεων στη δημιουργούμενη οντολογία.



Σχήμα 45. Ιεραρχία κλάσεων της οντολογίας που θα χρησιμοποιηθεί.

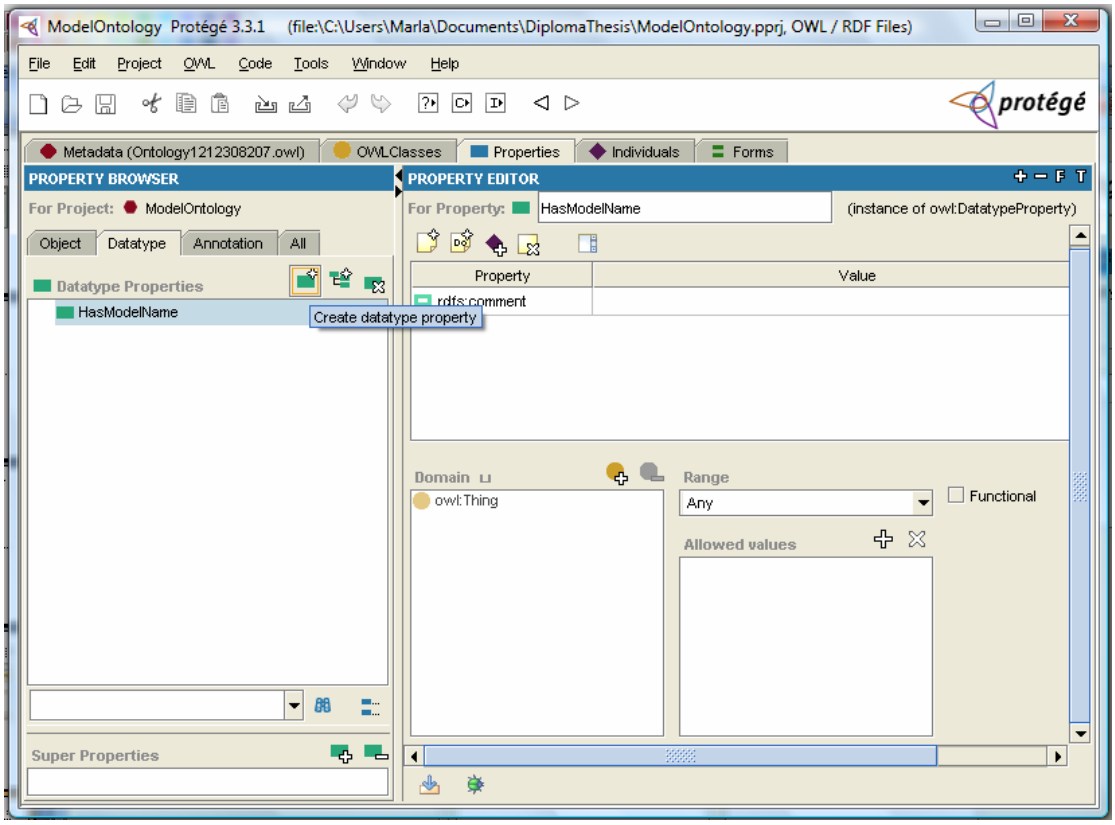
Παραπάνω φαίνεται η ιεραρχία των κλάσεων που έχει δημιουργηθεί για τις ανάγκες ελέγχου του συστήματος, όπως αυτή απεικονίζεται με τη βοήθεια του Protégé. Υπάρχουν τόσο ατομικές κλάσεις όσο και σύνθετες.

Αντίστοιχα για την προσθήκη μιας νέας Object Property αρκεί η επιλογή του tab OWL Properties και κατόπιν του tab Object. Με το πάτημα του εικονιδίου που φαίνεται στην παρακάτω εικόνα δημιουργούμε τη νέα Object Property.



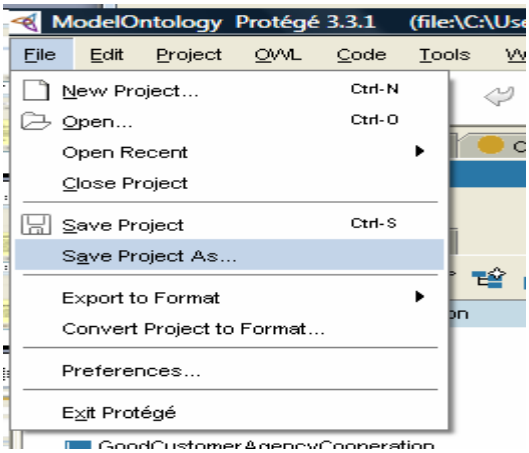
Σχήμα 46. Πρόσθηκη νέας Object Property και παρουσίαση των Object Properties της οντολογίας που θα χρησιμοποιηθεί.

Επιπλέον στην παραπάνω εικόνα φαίνεται και το σύνολο των Object Properties που έχουμε ορίσει για χρήση από το σύστημα. Αντίστοιχη είναι και η δημιουργία των Data Properties όπως δείχνει και η παρακάτω εικόνα:



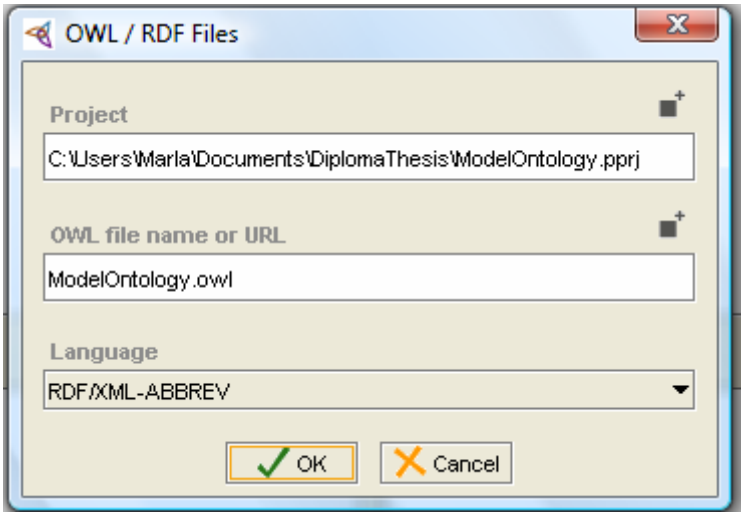
Σχήμα 47. Πρόσθηκη νέας Data Property και παρουσίαση των Data Properties της οντολογίας που θα χρησιμοποιηθεί.

Τέλος σώζουμε το project:



Σχήμα 48. Σώσιμο του project.

δίνοντας το όνομα που επιθυμούμε:



Σχήμα 49. Ορισμός του path και του ονόματος του project.

και είμαστε έτοιμοι να ξεκινήσουμε τον έλεγχο του υλοποιημένου συστήματος. Να σημειώσουμε ότι στο path που έχουμε ορίσει για το σώσιμο του project αποθηκεύονται τρία αρχεία με extensions .owl, .pprj και .repository. Το πρώτο από αυτά είναι αυτό που μας ενδιαφέρει.

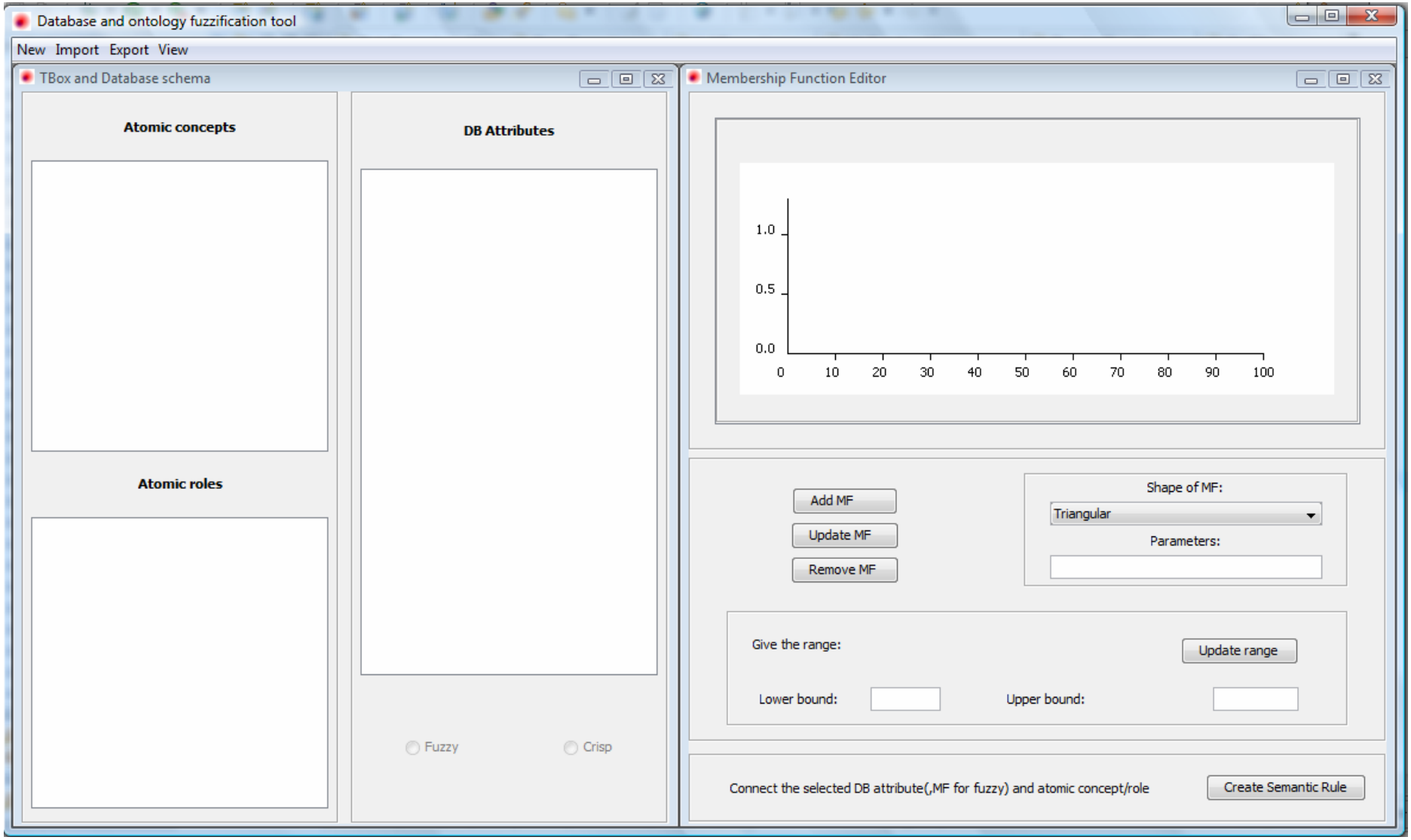
5.3 Λειτουργίες που προσφέρει το σύστημα

Πριν περιγράψουμε την ακριβή διαδικασία με την οποία ο χρήστης μπορεί να αλληλεπιδράσει με το λογισμικό θα σκιαγραφήσουμε τις λειτουργίες που αυτό προσφέρει. Πρόκειται για ένα σύστημα που λαμβάνει ως είσοδο μία βάση δεδομένων και ένα σώμα ορολογίας μιας οντολογίας (TBox) και δίνει ως έξοδο ένα σώμα ισχυρισμών μιας οντολογίας (ABox). Η σημασία έγκειται στο ότι το ABox περιλαμβάνει ένα σύνολο από ασαφείς ισχυρισμούς, δηλαδή ισχυρισμούς εκπεφρασμένους σε fuzzy-OWL DL και όχι σε OWL DL. Όμως προκειμένου να καθορίσει το σύστημα πώς θα κατανείμει αυτή την ασάφεια χρειάζεται τη συμβολή του χρήστη ο οποίος χρησιμοποιεί αυτό το interface για να κατευθύνει ανάλογα με τις επιθυμίες του την παραγωγή των ασαφών ισχυρισμών.

Ο χρήστης αφού προσδιορίσει με κατάλληλο τρόπο τις δύο παραπάνω εισόδους ορίζει μια σειρά από MF (membership functions, συναρτήσεις μέλους) . Κάθε MF ορίζεται σε συνάρτηση με ένα DB Attribute. Ωστόσο δεν ορίζονται MF για όλα τα DB Attributes. Κάθε DB Attribute διαθέτει δύο επιλογές: να παράγει fuzzy assertions και crisp assertions. Fuzzy assertions μπορούν να παράγουν μόνο τα DB Attributes τα οποία έχουν αριθμητικό πεδίο ορισμού. Πρόκειται για assertions τα οποία συσχετίζονται με ένα συντελεστή, ο οποίος εκφράζει την ασάφεια τους. Επομένως προκειμένου να οριστούν MF για κάποιο DB Attribute θα πρέπει ο χρήστης προηγουμένως να έχει επιλέξει την παραγωγή των fuzzy assertions για αυτό το DB Attribute. Το επόμενο βήμα είναι η παραγωγή ενός συνόλου από Semantic Rules, οι οποίοι αποτελούν δομές δεδομένων που σκοπό έχουν να κωδικοποιήσουν για το σύστημα τις επιθυμίες του χρήστη. Ένας Semantic Rule αποτελείται από τρία τμήματα τα οποία ο χρήστης πρέπει να καθορίσει. Το πρώτο τμήμα είναι ένα DB Attribute. Εάν πρόκειται για DB Attribute συσχετισμένο με παραγωγή crisp assertions το δεύτερο τμήμα είναι μια ατομική ιδιότητα (πρόκειται για data property) και το τρίτο τμήμα τίθεται αυτόματα από το σύστημα. Εάν πρόκειται για DB Attribute συσχετισμένο με παραγωγή fuzzy assertions το δεύτερο τμήμα είναι μία ατομική έννοια ή ένας ατομικός ρόλος (πρόκειται για object property) και το τρίτο τμήμα μία MF ορισμένη με κατάλληλο τρόπο από το χρήστη. Έχοντας ορίσει το σύνολο των παραπάνω δομών για όσα από τα DB Attributes ο χρήστης επιθυμεί δίνει εντολή για παραγωγή του ABox. Στη συνέχεια έχει πρόσβαση με δύο τρόπους στο ABox που παράγεται: είτε με απευθείας εποπτεία του .owl αρχείου από τον ίδιο ή από κάποιο εξωτερικό parser είτε με άμεση παρουσίαση στο χρήστη των ασαφών υποθέσεων σε racer form.

5.4 Έλεγχος του συστήματος με τη βοήθεια σεναρίων

Η φόρμα με την οποία έρχεται αρχικά σε επαφή ο χρήστης είναι η παρακάτω:


Database and ontology fuzzification tool
New Import Export View
TBox and Database schema
Atomic concepts
Atomic roles
DB Attributes
Fuzzy
Crisp
Membership Function Editor
1.0
0.5
0.0
0 10 20 30 40 50 60 70 80 90 100
Add MF
Update MF
Remove MF
Shape of MF:
Triangular
Parameters:
Give the range:
Update range
Lower bound:
Upper bound:
Connect the selected DB attribute(,MF for fuzzy) and atomic concept/role
Create Semantic Rule

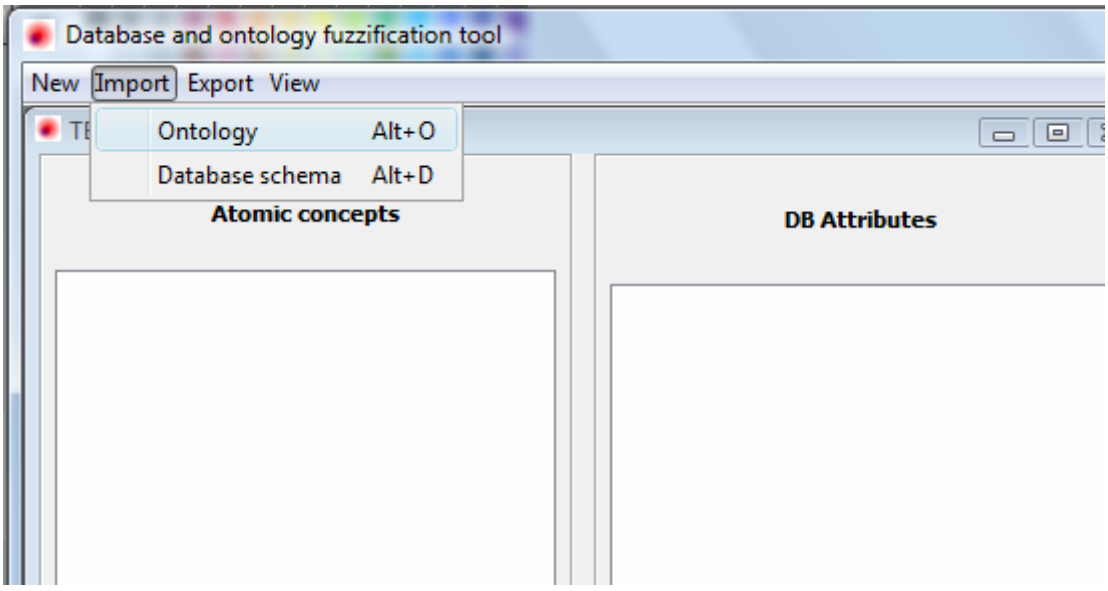
Σχήμα 50. Αρχική φόρμα του συστήματος.

Βασικό δομικό στοιχείο είναι μία μπάρα που αποτελείται από 4 μενού (τα New, Import, Export και View) τα οποία δίνουν τη δυνατότητα στο χρήστη να αλληλεπιδράσει με το λογισμικό μέσα από τις επιλογές που τα μενού του προσφέρουν. Τα άλλα δύο κύρια δομικά στοιχεία είναι τα δύο εσωτερικά frames (TBox and Database schema καθώς επίσης και Membership Function Editor) κάθε ένα από τα οποία περιλαμβάνει ένα σύνολο απο γραφικά components η χρησιμότητα των οποίων θα φανεί στη συνέχεια.

Προκειμένου να επιτευχθεί με επιτυχία η παραγωγή ασαφών ισχυρισμών απαιτείται αρχικά η τροφοδότηση του συστήματος αφενός μεν με ένα σύνολο αξιωμάτων ορολογίας τα οποία θα προέρχονται από μία οντολογία (TBox) αφετέρου δε το σχήμα μιας βάσης δεδομένων καθώς και τα δεδομένα που αυτή περιέχει. Ξεκινάμε με την ανάγνωση της οντολογίας.

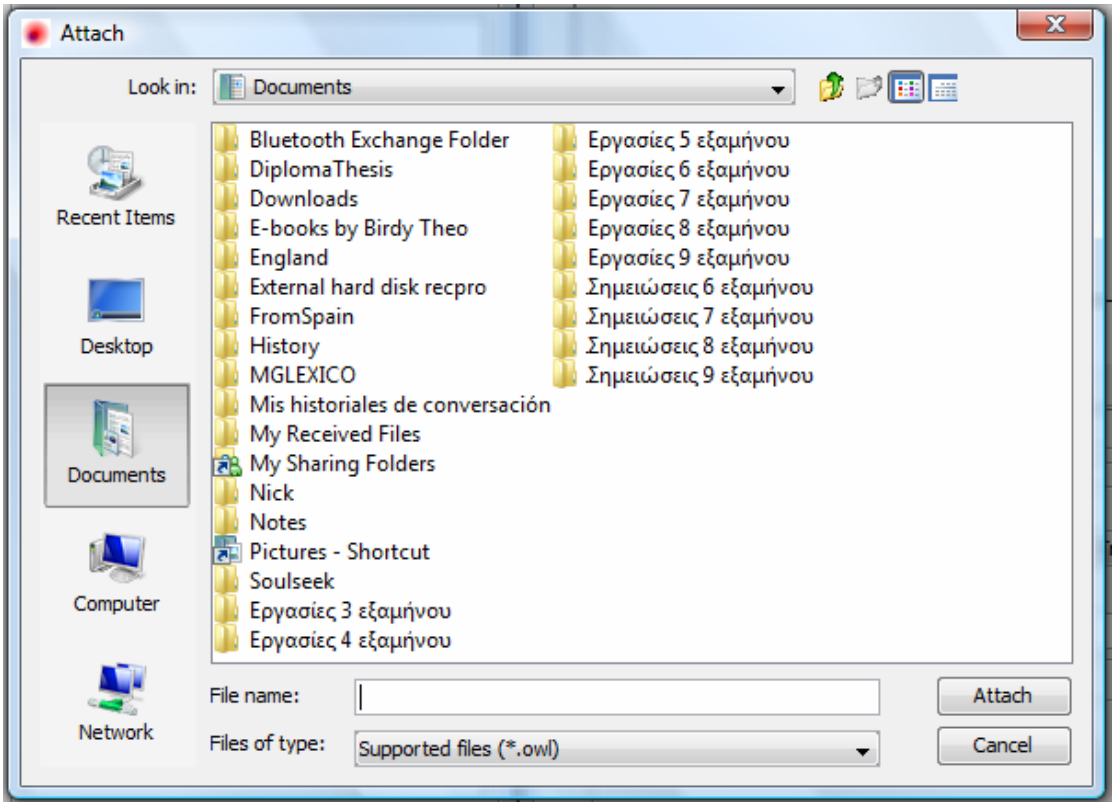
5.4.1 Υποσύστημα ανάγνωσης .owl αρχείου

Ο χρήστης υποδεικνύει στο λογισμικό ποια είναι η οντολογία της οποίας τις ατομικές έννοιες και ρόλους θα χρησιμοποιήσει. Για να το επιτύχει αυτό ακολουθεί την επιλογή Import→ Ontology μέσα από το μενού Import ή εναλλακτικά πιέζει τα πλήκτρα Alt+O, όπως φαίνεται στην παρακάτω εικόνα:

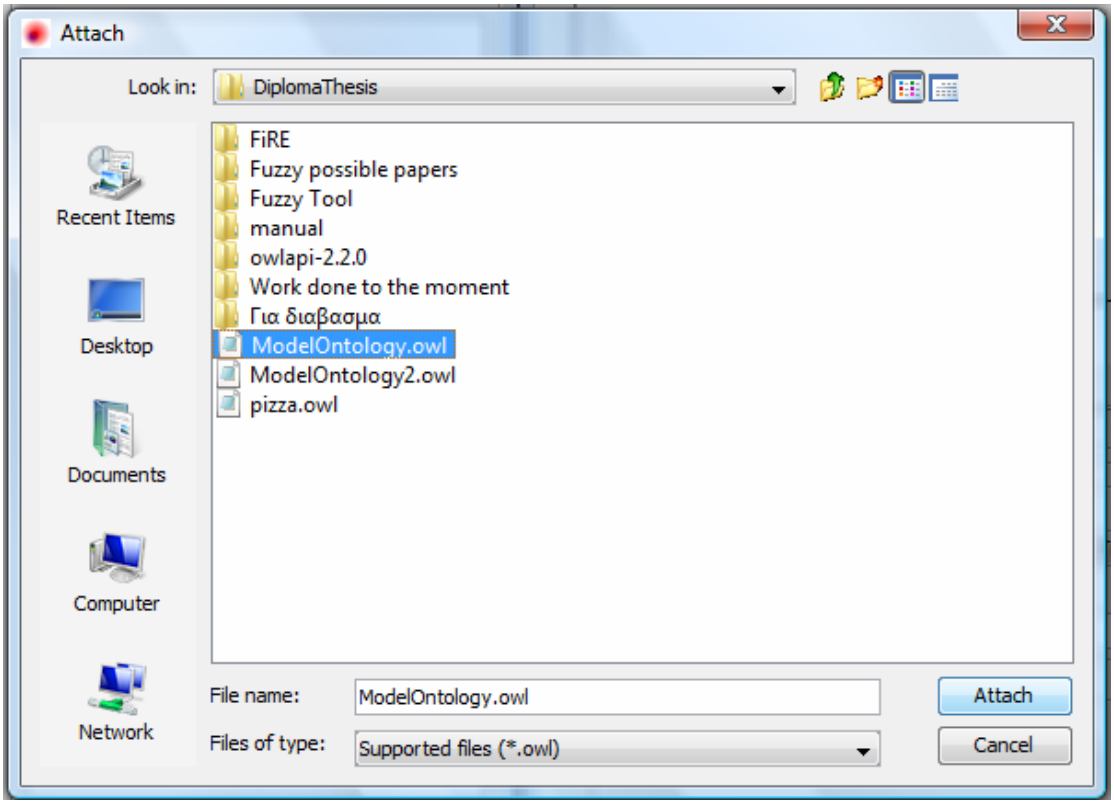


Σχήμα 51. Εντολή για την εισαγωγή οντολογίας από το χρήστη.

Το παράθυρο που εμφανίζεται ως αποτέλεσμα της παραπάνω ενέργειας είναι FileChooser της παρακάτω εικόνας. Με τη βοήθεια του ο χρήστης μπορεί να επισυνάψει το .owl αρχείο που βρίσκεται στο επιθυμητό μέρος. Πρόκειται για έναν file chooser ο οποίος υποστηρίζει μόνο .owl αρχεία, γι'αυτό και στο κύριο παράθυρο του file chooser το μόνο που θα δει να εμφανίζονται είναι φάκελοι και αρχεία με κατάληξη .owl. Επόμενο βήμα είναι η επιλογή του .owl αρχείου και το πάτημα του κουμπιού Attach (ή του πληκτρου enter μετά την επιλογή) προκειμένου να διαβαστεί από το σύστημα, όπως φαίνεται και στο σχήμα 52.

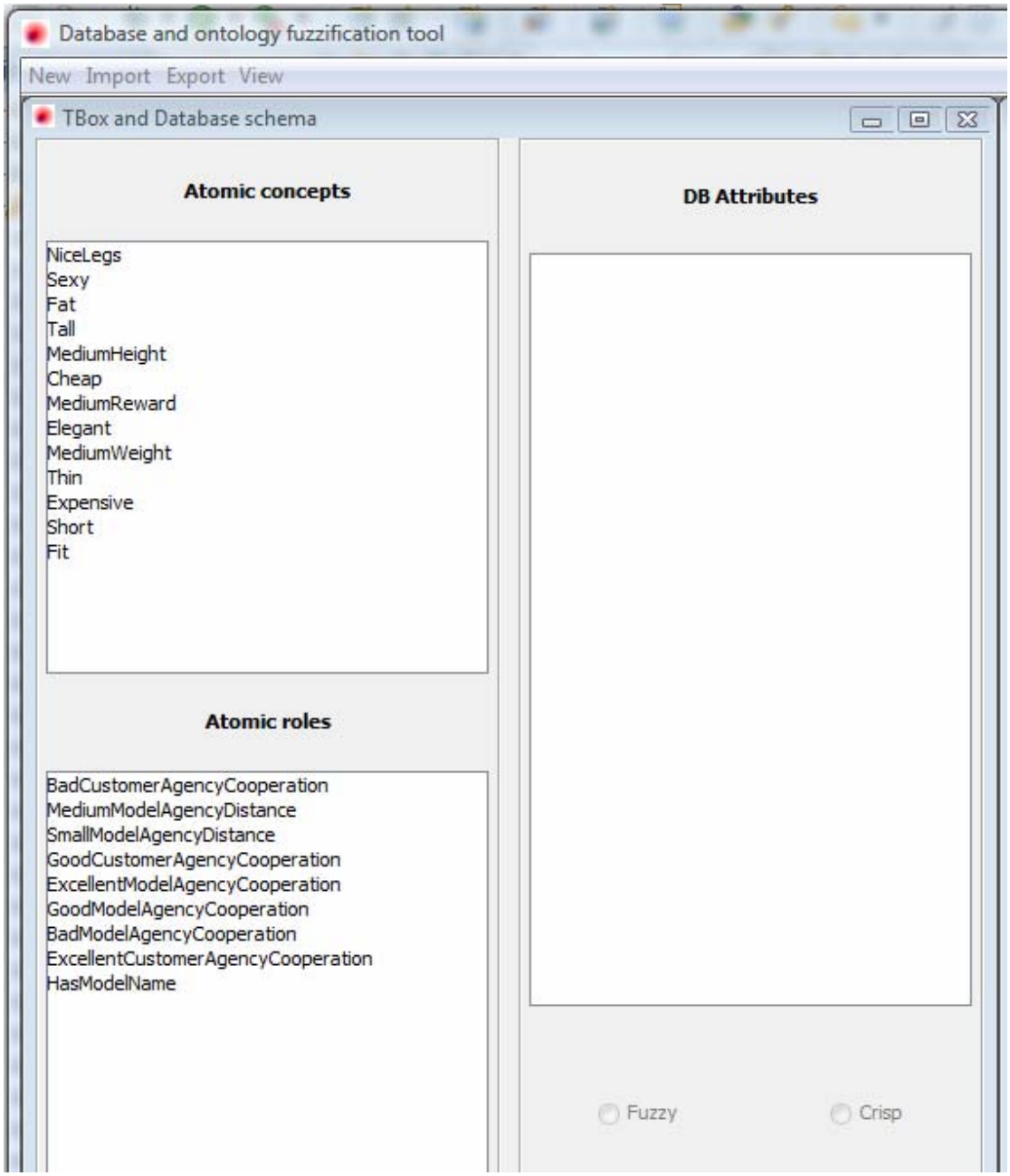


Σχήμα 52. Παράθυρο επιλογής .owl αρχείου



Σχήμα 53. Επιλογή .owl αρχείου

Το σχήμα 54 δίνει τη μορφή που λαμβάνει το interface του συστήματος (αριστερό frame) μετά την εισαγωγή της οντολογίας. Παρατηρούμε ότι στο αριστερό frame οι δύο JLists που φέρουν τον τίτλο Atomic concepts και Atomic Roles έχουν γεμίσει με ονόματα κλάσεων και ονόματα των Data και Object Properties αντίστοιχα. Στο σημείο αυτό να σημειώσουμε ότι οι Data και Object Properties προστίθενται στην ίδια λίστα χωρίς να υπάρχει κάποιο διακριτικό που να ξεχωρίζει τις μεν από τις δε. Όπως ήταν αναμενόμενο πρόκειται για τις ίδιες κλάσεις και ιδιότητες που εμφανίζονται στα σχήματα 45 και 46.

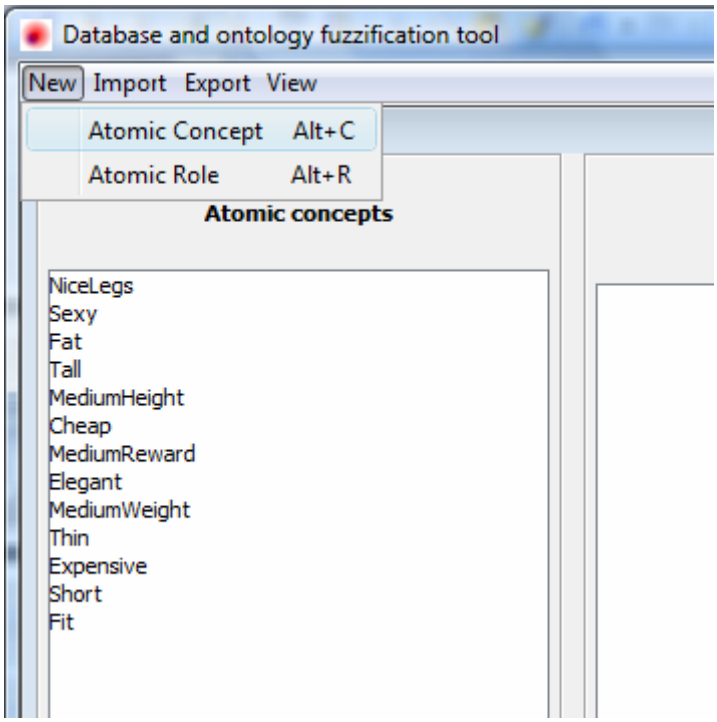


Σχήμα 54. Interface του συστήματος μετά την εισαγωγή της οντολογίας.

5.4.2 Υποσύστημα προσθήκης ατομικών εννοιών

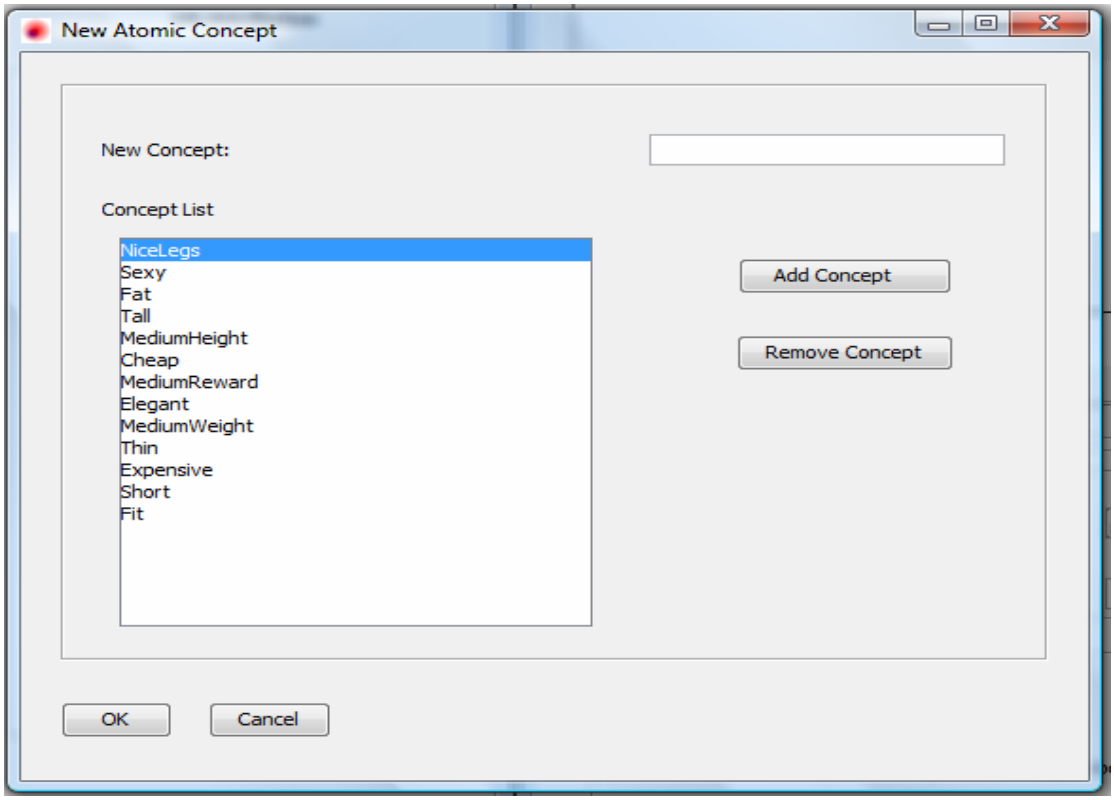
Ωστόσο ο χρήστης δεν είναι υποχρεωμένος να δημιουργήσει ένα .owl αρχείο προκειμένου το σύστημα να εξάγει από αυτό ατομικές έννοιες και ρόλους και να προχωρήσει σε περαιτέρω χρήση του. Έχει τη δυνατότητα να προσθέσει μόνος του τις επιθυμητές έννοιες και ρόλους μέσα από το interface που περιγράφεται παρακάτω. Για την εισαγωγή νέων ατομικών ρόλων

πραγματοποιεί την επιλογή μενού New → Atomic concept ή εναλλακτικά πατά τα πλήκτρα Alt+C.

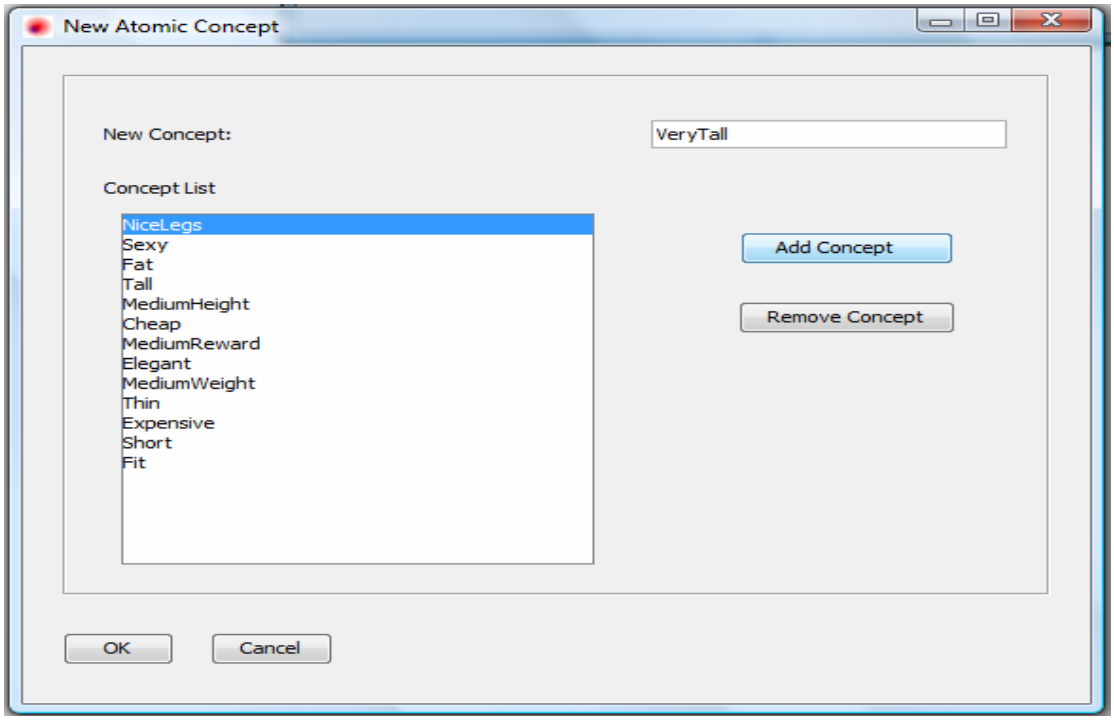


Σχήμα 55. Εντολή για την προσθήκη νέας ατομικής έννοιας από το χρήστη.

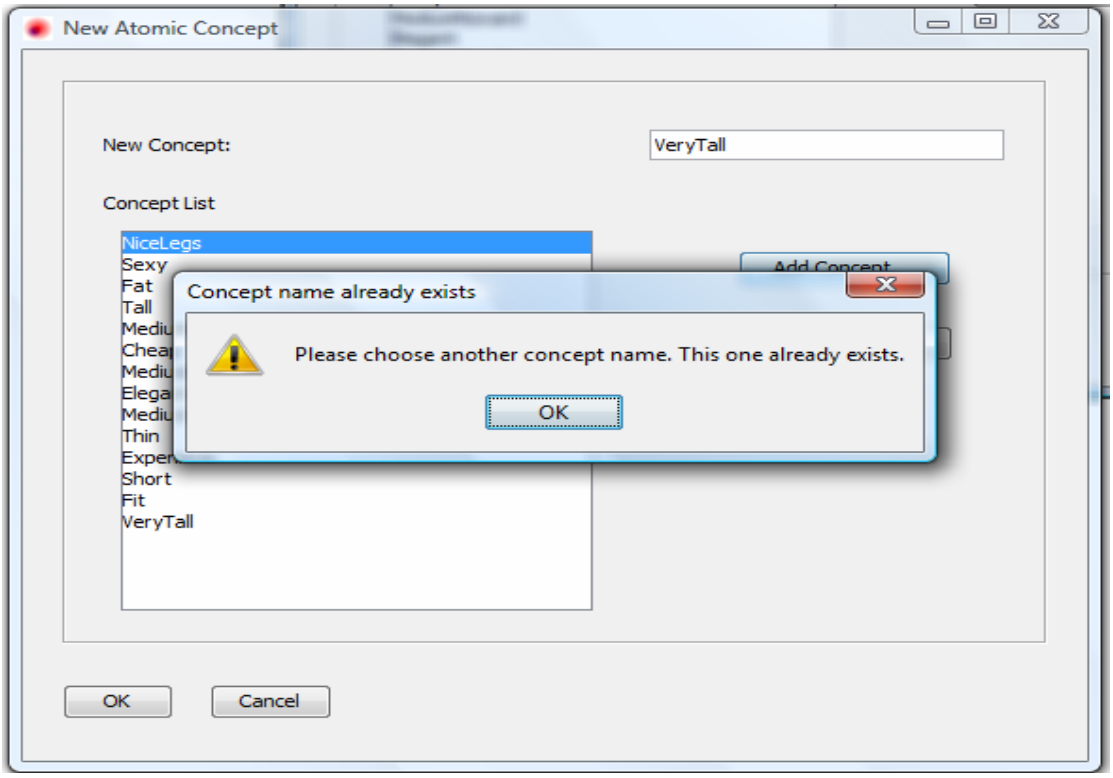
Το frame που εμφανίζεται ως αποτέλεσμα της παραπάνω εντολής είναι αυτό του σχήματος 56. Διαθέτει ένα textfield στο οποίο ο χρήστης εισάγει το όνομα της καινούριας έννοιας που επιθυμεί να προσθέσει, μία λίστα που εμφανίζει τις ήδη υπάρχουσες ατομικές έννοιες καθώς επίσης και δύο κουμπιά που του επιτρέπουν να προσθέτει νέες ή να αφαιρεί ήδη υπάρχουσες ατομικές κλάσεις. Τέλος τα δύο κουμπια OK και Cancel εκτελούν τις αναμενόμενες λειτουργίες, δηλαδή εφαρμόζουν τις αλλαγές που πραγματοποίησε ο χρήστης στη λίστα των ατομικών εννοιών ή απλά κλείνουν το παράθυρο χωρίς να επέρχεται καμία αλλαγή στη λίστα του βασικού interface αντίστοιχα. Όπως φαίνεται και στο σχήμα 57 η προσθήκη μιας ατομικής έννοιας απαιτεί την εισαγωγή του ονόματος της και το πάτημα του "Add concept". Ωστόσο τίθενται κάποιοι περιορισμοί στην προσθήκη νέων εννοιών. Το σύστημα δεν επιτρέπει την προσθήκη νέων εννοιών που διαθέτουν ίδιο όνομα με ήδη υπάρχουσες. Σε μία τέτοια περίπτωση το προειδοποιητικό μήνυμα του σχήματος 58 εμφανίζεται. Επιπλέον το σύστημα δεν επιτρέπει την εισαγωγή ονόματος ατομικής έννοιας με κενά όπως δείχνει το σχήμα 59.



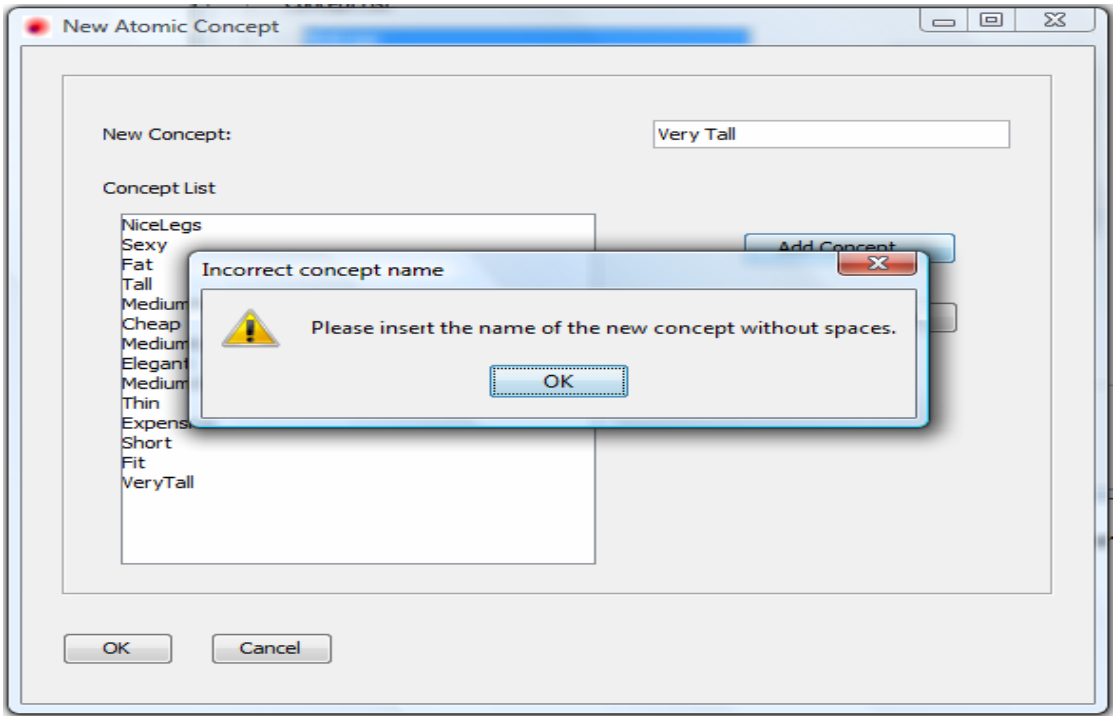
Σχήμα 56. Interface προσθήκης ατομικών εννοιών.



Σχήμα 57. Προσθήκη ατομικής έννοιας.

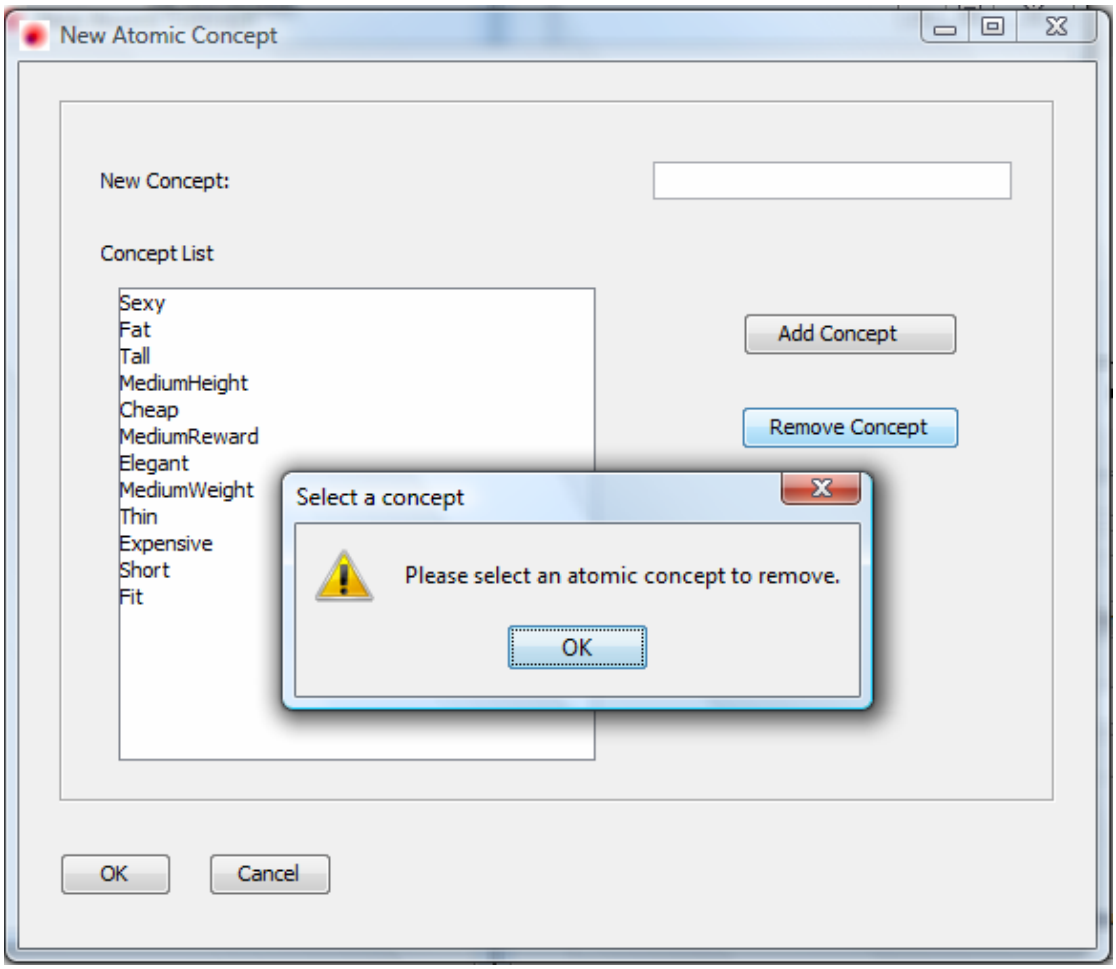


Σχήμα 58. Προειδοποιητικό μήνυμα για ήδη υπάρχον όνομα ατομικής έννοιας.



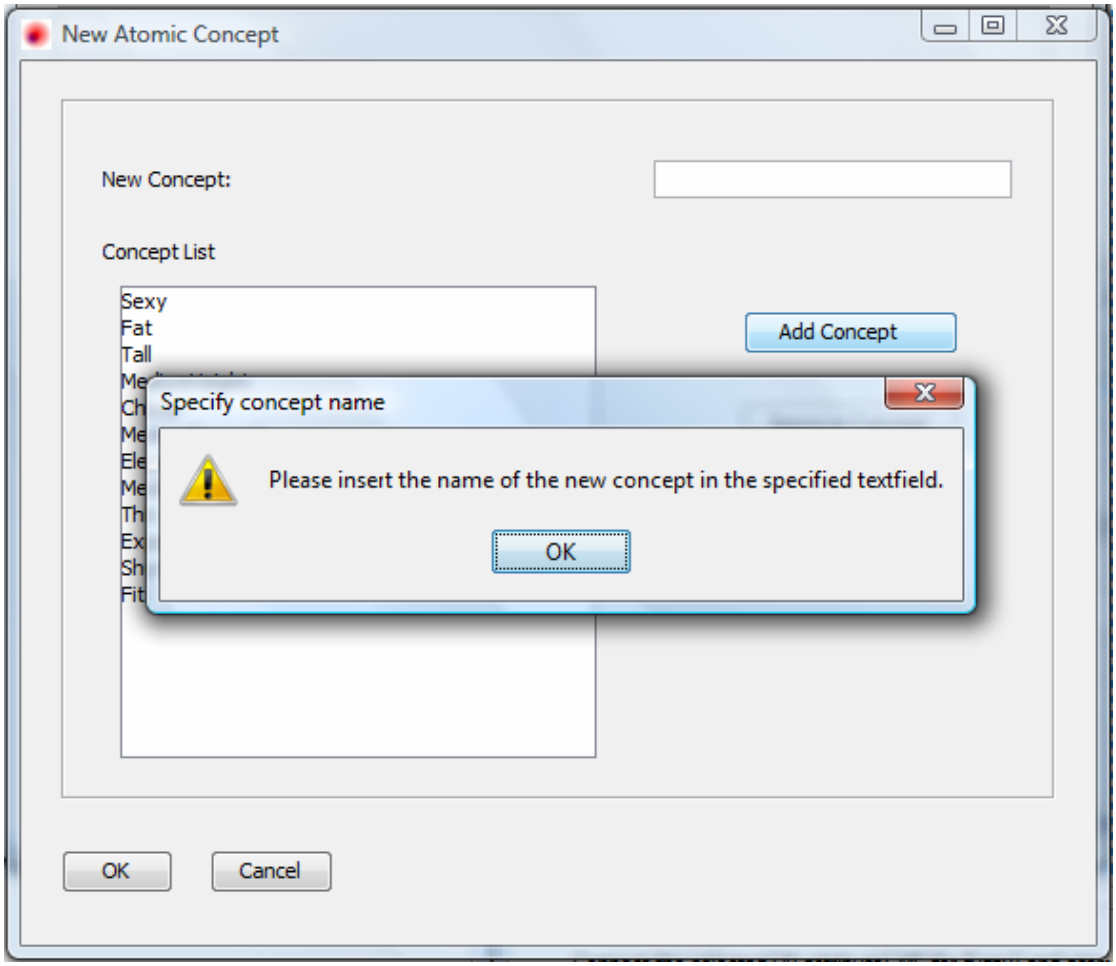
Σχήμα 59. Προειδοποιητικό μήνυμα για εισαγωγή ονόματος ατομικής έννοιας με κενά.

Το σύστημα προσφέρει τη δυνατότητα αφαίρεσης κάποιας από τις ήδη υπάρχουσες ατομικές έννοιες. Ωστόσο πρέπει πρώτα απαραίτητα ο χρήστης να έχει επιλέξει κάποια ατομική έννοια αλλιώς το προειδοποιητικό μήνυμα του σχήματος 60 κάνει την εμφάνιση του.



Σχήμα 60. Προειδοποιητικό μήνυμα για αφαίρεση ατομικής έννοιας που δεν έχει επιλεχτεί.

Ακόμη, όπως φαίνεται και στο σχήμα 61 είναι απαραίτητο ο χρήστης να προσδιορίσει ένα όνομα ατομικής έννοιας αλλιώς ένα προειδοποιητικό μήνυμα τον πληροφορεί γι'αυτό. Τέλος ο χρήστης με το πάτημα του "OK" εφαρμόζει τις προσθαφαιρέσεις εννοιών που πραγματοποίησε στην κύρια λίστα εννοιών του interface του συστήματος. Εάν πάλι επιθυμεί να αφήσει ανέπαφη την κύρια λίστα εννοιών πατά "Cancel" το οποίο κλείνει το interface προσθήκης εννοιών χωρίς τροποποιήσεις στο κύριο interface.

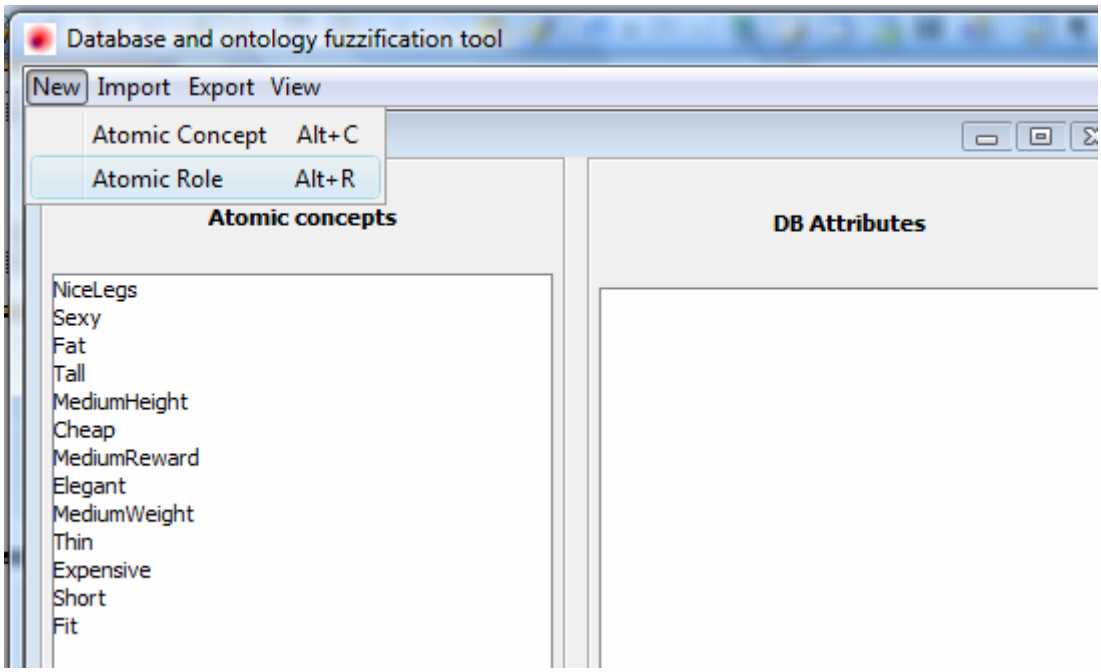


Σχήμα 61. Προειδοποιητικό μήνυμα για προσθήκη ατομικής έννοιας με κενό όνομα.

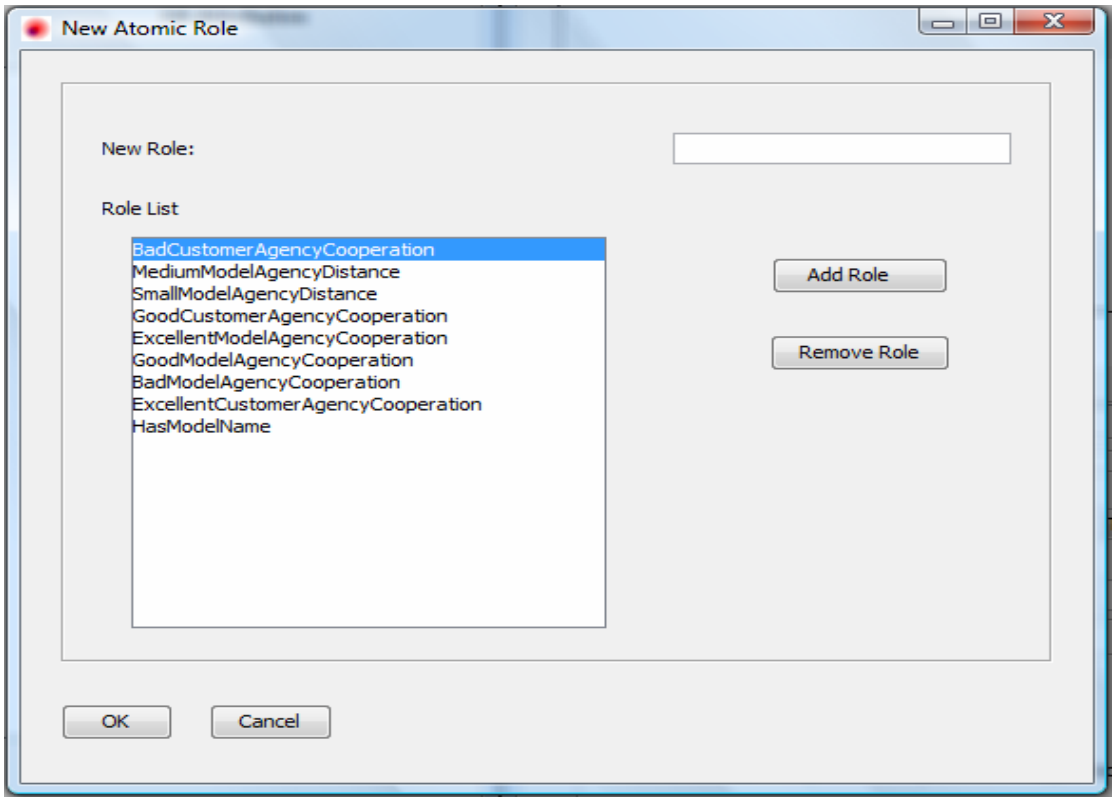
5.4.3 Υποσύστημα προσθήκης ατομικών ρόλων

Το υποσύστημα προσθήκης ατομικών ρόλων διαθέτει ακριβώς τον ίδιο σχεδιασμό και λειτουργικότητα, γι'αυτό και θα αναφερθούμε περισσότερο συνοπτικα σε αυτό. Η εντολή New→ Atomic Role (ή το πάτημα των πλήκτρων Alt+R) όπως δείχνει το σχήμα 62, εμφανίζει το interface προσθήκης ατομικών ρόλων του σχήματος 63. Η προσθήκη του ατομικού ρόλου γίνεται με τον ορισμό του ονόματος και το πάτημα του κουμπιού (σχήμα 64) ενώ περιορισμοί τίθενται για προσθήκη ατομικού ρόλου που διαθέτει ίδιο όνομα με κάποιον από τους ήδη υπάρχοντες (σχήμα 65) ή όνομα με κενά (σχήμα 66). Δεν επιτρέπεται η αφαίρεση ατομικού ρόλου που δεν έχει προσδιορισθεί (σχήμα 67) ούτε και η προσθήκη

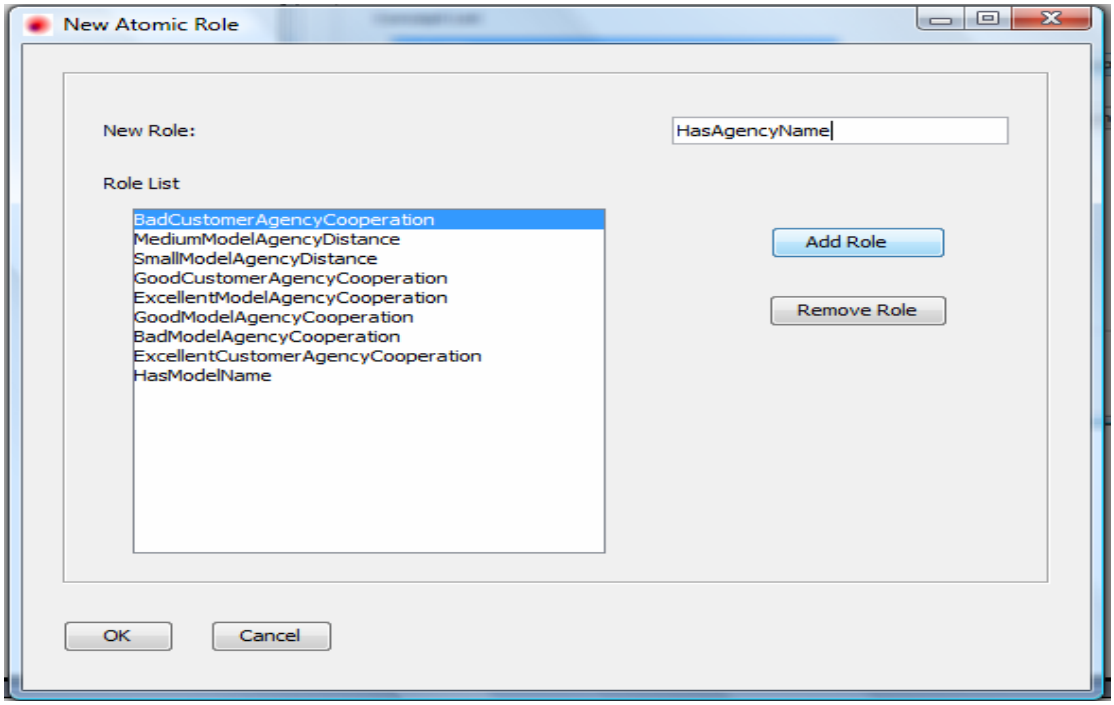
ατομικού ρόλου με κενό όνομα (σχήμα 68). Τέλος το "OK" πραγματοποιεί επιστροφή στο υπάρχον interface εφαρμόζοντας τις αλλαγές ενώ το "Cancel" χωρίς να τις εφαρμόζει.



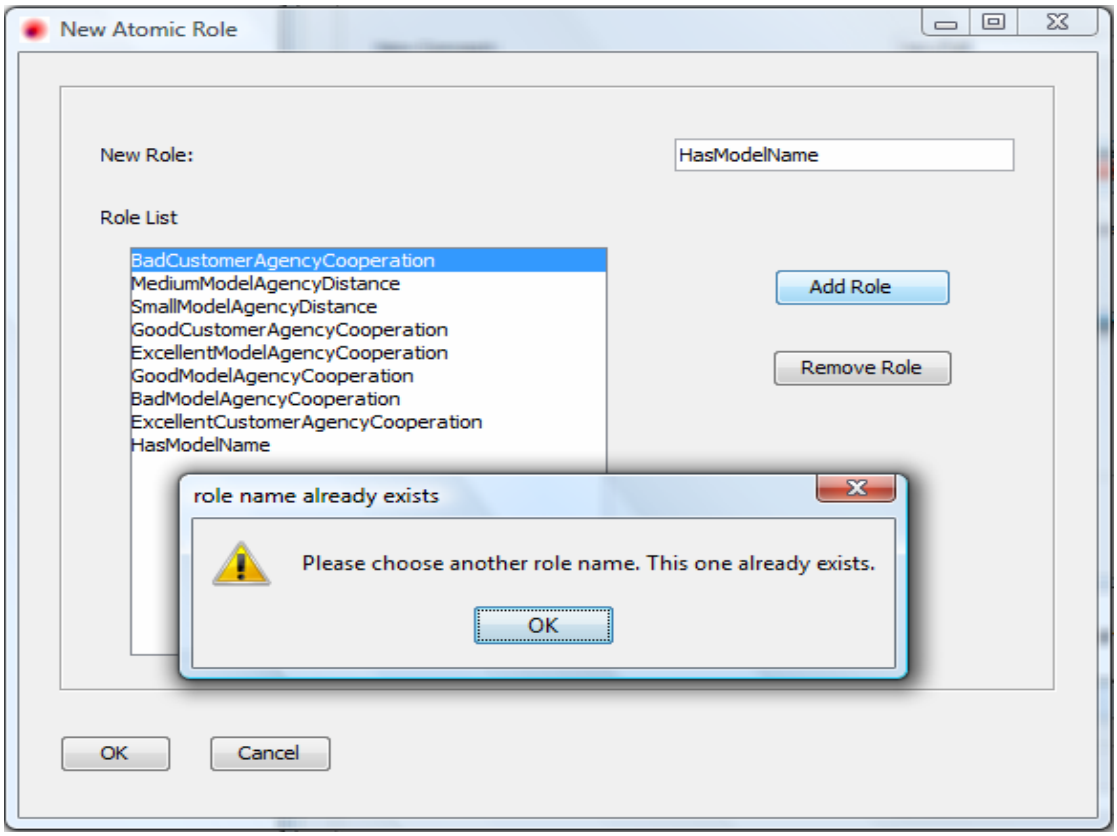
Σχήμα 62. Εντολή για την προσθήκη νέου ατομικού ρόλου από το χρήστη.



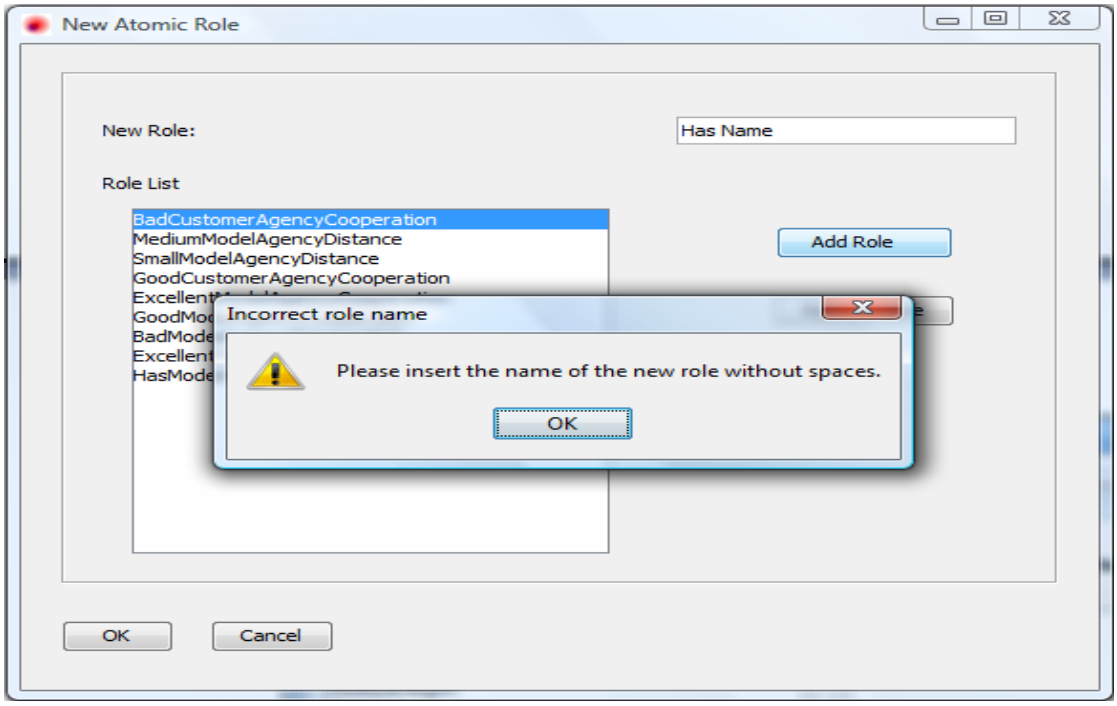
Σχήμα 63. Interface προσθήκης ατομικών ρόλων.



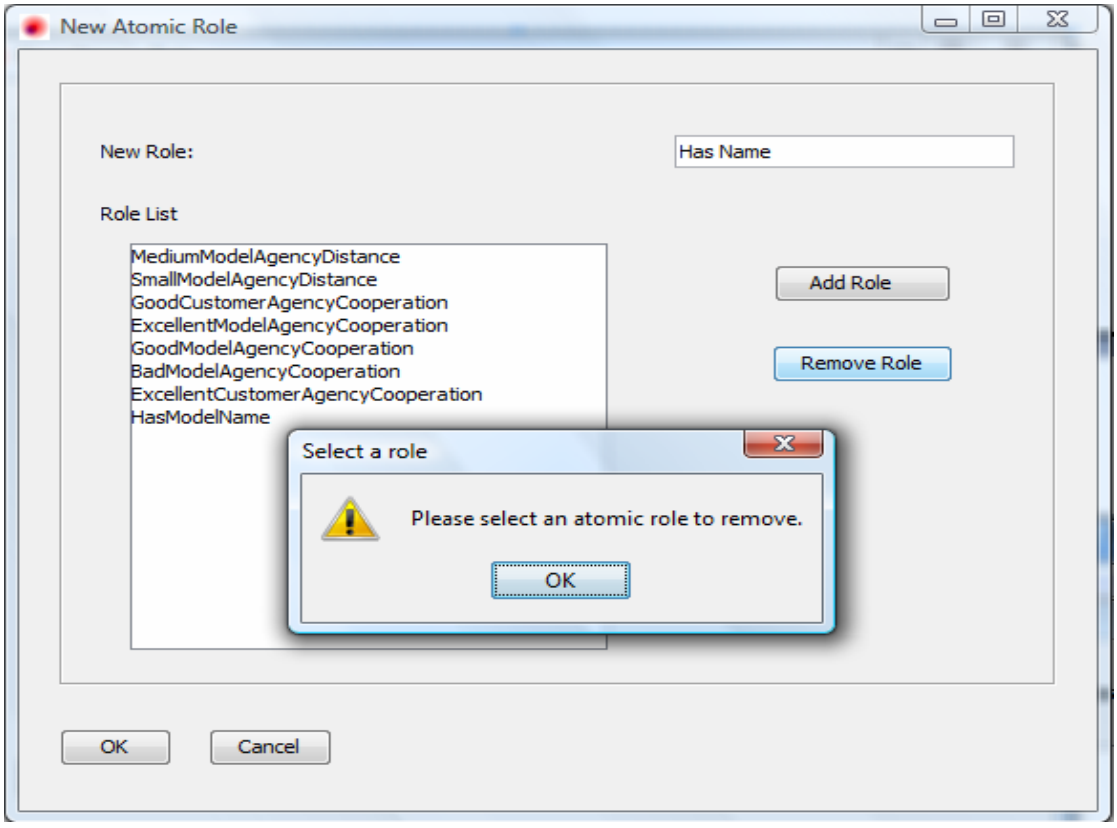
Σχήμα 64. Προσθήκη ατομικού ρόλου.



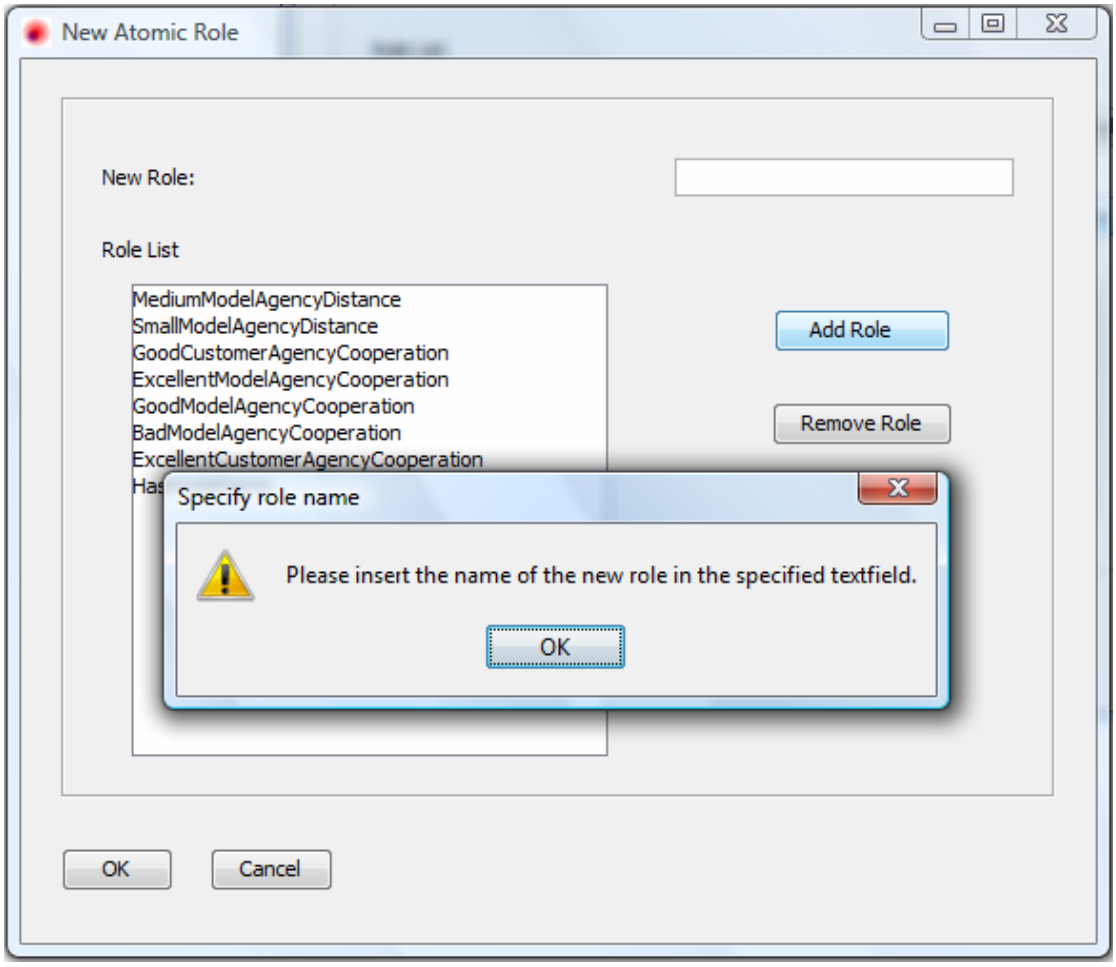
Σχήμα 65. Προειδοποιητικό μήνυμα για ήδη υπάρχον όνομα ατομικού ρόλου.



Σχήμα 66. Προειδοποιητικό μήνυμα για εισαγωγή ονόματος ατομικού ρόλου με κενά.



Σχήμα 67. Προειδοποιητικό μήνυμα για αφαίρεση ατομικού ρόλου που δεν έχει επιλεχτεί.



Σχήμα 68. Προειδοποιητικό μήνυμα για προσθήκη ατομικού ρόλου με κενό όνομα.

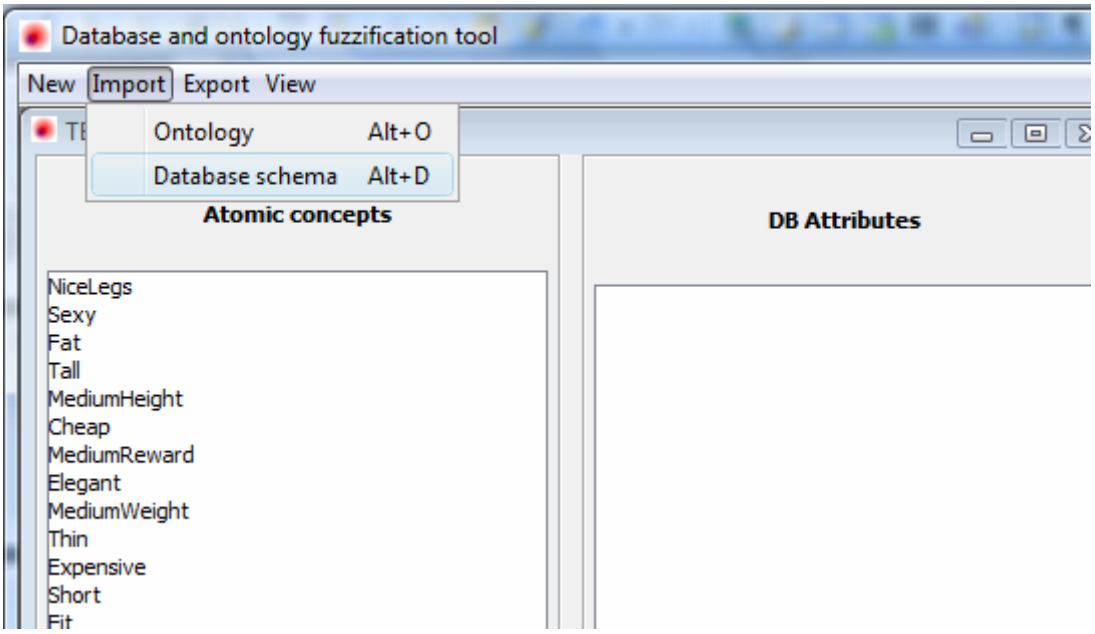
Συνοψίζοντας παρατηρούμε ότι το σύστημα προσφέρει στο χρήστη δύο δυνατότητες για εισαγωγή ατομικών εννοιών και ρόλων: είτε να τις εισάγει μέσω ενός .owl αρχείου και στη συνέχεια να τροποποιήσει τις λίστες όπως αυτός επιθυμεί είτε να εισάγει εξ'ολοκλήρου τις ατομικές έννοιες και ρόλους με το παραπάνω interface.

5.4.4 Υποσύστημα διασύνδεσης με mysql βάση δεδομένων

Προκειμένου ο χρήστης να χρησιμοποιήσει το σύστημα θα πρέπει απαραίτητα να διασυνδεθεί με μία βάση δεδομένων. Με τον τρόπο αυτό αφενός μεν θα αποκτήσει πρόσβαση στο σχήμα της βάσης αποφασίζοντας ποιο απο τα ιδιοχαρακτηριστικά θα επιλέξει για την παραγωγή των ασφών ισχυρισμών, αφετέρου δε προσφέρει στο σύστημα τα χρειαζούμενα στοιχεία για να

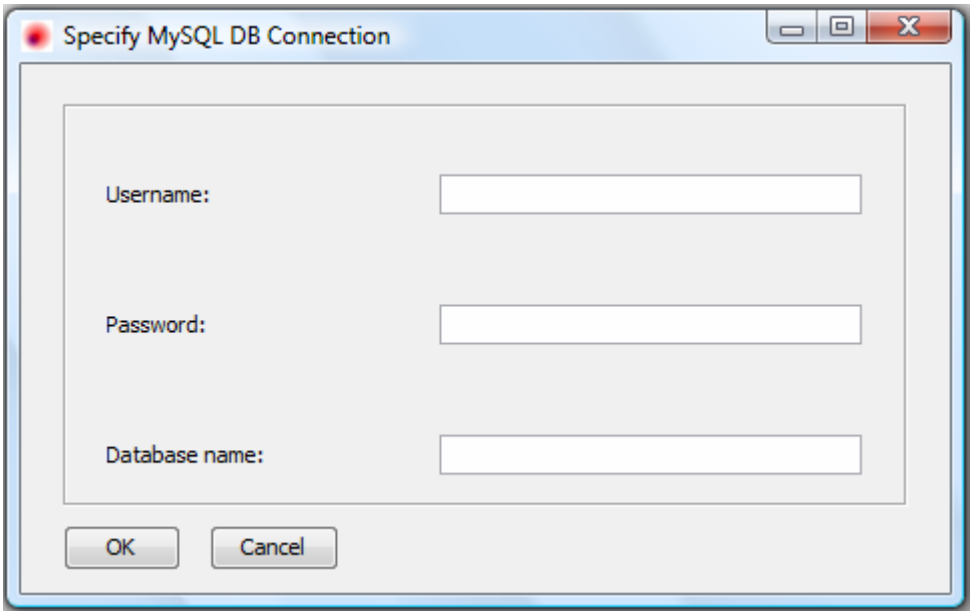
επικοινωνήσει με τη βάση δεδομένων και να ανακτήσει τις απαραίτητες πλειάδες για την παραγωγή των ασαφών ισχυρισμών.

Η εντολή για τη διασύνδεση με τη βάση δεδομένων δίνεται με την επιλογή μενού Import→Database schema ή με το πάτημα των πλήκτρων Alt+D όπως δείχνει το σχήμα 69.



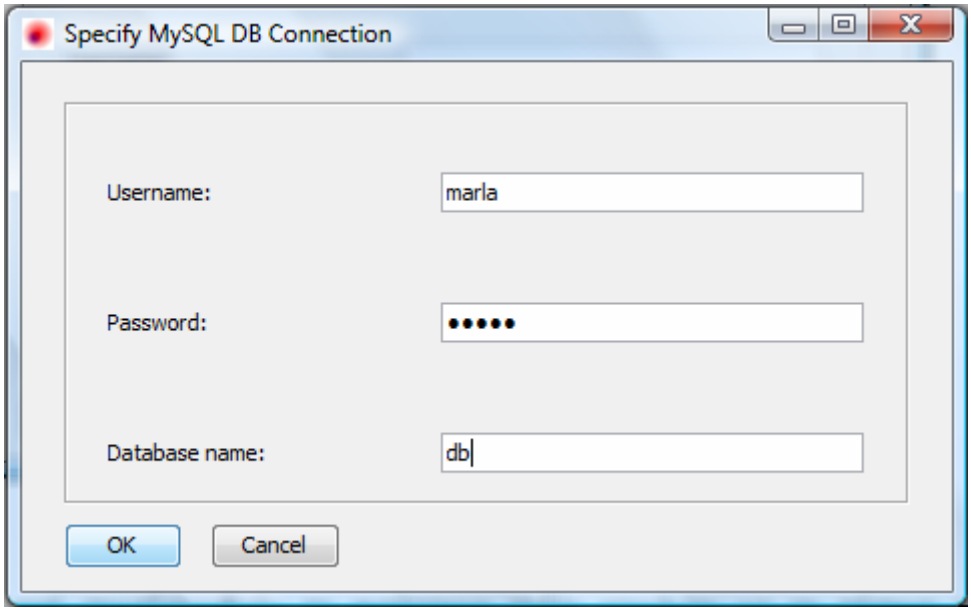
Σχήμα 69. Εντολή για διαδύνδεση του συστήματος με τη βάση δεδομένων.

Η εκτέλεση της παραπάνω εντολής εμφανίζει το interface του σχήματος 70, στο οποίο ο χρήστης καλείται να δώσει το username και password της mysql σύνδεσης που θα πραγματοποιήσει το πρόγραμμα, καθώς επίσης και το όνομα της βάσης δεδομένων της οποίας το σχήμα θα διαβαστεί. Σε περίπτωση που θα γίνει διασύνδεση με mysql account ο οποίος δε διαθέτει κωδικό αρκεί να μείνει κενό το αντίστοιχο πεδίο κειμένου.

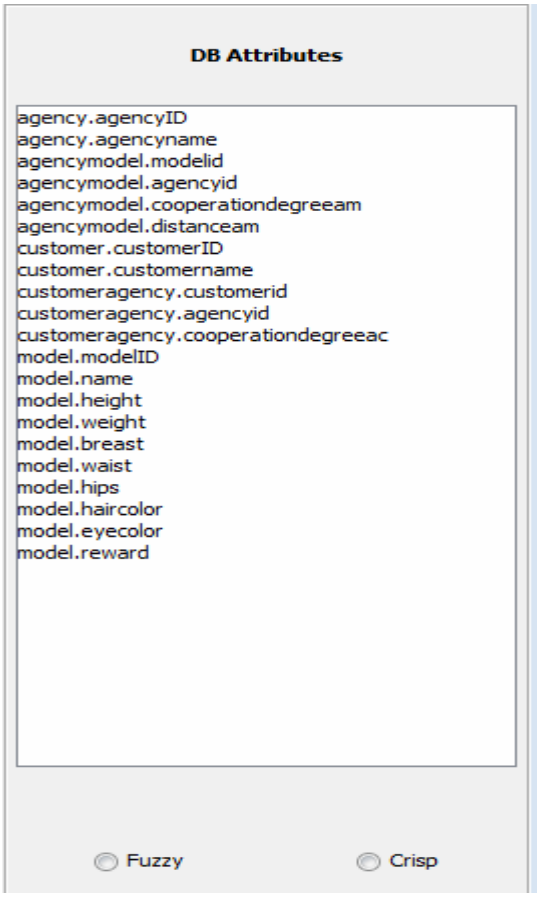


Σχήμα 70. Interface διασύνδεσης με τη βάση δεδομένων

Αφού συμπληρωθούν τα αντίστοιχα πεδία κειμένου με το πάτημα του κουμπιού "OK" ή απλά με το πάτημα του πλήκτρου enter (σχήμα 71) γίνεται η διασύνδεση και εμφανίζεται στη λίστα "DB Attributes" του κύριου interface το σύνολο των ιδιοχαρακτηριστικών της βάσης που έχει ζητήσει ο χρήστης (σχήμα 72).

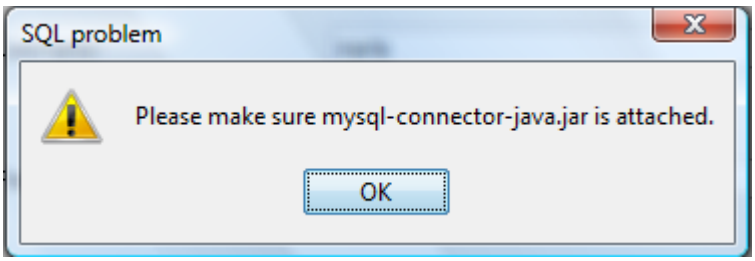


Σχήμα 71. Διασύνδεση με τη βάση δεδομένων από το χρήστη.

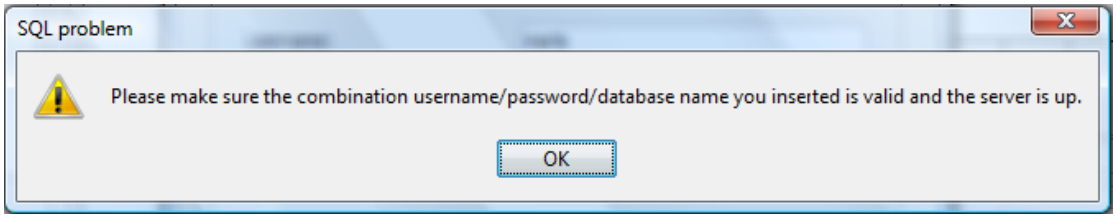


Σχήμα 72. Η λίστα DB Attributes μετά την εισαγωγή του σχήματος της βάσης δεδομένων

Επιπλέον το σύστημα χειρίζεται μία σειρά παθολογικών περιπτώσεων όπως το να μην έχει συμπεριληφθεί το αντίστοιχο mysql jar στις χρησιμοποιούμενες βιβλιοθήκες (σχήμα 73) ή σε περίπτωση που ο σύνδυασμός username/password/database που ο χρήστης έχει δώσει να μην επιτυγχάνει διασύνδεση με τη βάση δεδομένων (σχήμα 74). Ο τελευταίος έλεγχος περιλαμβάνει και την περίπτωση να μην είναι ενεργός ο mysql server στον υπολογιστή που εκτελείται η εφαρμογή.



Σχήμα 73. Προειδοποιητικό μήνυμα σε περίπτωση απουσίας των mysql κλάσεων.

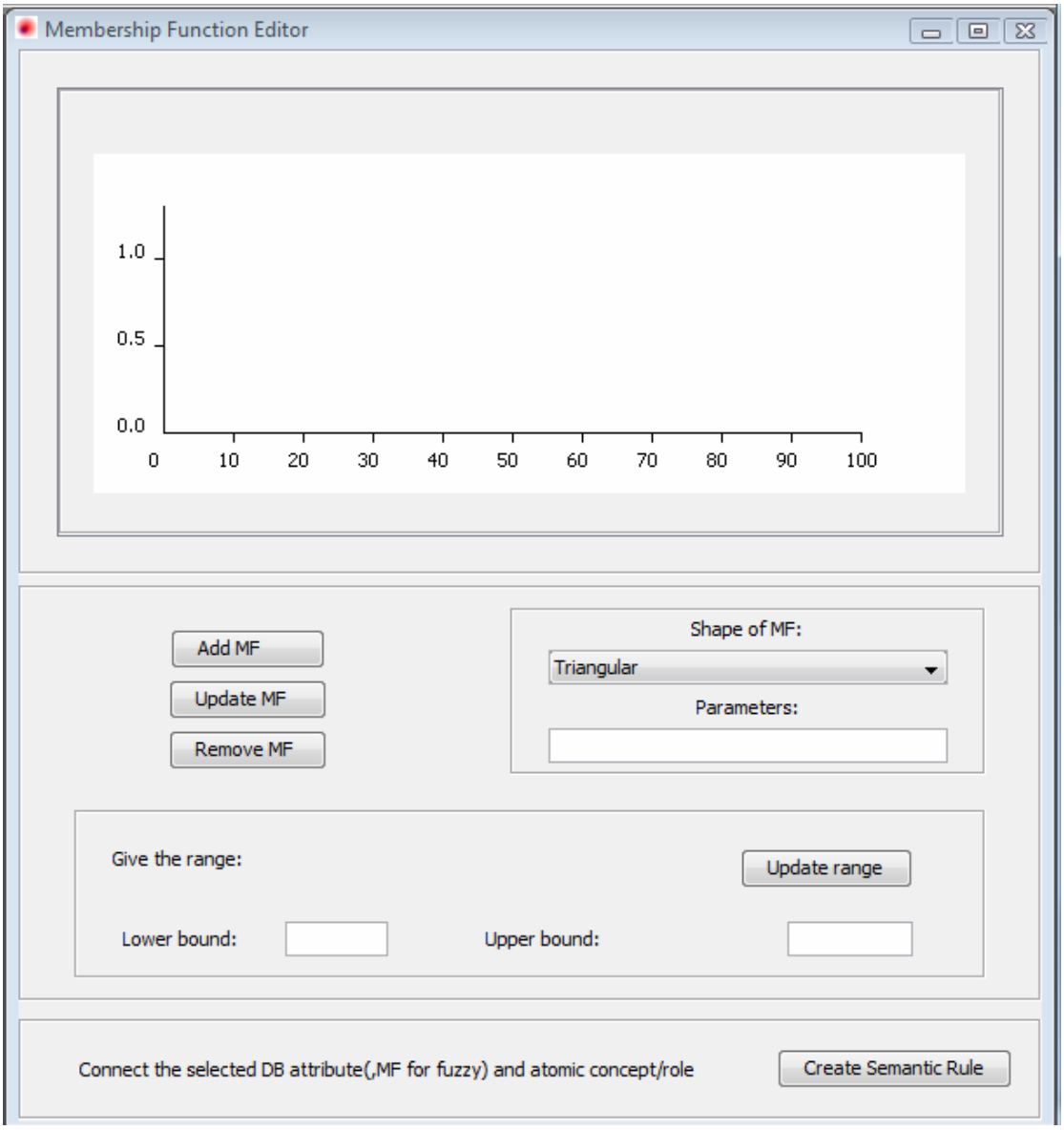


Σχήμα 74. Προειδοποιητικό μήνυμα σε περίπτωση λάθος συνδυασμού username/password/database από το χρήστη ή απουσίας server.

Στο σημείο αυτό να παρατηρήσουμε ότι συγκρίνοντας τα σχήματα 50 και 72 βλέπουμε τα radio buttons fuzzy και crisp απενεργοποιημένα και ενεργόποιημένα αντίστοιχα. Ο λόγος που συμβαίνει αυτό είναι ότι προτού κάνει ο χρήστης την αρχική διασύνδεση με τη βάση δεδομένων για την ανάκτηση του σχήματος της βάσης, ένα βήμα που είναι βασικής σημασίας, ο χρήστης δεν μπορεί να επιλέξει κάποιο από τα δύο radio buttons εφόσον δεν υπάρχει κάποιο DB Attribute στο οποίο αυτά θα αναφέρονται. Επιπλέον ο χρήστης μπορεί να ελέγξει ότι είναι αντίστοιχα απενεργοποιημένες οι επιλογές μενού Export→ABox, View→ABox και View→Semantic Rules πριν το γέμισμα της λίστας DB Attributes με ένα σχήμα βάσης δεδομένων επειδή ακριβώς οι παραπάνω ενέργειες έχουν ως προαπαιτούμενο τη διασύνδεση με τη βάση δεδομένων. Επιπλέον να επισημάνουμε ότι αυτό που απαιτείται για την ενεργοποίηση των παραπάνω εντολών δεν είναι απλά η εκτέλεση της εντολής Import→Database schema αλλά η επιτυχημένη διασύνδεση με τη βάση δεδομένων η οποία δεν περιλαμβάνει κάποια απο τα παθολογικά σενάρια που εξετάστηκαν παραπάνω.

5.4.5 Υποσύστημα επεξεργασίας και προβολής MF

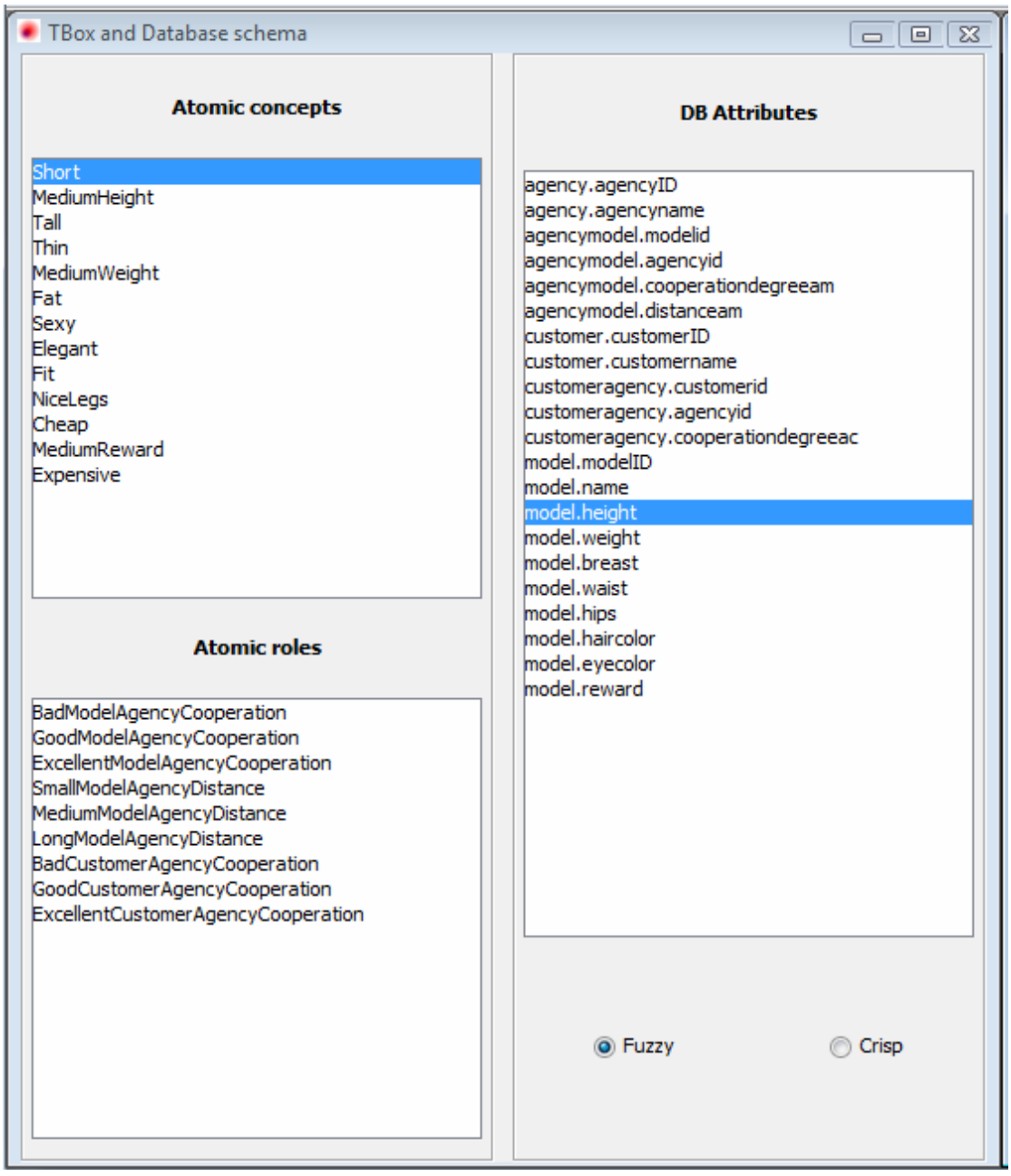
Το σύστημα προσφέρει στο χρήστη τη δυνατότητα ορισμού συναρτήσεων συμμετοχής και γραφικής αναπαράστασης τους. Το κομμάτι το οποίο σχετίζεται λειτουργικά με αυτή τη δυνατότητα είναι το δεξί frame (σχήμα 75) το οποίο με ένα σύνολο από λειτουργικά γραφικά components που περιλαμβάνει επιτρέπει στο χρήστη να ορίζει τις επιθυμητές MF (membership functions, συναρτήσεις συμμετοχής).



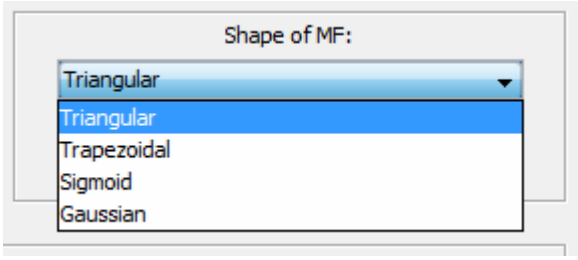
Σχήμα 75. Interface για τη δημιουργία MF.

Πριν ο χρήστης εισάγει μία MF θα πρέπει προηγούμενως να έχει επιλέξει ένα DB Attribute (από αυτά που έχει εισάγει με τον παραπάνω τρόπο) καθώς επίσης και το radio button "Fuzzy" που αντιστοιχεί σε αυτό το ιδιοχαρακτηριστικό της βάσης (σχήμα 76, δεξί μισό). Εάν είναι επιλεγμένο το radio button "Crisp" δε έχει νόημα η συσχέτιση κάποιας MF με το συγκεκριμένο DB Attribute καθώς θα χρησιμοποιηθεί για την παραγωγή crisp assertions τα οποία δεν περιέχουν αριθμητικούς συντελεστές για την παραγωγή των οποίων χρησιμοποιούνται οι MF. Μετά την επιλογή του επιθυμητού DB Attribute και του "Fuzzy" radio button ο χρήστης που θέλει να συσχετίσει μία MF με αυτό:

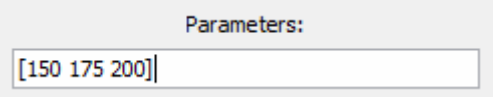
- επιλέγει τον τύπο της MF από την droplist επιλογών που επισημαίνεται με την ετικέτα “Shape of MF:” (σχήμα 77)
- εισάγει τις παραμέτρους της MF στο πεδίο κειμένου που επισημαίνεται από την ετικέτα “Parameters:” (σχήμα 78)
- εισάγει το κάτω και πάνω όριο στα πεδία κειμένου που βρίσκονται δεξιά από τις ταμπέλες “Lower bound:” και “Upper bound:” αντίστοιχα (σχήμα 79)
- πιέζει το κουμπί “Add MF” (σχήμα 80).



Σχήμα 76. Επιλογή του DB Attribute “model.height” πριν την προσθήκη MF γι’αυτό.

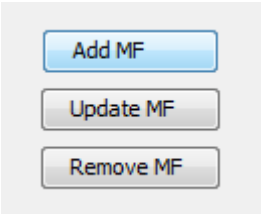


Σχήμα 77. Καθορισμός του σχήματος της MF.



Σχήμα 78. Καθορισμός των παραμέτρων της MF.

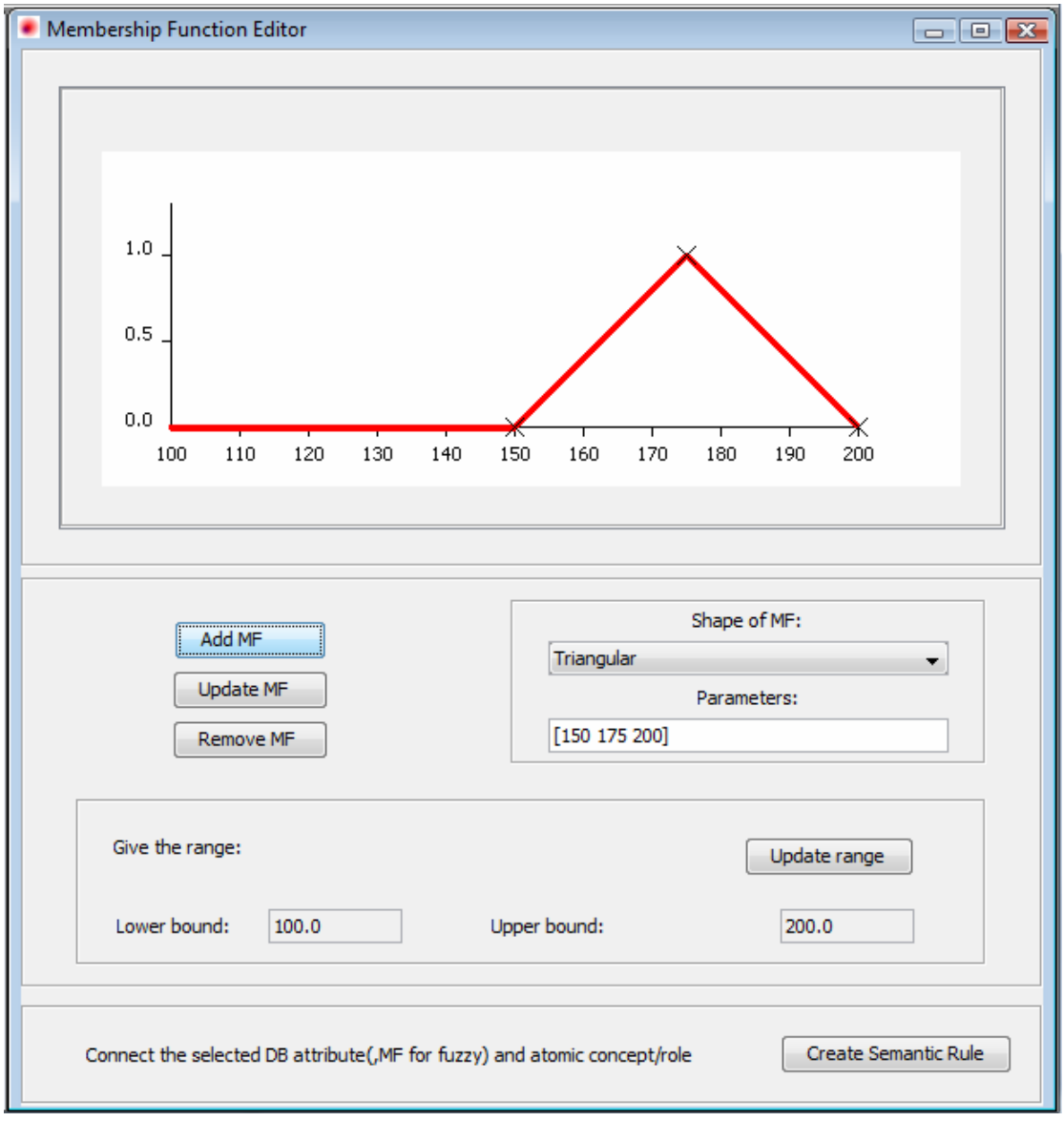
Σχήμα 79. Καθορισμός του κάτω και άνω ορίου προβολής της MF.



Σχήμα 80. Πάτημα κουμπιού για προσθήκη MF.

Το παραπάνω παράδειγμα απεικονίζει τη διαδικασία προσθήκης μιας MF για το DB Attribute Model.height, τριγωνικού σχήματος, με παραμέτρους 150, 175 και 200 (η σημασία των οποίων θα φανεί στη συνέχεια) και κάτω και άνω όριο προβολής 100 και 200 αντίστοιχα. Το σχήμα 81 δείχνει το αποτέλεσμα της παραπάνω προσθήκης.

Ωστόσο κάθε φορά που πατιέται το κουμπί “Add MF” το σύστημα εκτελεί ένα σύνολο από ελέγχους προκειμένου να βεβαιωθεί ότι η MF θα προστεθεί με συνεπή τρόπο. Οι περιορισμοί αυτοί είναι δύο ειδών: πρόκειται για τους γενικούς περιορισμούς που ελέγχονται κάθε φορά ανεξαρτήτως του σχήματος της MF που έχει επιλεχθεί και τους περιορισμούς που σχετίζονται με το σχήμα της MF που προορίζεται για προσθήκη.

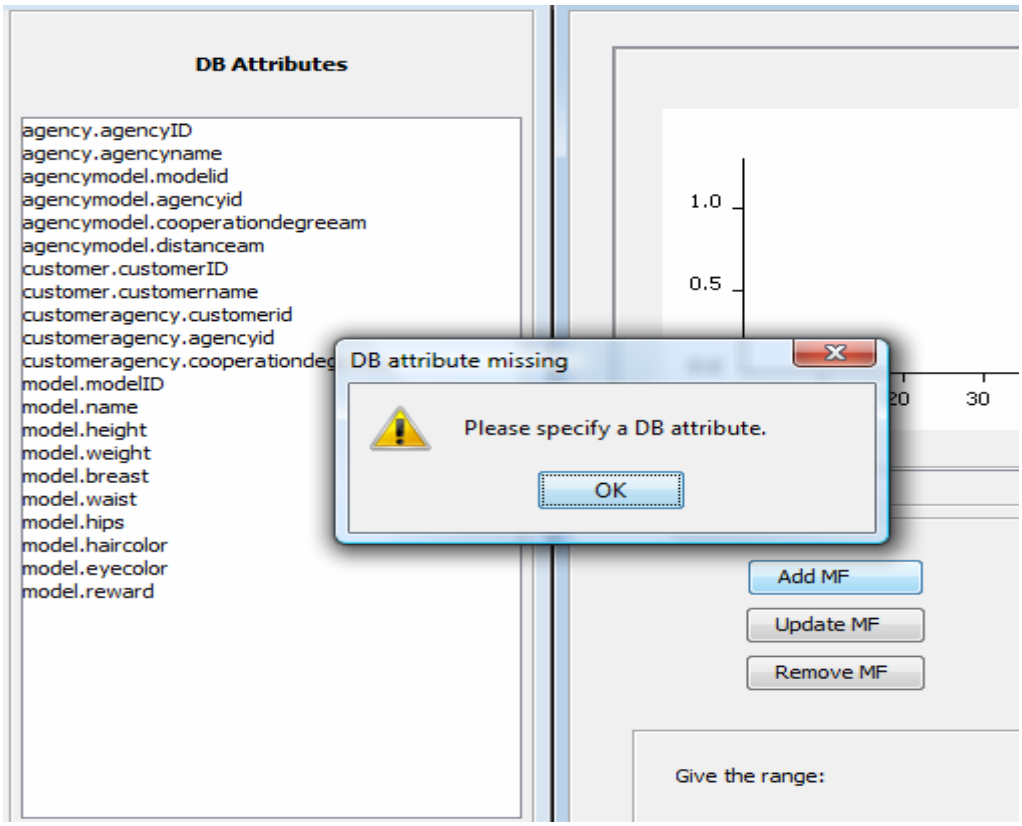


Σχήμα 81. Προσθήκη της MF με τις απεικονιζόμενες παραμέτρους.

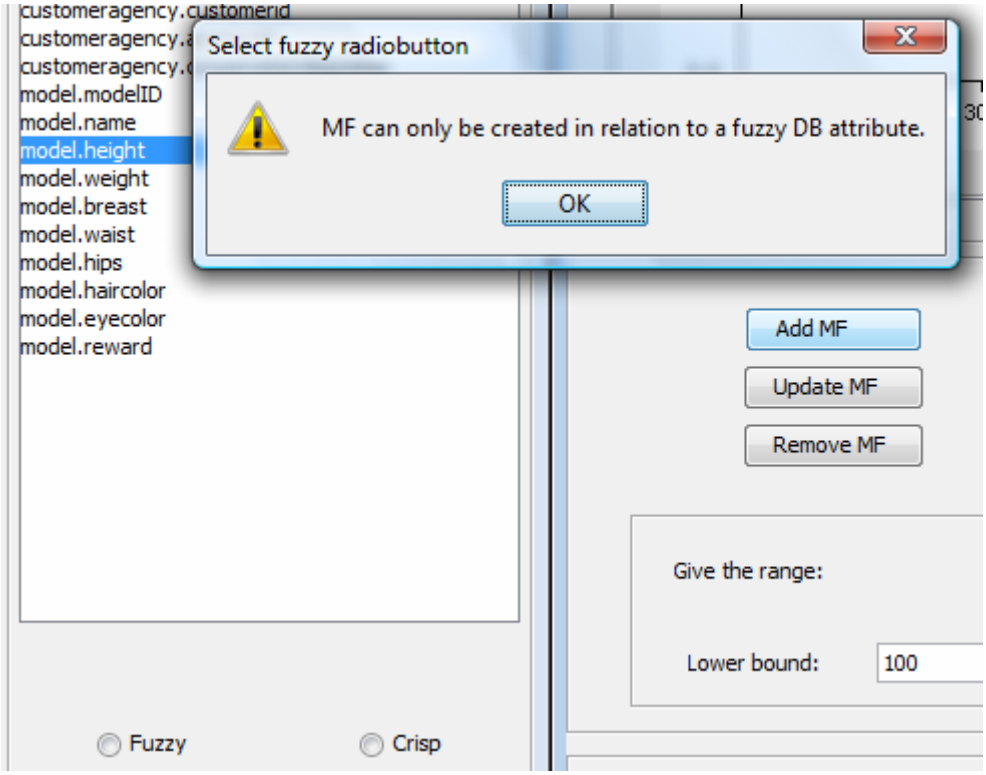
Σε περίπτωση που ο χρήστης προχωρήσει σε διαδικασία προσθήκης MF χωρίς προηγούμενως να έχει προσδιορίσει κάποιο DB Attribute εμφανίζεται το προειδοποιητικό μήνυμα του σχήματος 82.

Εάν πάλι ο χρήστης επιλέξει ένα DB Attribute αλλά έχει ξεχάσει να προσδιορίσει ότι θα παραχθούν fuzzy assertions από αυτό, παίρνει το προειδοποιητικό μήνυμα του σχήματος 83.

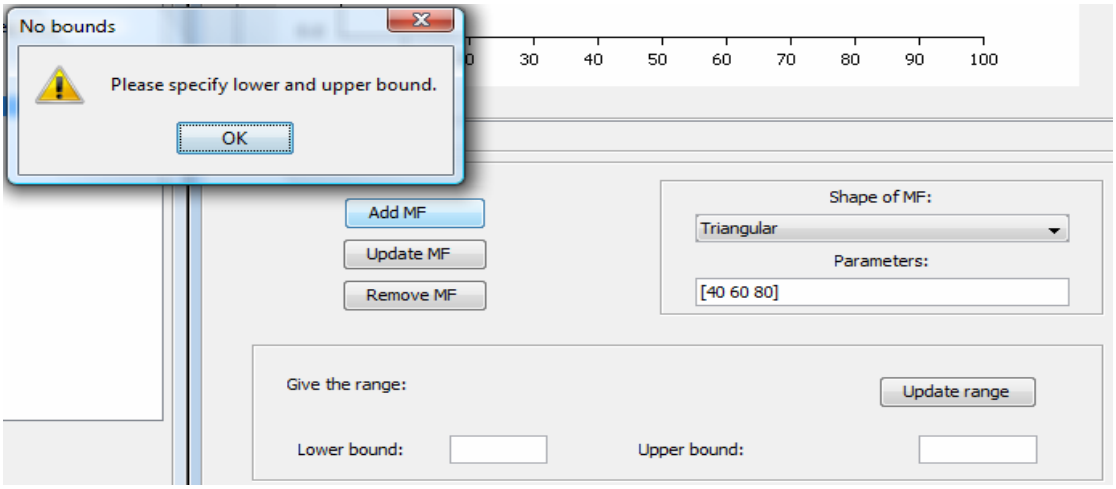
Εάν ο χρήστης κατά την προσθήκη της MF λησμονήσει να προσδιορίσει τα όρια λαμβάνει το μήνυμα του σχήματος 84 ενώ για όρια τα οποία έχει εισάγει αλλά τα οποία δεν είναι αριθμοί αντιμετωπίζει το μήνυμα του σχήματος 85. Τέλος το σύστημα δεν επιτρέπει στο χρήστη να ορίσει το κάτω όριο μεγαλύτερο από το πάνω όριο όπως δείχνει το σχήμα 86. Στο σημείο αυτό να προσθέσουμε μια παρατήρηση για τα προσδιοριζόμενα όρια: αυτό το οποίο ορίζουν δεν είναι κάποια λειτουργική παράμετρος της συνάρτησης που θα διαφοροποιήσει τις τιμές της αλλά το «παράθυρο» προβολής της γραφικής παράστασης της συνάρτησης το οποίο ο χρήστης βλέπει.



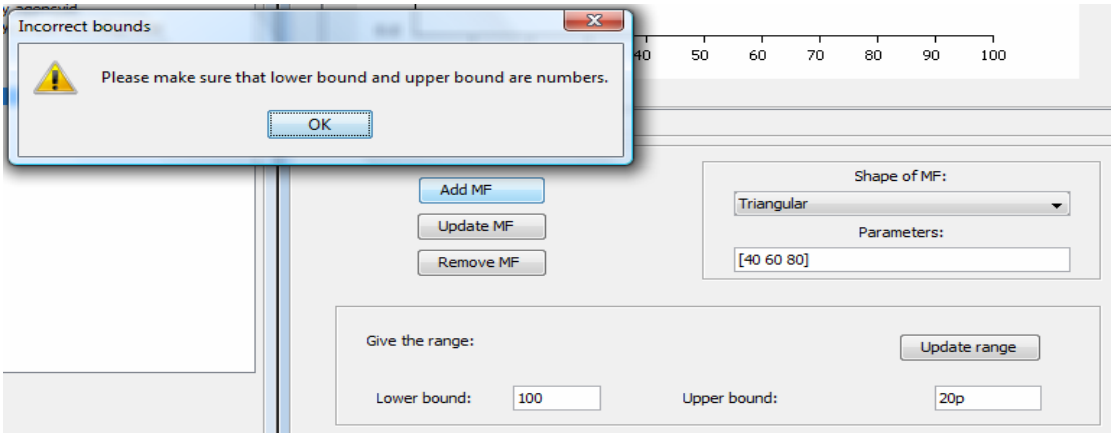
Σχήμα 82. Προειδοποιητικό μήνυμα επιλογής DB Attribute για προσθήκη MF.



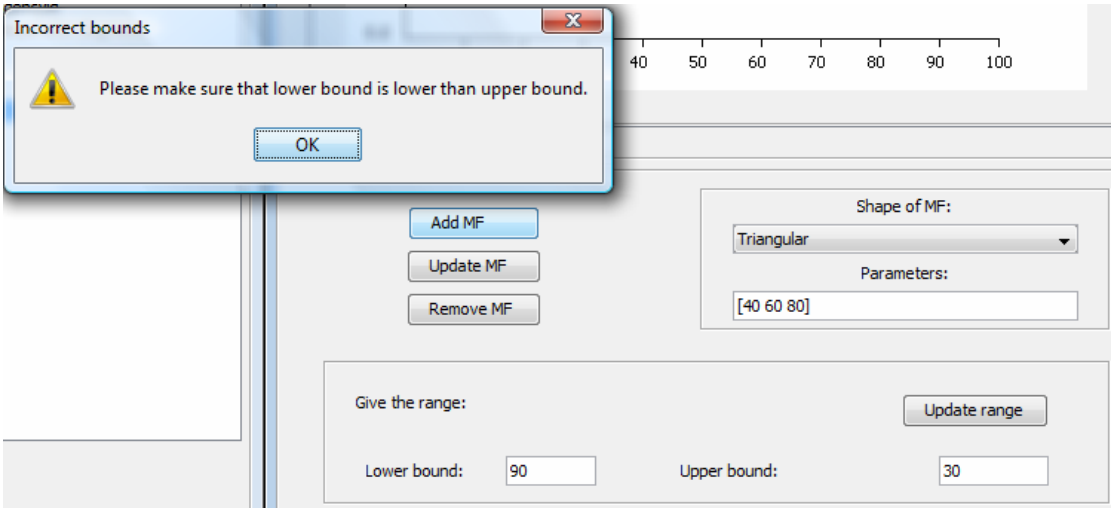
Σχήμα 83. Προειδοποιητικό μήνυμα επιλογής "Fuzzy" radio button για προσθήκη MF.



Σχήμα 84. Προειδοποιητικό μήνυμα προσδιορισμού κάτω και άνω ορίου για προσθήκη MF.

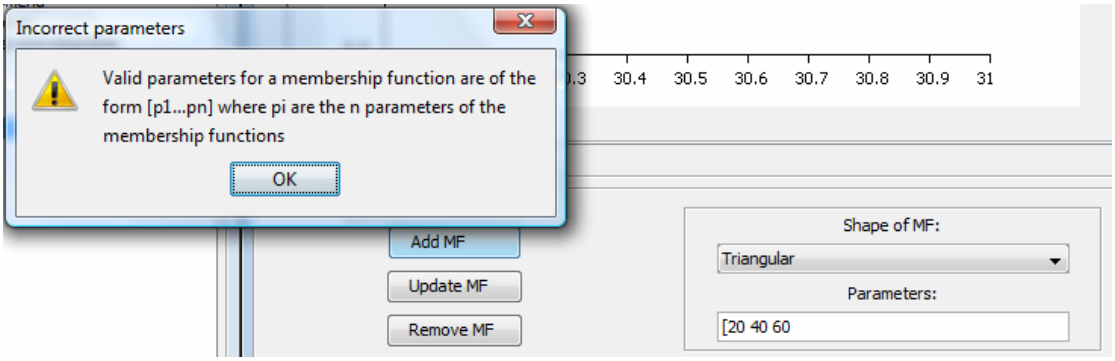


Σχήμα 85. Προειδοποιητικό μήνυμα λάθος ορίων για προσθήκη MF.



Σχήμα 86. Προειδοποιητικό μήνυμα για κάτω όριο μεγαλύτερο από το πάνω όριο.

Επιπλέον το σύστημα πραγματοποιεί έλεγχο των παραμέτρω που ο χρήστης εισάγει. Οι παράμετροι πρέπει να είναι της μορφής $[p_1 \ldots p_n]$ όπου p_i είναι ένας πραγματικός αριθμός και το n εξαρτάται από το σχήμα της MF που χρησιμοποιείται κάθε φορά. Το σχήμα 87 δείχνει τι συμβαίνει σε περίπτωση που οι παράμετροι που έχουν εισαχθεί δεν υπακούουν στην παραπάνω μορφή.



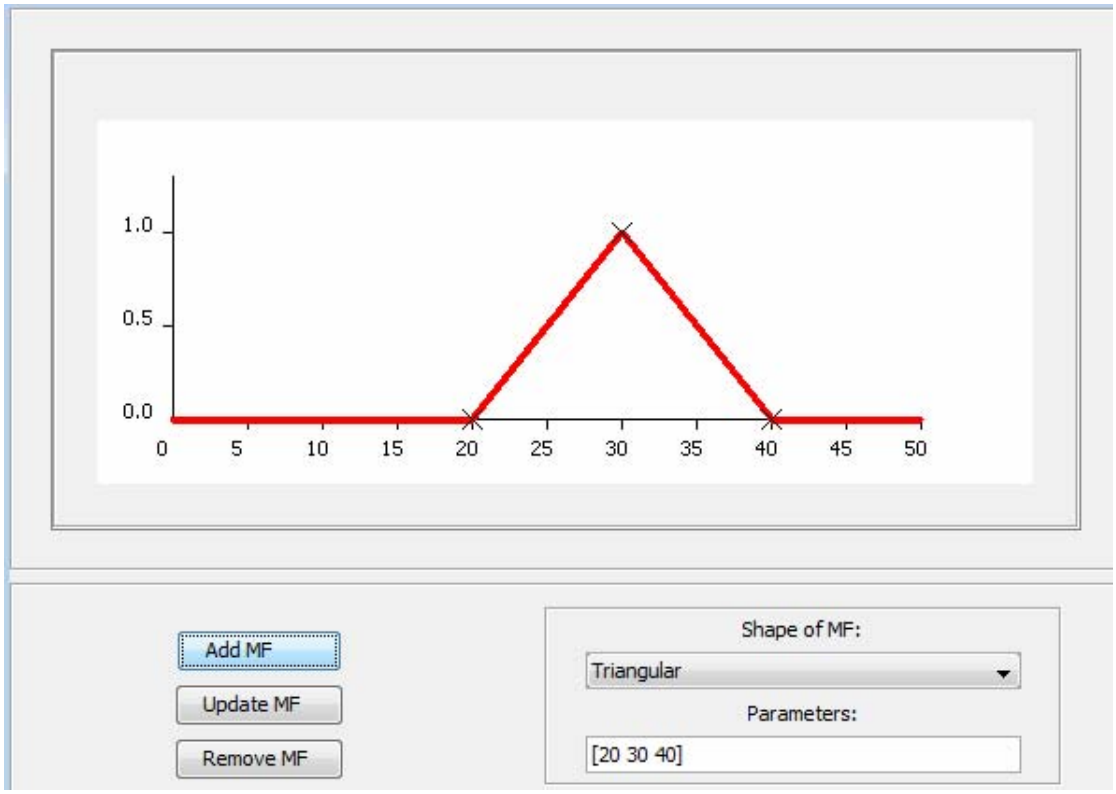
Σχήμα 87. Προειδοποιητικό μήνυμα λάθος παραμέτρων ανεξαρτήτως σχήματος MF.

Στο σημείο αυτό θα παρουσιάσουμε τα διαφορετικά σχήματα MF που διαθέτει το σύστημα καθώς και τις παραμέτρους που σχετίζονται με αυτά. Επιπλέον θα περιγράψουμε τη δυνατότητα που έχει ο χρήστης να τροποποιεί τις MF με το ποντίκι του χωρίς να καθορίζει αριθμητικά τις παραμέτρους.

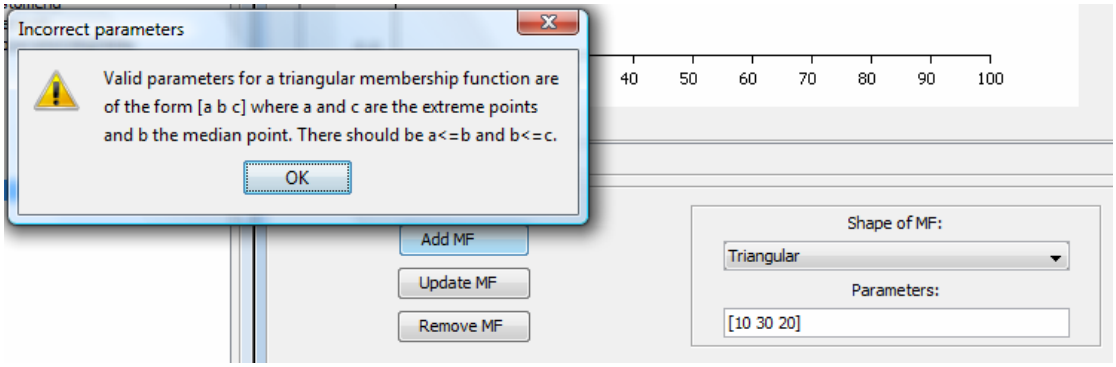
5.4.5.1 Τριγωνική MF

Η τριγωνική MF, όπως μαρτυρά και το όνομα της, είναι ένα τρίγωνο το οποίο έχει τη βάση του στο 0 και του οποίου η κορυφή αντιστοιχεί στην τιμή 1. Περισσότερες λεπτομέρειες για τον τύπο αυτής της συνάρτησης δίνονται στο κεφάλαιο 4. Όσον αφορά τις παραμέτρους της, αυτές είναι τρεις: οι a,b και c. Ο χρήστης τις εισάγει στο πεδίο κειμένου με τη σειρά [a b c] όπου οι παράμετροι a και c αντιστοιχούν στα ακριανά σημεία του τριγώνου ενώ η παράμετρος b στο μεσαίο σημείο αυτού. Το σχήμα 88 δείχνει περισσότερο εποπτικά τη χρήση των παραμέτρων.

Ωστόσο σε περίπτωση που ο χρήστης επιχειρήσει να προσθέσει μία τριγωνική MF της οποίας οι παράμετροι δεν είναι 3 αριθμοί, ή για της οποίας τις παραμέτρους a,b,c δεν ισχύει a<=b και b<=c το μήνυμα λάθους του σχήματος 89 κάνει την εμφάνιση του.



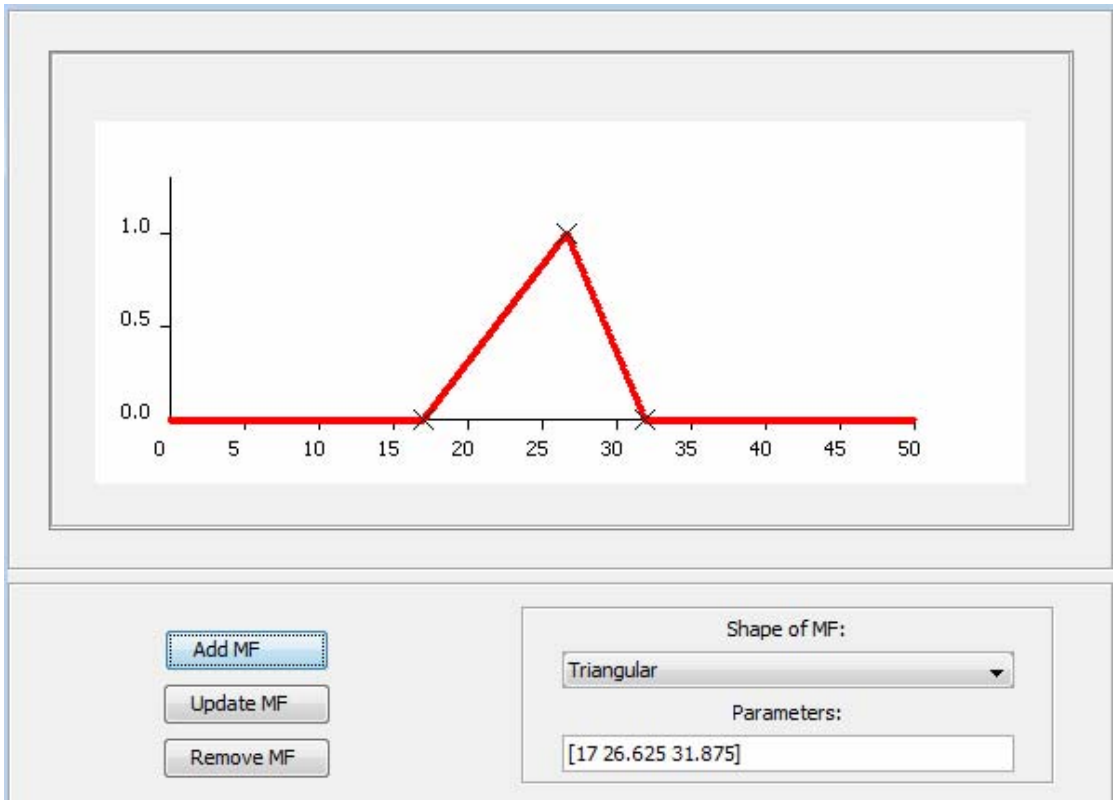
Σχήμα 88. Μία τριγωνική MF με τις αντίστοιχες παραμέτρους της.



Σχήμα 89. Μήνυμα λάθους για προσθήκη τριγωνικής MF με ασυνεπείς παραμέτρους.

Επιπλέον παρατηρούμε ότι σε 3 σημεία της γραφικής παράστασης της συνάρτησης έχουν σχεδιαστεί 3 μαύρα σημάδια. Σκοπό έχουν να βοηθήσουν τον χρήστη να τροποποιήσει την τριγωνική MF με τις κινήσεις του ποντικιού του και χωρίς να χρειάζεται να δώσει ακριβείς αριθμητικές παραμέτρους. Η μετακίνηση του αριστερού σταυρού έχει ως αποτέλεσμα την αλλαγή της παραμέτρου a. Η μετακίνηση του μεσαίου και του δεξιού σταυρού των

παραμέτρων b και c αντίστοιχα. Ωστόσο τίθενται κάποιοι περιορισμοί: η νέα παράμετρος a δεν μπορεί να είναι μικρότερη από το κάτω όριο ή μεγαλύτερη από την παράμετρο b, η νέα παράμετρος b θα πρέπει να βρίσκεται μεταξύ των παραμέτρων a και c ενώ η νέα παράμετρος c μεταξύ της b και του άνω ορίου. Επιπλέον καθώς τροποποιείται η MF τροποποιούνται και οι παράμετροι που εμφανίζονται στο αντίστοιχο πεδίο κειμένου. Το σχήμα 90 δείχνει μια MF που έχει τροποποιηθεί με αυτόν τον τρόπο με τις αντίστοιχες τροποποιημένες παραμέτρους.

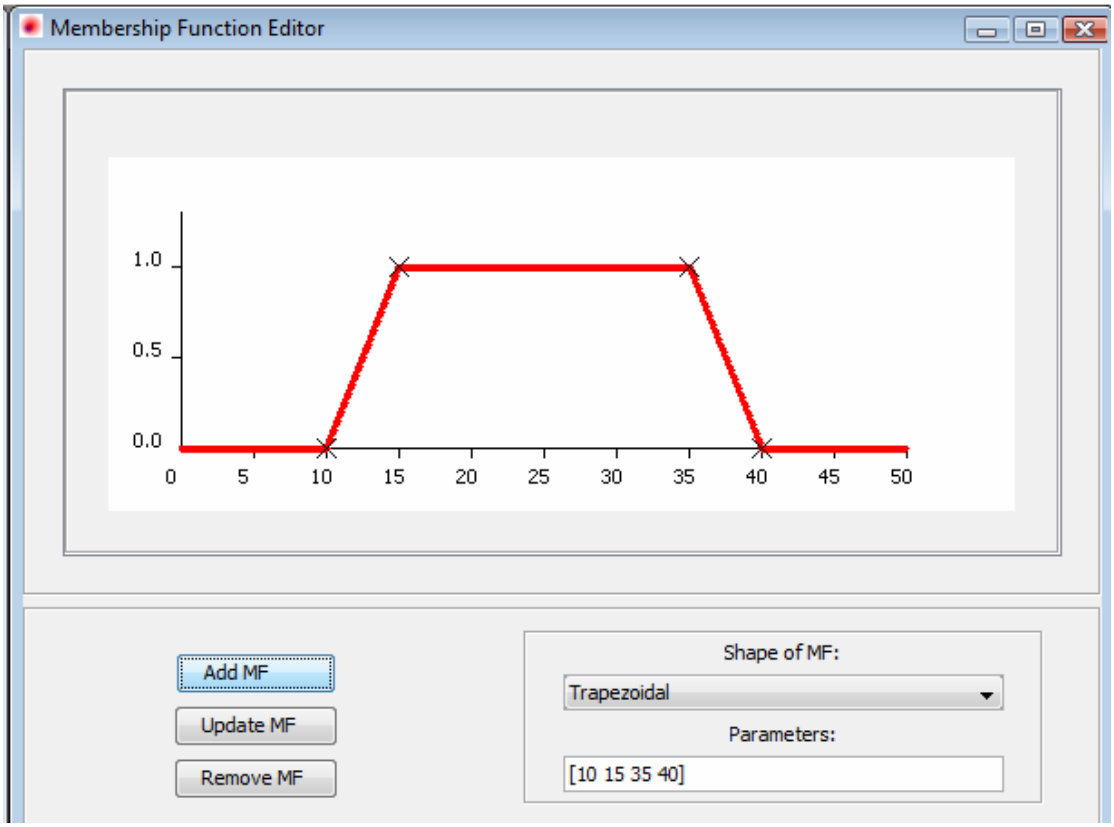


Σχήμα 90. Τροποποίηση τριγωνικής MF με τη βοήθεια του ποντικιού.

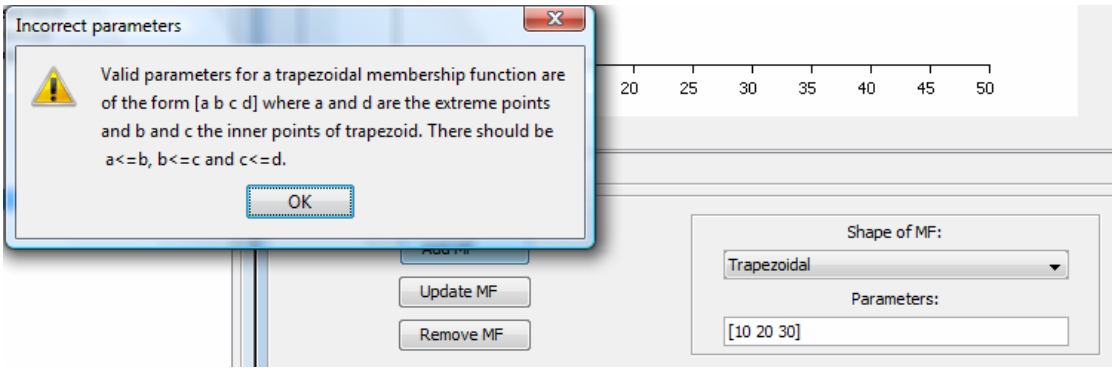
5.4.5.2 Τραπεζοειδής MF

Η τραπεζοειδής MF έχει σχήμα τραπεζίου με τη βάση του να ξεκινά στο 0 και την άνω πλευρά του να βρίσκεται στο 1. Ο ακριβής τύπος της συνάρτησης μπορεί να βρεθεί στο κεφάλαιο 4. Διαθέτει 4 παραμέτρους: τις a,b,c και d. Οι a και d δείχνουν το αριστερό και αντίστοιχα δεξιό ακριανό σημείο της συνάρτησης ενώ οι b και c το αριστερό και δεξιό άκρο της άνω πλευράς του τραπεζίου. Ο χρήστης εισάγει παραμέτρους για μία τραπεζοειδή συνάρτηση στη μορφή [a b c d] όπως δείχνει και το σχήμα 91.

Ο χρήστης θα λάβει ένα μήνυμα λάθους εάν επιχειρήσει να προσθέσει τραπεζοειδή MF με λιγότερες ή περισσότερες από 4 παραμέτρους ή παραμέτρους a,b,c,d που δεν τηρούν τους περιορισμούς a<=b,b<=c και c<=d. Ότι φαίνεται στο σχήμα 92 είναι αυτό που θα συμβεί σε μία τέτοια περίπτωση.

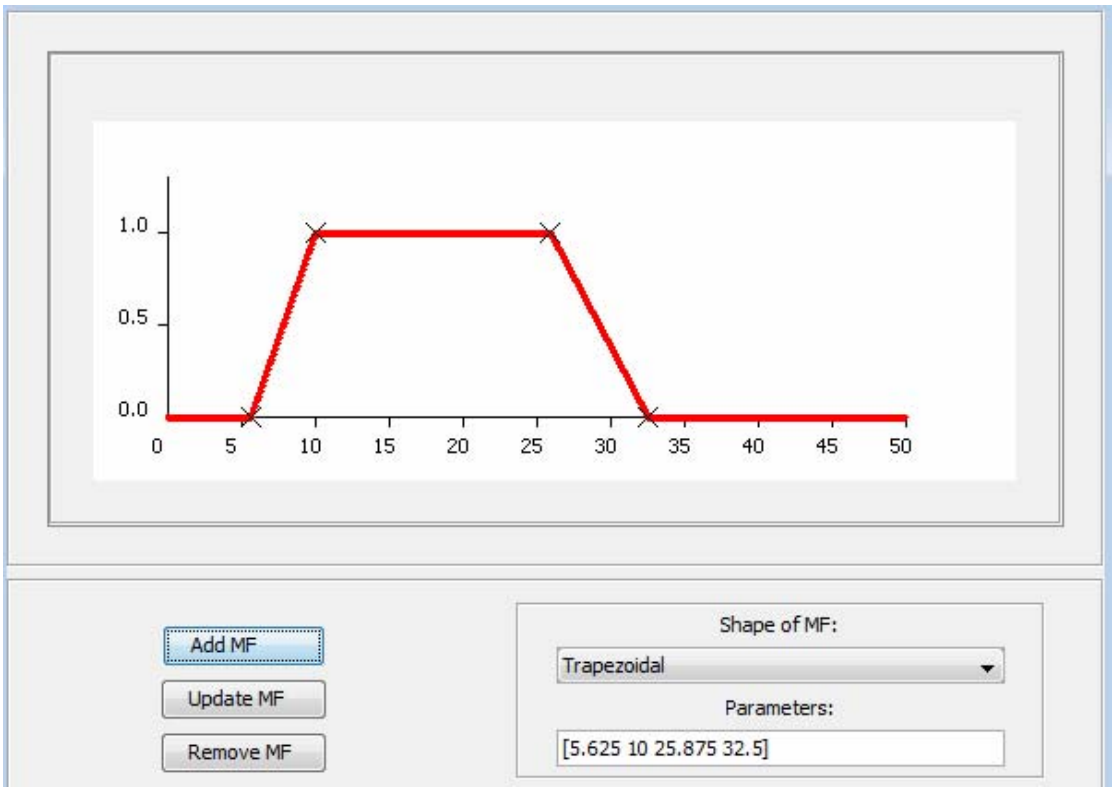


Σχήμα 91. Μια τραπεζοειδής MF με τις αντίστοιχες παραμέτρους της.



Σχήμα 92. Μήνυμα λάθους για προσθήκη τραπεζοειδούς MF με ασυνεπείς παραμέτρους.

Και πάλι τα 4 σημάδια αντιστοιχούν στις 4 παραμέτρους που ο χρήστης μπορεί να τροποποιήσει με τη βοήθεια του ποντικιού του. Οι εύλογοι περιορισμοί που τίθενται είναι οι εξής: η νέα παράμετρος a οφείλει να είναι μεταξύ του κάτω ορίου και της b, η νέα παράμετρος b μεταξύ των a και c, η νέα παράμετρος c μεταξύ των b και d ενώ η νέα παράμετρος d μεταξύ των c και του άνω ορίου. Το σχήμα 93 δίνει τη MF του σχήματος 91 την οποία ο χρήστης έχει τροποποιήσει σύροντας ελαφρά με το ποντίκι του τους 4 σταυρούς προς τα αριστερά. Παρατηρούμε ότι έχουν αλλάξει επίσης οι αριθμητικές τιμές των παραμέτρων στο πεδίο κειμένου.



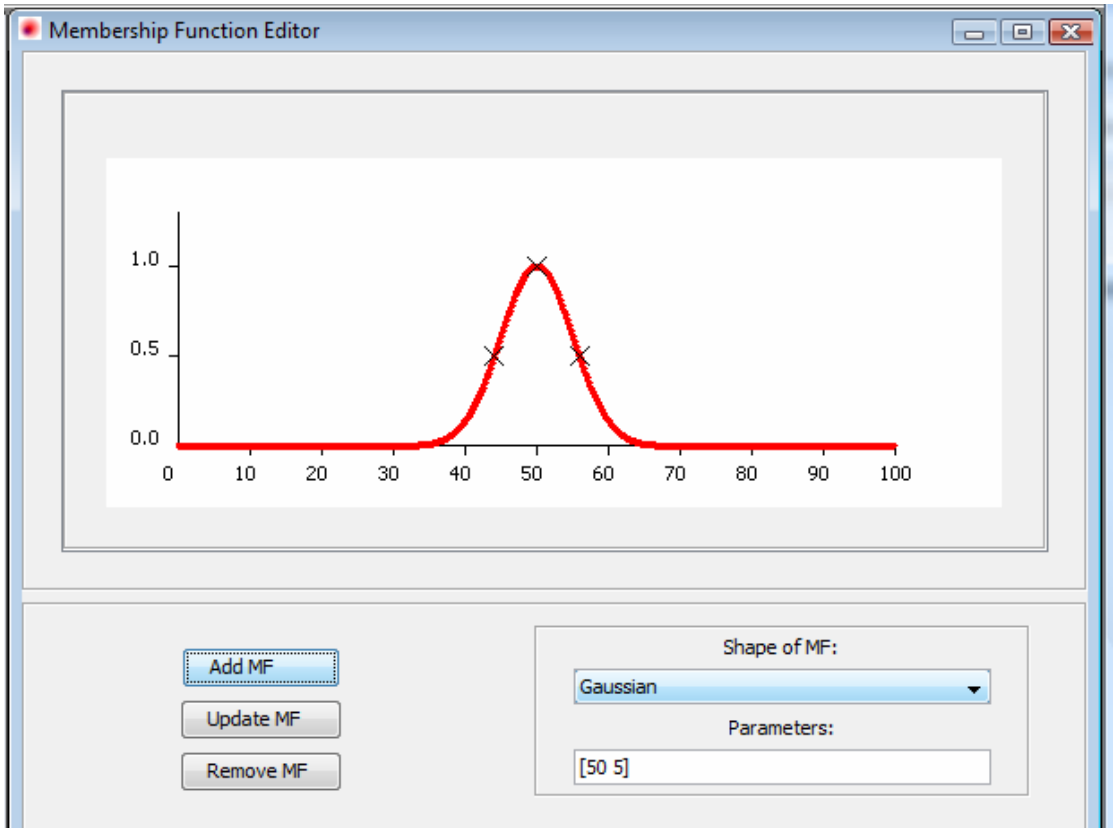
Σχήμα 93. Τροποποίηση τραπεζοειδούς MF με τη βοήθεια του ποντικιού.

5.4.5.3 Γκαουσιανή MF

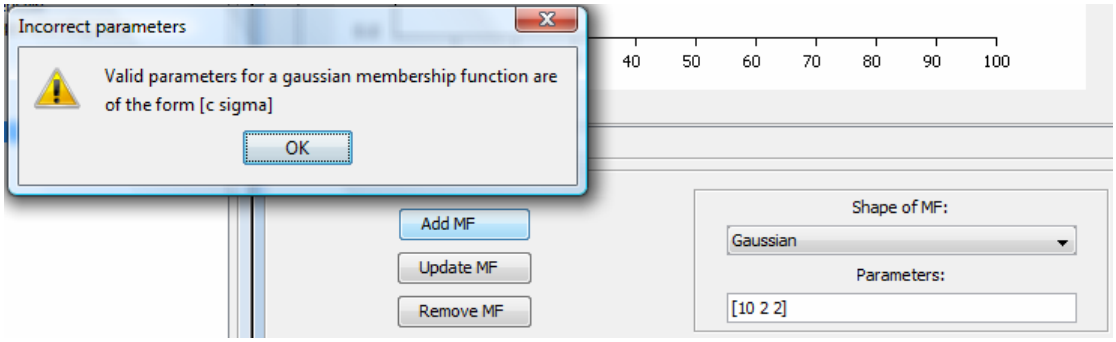
Η γκαουσιανή συνάρτηση υπακούει στον τύπο $e^{\frac{(x-c)^2}{-2\sigma^2}}$. Διαθέτει δύο παραμέτρους: την c η οποία καθορίζει τη x συντεταγμένη που παρατηρείται η κορυφή της συνάρτησης και τη σ η οποία καθορίζει το άνοιγμα της «καμπάνας». Ο χρήστης εισάγει τις παραμέτρους στη μορφή [c σ]. Ένα τέτοιο παράδειγμα δείχνει το σχήμα 93.

Προκειμένου ο χρήστης να μη λάβει το μήνυμα λάθους του σχήματος 95 θα πρέπει να εισάγει ακριβώς 2 παραμέτρους κατά την προσθήκη της γκαουσιανής MF.

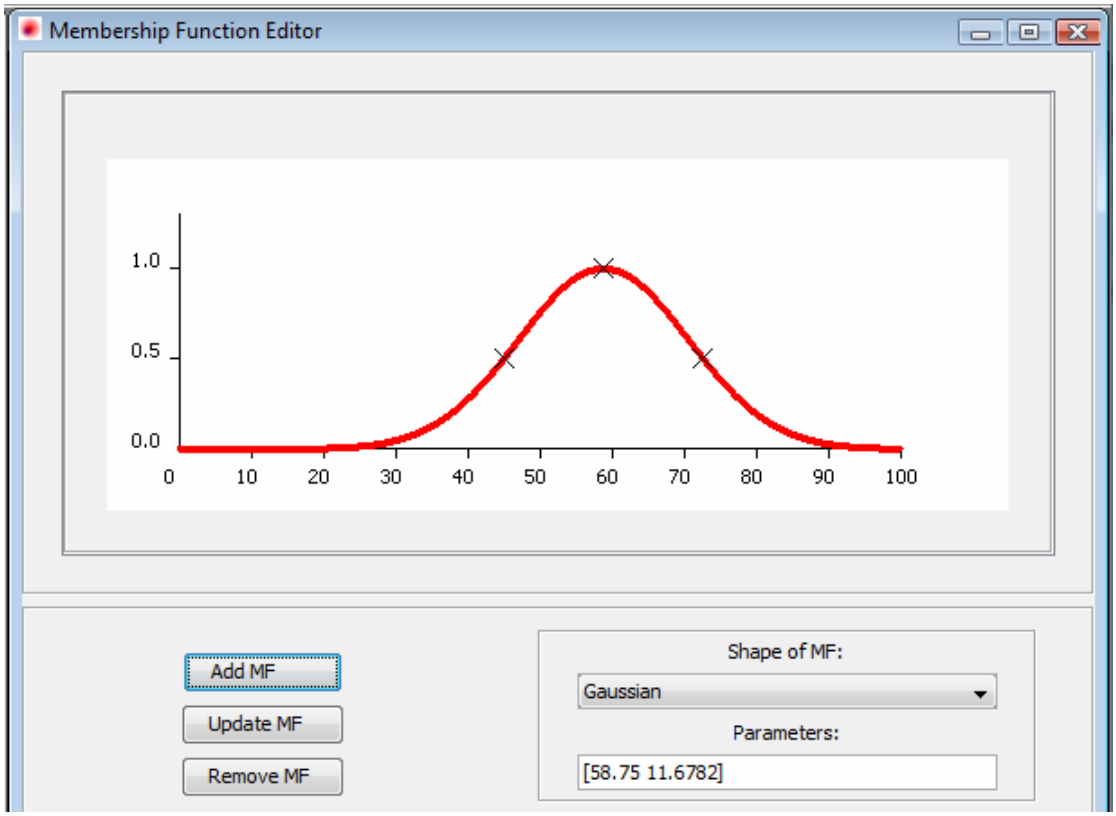
Όσον αφορά τη λειτουργία που επιτρέπει στο χρήστη να τροποποιεί με το ποντίκι του τη MF αυτή τη φορά εμφανίζονται 3 μαύροι σταυροί. Ο μεσαίος σταυρός αντιστοιχεί στην παράμετρο c και η μετακίνηση του αριστερά ή δεξιά μετατοπίζει αντίστοιχα το κέντρο της συνάρτησης. Οι σταυροί που βρίσκονται αριστερά και δεξιά επιτρέπουν στο χρήστη να αυξομειώνει το άνοιγμα της «καμπάνας». Όταν οι σταυροί απομακρύνονται από το κέντρο της συνάρτησης αυξάνει το πλάτος της ενώ όταν το προσεγγίζουν μειώνεται το πλάτος της. Το σχήμα 96 δείχνει τη MF του σχήματος 94 μετατοπισμένο προς τα δεξιά και με μεγαλύτερο άνοιγμα. Οι ανάλογες αλλαγές των παραμέτρων απεικονίζονται στο πεδίο κειμένου.



Σχήμα 94. Μια γκαουσιανή MF με τις αντίστοιχες παραμέτρους της.



Σχήμα 95. Μήνυμα λάθους για προσθήκη γκαουσιανής MF με ασυνεπείς παραμέτρους.



Σχήμα 96. Τροποποίηση γκαουσιανής MF με τη βοήθεια του ποντικιού.

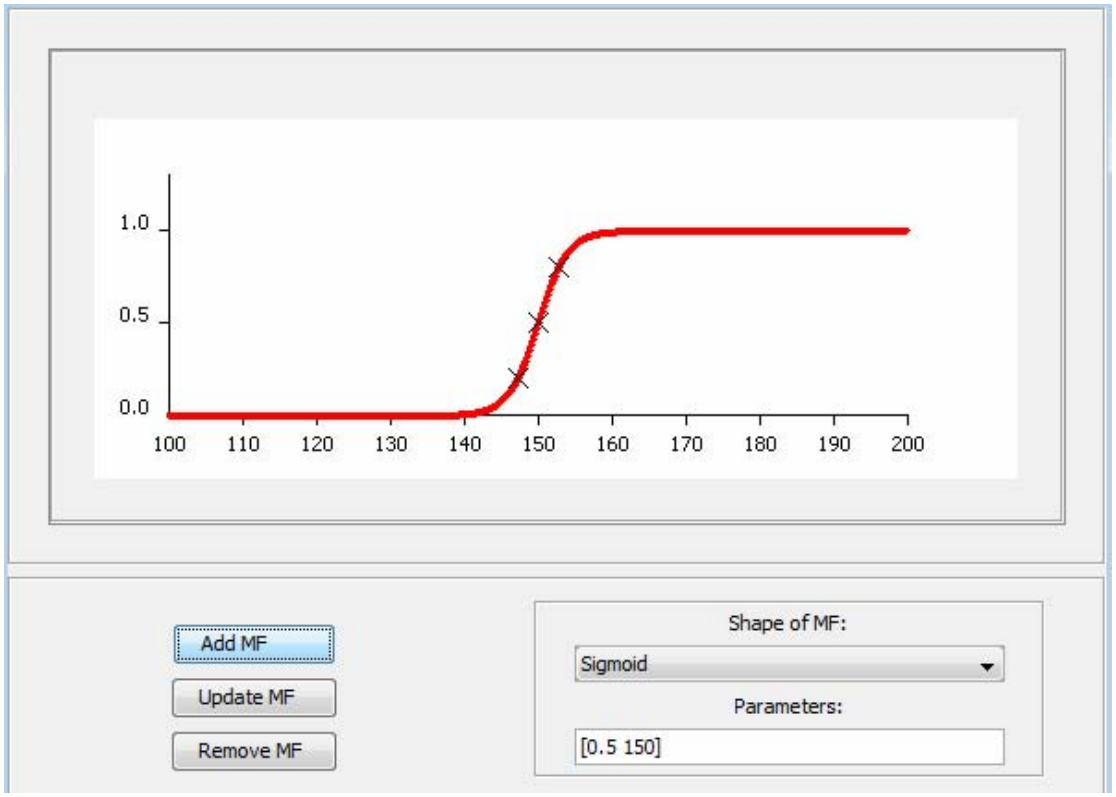
5.4.5.4 Σιγμοειδής MF

Η σιγμοειδής MF υπακούει στον τύπο $\frac{1}{1+e^{-\frac{a}{x-c}}}$. Έχει δύο παραμέτρους: την α που καθορίζει το πλάτος της συνάρτησης και τη c που είναι η συντεταγμένη x στο σημείο όπου

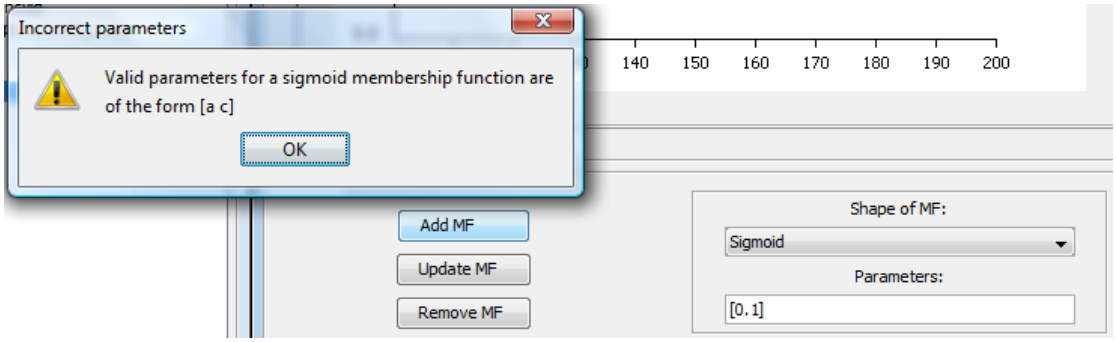
ισχύει y=0.5. Ο χρήστης εισάγει τις δύο αυτές παραμέτρους στη μορφή [α c]. Το σχήμα 97 απεικονίζει την προσθήκης μιας σιγμοειδούς MF.

Και πάλι υπάρχει ο περιορισμός για δύο παραμέτρους ο οποίος εάν δεν τηρηθεί έχει ως αποτέλεσμα το μήνυμα λάθους του σχήματος 100.

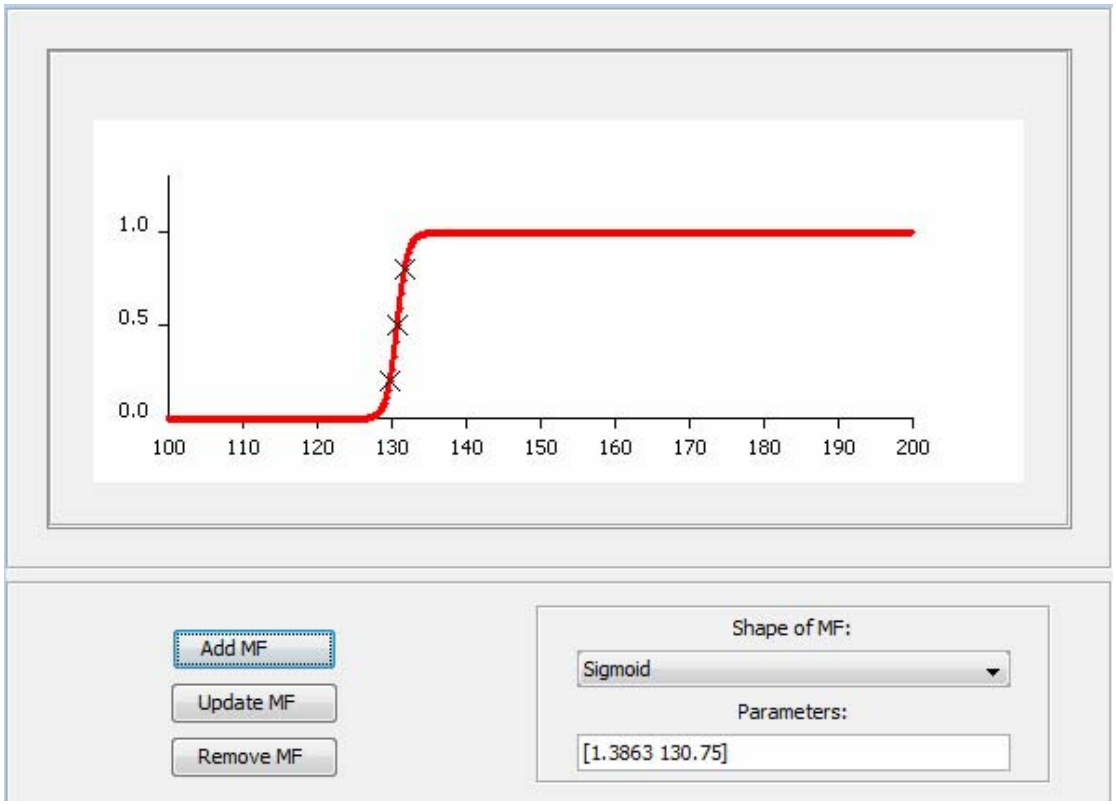
Τελος και αυτός ο τύπος συνάρτησης προσφέρει στο χρήστη τη δυνατότητα τροποποίησης με τη βοήθεια του ποντικιού. Διαθέτει όπως και η γκαουσιανή συνάρτηση 3 σημάδια: ένα στο κέντρο, ένα αριστερά και ένα δεξιά. Και πάλι το κεντρικό σημάδι μετατοπίζει τη συνάρτηση αριστερά ή δεξιά ανάλογα με την κατεύθυνση προς την οποία σύρεται. Τα αριστερά και δεξιά σημάδια προκαλούν κατά τη μετακίνηση τους αυξομείωση του πλάτους της συνάρτησης ανάλογα με το αν απομακρύνονται ή προσεγγίζουν το κέντρο της συνάρτησης. Το σχήμα 99 δείχνει τη MF του σχήματος 101 μετατοπισμένη προς τα αριστερά και συρρικνωμένη.



Σχήμα 97. Μια σιγμοειδής MF με τις αντίστοιχες παραμέτρους της.



Σχήμα 98. Μήνυμα λάθους για προσθήκη σιγμοειδούς MF με ασυνεπείς παραμέτρους.



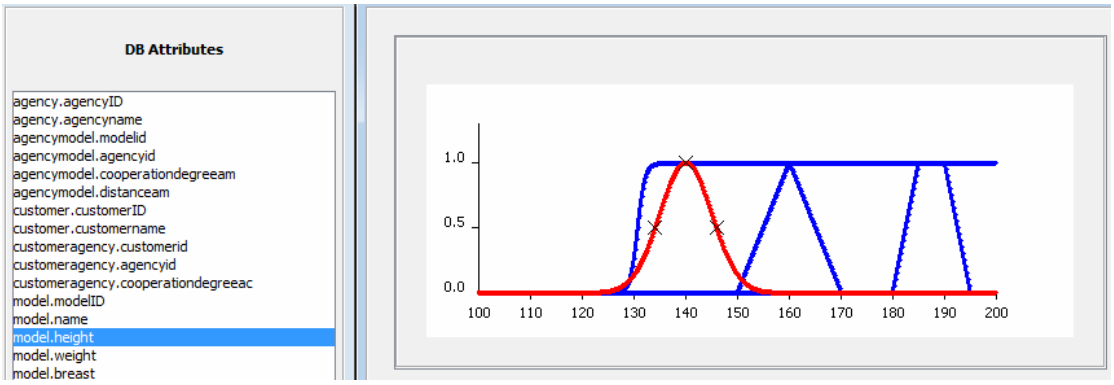
Σχήμα 99 . Τροποποίηση σιγμοειδούς MF με τη βοήθεια του ποντικιού.

Το panel το οποίο φιλοξενεί τη γραφική παράσταση των MF έχει τη δυνατότητα να φιλοξενήσει όσες MF ο χρήστης επιθυμεί διαφορετικών σχημάτων και παραμέτρων. Ωστόσο οι εν λόγω MF θα διαθέτουν τα ίδια όρια (τα οποία όπως έχουμε σημειώσει αποτελούν τα «παράθυρα» προβολής των γραφικών τους παραστάσεων) και θα αναφέρονται στο ίδιο DB

Attribute. Άμεση συνέπεια του παραπάνω είναι σε κάθε DB Attribute να αντιστοιχούν ένα σύνολο από MF που ο χρήστης έχει ορίσει. Σε περίπτωση που ο χρήστης έχει π.χ. ορίσει ένα σύνολο από MF για το DB Attribute Model.weight και στη συνέχεια μεταβεί στο DB Attribute Model.height επιλέγοντας το κατά την επιστροφή του στο Model.weight θα δει στο σύστημα αξόνων τις συναρτήσεις που πριν είχε ορίσει για το Model.weight.

Επιπλέον όπως συνδέεται μοναδικά ένα σύνολο MF με ένα DB Attribute υπάρχει και μοναδική σύνδεση μεταξύ ενός ζεύγους ορίων με αυτό το DB Attribute. Τα όρια αυτά είναι ίδια για όλες τις MF που αντιστοιχούν στο συγκεκριμένο DB Attribute και απεικονίζουν το κομμάτι εκείνο της MF που αντιστοιχεί στα όρια ακόμα και αν αυτό είναι μηδενικό. Αυτό σημαίνει ότι ο χρήστης ορίζει για κάθε DB Attribute μία φορά το ζεύγος των ορίων κατά την προσθήκη της πρώτης MF. Όλες οι επόμενες MF για το συγκεκριμένο DB Attribute προστίθενται με τα όρια τους αυτόματα προκαθορισμένα. Επιπλέον να προσθέσουμε ότι πρακτικά το πεδίο ορισμού των MF είναι το $(-\infty, +\infty)$, απλά πρόκειται για MF που στο μεγαλύτερο κομμάτι τους έχουν τιμή 0.

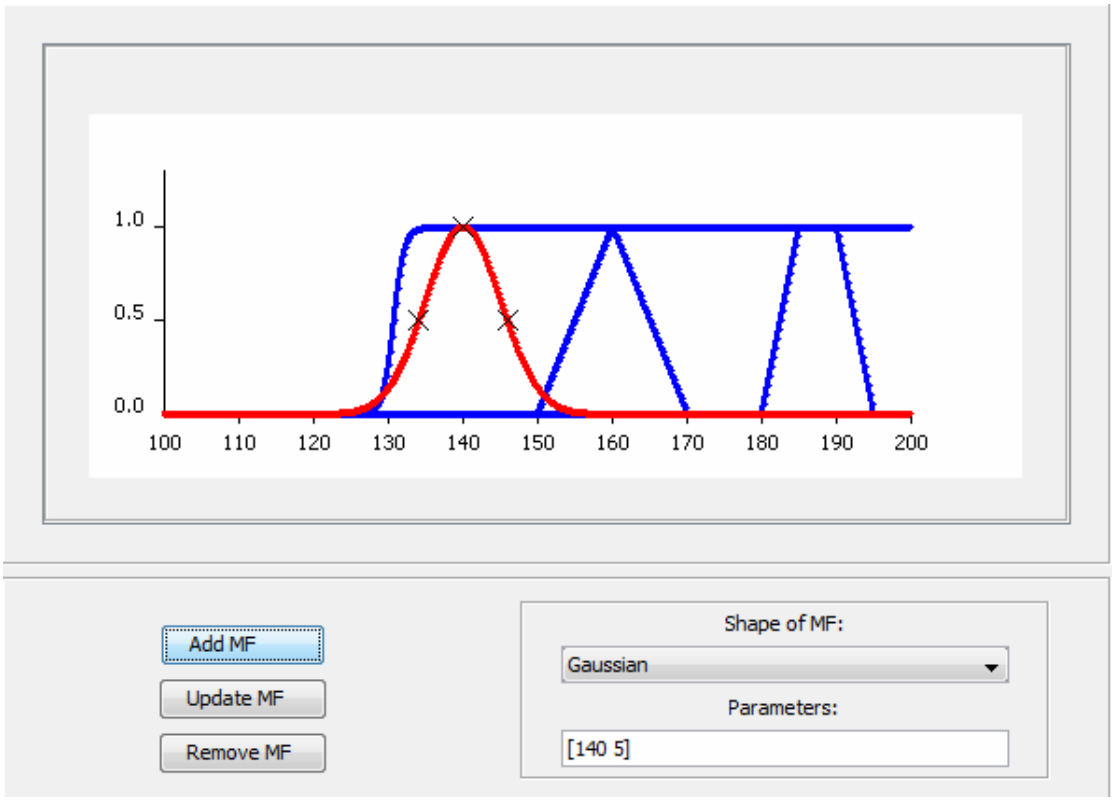
Όσον αφορά την απεικόνιση στο interface του συστήματος τώρα, από τις συναρτήσεις που απεικονίζει το σύστημα αξόνων όλες είναι ζωγραφισμένες με μπλε χρώμα εκτός από μία η οποία είναι ζωγραφισμένη με κόκκινο χρώμα και αυτό ακριβώς σηματοδοτεί ότι είναι η επιλεγμένη. Η επιλεγμένη συνάρτηση είναι επίσης η μόνη που διαθέτει σημάδια για την τροποποίηση της με το ποντίκι. Ο χρήστης μπορεί να επιλέξει μία συνάρτηση κάνοντας κλικ με το ποντίκι πάνω στο γράφημα της. Μια τέτοια περίπτωση απεικονίζει το σχήμα 100 για το DB Attribute model.height.



Σχήμα 100. Ένα DB Attribute μαζί με τις MF που έχουν οριστεί γι'αυτό.

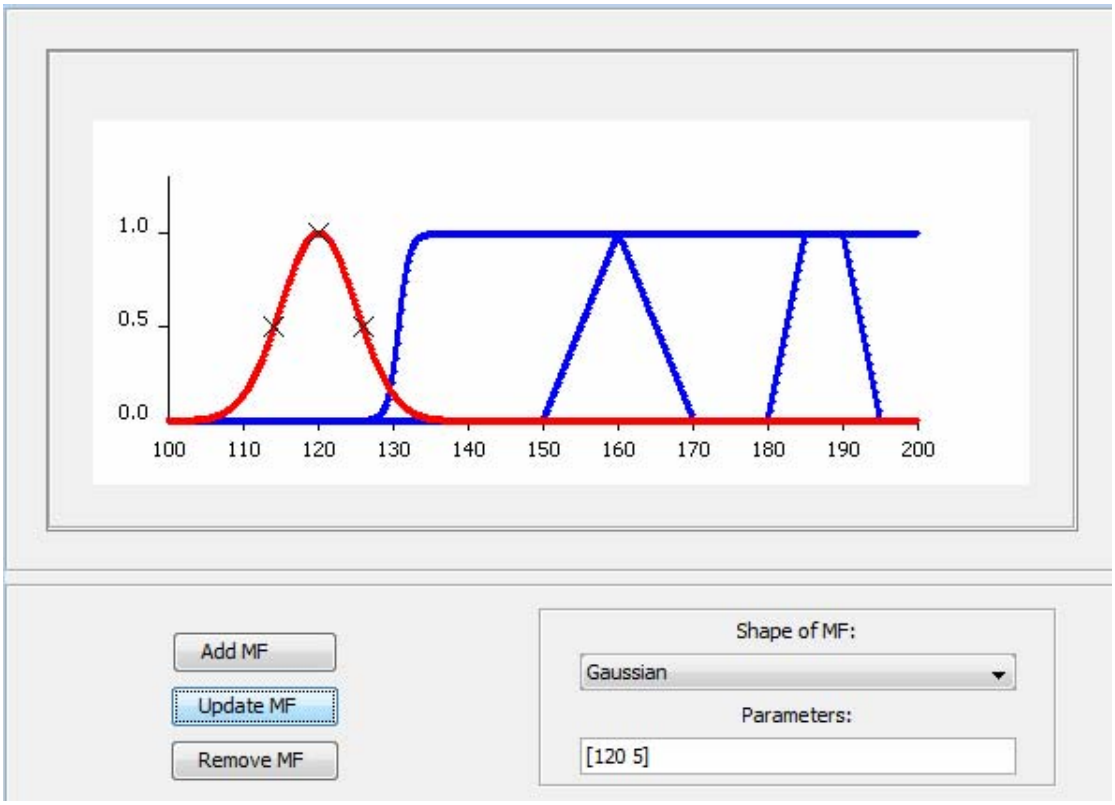
Όσον αφορά την επεξεργασία των MF το σύστημα δίνει τη δυνατότητα στο χρήστη να ενημερώνει την επιλεγμένη MF με νέες παραμέτρους τις οποίες εισάγει στη θέση των

παλαιών στο αντίστοιχο πεδίο κειμένου. Μία τέτοια περίπτωση φαίνεται μεταξύ των σχημάτων 101 και 102 στα οποία η γκαουσιανή συνάρτηση έχει υποστεί ενημέρωση της πρώτης παραμέτρου της. Επιπλέον ενημέρωση μπορεί να γίνει και στον τυπο της συνάρτησης (τριγωνική, τραπεζοειδής κτλ.) με την προϋπόθεση βέβαια οι παράμετροι να βρίσκονται στην κατάλληλη μορφή.

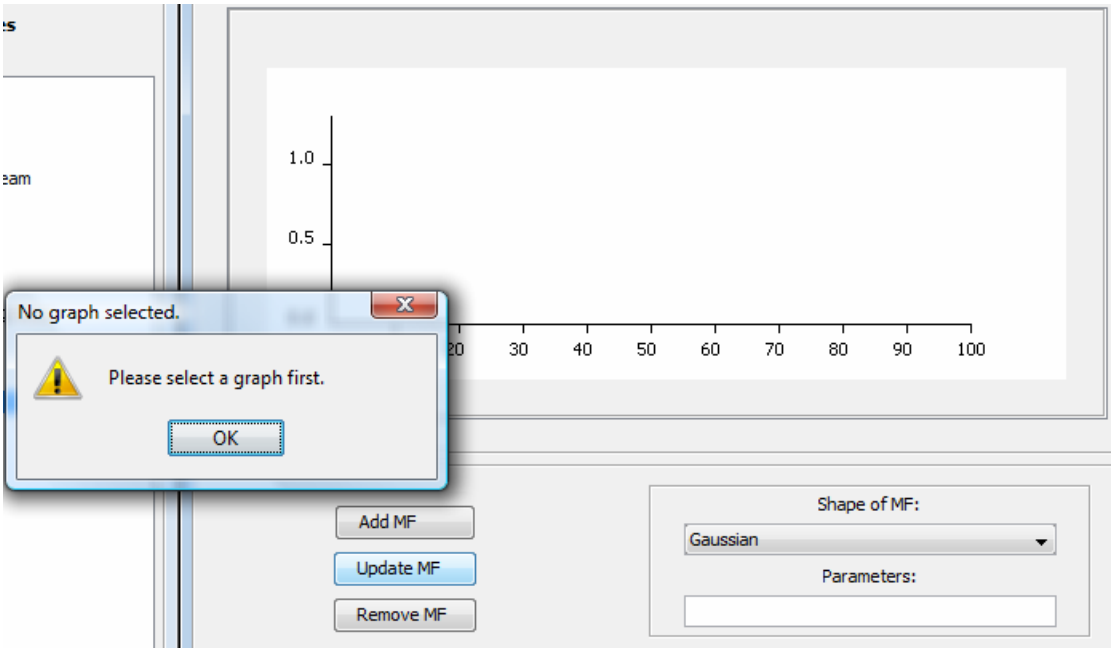


Σχήμα 101. MF πριν την ενημέρωση

Βέβαια απαραίτητη προϋπόθεση για τη λειτουργία του κουμπιού “Update MF” είναι να υπάρχει μία επιλεγμένη συνάρτηση όπως δείχνει και το σχήμα 103.



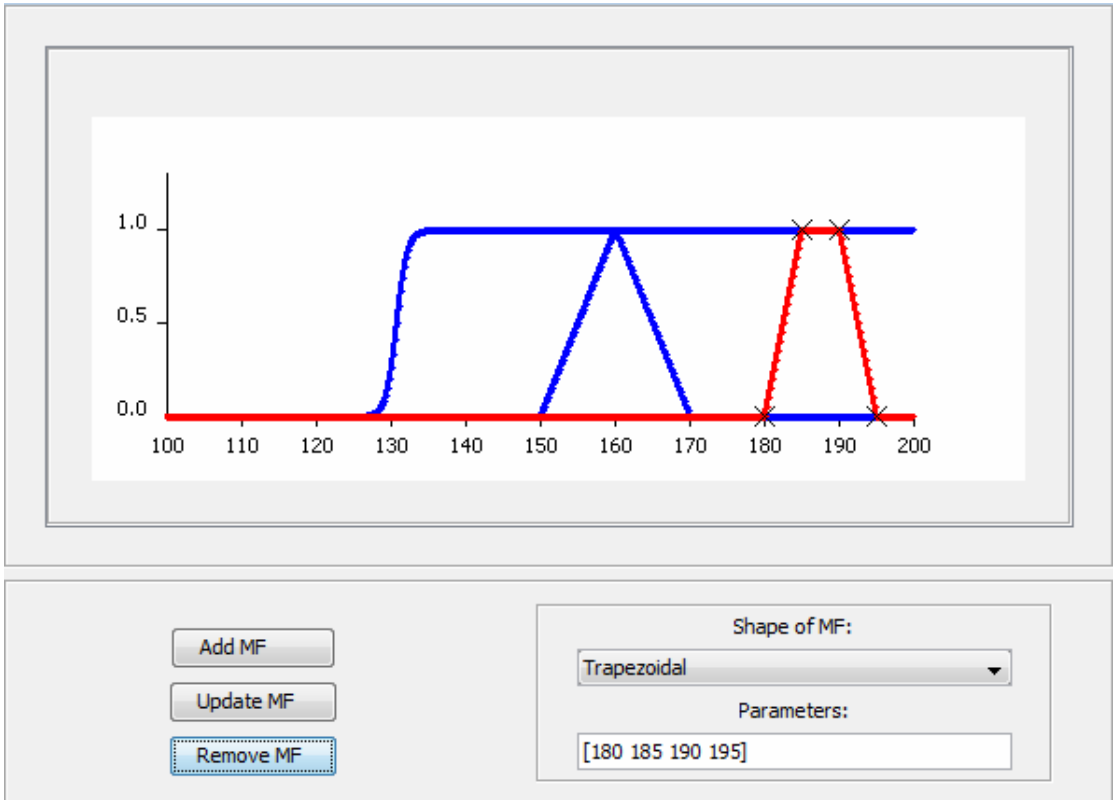
Σχήμα 102. MF μετά την ενημέρωση



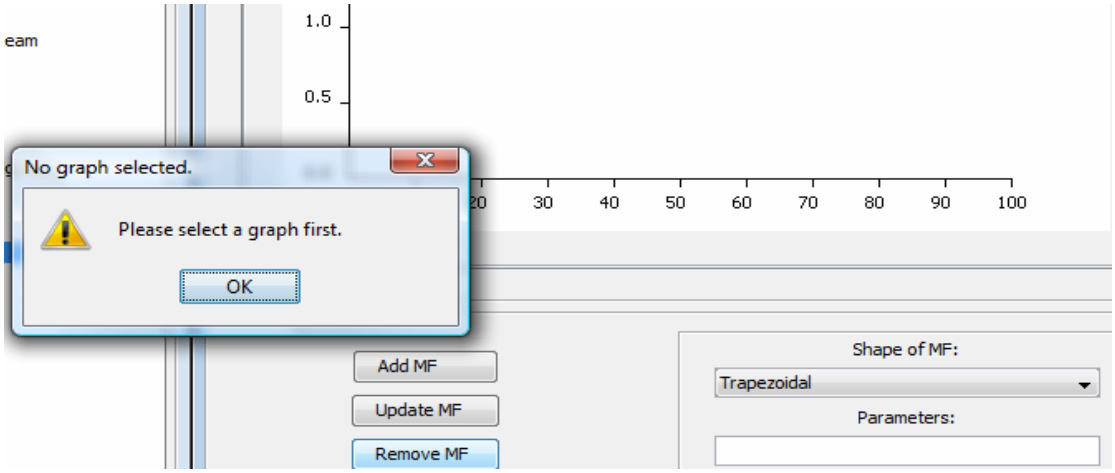
Σχήμα 103. Προειδοποιητικό μήνυμα επιλογής MF για ενημέρωση συνάρτησης.

Τέλος να τονίσουμε ότι το σύνολο των ελέγχων σχετικά με τις παραμέτρους που πραγματοποιούνται στην περίπτωση της προσθήκης συνάρτησης πραγματοποιούνται και εδώ. Πρόκειται για τους περιορισμούς που πρέπει να ικανοποιούν οι παράμετροι και περιγράφηκαν λίγο πιο πάνω.

Εκτός από τη δυνατότητα ενημέρωσης το σύστημα δίνει και τη δυνατότητα αφαίρεσης μιας MF με το πάτημα του κουμπιού “Remove MF”. Για παράδειγμα στο σχήμα 104 παρατηρούμε τα γραφήματα των MF μετά την απομάκρυνση της γκαουσιανής συνάρτησης που προηγουμένως είχαμε ενημερώσει. Το σχήμα 105 δείχνει τι συμβαίνει εάν πατήσουμε το “Remove MF” χωρίς να υπάρχει μία επιλεγμένη συνάρτηση. Επιπλέον παρατηρούμε ότι μετά την αφαίρεση της συνάρτησης επιλέγεται αυτόματα η συνάρτηση που είχε πιο πρόσφατα επεξεργαστεί και ενημερώνεται το σχήμα και οι παράμετροι της στη λίστα επιλογής και το πεδίο κειμένου αντίστοιχα.



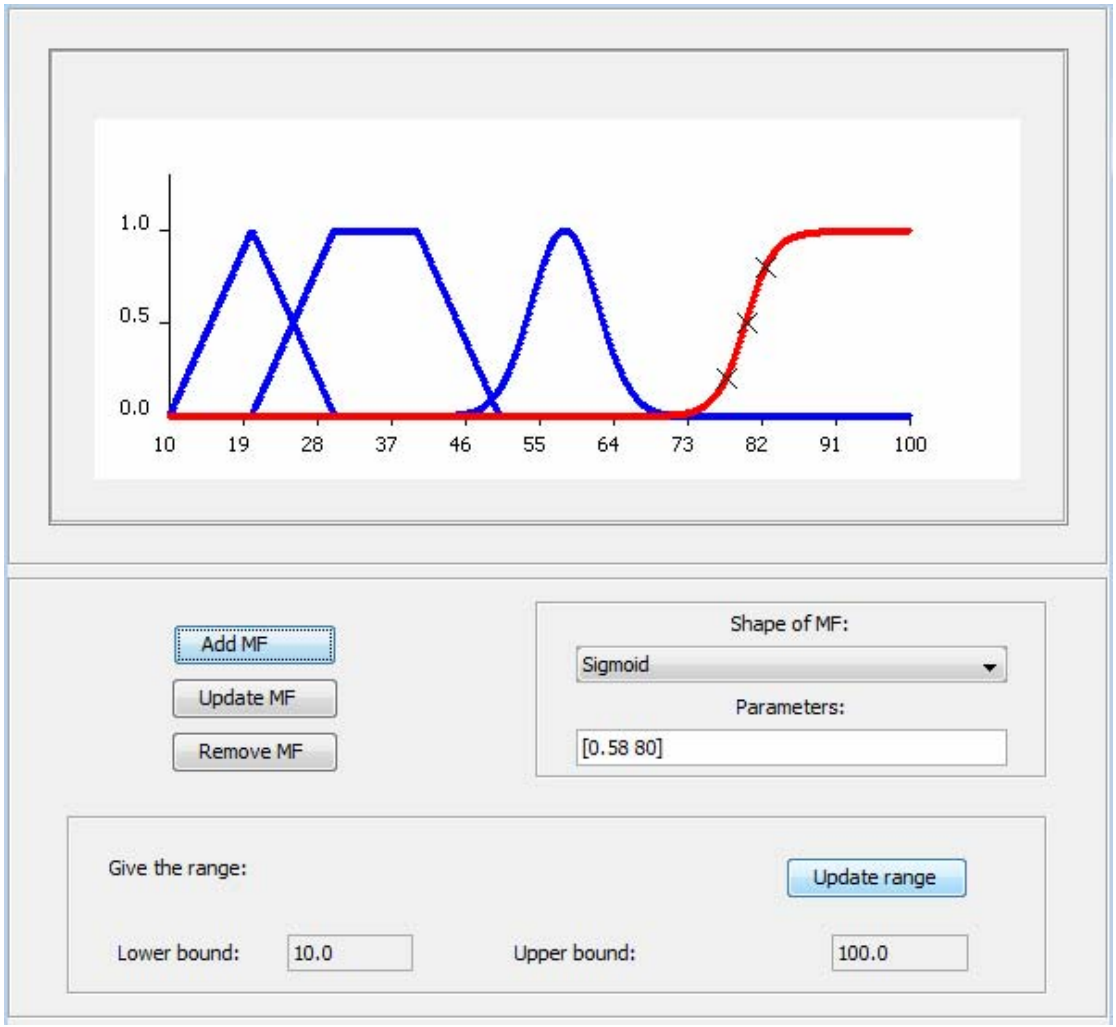
Σχήμα 104. Μετά την απομάκρυνση της γκαουσιανής συνάρτησης από το σχήμα 102.



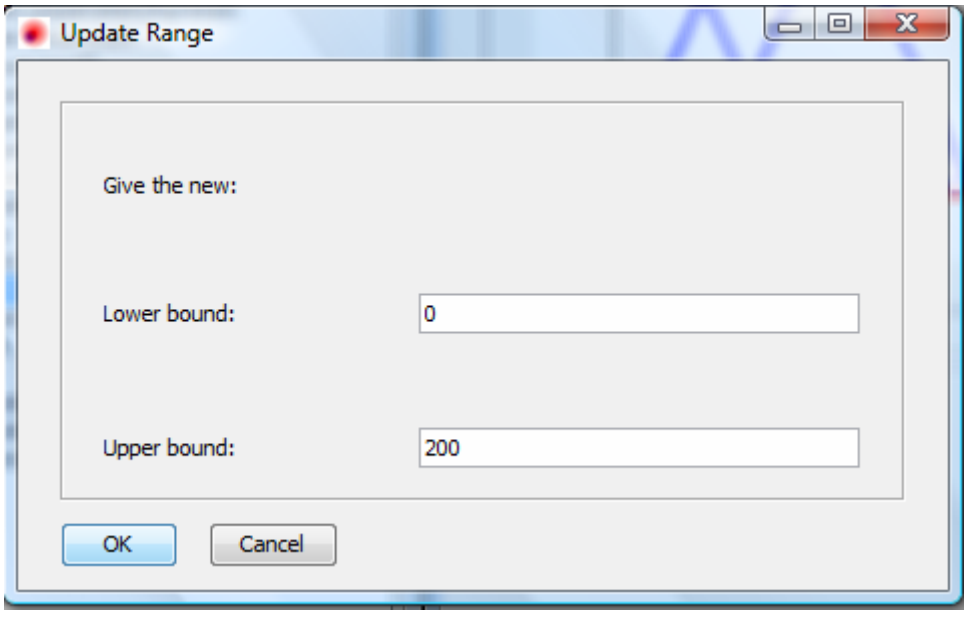
Σχήμα 105. Προειδοποιητικό μήνυμα επιλογής MF για απομάκρυνση συνάρτησης.

Ο χρήστης από τη στιγμή που ορίζει ένα ζεύγος ορίων για ένα DB Attribute δεν έχει τη δυνατότητα να το αλλάξει άμεσα. Αυτό οφείλεται στο ότι τα αντίστοιχα πεδία κειμένου γίνονται μη επεξεργάσιμα μετά την πρώτη επιτυχημένη προσθήκη MF. Ωστόσο αυτό δε σημαίνει ότι είναι υποχρεωμένος μέχρι το τέλος της εργασίας του να δουλεύει με τα ίδια όρια ενός DB Attribute που όρισε στην αρχή. Με τη βοήθεια του κουμπιού “Update range” έχει τη δυνατότητα να αλλάξει τα όρια και κατ’επέκταση το κομμάτι των MF των οποίων έχει άμεση εποπτεία στο panel.

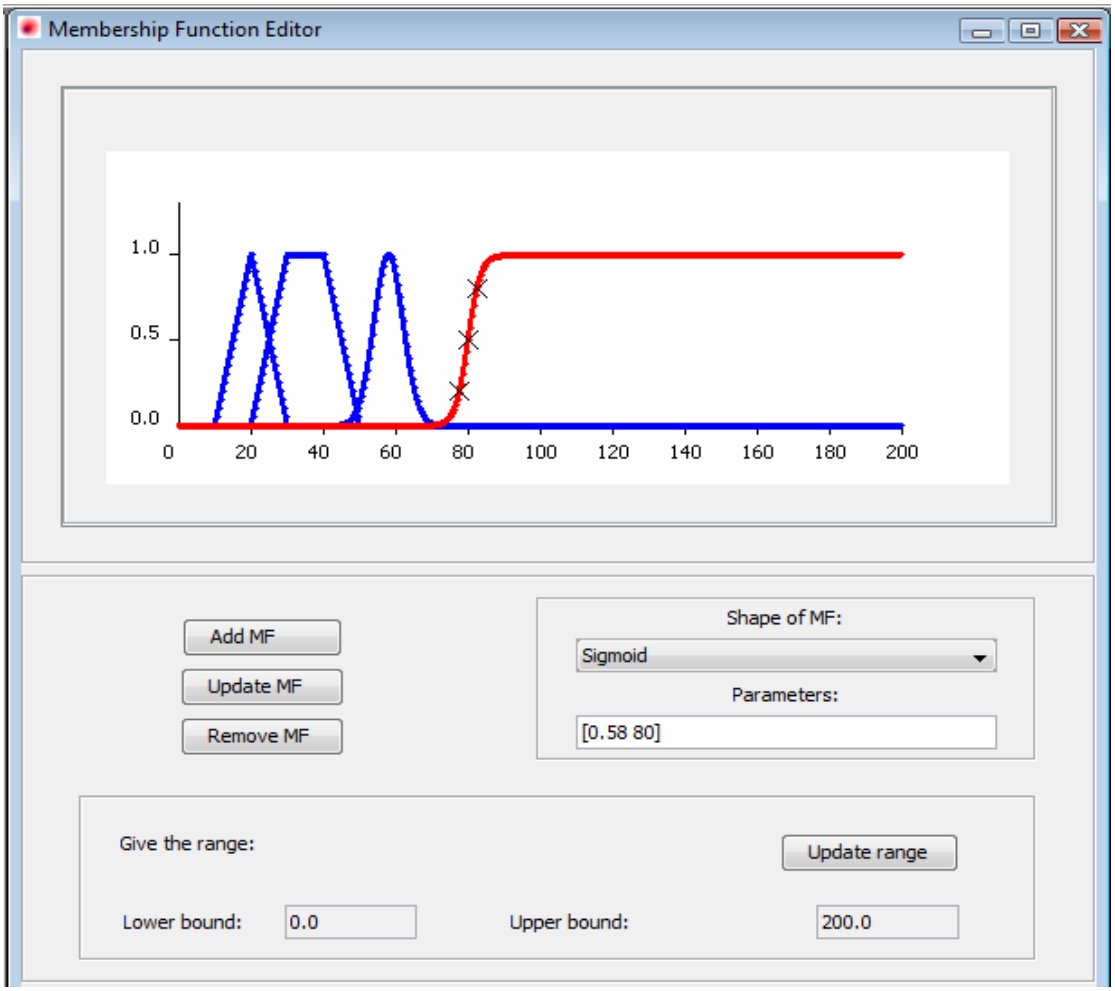
Το σχήμα 106 δείχνει ένα σύνολο από MF με όρια προβολής από 10 εώς 100. Με το πάτημα του κουμπιού “Update range” εμφανίζεται το frame του σχήματος 107. Θέτουμε τα νέα όρια σε 0 και 200. Το σχήμα 108 δείχνει το σύνολο των MF με τα νέα όρια. Όταν ο χρήστης εισάγει τα νέα όρια γίνεται έλεγχος για κενά όρια (σχήμα 109), έλεγχος προκειμένου το κάτω όριο να μην είναι μεγαλύτερο ή ίσο από το πάνω (σχήμα 110) καθώς και έλεγχος έτσι ώστε τα δύο όρια να είναι αριθμοί (σχήμα 111). Τέλος ο χρήστης εάν αλλάξει γνώμη και δεν επιθυμεί πλέον να τροποποιήσει τα όρια δύναται με το πάτημα του “Cancel” να διατηρήσει τα προηγούμενα όρια.



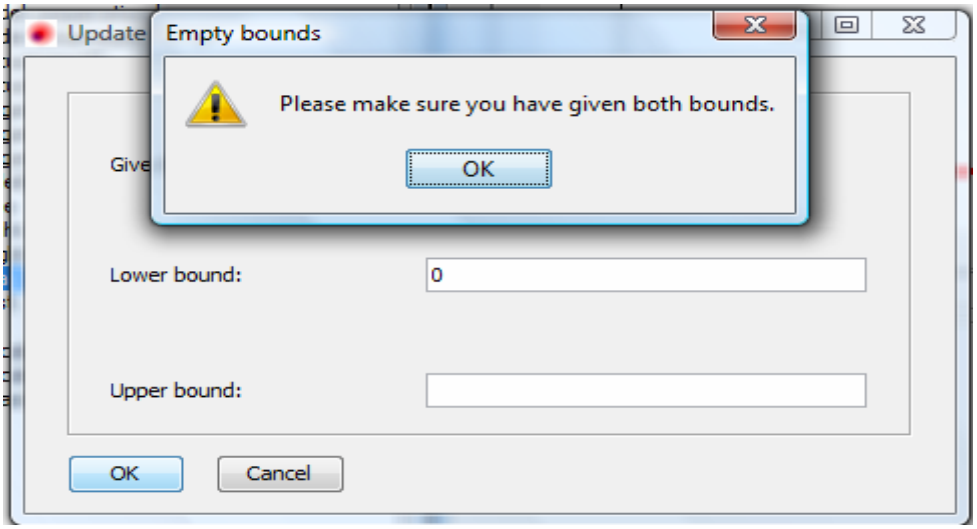
Σχήμα 106. MFs πριν την αλλαγή των ορίων.



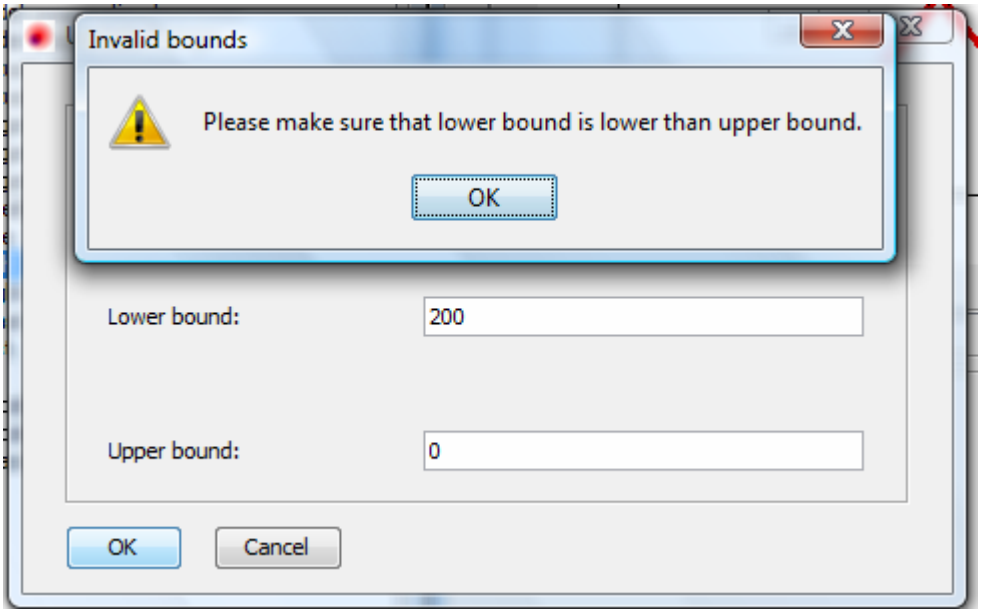
Σχήμα 107. Φόρμα αλλαγής ορίων.



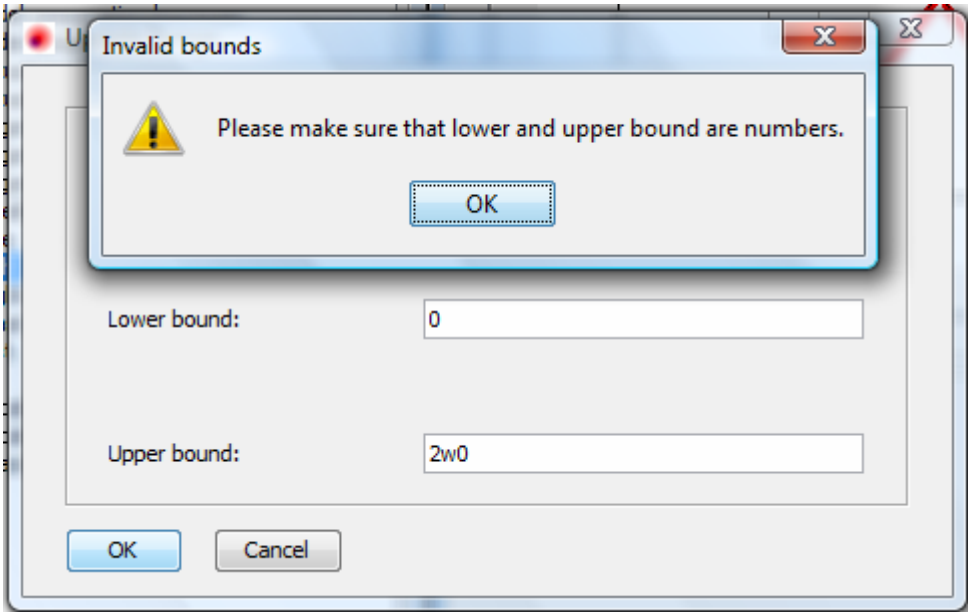
Σχήμα 108. MFs μετά την αλλαγή των ορίων.



Σχήμα 109. Προειδοποιητικό μήνυμα για εισαγωγή και των δύο ορίων.



Σχήμα 110. Προειδοποιητικό μήνυμα για κάτω όριο μικρότερο από άνω όριο.



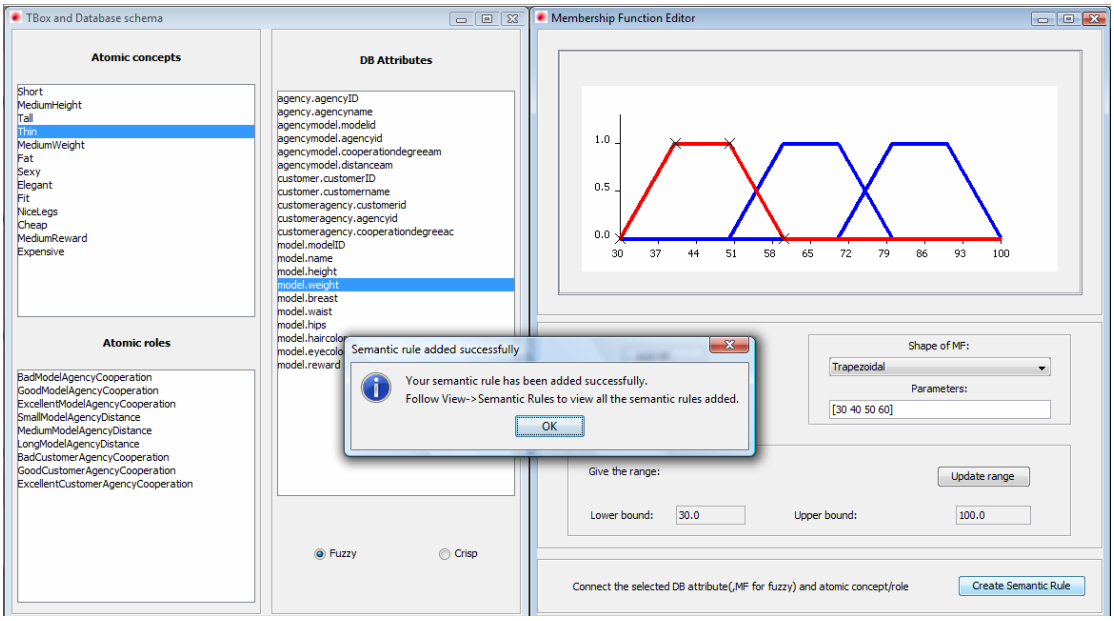
Σχήμα 111. Προειδοποιητικό μήνυμα για εισαγωγή αριθμών στη θέση των ορίων.

5.4.6 Υποσύστημα δημιουργίας, προβολής και αφαίρεσης SR

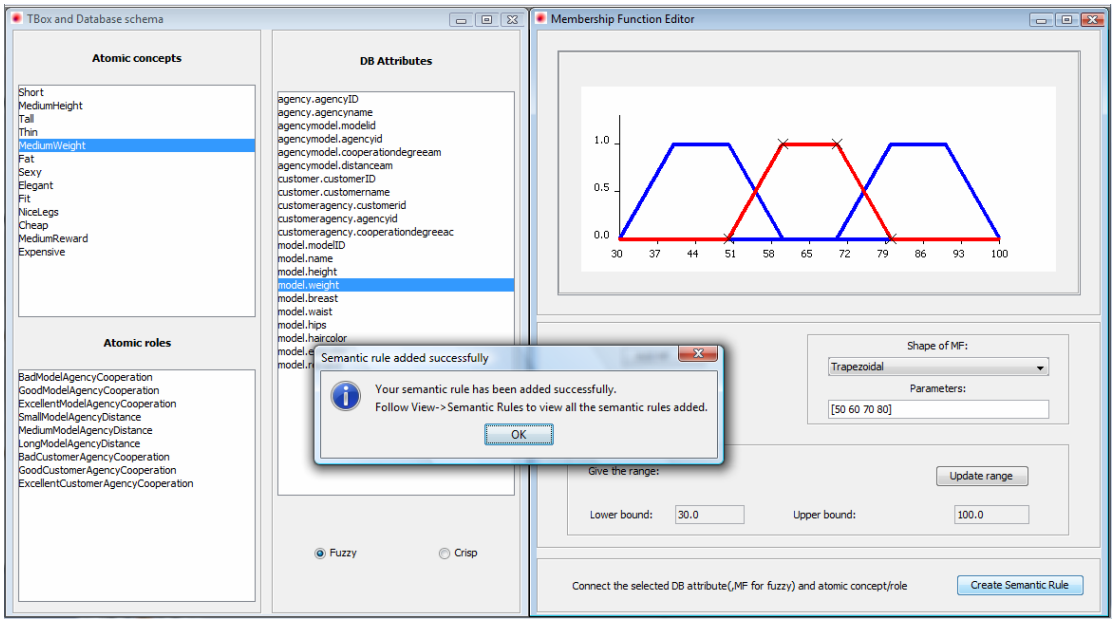
Όπως έγινε κατανοητό από τα παραπάνω απώτερος σκοπός του χρήστη είναι η δημιουργία SR, δηλαδή των δομών εκείνων που θα επιτρέψουν στο χρήστη να δώσει ακριβείς οδηγίες στο σύστημα σχετικά με το ABox που αυτό θα παράγει. Ένας SR αποτελείται από τρία τμήματα: ένα DB Attribute το οποίο βρίσκεται στην αντίστοιχη λίστα στο αριστερό frame, μία MF και μία ατομική έννοια ή ρολο. Επιπλέον ο SR εμπίπτει σε δύο κατηγορίες, ανάλογα με το ποιο από τα δύο radio buttons είναι επιλεγμένα για το DB Attribute: το "Fuzzy" για την παραγωγή των ασαφών ισχυρισμών ή το "Crisp" για την παραγωγή αντίστοιχα των μη ασαφών ισχυρισμών.

Αυτό που απαιτείται από το χρήστη για την παραγωγή του SR είναι να έχει ταυτόχρονα επιλεγμένο ένα DB Attribute, μια ατομική έννοια ή ρόλο καθώς και μία MF την οποία έχει προηγουμένως δημιουργήσει. Στο σημείο αυτό να επισημάνουμε ότι δεν επιτρέπεται στο χρήστη να κάνει μια επιλογή ταυτόχρονα από τη λίστα των atomic concepts και τη λίστα των atomic roles. Αυτό είναι λογικό εφόσον ένας SR δεν μπορεί να δημιουργηθεί συναρτήσει-ταυτόχρονα- και ενός ατομικού ρόλου και μιας ατομικής έννοιας.

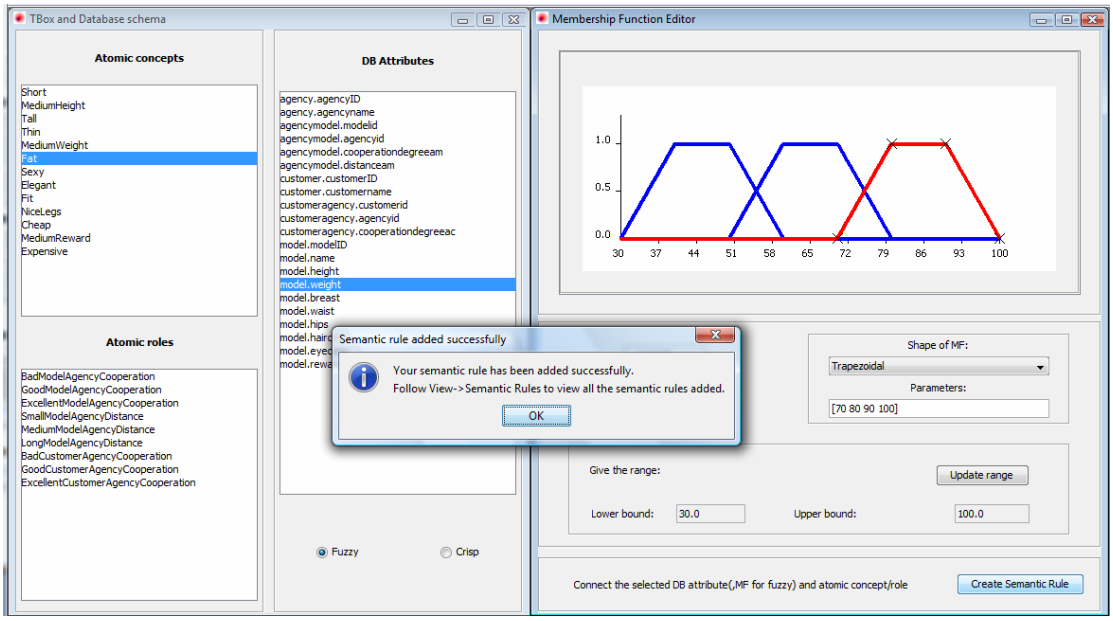
Το πρώτο παράδειγμα που θα δώσουμε θα αφορά την επιτυχημένη δημιουργία ενός SR που θα απαρτίζεται από το atomic concept Thin, το DB Attribute Model.weight καθώς επίσης και μία MF που έχει δημιουργηθεί γι'αυτό το σκοπό. Το σχήμα 112 δείχνει αυτήν την περίπτωση. Ακολουθούν στα σχήματα 113 και 114 η δημιουργία άλλων δύο SR με ίδιο DB Attribute αλλά διαφορετικές ατομικές έννοιες (MediumWeight και Fat) και MF. Το μέγεθος των σχημάτων δεν επιτρέπει να ειδωθούν πολλές λεπτομέρειες ώστοσο τα βασικά δομικά στοιχεία του κανόνα που είναι επιλεγμένα (atomic concept, DB Attribute και MF) είναι αρκετά ευδιάκριτα. Ένα μήνυμα που φαίνεται σε μεγαλύτερο μέγεθος στο σχήμα 115 πληροφορεί το χρήστη για την επιτυχημένη δημιουργία του κανόνα.



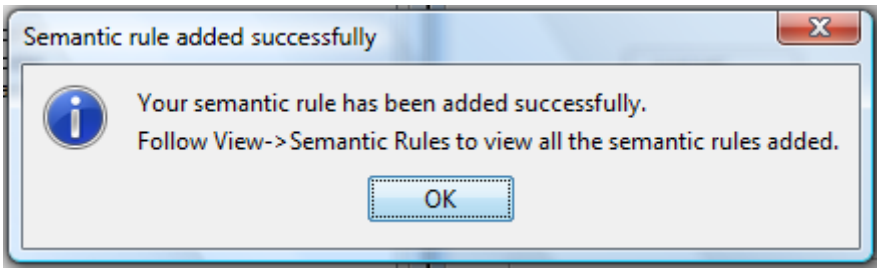
Σχήμα 112. Επιτυχημένη δημιουργία SR 1.



Σχήμα 113. Επιτυχημένη δημιουργία SR 2.



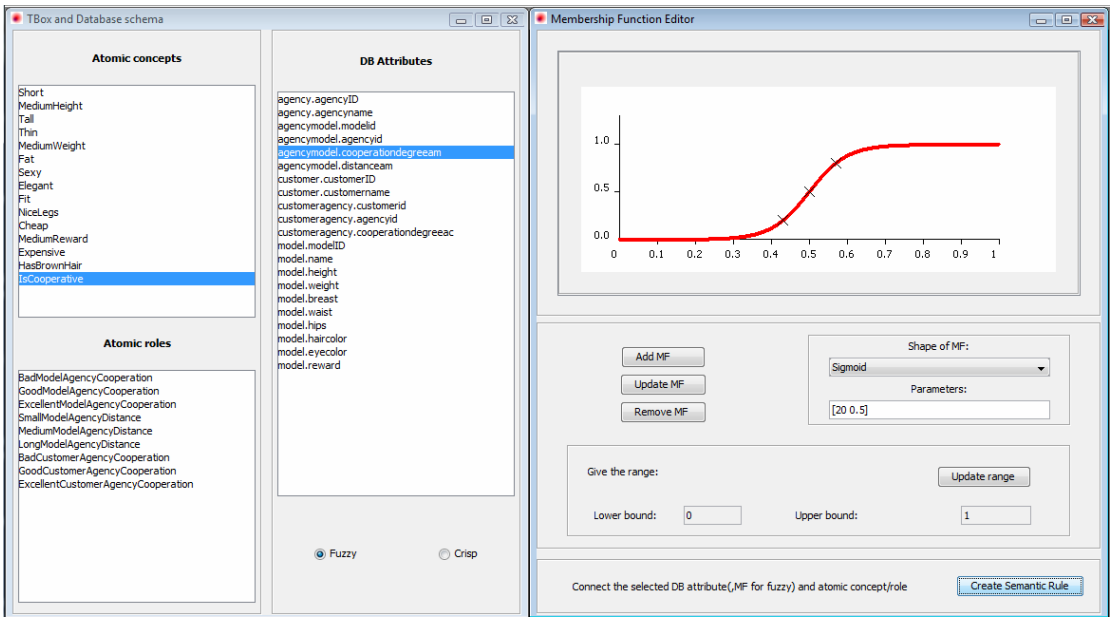
Σχήμα 114. Επιτυχημένη δημιουργία SR 3.



Σχήμα 115. Μήνυμα του συστήματος για επιτυχημένη δημιουργία του σημσιολογικού κανόνα

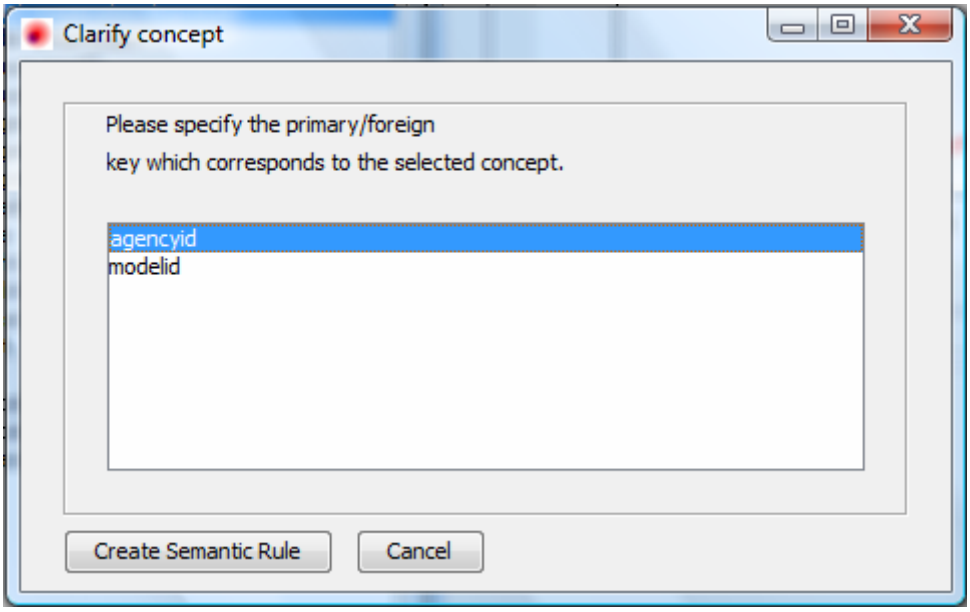
Σε όλες τις παραπάνω περιπτώσεις το DB Attribute άνηκε σε μία σχέση στην οποία το πρωτεύον κλειδί απαρτιζόταν από μία στήλη (στην περίπτωση του Model.weight το Model.ModelID). Σε μία τέτοια περίπτωση είναι αρκετά σαφής η παραγωγή του ABox. Εάν όμως το πρωτεύον κλειδί απαρτίζεται από δύο στήλες (π.χ. στο AgencyModel, το ModelID και το AgencyID) το σύστημα αδυνατεί να γνωρίζει ποια από τις δύο στήλες θα χρησιμοποιηθεί στην παραγωγή του ABox. Με άλλα λόγια στα πλαίσια του προηγούμενου παραδείγματος το σύστημα δεν ξέρει εάν η ατομική έννοια που χρησιμοποιήθηκε στην παραγωγή του κανόνα θα αφορά τα μοντέλα ή τα πρακτορεία. Για το λόγο αυτό κατά την παραγωγή ενός τέτοιου είδους κανόνα το σύστημα ζητά από το χρήστη να διασαφηνίσει σε

ποια από τις δύο (ή περισσότερες) στήλες αναφέρεται προτού προχωρήσει στην παραγωγή του. Το σχήμα 116 δείχνει μία τέτοια περίπτωση.

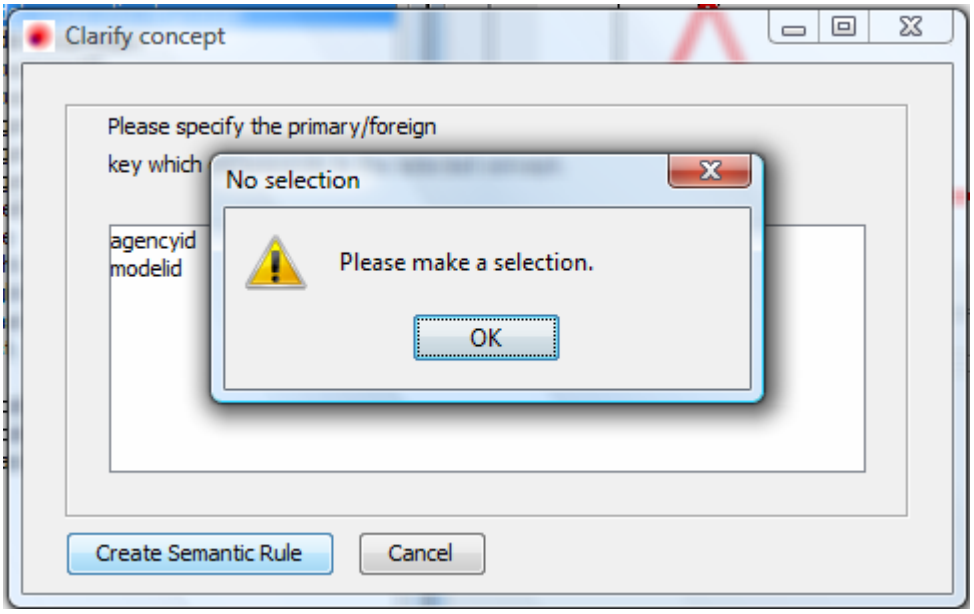


Σχήμα 116. Δημιουργία SR 4.

Προκειμένου να διευκρινιστεί η παραπάνω πληροφορία κατά το πάτημα του κουμπιού "Create Semantic Rule" εμφανίζετια το frame του σχήματος 117. Στο interface αυτό το σύστημα αφού πρώτα έχει πραγματοποιήσει μια επικοινωνία με τη βάση προκειμένου να διαπιστώσει ποιες είναι οι στήλες που απαρτίζουν το πρωτεύον κλειδί, ρωτά το χρήστη με βάση ποια από τις δύο στήλες θα γίνει η παραγωγή των ασαφών ισχυρισμών. Με το πάτημα του κουμπιού "Create Semantic Rule" (του interface του σχήματος 117 πλέον) έχουμε το μήνυμα επιτυχημένης επιβεβαίωσης προσθήκης του SR. Απαραίτητη προϋπόθεση βέβαια είναι να έχει επιλεγεί ακριβώς ένα DB Attribute από αυτά που εμφανίζονται στη λίστα, αλλιώς έχουμε την περίπτωση του σχήματος 118. Επιπλέον ο χρήστης έχει τη δυνατότητα σε περίπτωση που διαπιστώσει ότι δεν επιθυμεί πλέον τη δημιουργία του κανόνα να εγκαταλείψει το interface χωρίς να γίνει καμία ενέργεια με το πάτημα του κουμπιού "Cancel". Στο σημείο αυτό να τονίσουμε ότι το παραπάνω interface δεν εμφανίζεται σε περίπτωση που το πρωτεύον κλειδί απαρτίζεται από μία μόνο στήλη καθώς δεν έχει νόημα ο χρήστης να επιλέξει ανάμεσα σε μία στήλες.

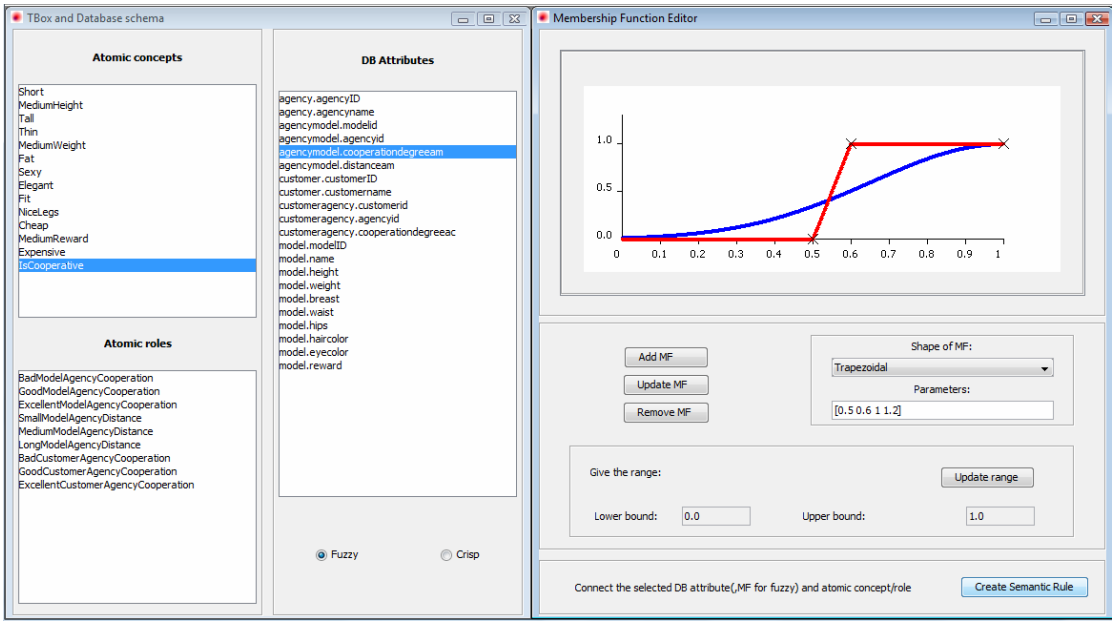


Σχήμα 117. Interface διευκρίνισης ατομικής έννοιας.

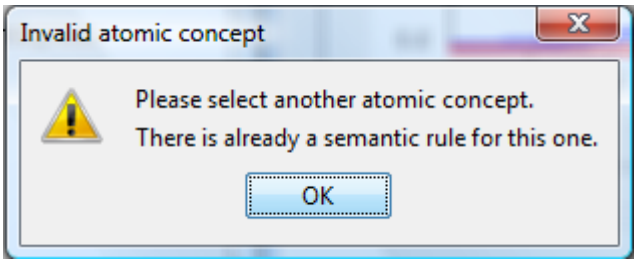


Σχήμα 118. Προειδοποιητικό μήνυμα σε περίπτωση κενής επιλογής.

Επιπλέον στην περίπτωση κατά την οποία ο χρήστης επιχειρήσει να δημιουργήσει έναν κανόνα που θα αναφέρεται στο ίδιο DB Attribute (σχήμα 119) αυτό δε γίνεται αποδεκτό απο το σύστημα (σχήμα 120) καθώς μια τέτοια περίπτωση θα προκαλούσε ασυνέπεια στην παραγωγή του ABox.



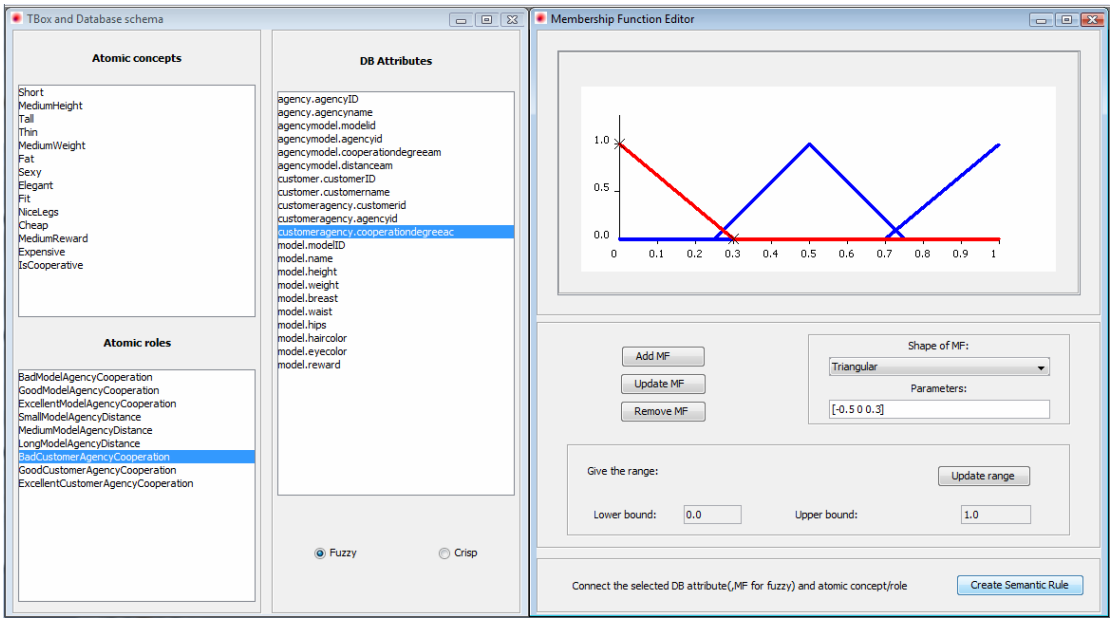
Σχήμα 119. Απόπειρα δημιουργίας SR για την ίδια ατομική έννοια.



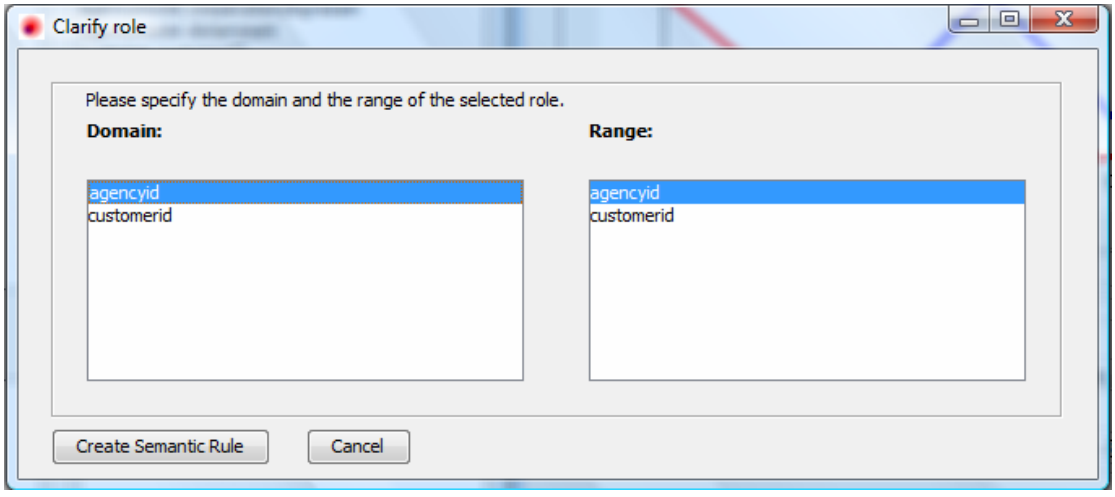
Σχήμα 120. Προειδοποιητικό μήνυμα ως αποτέλεσμα της εντολής του σχήματος 119.

Προχωράμε στη δημιουργία SR αλλά αυτή τη φορά για ατομικούς ρόλους. Στην περίπτωση των ατομικών ρόλων, εφόσον ένας ρόλος εξ'ορισμού αναφέρεται σε δύο άτομα, ανακύπτει το πρόβλημα προσδιορισμού του ποιο από τα δύο άτομα είναι το domain και ποιο το range. Δηλαδή ο πίνακας της βάσης δεδομένων στον οποίο αναφέρεται η παραγωγή του κανόνα περιέχει πρωτεύον κλειδί που απαρτίζεται από τουλάχιστον δύο στήλες, αλλά ποια από τις δύο αναφέρεται στο domain και ποια στο range. Πρόκειται για μία πληροφορία την οποία χρειάζεται το σύστημα για να κάνει σωστή παραγωγή ασαφών υποθέσεων καθώς μια εναλλαγή domain και range συνεπάγεται αλλαγή στη σημασιολογία. Π.χ. ο ρόλος BadCustomerAgencyCooperation με domain το CustomerID και range το AgencyID ερμηνεύεται ως ο βαθμός συνεργασίας από την οπτική γωνία του πελάτη ενώ αντίστροφα από την οπτική γωνία του πρακτορείου. Έτσι προκειμένου ο χρήστης να ορίσει με σαφήνεια τις επιθυμίες του εμφανίζεται το interface του σχήματος 122. Ο χρήστης κάνει τις επιλογές του

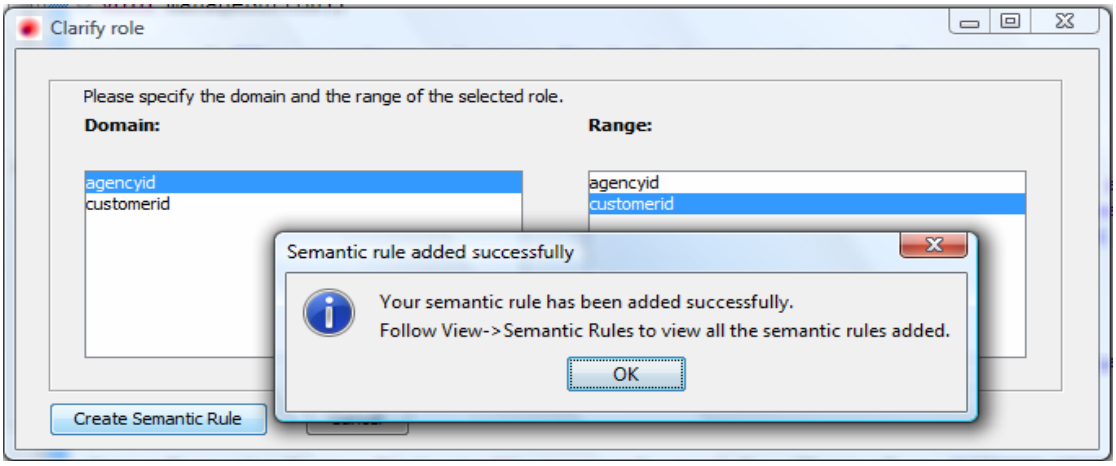
ανάλογα με τη σημασιολογία που θέλει να έχει το προς παραγωγή ABox και όπως φαίνεται στο σχήμα 123 ο SR δημιουργείται επιτυχημένα. Ωστόσο σε περίπτωση που ο χρήστης επιλέξει ως domain και range την ίδια στήλη εμφανίζεται το προειδοποιητικό μήνυμα του σχήματος 124.



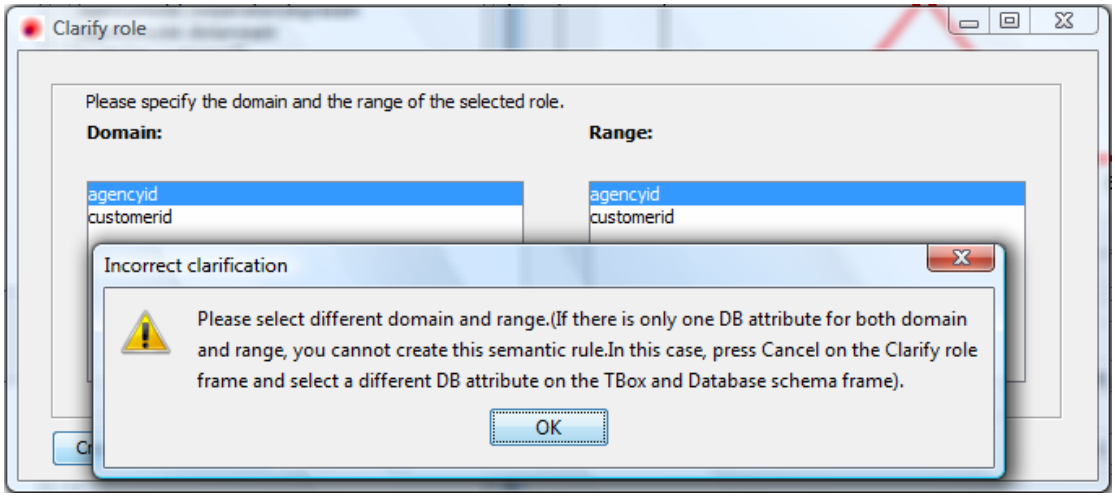
Σχήμα 121. Δημιουργία SR 5.



Σχήμα 122. Interface διευκρίνισης του SR 5.

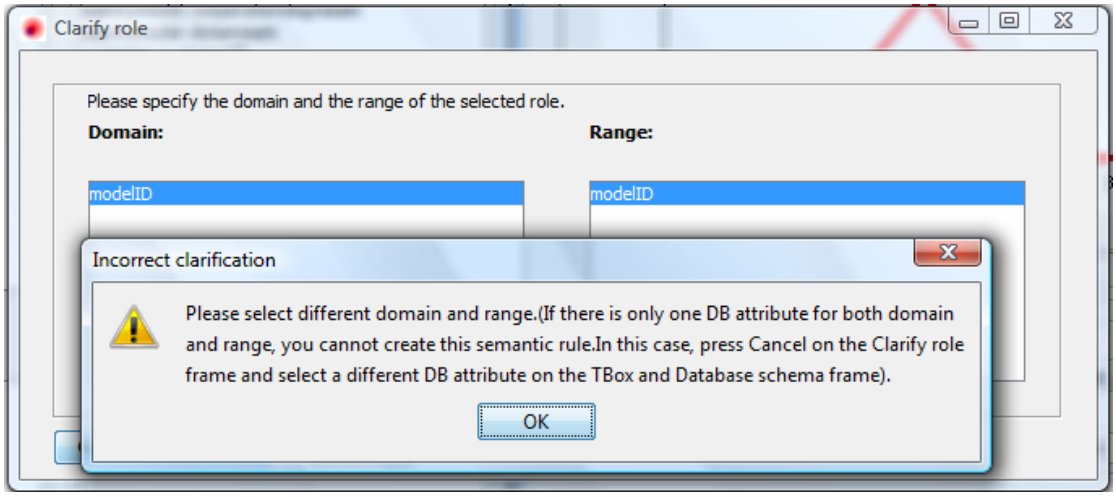


Σχήμα 123. Επιτυχημένη δημιουργία SR 5.



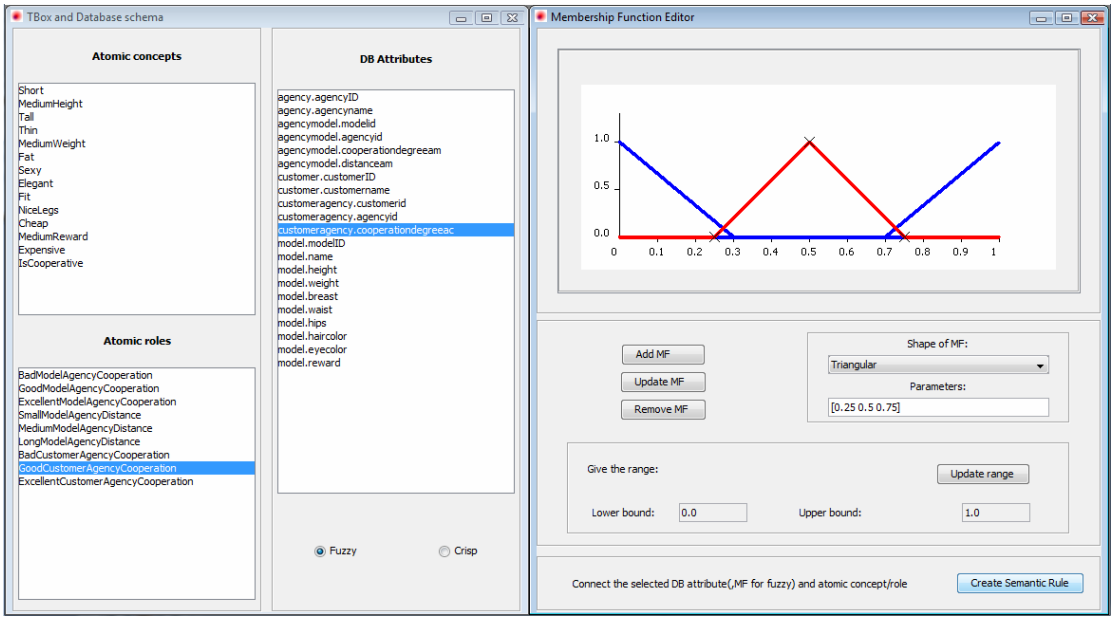
Σχήμα 124. Προειδοποιητικό μήνυμα σε περίπτωση επιλογής ίδιας στήλης για domain και range.

Ωστόσο παρατηρούμε ότι το προειδοποιητικό μήνυμα του σχήματος 124 περιέχει ένα επιπλέον μήνυμα μέσα σε ένα ζεύγος παρενθέσεων. Το μήνυμα αυτό γίνεται περισσότερο κατανοητό στο context του επόμενου σχήματος. Σε αυτή την περίπτωση ο χρήστης έχει επιχειρήσει να δημιουργήσει ένα SR με ατομικό ρόλο, ο οποίος όμως αναφέρεται σε ένα DB Attribute το οποίο έχει αντίστοιχο πρωτεύον κλειδί μιας στήλης. Αυτό σημαίνει ότι το domain και το range θα πρέπει να αντιστοιχιθούν στο ίδιο άτομο, πράγμα μη επιθυμητό. Έτσι εμφανίζεται το μήνυμα του σχήματος 124 που πληροφορεί το χρήστη ότι δεν έχει τη δυνατότητα να παράγει αυτόν τον SR και τον καλεί να πατήσει “Cancel” στο frame “Clarify role” και να δημιουργήσει το SR με κάποιο άλλο DB Attribute.

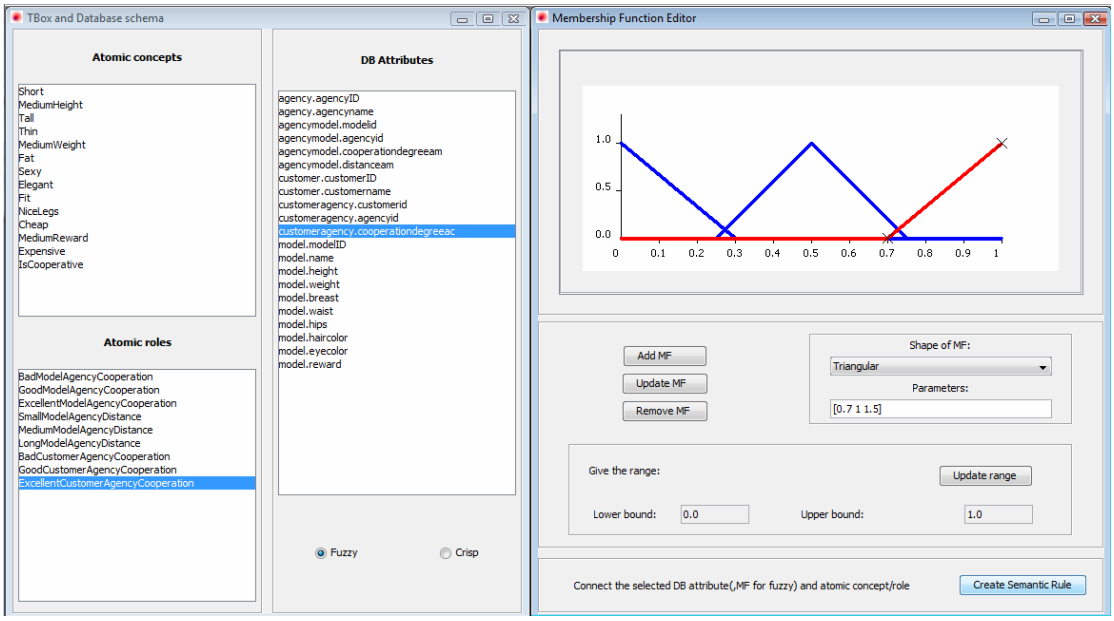


Σχήμα 125. Προειδοποιητικό μήνυμα σε περίπτωση που το πρωτεύον κλειδί αποτελείται από μία μόνο στήλη.

Με τον ίδιο τρόπο προχωράμε στην παραγωγή του SR 6 ο οποίος συνδυάζει το DB Attribute CustomerAgency.CooperationDegreeAC, τον ατομικό ρόλο GoodCustomerAgencyCooperation και την επιλεγμένη MF. Το σχήμα 126 αντιστοιχεί στον κανόνα 6 και το σχήμα 127 στον κανόνα 7 ο οποίος αναφέρεται στο ίδιο DB Attribute, στον ρόλο ExcellentCustomerAgencyCooperation και σε μία τρίτη MF.

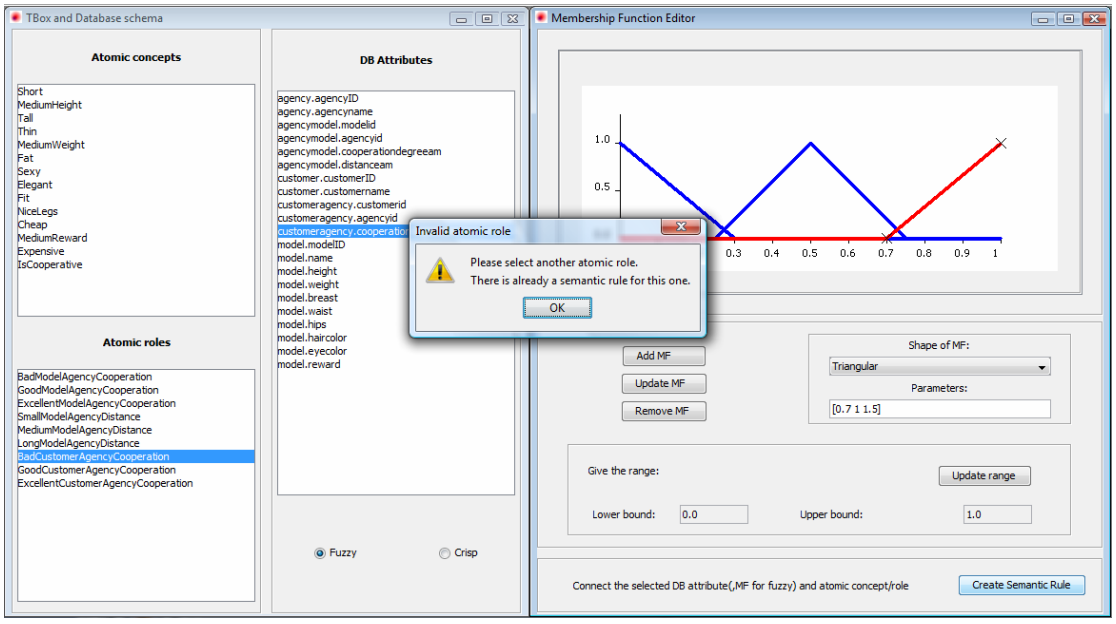


Σχήμα 126. Δημιουργία SR 6.

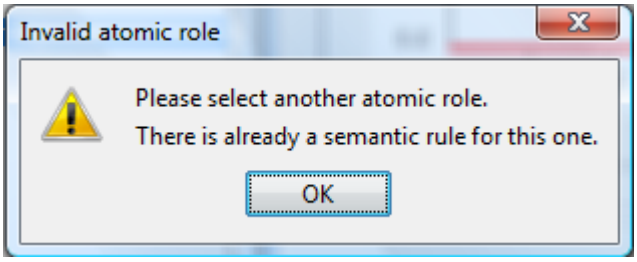


Σχήμα 127. Δημιουργία SR 7.

Τέλος όπως και στην περίπτωση των κανόνων που αναφέρονται σε ατομική έννοια έτσι και εδώ δεν επιτρέπεται η δημιουργία δύο SR που αναφέρονται στον ίδιο ατομικό ρόλο. Αυτό συνέβη και στην περίπτωση του σχήματος 128 του οποίου το προειδοποιητικό μήνυμα βλέπουμε καθαρότερα στο σχήμα 129.

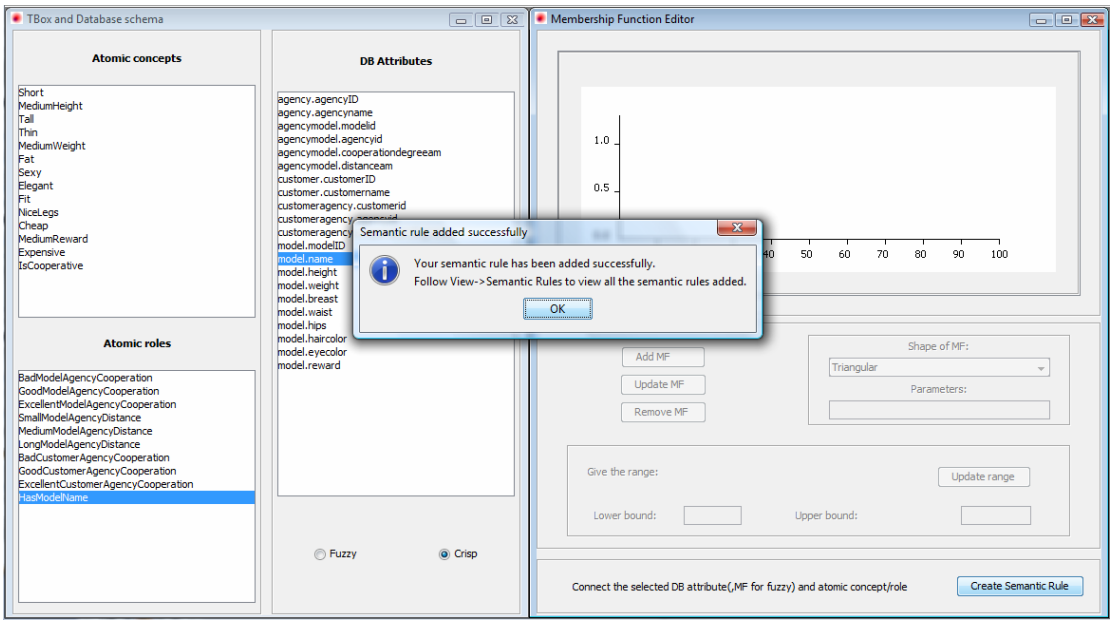


Σχήμα 128. Απόπειρα δημιουργίας SR για την ίδια ατομική έννοια.

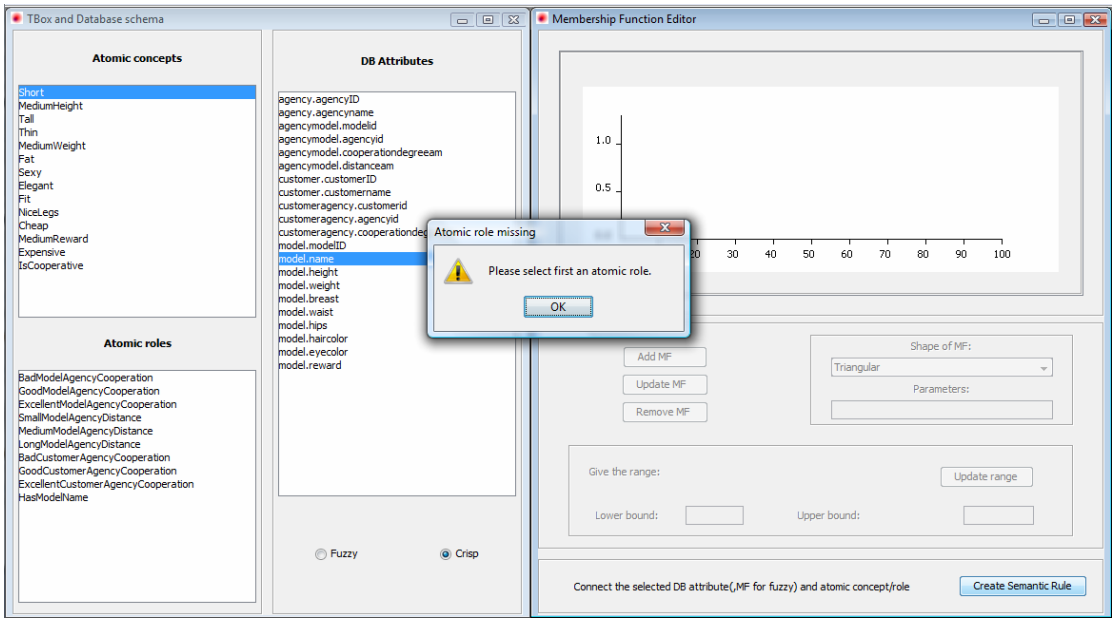


Σχήμα 129. Προειδοποιητικό μήνυμα ως αποτέλεσμα της εντολής του σχήματος 128.

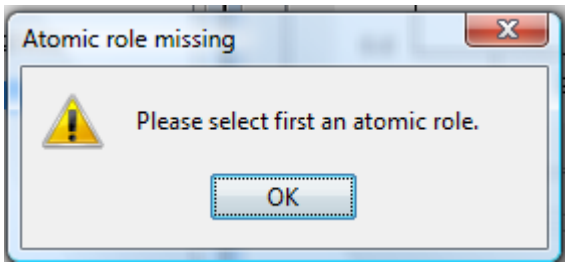
Εάν παρατηρήσουμε τα σχήματα που αντιστοιχούν στην επιτυχημένη παραγωγή των πρώτων SR 1,2,3,4,5,6,7 θα δούμε ότι σε κάθε περίπτωση είναι επιλεγμένο το radio button “Fuzzy”. Τι γίνεται ωστόσο στην περίπτωση που για κάποιο DB Attribute είναι επιλεγμένο το radio button “Crisp”; Αυτό το οποίο συμβαίνει είναι η παραγωγή SR χωρίς τη βοήθεια ορισμού μιας συνάρτησης MF (εφόσον crisp assertions είναι αυτά που θα παραχθούν) και με τον ορισμό εκτός από το DB Attribute ενός ατομικού ρόλου που προέρχεται από την αντίστοιχη λίστα και όχι ατομικής έννοιας. Το σχήμα 130 δείχνει την παραγωγή ενός τέτοιου SR. Σε περίπτωση που ο χρήστης επιχειρήσει να δημιουργήσει ένα σημασιολογικά κανόνα με επιλεγμένο το “Crisp” radio button και μία ατομική έννοια θα λάβει το προειδοποιητικό μήνυμα του σχήματος 131 που φαίνεται καθαρότερα στο σχήμα 133.



Σχήμα 130. Επιτυχημένη παραγωγή SR 8.

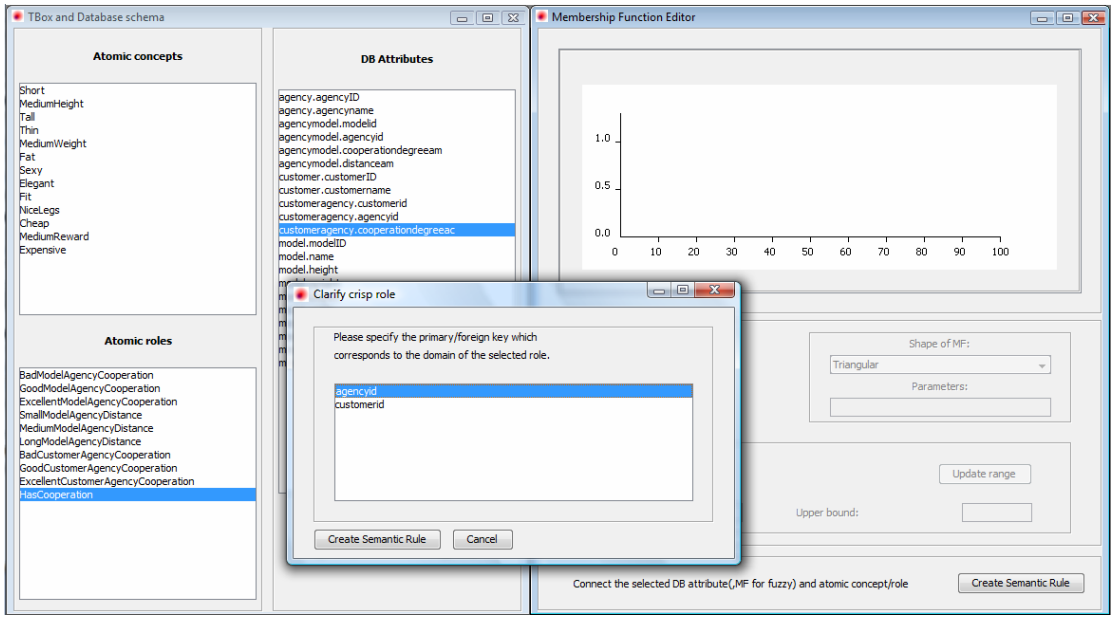


Σχήμα 131. Προσπάθεια παραγωγής crisp κανόνα με ατομική έννοια



Σχήμα 132. Προειδοποιητικό μήνυμα ως αποτέλεσμα της εντολής του σχήματος 131.

Εάν ο χρήστης επιθυμεί την παραγωγή ενός crisp SR ο οποίος αφορά DB Attribute το οποίο με τη σειρά του έχει πρωτεύον κλειδί δύο ή περισσότερων στηλών θα πρέπει να διευκρινίσει στο σύστημα ποια από τις εν λόγω στήλες θα χρησιμοποιηθεί στην παραγωγή των crisp assertions. Μια τέτοια περίπτωση απεικονίζει το σχήμα 133. Το σχήμα 134 δείχνει με περισσότερη σαφήνεια το νέο interface που αναδύεται καθώς επίσης και την επιτυχημένη έκβαση της δημιουργίας του κανόνα. Ο λόγος για τον οποίο ζήτείται αυτή η επιπλέον πληροφορία από το χρήστη είναι για να καθοριστεί σε ποια άτομα θα αναφέρονται τα assertions που θα δημιουργηθούν.

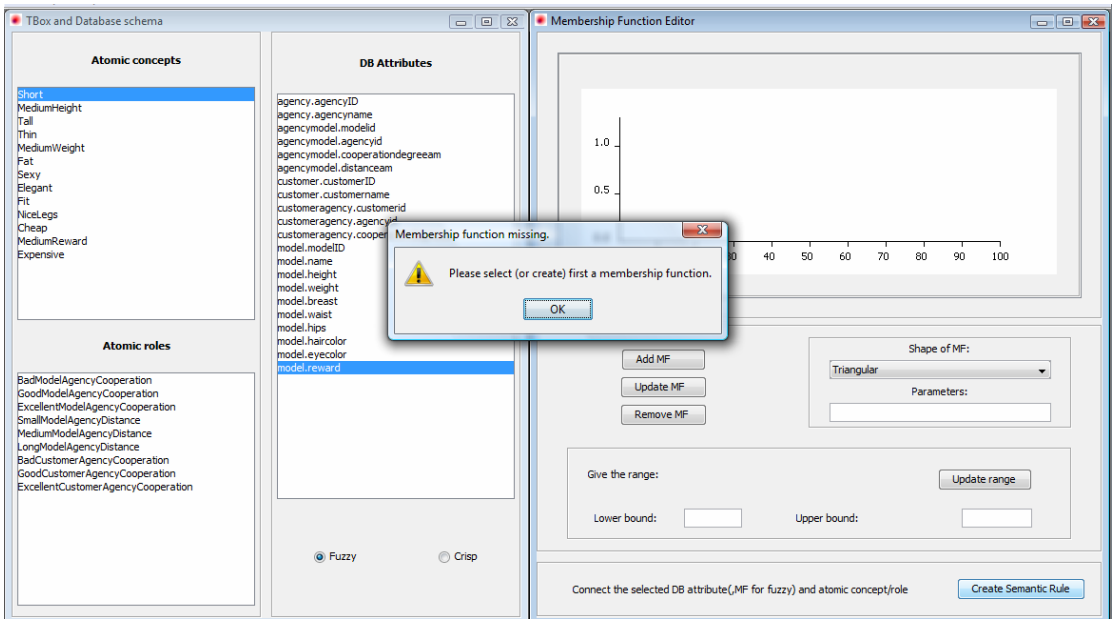


Σχήμα 133. Δημιουργία crisp SR με DB Attribute που έχει πρωτεύον κλειδί δύο ή περισσότερων στηλών.

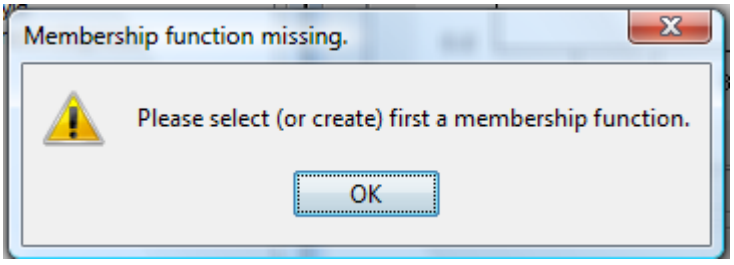
Σχήμα 134. Επιτυχημένη δημιουργία του SR του σχήματος 133.

Στο σημείο αυτό θα παραθέσουμε κάποιους γενικούς περιορισμούς που τίθενται κατά τη δημιουργία SR. Όπως δείχνει το σχήμα 133 δεν επιτρέπεται η δημιουργία SR χωρίς να έχει προηγουμένως επιλεγεί κάποια ατομική έννοια ή ρόλος, για την περιπτωση που το fuzzy radio button είναι επιλεγμένο. Επιπλέον πάλι για το fuzzy radio button είναι απαραίτητο να

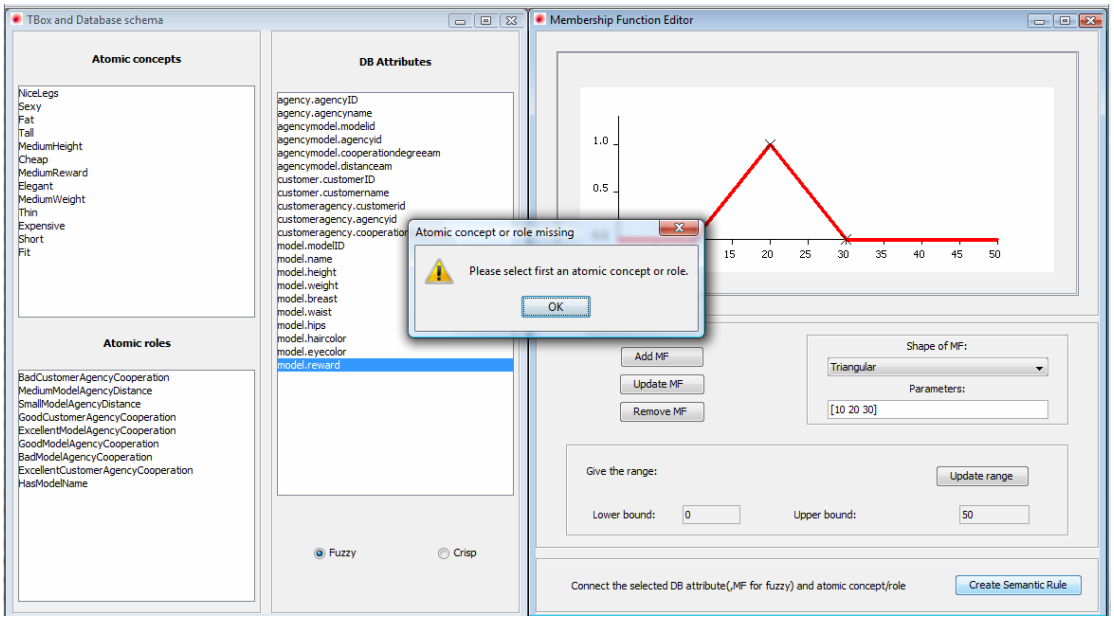
είναι επιλεγμένη (αφού πρώτα έχει δημιουργηθεί απαραίτητα) μία MF, κάτι που το σχήμα 135 δείχνει λεπτομερέστερα. Τα σχήματα 136 και 138 δείχνουν τα προειδοποιητικά μηνύματα των σχημάτων 135 και 137 με ευκρινέστερο τρόπο.



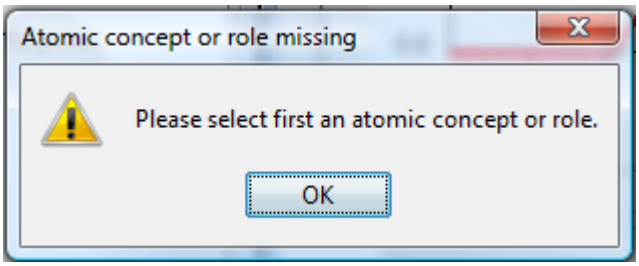
Σχήμα 135. Απόπειρα δημιουργίας SR χωρίς επιλεγμένη MF.



Σχήμα 136. Το προειδοποιητικό μήνυμα του σχήματος 135.



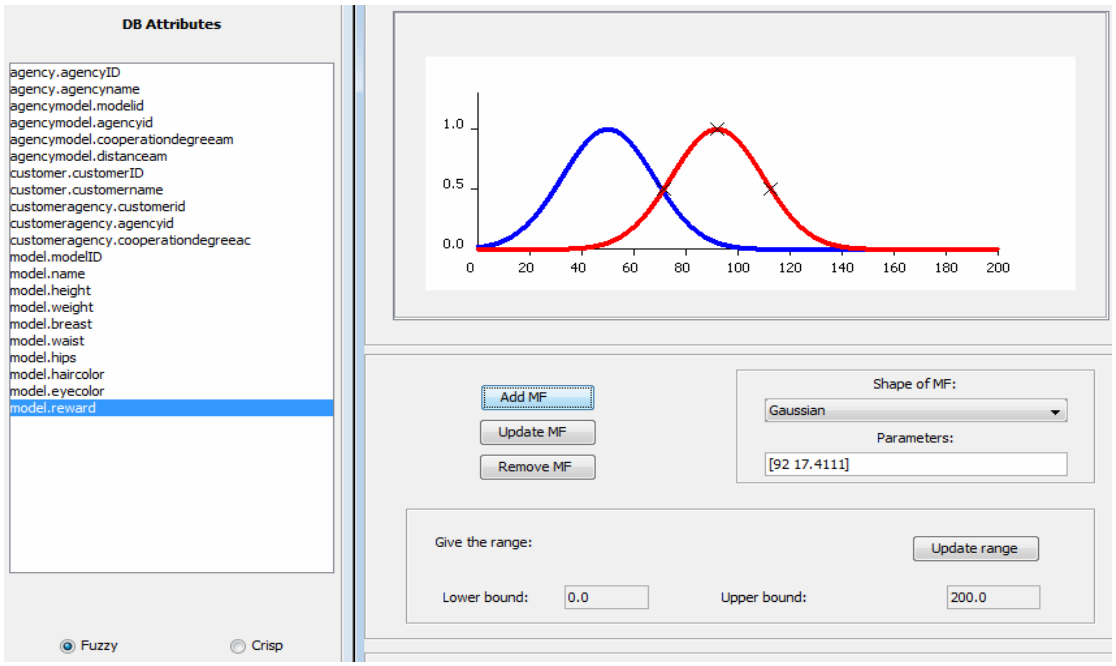
Σχήμα 137. Απόπειρα δημιουργίας SR χωρίς επιλεγμένη ατομική έννοια ή ρόλο.



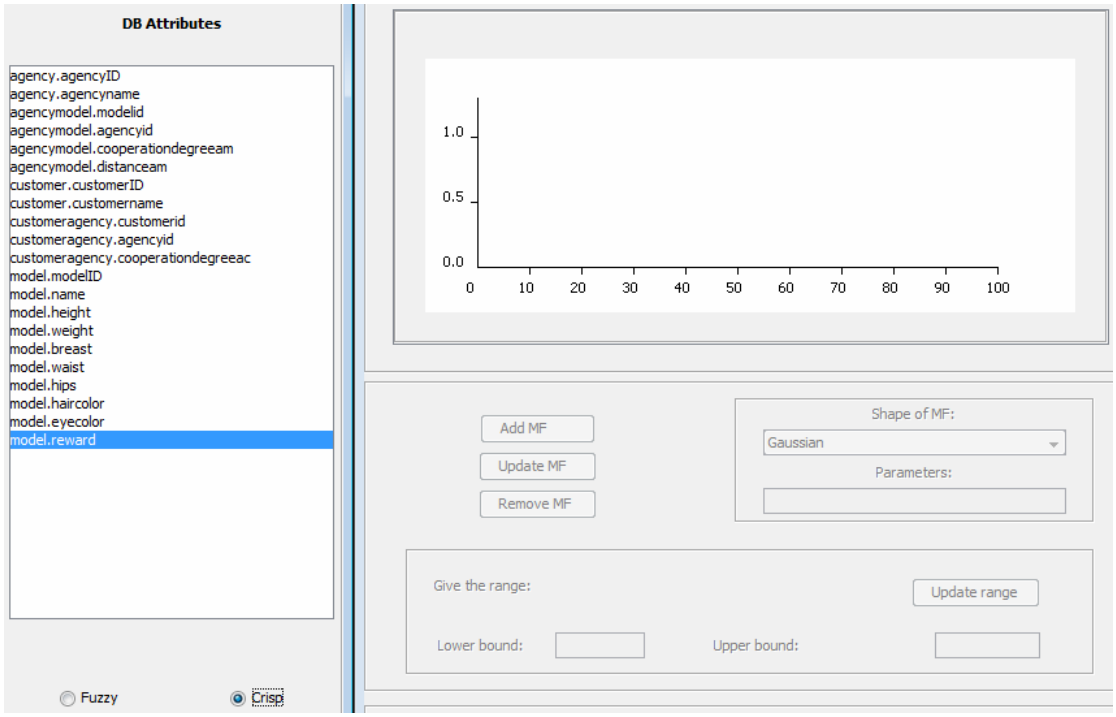
Σχήμα 138. Το προειδοποιητικό μήνυμα του σχήματος 137.

Στο σημείο αυτό θα εξετάσουμε τον τρόπο με το τον οποίο αλληλεπιδρούν τα radio buttons "Fuzzy" και "Crisp" με το υπόλοιπο γραφικό κομμάτι του interface. Αρχικά ορίζουμε δύο συναρτήσεις MF για ένα DB Attribute με επιλεγμένο το "Fuzzy" radio button. Το σχήμα 139 δείχνει το αναμενόμενο αποτέλεσμα. Στη συνέχεια πατάμε το "Crisp" radio button. Το σχήμα 140 δείχνει ότι το υποσύστημα επεξεργασίας MF απενεργοποιείται και οι ορισμένες MF απομακρύνονται, πράγμα λογικό εφόσον ένα DB Attribute για να χρησιμοποιηθεί για την παραγωγή crisp assertions δε χρειάζεται τίποτα από αυτά. Κατόπιν μεταβαίνουμε σε ένα άλλο DB Attribute. Εφόσον το "Crisp" radio button εξακολουθεί να είναι επιλεγμένο το υποσύστημα επεξεργασίας MF δεν ενεργοποιείται εκ νέου (σχήμα 141). Κατόπιν επανεπιλέγουμε το DB Attribute του σχήματος 138 και επιλέγουμε το "Fuzzy" radio button

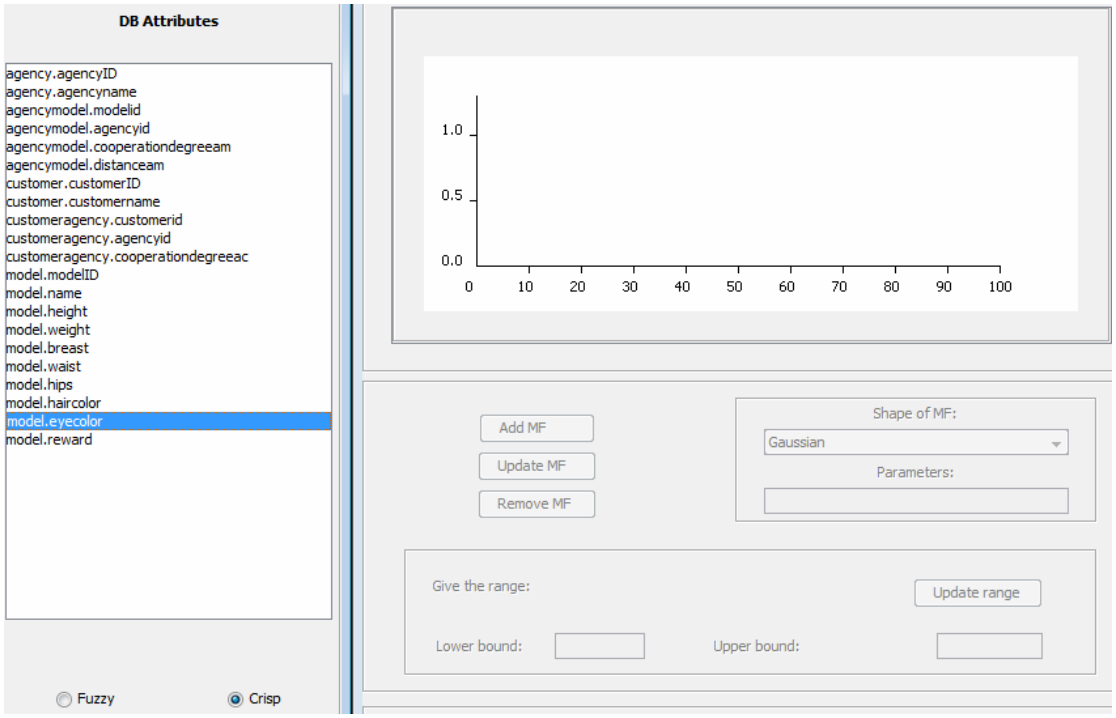
εκ νέου. Το σχήμα 142 μας δείχνει ότι το σύστημα έχει αποθηκεύσει τις MF που είχαν ορίσει πιο πριν για το “Fuzzy” radio button.



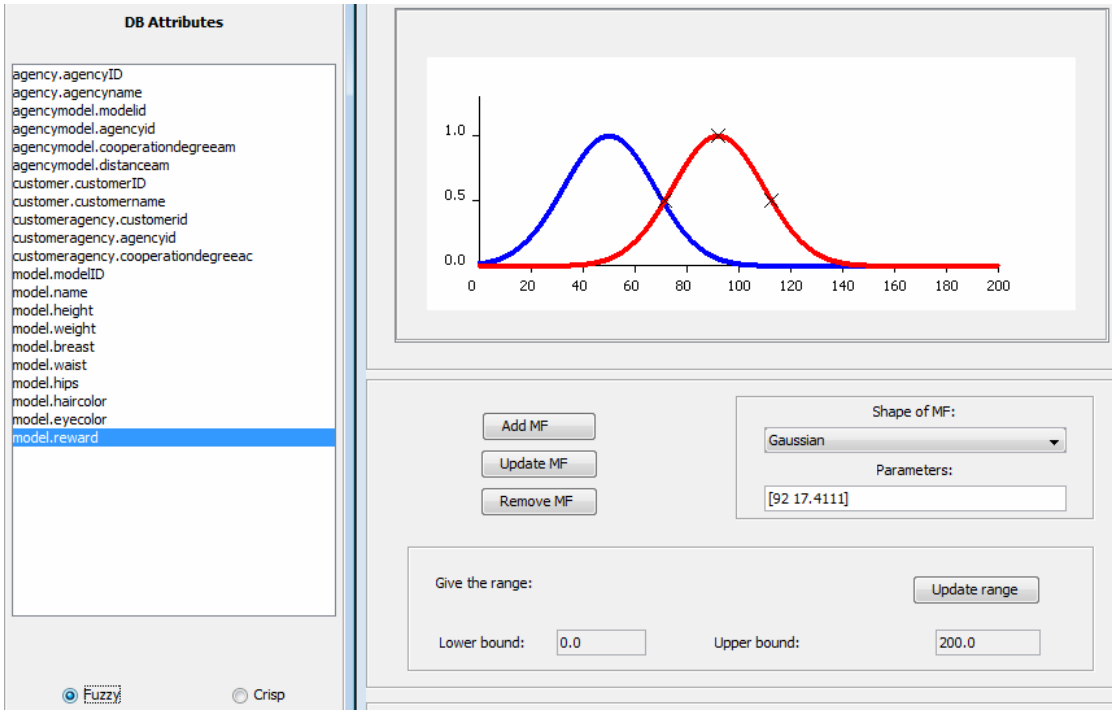
Σχήμα 139. “Fuzzy” radio button



Σχήμα 140. “Crisp” radio button.

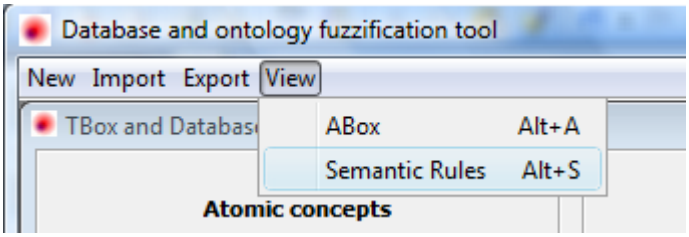


Σχήμα 141. "Crisp" radio button για διαφορετικό attribute.



Σχήμα 142. "Fuzzy" radio button εκ νέου. Οι ορισμένες MF δεν έχουν χαθεί.

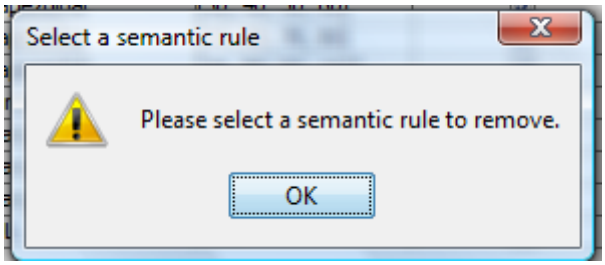
Ωστόσο τι συμβαίνει σε περίπτωση που ο χρήστης ξεχάσει ποιοι είναι οι SR που έχει ορίσει; Ή σε περίπτωση που επιθυμήσει να τροποποιήσει κάποιον από αυτούς διαγράφοντας τον και τοποθετώντας στη θέση του έναν καινούριο; Στο σημείο αυτό εξηγούμε πώς μπορούν να γίνουν τα παραπάνω ακολουθώντας τη συμβουλή που το σύστημα δίνει στο χρήστη μετά από κάθε επιτυχημένη παραγωγή κανόνα. Δηλαδή την επιλογή μενού View→Semantic Rules ή πατώντας τα πλήκτρα Alt+S, όπως δείχνει το σχήμα 143.



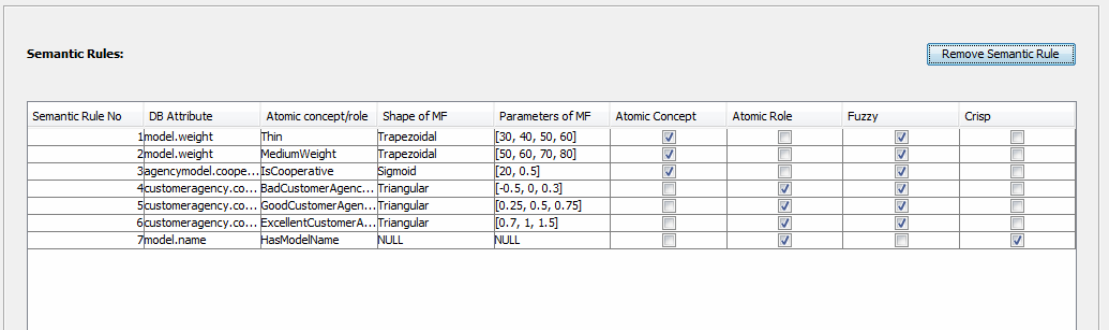
Σχήμα 143. Εντολή για προβολή των SR.

Το αντίστοιχο interface που εμφανίζεται στο χρήστη είναι αυτό του σχήματος 144. Στο frame αυτό που επιτρέπει την προβολη των SR φαίνονται οι σημασιολογικοί κανόνες που προστέθηκαν από τα σχήματα 112, 113, 114, 116, 119, 126, 127 και 130 με αύξοντες αριθμούς 1, 2, 3, 4, 5, 6, 7 και 8 αντίστοιχα. Το frame αυτό παρουσιάζει όλα όσα πρέπει να γνωρίζει ο χρήστης για τους υπάρχοντες SR: τον αύξοντα αριθμό τους, το DB Attribute, το atomic concept ή role, το σχήμα της MF και τις παραμέτρους τις, το αν εμπλέκεται atomic concept ή atomic role καθώς επίσης και το αν θα προέλθουν από αυτό το SR fuzzy ή crisp assertions. Ο χρήστης έχει τη δυνατότητα να αφαιρέσει ένα SR είτε επειδή δε θέλει να συμπεριληφθεί στην παραγωγή των assertions είτε επειδή θέλει να τον αντικαταστήσει με άλλον. Η δυνατότητα αυτή του προσφέρεται από το κουμπί “Remove Semantic Rule”. Ωστόσο πρέπει να είναι προσεκτικός γιατί θα πρέπει να έχει επιλέξει προηγούμενως έναν κανόνα αλλιώς θα λάβει το μήνυμα του σχήματος 145. Τέλος το σχήμα 146 παρουσιάζει το σύνολο των SR από τους οποίους έχουμε αφαιρέσει τον τρίτο. Παρατηρούμε ότι δεν εμφανίζεται πλέον στην παρουσίαση τους. Επιπλέον δε θα εφαρμοστεί για την παραγωγή του ABox εφόσον έχει αφαιρεθεί.

Σχήμα 144. Προβολή υπάρχοντων SR.



Σχήμα 145. Προειδοποιητικό μήνυμα για επιλογή SR προς απομάκρυνση.



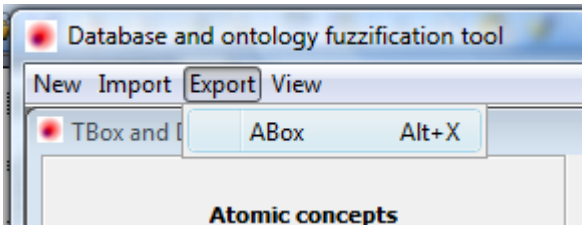
Σχήμα 146. Μετά την αφαίρεση του τρίτου SR.

5.4.7 Υποσύστημα παραγωγής και προβολής ABox

Στο σημείο αυτό θα προχωρήσουμε στο τελικό στάδιο της εφαρμογής που είναι η παραγωγή του ABox, κάτι που άλλωστε αποτελεί τον τελικό στόχο μας. Το ABox παράγεται βασιζόμενο στους SR που έχουν αποθηκευθεί. Διακρίνουμε τρεις περιπτώσεις παραγωγής ισχυρισμών που θα συμπεριληφθούν στο ABox:

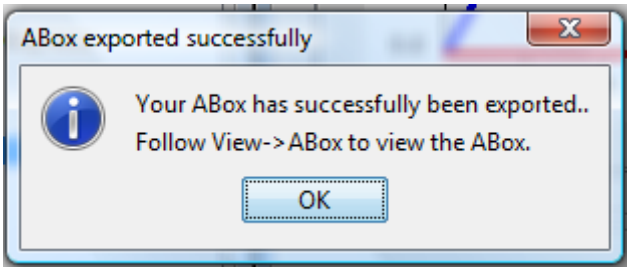
- Παραγωγή ισχυρισμών που συνδέουν ένα άτομο με κάποια Class και κάποιο συντελεστή ασάφειας.
- Παραγωγή ισχυρισμών που συνδέουν δύο άτομα με ένα Object Property και κάποιο συντελεστή ασάφειας.
- Παραγωγή ισχυρισμών που συνδέουν ένα άτομο με μία τιμή σε ένα Data Property χωρίς κάποιο συντελεστή ασάφειας.

Προκειμένου να παραχθεί το ABox ο χρήστης ακολουθεί την εντολή Export→ABox ή εναλλακτικά πατά τα κουμπιά Alt+X, όπως φαίνεται στο σχήμα 147.



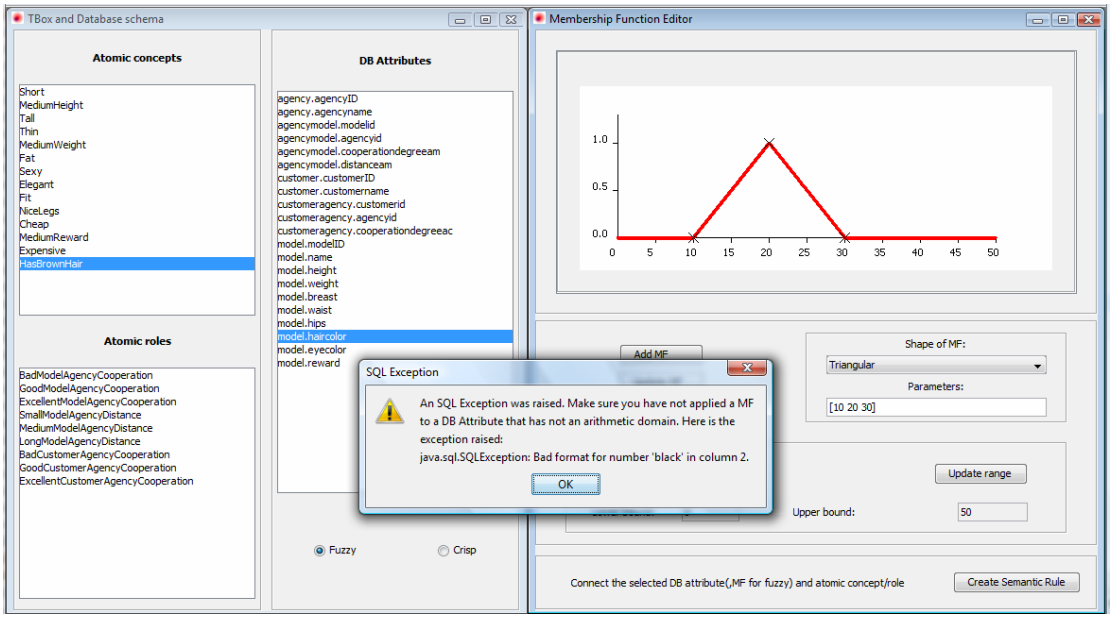
Σχήμα 147. Εντολή για παραγωγή του ABox.

Εάν όλα πάνε καλά το επιβεβαιωτικό μήνυμα για επιτυχημένη παραγωγή του ABox του σχήματος 148 κάνει την εμφάνιση του.

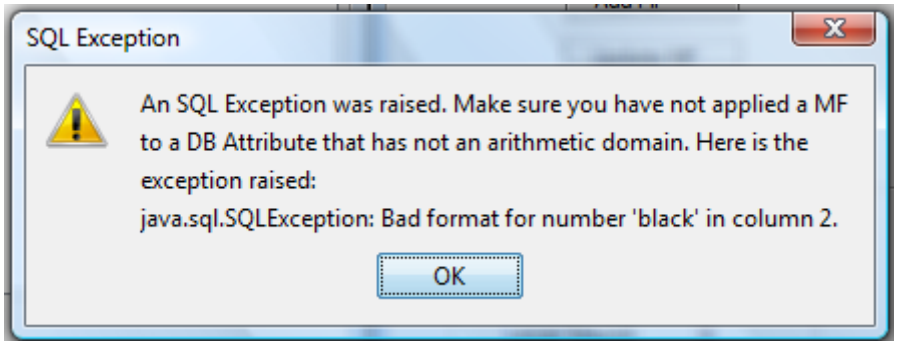


Σχήμα 148. Επιβεβαιωτικό μήνυμα επιτυχημένης παραγωγής Abox.

Στο σημείο αυτό να τονίσουμε ότι έχουμε υποθέσει ότι ο μηχανικός-χρήστης στον οποίο απευθύνεται το παραπάνω σύστημα θα προχωρήσει σε παραγωγή έυλογων SR. Με αυτό εννοούμε ότι η δημιουργία ενός SR με μία MF (δηλαδή με “Fuzzy” radio button) και με ένα DB Attribute που δεν έχει αριθμητικό domain είναι άτοπη. Αυτό συμβαίνει επειδή μόνο αριθμοί μπορούν να χρησιμέψουν ως είσοδο σε MF οι οποίες θα επιστρέψουν μια αριθμητική τιμή ως αποτέλεσμα. Το σχήμα 149 παρουσιάζει έναν τέτοιο χωρίς νόημα SR. Ο χρήστης έχει ορίσει μία MF για το DB Attribute Model.HairColor το οποίο στη βάση δεδομένων όπως έχει ειπωθεί πιο πάνω έχει domain VARCHAR(30). Το αποτέλεσμα είναι ένα προειδοποιητικό μήνυμα που φαίνεται ευκρινέστερα στο σχήμα 150. Αυτό το οποίο αναμένεται από το χρήστη σε μία τέτοια περίπτωση είναι η διαγραφή του SR με τον τρόπο που περιγράφηκε παραπάνω και η εκ νέου δημιουργία του, αυτή τη φορά όμως με επιλεγμένο το “Crisp” radio button.



Σχήμα 149. Παραγωγή SR με μη αριθμητικό DB Attribute και MF.



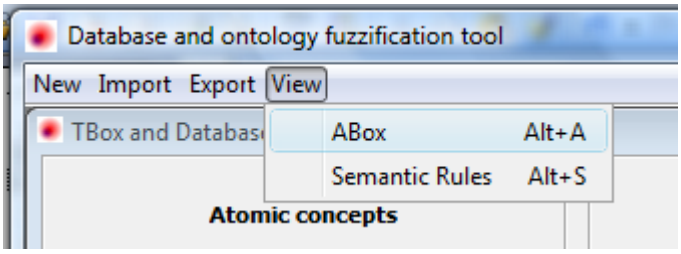
Σχήμα 150. Προειδοποιητικό μήνυμα του SR του σχήματος 149 κατά την παραγωγή του Abox.

Μετά το επιβεβαιωτικό μήνυμα του σχήματος 148 το ABox έχει μεν παραχθεί αλλά πώς μπορεί να έρθει ο χρήστης σε επαφή με αυτό; Υπάρχουν δύο τρόποι: ο ένας είναι να ανοίξει το αρχείο tmp/MyABox.owl που έχει δημιουργηθεί στο path της εφαρμογής. Το αρχείο παράγεται σε OWLXMLOntologyFormat και ένα απόσπασμα του δίνει το σχήμα 151. Από αυτό το σημείο και μετά ο χρήστης μπορεί είτε να το επεξεργαστεί με κάποιον parser είτε να το εποπτεύσει άμεσα.

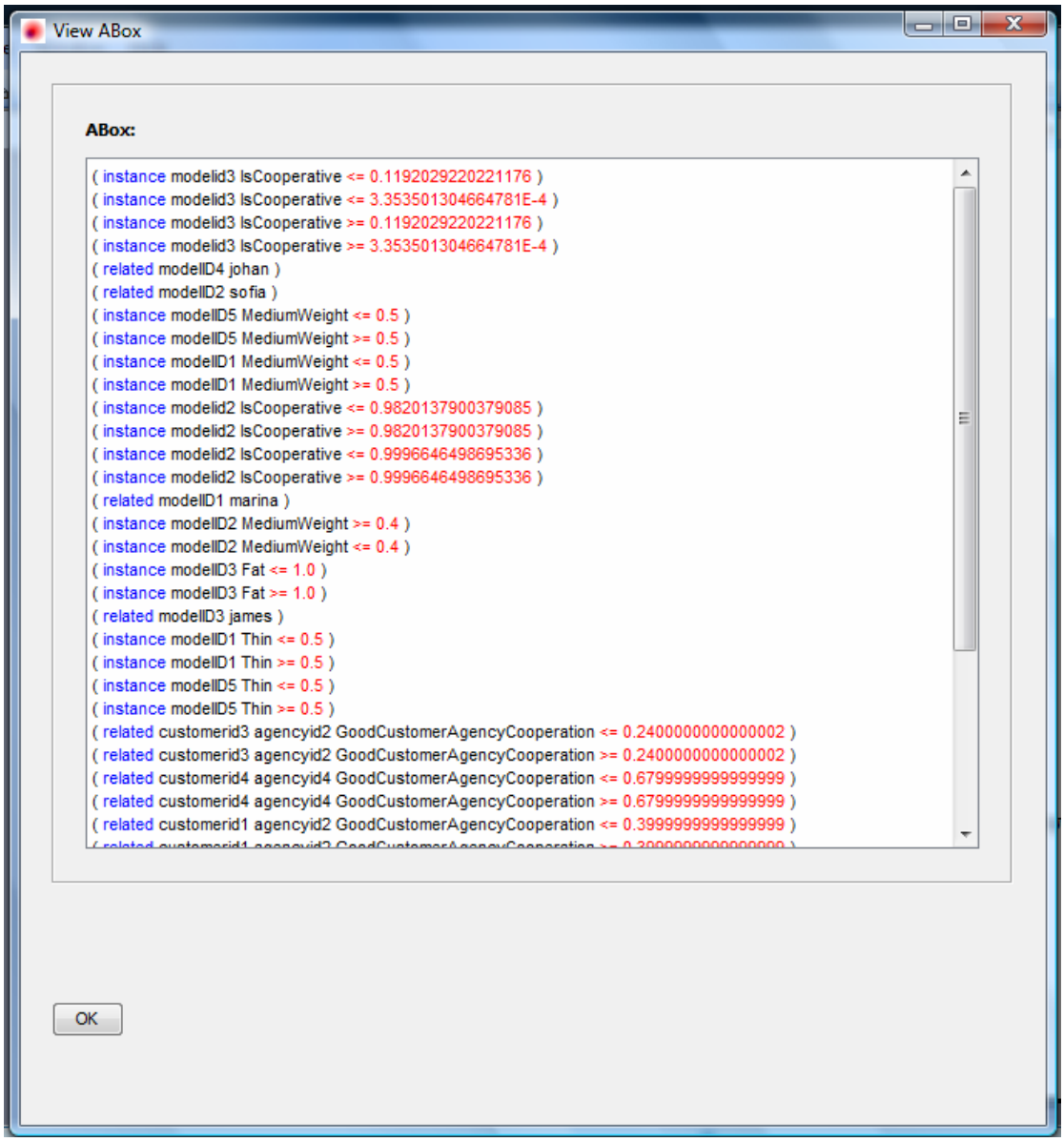
```
<DataPropertyAssertion>
    <DataProperty URI="&MyABox;HasModelName"/>
    <Individual URI="&MyABox;modelID3"/>
    <Constant datatypeURI="&xsd;string">james</Constant>
  </DataPropertyAssertion>
  <ClassAssertion>
    <Annotation annotationURI="&MyABox;hasFuzziness">
      <Constant datatypeURI="&xsd;string">geq 0.5</Constant>
    </Annotation>
    <Annotation annotationURI="&MyABox;hasFuzziness">
      <Constant datatypeURI="&xsd;string">leq 0.5</Constant>
    </Annotation>
    <OWLClass URI="&MyABox;Thin"/>
    <Individual URI="&MyABox;modelID5"/>
  </ClassAssertion>
  <ObjectPropertyAssertion>
    <Annotation annotationURI="&MyABox;hasFuzziness">
      <Constant datatypeURI="&xsd;string"
        >leq 0.9666666666666667</Constant>
    </Annotation>
    <Annotation annotationURI="&MyABox;hasFuzziness">
      <Constant datatypeURI="&xsd;string"
        >geq 0.9666666666666667</Constant>
    </Annotation>
    <ObjectProperty URI="&MyABox;BadCustomerAgencyCooperation"/>
    <Individual URI="&MyABox;customerid4"/>
    <Individual URI="&MyABox;agencyid1"/>
  </ObjectPropertyAssertion>
```

Σχήμα 151. Απόσπασμα του .owl αρχείου που αντιπροσωπεύει το ABox.

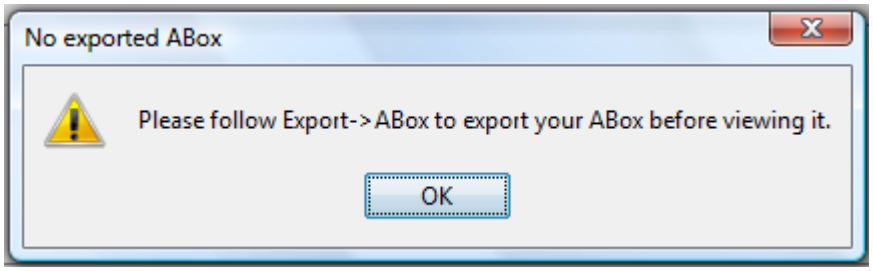
Επιπλέον υπάρχει η δυνατότητα να παρουσιαστεί το ABox σε μία πιο εύληπτη μορφή. Ο χρήστης επιλέγει View→ABox όπως δείχνει το σχήμα 152 ή εναλλακτικά πατά τα κουμπιά Alt+A. Το αποτέλεσμα απεικονίζεται στο σχήμα 153. Ωστόσο για την εκτέλεση αυτής της εντολής απαραίτητο είναι να έχει προηγηθεί η Export→ABox αλλιώς εμφανίζεται το μήνυμα του σχήματος 154.



Σχήμα 152. Εντολή για προβολη του ABox.

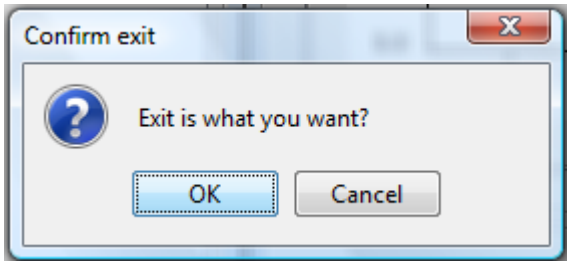


Σχήμα 153. Προβολή ABox.



Σχήμα 154. Προειδοποιητικό μήνυμα για εξαγωγή ABox πριν την προβολή του.

Κλείνοντας να τονίσουμε ότι ο χρήστης δεν έχει τη δυνατότητα να κλείσει κανένα από τα δύο εσωτερικά frames γιατί κάτι τέτοιο καθιστά το σύστημα μη λειτουργικό. Το σχήμα 155 είναι η τελευταία επαφή που έχει ο χρήστης με το σύστημα:



Σχήμα 155. Μήνυμα επιβεβαίωσης εξόδου.

6

Περιγραφή κώδικα του συστήματος

6.1 Περιγραφή Κλάσεων

Πριν ξεκινήσουμε την περιγραφή του λογισμικού στο σημείο αυτό να επισημάνουμε ότι όλες οι κλάσεις διαθέτουν ένα σύνολο private μεταβλητών που δίνονται. Γι'αυτό το σύνολο μεταβλητών έχουν οριστεί οι αντίστοιχοι setters και getters τους οποίους δε θα αναφέρουμε ξεχωριστά στην περιγραφή των μεθόδων αλλά των οποίων η ύπαρξη και λειτουργία εξυπακούεται.

6.1.1 FuzzyFrame.java

```
public class FuzzyFrame extends JFrame implement ActionListener,
WindowListener
```

Η κλάση αυτή είναι καταρχήν η κλάση που περιλαμβάνει τη main μέθοδο η οποία εκκινεί την εφαρμογή. Ουσιαστικά πρόκειται για την «πατρική» κλάση του συστήματος η οποία δημιουργεί το σκελετό του αρχικού interface, είναι υπεύθυνη για τα functionalities του μενού και κάνει τις πρώτες βασικές αρχικοποιήσεις.

6.1.1.1 Πεδία

- `private JDesktopPane desktop;`

Πρόκειται για τον υποδοχέα εκείνο ο οποίος φιλοξενεί τα δύο JInternalFrames, το RightFrame και το LeftFrame.

- `private JMenuItem menuItem1,menuItem2,menuItem3,menuItem4,menuItem5,menuItem6,menuItem7;`

Πρόκειται για το σύνολο των επιλογών μενού που θα χρησιμέψουν για την εκτέλεση εντολών.

- `private JFileChooser fc;`

Πρόκειται για το frame εκείνο που χρησιμεύει για την επιλογή και επεξεργασία αρχείων-σε αυτή την περίπτωση .owl αρχείων που θα χρησιμέψουν για την ανάκτηση ατομικών εννοιών και ρόλων.

- `private LeftFrame frameLeft;`

Πρόκειται για το κύριο αριστερό frame του συστήματος.

- `private RightFrame frameRight;`

Πρόκειται για το κύριο δεξιό frame του συστήματος.

- `private SemanticRulesManager semanticRulesManager;`

Πρόκειται για το διαχειριστή των SR, μία κλάση που είναι υπεύθυνη για την επεξεργασία του συνόλου των SR.

- `private DBManager dbMan;`

Πρόκειται για το διαχειριστή της επικοινωνίας του συστήματος με τη βάση δεδομένων.

- `private DBConnectionFrame frameDBConnection;`

Πρόκειται για το frame μέσω του οποίου ο χρήστης εγκαθιδρύει την επικοινωνία με τη βάση δεδομένων.

- `private ABoxManager aboxman;`

Πρόκειται για το διαχειριστή του σώματος ισχυρισμών που το σύστημα θα παράγει.

- `private URI physicalURI;`

Πρόκειται για το φυσικό URI της OWLOntology (δηλαδή του ABox) που το σύστημα θα δημιουργήσει.

6.1.1.2 Μέθοδοι

- `public FuzzyFrame()`

Είναι ο κατασκευαστής της κλάσης. Εφόσον πρόκειται για subclass της JFrame χρησιμοποιεί τον πατρικό constructor και θέτει τον τίτλο. Στη συνέχεια τοποθετεί το JFrame σε συγκεκριμένη θέση ως προς τα όρια της οθόνης. Θέτει το εικονίδιο το οποίο βρίσκεται σε σχετικό path του σκληρού δίσκου καθώς επίσης και το χρώμα του background. Χρησιμοποιεί ένα αντικείμενο της κλάσης GridBagLayout προκειμένου να ρυθμίσει τη θέση του frameLeft και του frameRight. Καλεί τις δύο μεθόδους που τα δημιουργούν, τα τοποθετεί στις θέσεις τους και στη συνέχεια δημιουργεί και θέτει το μενού. Τέλος προσθέτει έναν WindowListener γι'αυτό το JFrame.

- `protected JMenuBar createMenuBar()`

Τοποθετεί τα τέσσερα μενού "New","Import","Export" και "View" στην μπάρα του αντικειμένου της FuzzyFrame μαζί με τα μέλη του κάθε μενού. Προσφέρει στο μέλος κάθε μενού τη δυνατότητα να επιλέγετα όχι μόνο με τη γνωστή διαδρομή του ποντικιού αλλά και με τον κατάλληλο συνδυασμό πλήκτρων, διαφορετικών για κάθε menu item. Επιπλέον προσθέτει σε κάθε menu item τον ActionListener που θα εκτελέσει τις κατάλληλες εντολές ανάλογα με το ποιο menu item επιλέγεται.

- `protected void createLeftFrame(GridBagLayout gridbagMain, GridBagConstraints constraintsMain)`

Δημιουργεί το frameLeft του συστήματος. Επιπλέον το τοποθετεί στην αριστερή θέση του desktop χρησιμοποιώντας το gridbagMain που έχει περαστεί ως παράμετρος και τέλος το θέτει ορατό.

- `protected void createRightFrame(GridBagLayout gridbagMain, GridBagConstraints constraintsMain)`

Δημιουργεί το frameRight του συστήματος. Επιπλέον το τοποθετεί στη δεξιά θέση του desktop χρησιμοποιώντας το gridbagMain που έχει περαστεί ως παράμετρος και τέλος το θέτει ορατό.

- `public void actionPerformed(ActionEvent e)`

Ανιχνεύει ποιο component προκάλεσε το γεγονός και εκτελεί την αντίστοιχη ενέργεια καλώντας μία ξεχωριστή μέθοδο για κάθε component. Η υλοποίηση αυτής της μεθόδου είναι υποχρεωτική εφόσον η κλάση FuzzyFrame υλοποιεί το interface ActionListener.

- `protected void createNewAtomicConceptFrame(DefaultListModel conceptListModel)`

Είναι η μέθοδος που αναλαμβάνει την εκτέλεση της εντολής μενού New→Atomic Concept.

Δημιουργεί ένα αντικείμενο της κλάσης NewAtomicConceptFrame χρησιμοποιώντας ως όρισμα το conceptListModel (που είναι υπεύθυνο για τα ατομικά concepts που εμφανίζονται στο χρήστη) και το καθιστά ορατό.

- `protected void createNewAtomicRoleFrame(DefaultListModel roleListModel)`

Είναι η μέθοδος που αναλαμβάνει την εκτέλεση της εντολής μενού New→Atomic Role.

Δημιουργεί ένα αντικείμενο της κλάσης NewAtomicRoleFrame χρησιμοποιώντας ως όρισμα το roleListModel (που είναι υπεύθυνο για τους ατομικούς ρόλους που εμφανίζονται στο χρήστη) και το καθιστά ορατό.

- `protected void createNewOwlFileChooser()`

Είναι η μέθοδος που αναλαμβάνει την εκτέλεση της εντολής μενού Import→Ontology. Δημιουργεί ένα αντικείμενο της κλάσης JFileChooser, στο οποίο προσθέτει ένα αντικείμενο της κλάσης OWLFilter προκειμένου ο χρήστης να μπορεί να επιλέξει μόνο .owl αρχεία. Εμφανίζει τον fc και σε περίπτωση που ο χρήστης πατήσει το κουμπι του "Attach" για κάποιο αρχείο γίνονται τα παρακάτω. Λαμβάνεται το επιλεγμένο αρχείο και στη συνέχεια δημιουργείται ένα αντικείμενο της κλάσης MyOntologyManager που παίρνει ως όρισμα το URI του επιλεγμένου αρχείου. Με τη βοήθεια αυτού του αντικειμένου γίνεται η ανάκτηση των ατομικών εννοιών και ρόλων από το επιλεγμένο αρχείο σε μορφή ArrayList και τα ανακτημένα αυτά περιεχόμενα της οντολογίας χρησιμοποιούνται για την ανανέωση της λίστας ατομικών εννοιών και ρόλων του frameLeft.

- `protected void createDBConnectionFrame(DefaultListModel attributesListModel)`

Είναι η μέθοδος που αναλαμβάνει την εκτέλεση της εντολής μενού Import→Database schema. Δημιουργεί ένα αντικείμενο της κλάσης DBConnectionFrame το οποίο και καθιστά ορατό. Επιπλέον δημιουργεί ένα αντικείμενο της κλάσης SemanticRulesManager το οποίο και θέτει στο frameRight.

- `protected void createViewSemanticRulesFrame()`

Είναι η μέθοδος που αναλαμβάνει την εκτέλεση της εντολής μενού View→Semantic Rules. Δημιουργεί ένα αντικείμενο της κλάσης ViewSemanticRulesFrame και το καθιστά ορατό.

- `protected void createABoxManager()`

Είναι η μέθοδος που αναλαμβάνει την εκτέλεση της εντολής μενού Export→ABox. Αρχικά λαμβάνει τον DBManager από το frame που επικοινώνησε με το χρήστη για τη ΒΔ και στη συνέχεια δημιουργεί ένα αντικείμενο της κλάσης ABoxManager χρησιμοποιώντας ως ορίσματα τον διαχειριστή της ΒΔ και το διαχειριστή των Semantic Rules. Σε περίπτωση που εγερθεί μία SQLException αυτή πιάνεται και εμφανίζεται στο χρήστη το αντίστοιχο μήνυμα. Ένας πιθανός λόγος μιας SQLException είναι να έχει ορίσει ο χρήστης να εφαρμοστεί μια

MF σε ένα DB Attribute που δεν έχει αριθμητικό domain. Επίσης και οι περισσότερο γενικές Exception πιάνονται και εμφανίζονται στο χρήστη.

- `protected void createABoxFrame()`

Είναι η μέθοδος που αναλαμβάνει την εκτέλεση της εντολής μενού View→ABox. δημιουργεί ένα αντικείμενο της κλάσης ABoxFrame, το καθιστά ορατό και στη συνέχεια καλεί τη μέθοδο του διαχειριστή ABox με όρισμα το StyledDocument του ABoxFrame και το physicalURI (περιλαμβάνει το φυσικό URI του παραχθέντος ABox) προκειμένου να εγγραφεί το ABox σε φιλική για το χρήστη μορφή στο JTextPane του ABoxFrame. Εάν δεν έχει ήδη εξαχθεί το ABox μια NullPointerException θα εγερθεί και θα εμφανιστεί στο χρήστη το αντίστοιχο μήνυμα.

- `private static void createAndShowGUI()`

Είναι η μέθοδος που αναλαμβάνει τη δημιουργία και την εμφάνιστη του GUI. Θέτει το LookAndFeel της τρέχουσας πλατφόρμας και δημιουργεί ένα αντικείμενο της κλάσης FuzzyFrame. Ορίζει να μη γίνεται τίποτα εάν ο χρήστης επιχειρήσει να κλείσει το frame και τέλος το καθιστά ορατό.

- `public static void main(String[] args)`

Πρόκειται για την κύρια μέθοδο η οποία εκκινεί την εφαρμογή χρησιμοποιώντας τη μέθοδο createAndShowGUI( ).

- `public void windowActivated(WindowEvent e)`
- `public void windowClosed(WindowEvent e)`
- `public void windowDeactivated(WindowEvent e)`
- `public void windowDeiconified(WindowEvent e)`
- `public void windowIconified(WindowEvent e)`
- `public void windowOpened(WindowEvent e)`

Η υλοποίηση των 6 παραπάνω μεθόδων είναι υποχρεωτική εφόσον η κλάση FuzzyFrame υλοποιεί το interface WindowListener. Δεν υλοποιούν κάποια λειτουργία.

- `public void windowClosing (WindowEvent e)`

Η παραπάνω μέθοδος εκτελείται όταν ο χρήστης κλείνει το παράθυρο, κατά τη διάρκεια του κλεισίματος όχι όταν το έχει ήδη κλείσει. Εμφανίζει στο χρήστη ένα Confirm Dialog προκειμένου να βεβαιωθεί ότι θέλει όντως να εγκαταλείψει την εφαρμογή.

6.1.2 LeftFrame.java

`public class LeftFrame extends JInternalFrame implements ActionListener, ListSelectionListener`

Πρόκειται για την κλάση που σχεδιάζει το αριστερό JInternalFrame του interface, δηλαδή εμφανίζει στο χρήστη τις ατομικές έννοιες, τους ατομικούς ρόλους και τα DB Attributes ενώ έχει και μερικά βασικά functionalities.

6.1.2.1 Πεδία

- `private DefaultListModel conceptListModel,roleListModel, attributesListModel;`

Πρόκειται για τις δομές δεδομένων εκείνες που κρατούν τα σύνολα των ατομικών εννοιών, ατομικών ρόλων και DB Attributes που εμφανίζονται στο χρήστη.

- `private JRadioButton fuzzyButton,crispButton,dummyButton;`

Πρόκειται για radio buttons (που στη συνέχεια θα τοποθετηθούν στο ίδιο button group) και τα οποία σχετίζονται με το χαρακτήρα του DB Attribute που είναι κάθε φορά επιλεγμένος.

- `private JPanel pane1,pane2,pane3;`

Πρόκειται για JPanels τα οποία σχετίζονται με τη γραφική παρουσίαση του interface.

- `private RightFrame frameRight;`

Πρόκειται για ένα αντικείμενο τύπου RightFrame το οποίο χρησιμεύει ως αναφορά στο δεξιό frame και επιτρέπει την κλήση πολλών μεθόδων του.

- `private JList roleList,conceptList,attributesList;`

Πρόκειται για τις λίστες εκείνες που εμφανίζουν τα σύνολα των ατομικών εννοιών, ατομικών ρόλων και DB Attributes στο χρήστη και του επιτρέπουν να τα επιλέξει.

- `private Hashtable<String, MyBounds> hashBounds= new Hashtable<String, MyBounds>();`

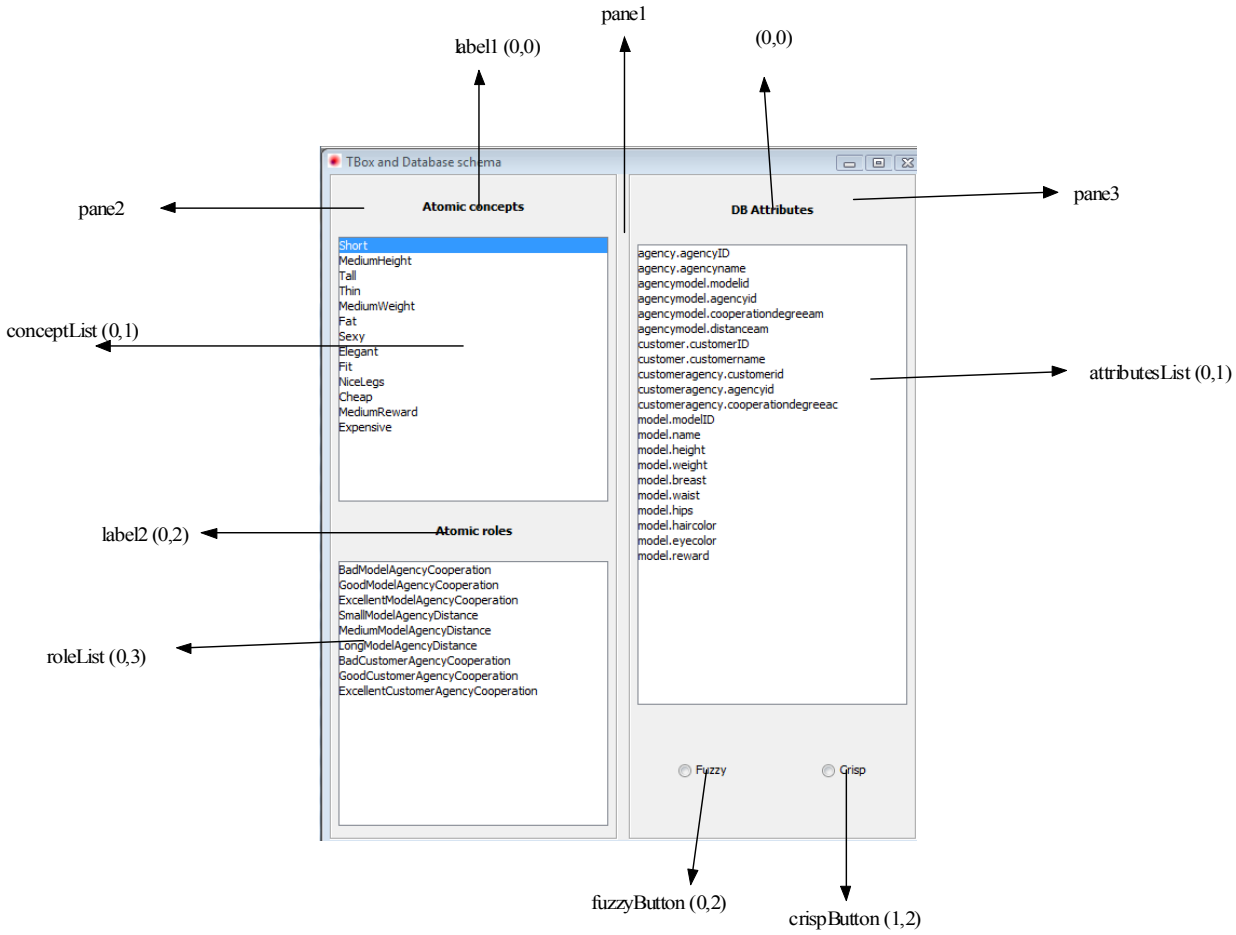
Πρόκειται για μία δομή HashTable η οποία συσχετίζει μοναδικά ένα String (το οποίο σε αυτή την περίπτωση αντιπροσωπεύει κάποιο DB Attribute) με ένα αντικείμενο τύπου MyBounds. Η δομή αυτή σχετίζεται με τη σωστή απεικόνιση της γραφικής παράστασης των MF στο δεξί frame.

6.1.2.2 Μέθοδοι

- `public LeftFrame()`

Είναι ο κατασκευαστής της κλάσης. Εφόσον πρόκειται για subclass της JFrame χρησιμοποιεί τον πατρικό constructor και θέτει τον τίτλο. Θέτει το εικονίδιο το οποίο βρίσκεται σε σχετικό path, αφού πρώτα το έχει δημιουργήσει με τη βοήθεια της αντίστοιχης μεθόδου. Ακόμη δίνει στο παρόν αντικείμενο την ικανότητα να ελαχιστοποιείται, να μεγιστοποιείται, να κλείνει καθώς και να γίνεται iconify.Επιπλέον ορίζει να μη γίνεται τίποτα κατά το κλείσιμο του παραθύρου μια και δε θέλουμε να κλείσει ο χρήστης το frame έστω και ηθελημένα. Δημιουργεί τρία panel εκ των οποίων τα 2 με ευδιάκριτα όρια. Όπως φαίνεται και στο σχήμα 156 το pane1 είναι αυτό που φιλοξενεί τα άλλα δύο. Δημιουργείται ένας GridLayoutManager ο οποίος τίθεται στο pane1 και δύο GridBagLayoutManager για τα pane2 και pane3. Ακολουθεί η δημιουργία των γραφικών components τους και η επικόλληση τους στα pane2 και pane3 σύμφωνα με τις συντεταγμένες που δίνει το σχήμα 156. Ιδιαίτερα για τα labels “Atomic concepts”, “Atomic Roles” και “DB Attributes” χρησιμοποιείται έντονη γραμματοσειρά.

Στο σημείο αυτό θα κάνουμε μία αναφορά στο διαχειριστή GridBagLayout καθώς επίσης και τις εντολές που σχετίζονται με αυτόν, επειδή κατά την ανάπτυξη του συστήματος είναι ο κατεξοχήν διαχειριστής διάταξης που χρησιμοποιούμε. Ο GridBagLayout manager καθορίζει που θα τοποθετηθεί κάθε συστατικό. Αυτό το πετυχαίνει χρησιμοποιώντας ένα αντικείμενο της κλάσης GridBagConstraints. Συγκεκριμένα θέτει τα πεδία αυτού του αντικειμένου: τα πεδία gridx και gridy σχετίζονται με τις «συντεταγμένες» των components, τα πεδία gridwidth και gridheight σχετίζονται με την έκταση των components στην οριζόντια και κάθετη κατεύθυνση ενώ τα weightx και weighty με το ποσοστό της οριζόντιας ή της κάθετης διάστασης που καταλαμβάνει το κάθε ένα από αυτά. Επιπλέον υπάρχουν το πεδίο fill που καθορίζει εάν τα συστατικά θα επεκταθούν προς κάποια κατεύθυνση και προς ποια και το πεδίο anchor που ορίζει ως προς ποια κατεύθυνση το συστατικό θα τοποθετηθεί. Επίσης υπάρχει το πεδίο insets το οποίο χρησιμοποιείται εάν επιθυμούμε να εισάγουμε περιθώρια στο γραφικό συστατικό. Τέλος για να συσχετίσουμε το αντικείμενο κλάσης GridBagConstraints με το γραφικό συστατικό χρησιμοποιείται η μέθοδος setConstraints της κλάσης GridBagLayout.



Σχήμα 156. Γραφική διάταξη αριστερού frame.

Συνεχίζοντας την περιγραφή του κατασκευαστή θα περιγράψουμε λίγο πιο λεπτομερώς την περιγραφή μιας JList καθώς πρόκειται για ένα component που χρησιμοποιείται και σε άλλα σημεία της κατασκευής του συστήματος. Αρχικά πρέπει να δημιουργηθεί ένα αντικείμενο της κλάσης DefaultListModel το οποίο είναι υπεύθυνο για τα περιεχόμενα της λίστας. Στη συνέχεια χρησιμοποιώντας αυτό κατασκευάζεται ένα αντικείμενο τύπου JList. Αναφερόμενοι σε αυτό το αντικείμενο ορίζουμε να επιτρέπεται στο χρήστη μοναδική επιλογή στοιχείου καθώς επίσης και κατακόρυφη παρουσίαση διάταξης. Δεν τίθεται κάποιος περιορισμός στοιχείων ως προς τον αριθμό στοιχείων που θα είναι ορατά, ενώ η όλη λίστα τοποθετείται σε ένα scroll pane για να δίνεται η δυνατότητα στο χρήστη να έχει εποπτεία σε λίστες μεγάλου μεγέθους. Η παραπάνω διαδικασία επαναλαμβάνεται για τη λίστα ατομικών εννοιών, ατομικών ρόλως καθώς επίσης και για τη λίστα των ιδιοχαρακτηριστικών της βάσης δεδομένων. Επιπλέον σε κάθε λίστα προστίθεται και ένας ListSelectionListener ο οποίος ενεργοποιεί κάποια μέθοδο κάθε φορά που ένα στοιχείο της λίστας επιλέγεται. Όσον αφορά τα radio buttons τώρα, αυτά δημιουργούνται απενεργοποιημένα και επιπλέον ανήκουν στο ButtonGroup bGroup. Ο σκοπός του bGroup είναι να είναι κάθε φορά το πολύ ένα radio button ενεργοποιημένο, δηλαδή να είναι αλληλοαποκλειόμενα. Επιπλέον δημιουργούμε και ένα radio button χωρίς λειτουργική χρησιμότητα το dummyButton, το οποίο δε φαίνεται σε κάποιο σημείο του interface και του οποίου ο λόγος ύπαρξης θα διευκρινιστεί αργότερα.

- `public void renewConceptListModel(ArrayList concepts)`

Πρόκειται για μία μέθοδο η οποία δέχεται ως όρισμα ένα ArrayList και η οποία αφού αφαιρέσει όλες τις ατομικές έννοιες του conceptListModel προσθέτει ένα ένα σε αυτό τα στοιχεία του ορίσματος.

- `public void renewRoleListModel(ArrayList roles)`

Πρόκειται για μία μέθοδο η οποία δέχεται ως όρισμα ένα ArrayList και η οποία αφού αφαιρέσει όλους τους ατομικούς ρόλους του roleListModel προσθέτει ένα ένα σε αυτό τα στοιχεία του ορίσματος.

- `protected ImageIcon createImageIcon(String path,String description)`

Πρόκειται για μία μέθοδο η οποία με όρισμα ένα path και μία ασήμαντη περιγραφή επιστρέφει ένα αντικείμενο της κλάσης ImageIcon. Χρησιμοποιείται για να θέσει το εικονίδιο του frame ενώ ουσιαστικά αυτό που κάνει είναι να αναζητήσει το εικονίδιο στο path που δίνεται και εκτός απροόπτου να κατασκευάσει το αντικείμενο της ImageIcon.

- `public void actionPerformed(ActionEvent e)`

Ανιχνεύει ποιο component προκάλεσε το γεγονός και εκτελεί την αντίστοιχη ενέργεια καλώντας μία ξεχωριστή μέθοδο για κάθε component. Η υλοποίηση αυτής της μεθόδου είναι υποχρεωτική εφόσον η κλάση LeftFrame υλοποιεί το interface ActionListener.

- `public void manageFuzzyButton()`

Πρόκειται για τη μέθοδο που εκτελείται κάθε φορά που επιλέγεται το “Fuzzy” radio button. Λαμβάνει την τρέχουσα επιλεγμένη τιμή από τη λίστα των DB Attributes και τη χρησιμοποιεί ως όρισμα για να καλέσει τη μέθοδο activateAllComponents του frameRight. Σε περίπτωση που δεν έχει επιλεγεί κάποιο DB Attribute εγείρεται μία εξαίρεση η οποία πιάνεται και ενημερώνει το χρήστη ότι πρέπει να έχει επιλέξει πρώτα κάποιο ιδιοχαρακτηριστικό προκειμένου να επιλέξει το “Fuzzy” radio button. Στο σημείο αυτό χρησιμοποιείται το dummyButton προκειμένου να επιλεγεί κάποιο τρίτο κουμπί του bGroup και να μην εμφανίζεται επιλεγμένο κανένα από τα άλλα δύο. Δυστυχώς η χρήση της εντολής `fuzzyButton.setSelected(false)` δε βοηθά επειδή ενδεχομένως το πάτημα του χρήστη διαρκεί περισσότερο από την εκτέλεση των εντολών.

- `public void manageCrispButton()`

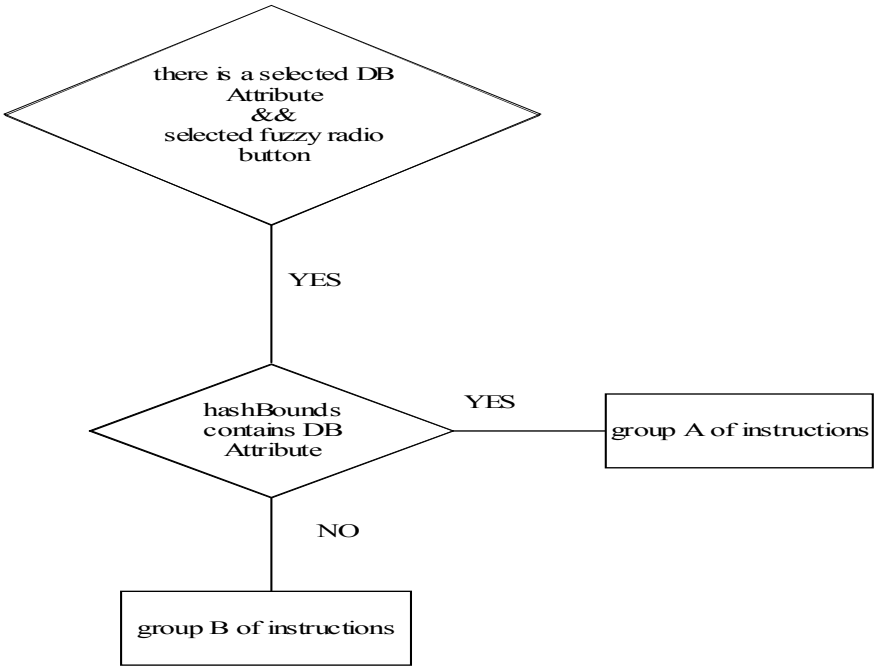
Πρόκειται για τη μέθοδο που εκτελείται κάθε φορά που επιλέγεται το “Crisp” radio button. Λαμβάνει την τρέχουσα επιλεγμένη τιμή από τη λίστα των DB Attributes και τη χρησιμοποιεί ως όρισμα για να καλέσει τη μέθοδο deactivateAllComponents του frameRight. Σε περίπτωση που δεν έχει επιλεγεί κάποιο DB Attribute εγείρεται μία εξαίρεση η οποία πιάνεται και ενημερώνει το χρήστη ότι πρέπει να έχει επιλέξει πρώτα κάποιο ιδιοχαρακτηριστικό προκειμένου να επιλέξει το “Crisp” radio button. Η χρήση του dummyButton γίνεται όπως και παραπάνω.

- `public void valueChanged(ListSelectionEvent e)`

Ανιχνεύει από ποια JList προήλθε το γεγονός και δρα ανάλογα. Εάν πρόκειται για τη λίστα ατομικών εννοιών εντοπίζεται το επιλεγμένο στοιχείο της λίστας ατομικών ρόλων και αποεπιλέγεται. Εάν πάλι πρόκειται για τη λίστα ατομικών ρόλων εντοπίζεται το επιλεγμένο στοιχείο της λίστας ατομικών εννοιών και αποεπιλέγεται αυτό. Με τον τρόπο αυτό επιτυγχάνουμε να μην είναι ταυτόχρονα επιλεγμένο ένα στοιχείο της λίστας ρόλων και ένα στοιχείο της λίστας εννοιών. Τέλος η περίπτωση της λίστας DB Attributes καλύπτεται με την κλήση μιας μεθόδου. Η υλοποίηση αυτής της μεθόδου είναι υποχρεωτική εφόσον η κλάση LeftFrame υλοποιεί το interface ListSelectionListener.

- `public void manageAttributesList()`

Πρόκειται για τη μέθοδο που καλείται κάθε φορά που επιλέγεται ένα στοιχείο από τα DB Attributes. Το παρακάτω flow diagram δείχνει τη λειτουργία της μεθόδου. Το group A περιλαμβάνει την ανάκτηση του αντικειμένου b της κλάσης MyBounds από το hashBounds, θέτει το b στο frameRight, θέτει το επιλεγμένο DB Attribute στο dPane του frameRight, θέτει το b στο dPane του frameRight και καλεί τις μεθόδους showParameters() και refresh() πάλι του frameRight. Το group B από την άλλη καλεί τις μεθόδους resetBounds(), resetParameters() και refresh() του frameRight. Το τι κάνουν οι προαναφερθείσες μέθοδοι δίνεται στην αντίστοιχη περιγραφή μεθόδων.



Σχήμα 157. Μέθοδος manageAttributesList().

- `public String getSelectedDBAttribute()`

Σε περίπτωση επιλεγμένου DB Attribute το επιστρέφει αλλιώς επιστρέφει το άδειο String “”.

- `public String getSelectedAtomicConceptOrRole()`

Επιστρέφει την επιλεγμένη ατομική έννοια ή ατομικό ρόλο. Σε περίπτωση που τίποτα από τα δύο δεν είναι επιλεγμένο επιστρέφει το άδειο String “”.

- `public String getSelectedAtomicRole()`

Σε περίπτωση επιλεγμένου ατομικού ρόλου τον επιστρέφει αλλιώς επιστρέφει το άδειο String “”.

- `public String getSelectedType()`

Επιστρέφει “concept” εάν είναι επιλεγμένη κάποια ατομική έννοια ή “role” εάν είναι επιλεγμένος κάποιος ατομικός ρόλος. Σε περίπτωση που τίποτα από τα δύο δεν είναι επιλεγμένο επιστρέφει το άδειο String “”.

- `public boolean isBounded(String dbAttribute)`

Ελέγχει εάν το dbAttribute περιέχεται στο hashBounds και αποκρίνεται ανάλογα.

- `public void putIntoHashBounds(String selection,MyBounds b)`

Εισάγει το ζεύγος (selection,b) στο hashBounds.

- `public void removeFromHashBounds(String selection)`

Αφαιρεί το ζεύγος (selection,value) όπου value είναι το αντικείμενο της κλάσης MyBounds με key το selection από το hashBounds.

- `public MyBounds getFromHash(String selection)`

Επιστρέφει το ζεύγος (selection,value) όπου value είναι το αντικείμενο της κλάσης MyBounds με key το selection από το hashBounds.

6.1.3 RightFrame.java

```
public class RightFrame extends JInternalFrame implements ActionListener, MouseListener
```

Πρόκειται για την κλάση που σχεδιάζει το δεξί JInternalFrame του interface, δηλαδή εμφανίζει στο χρήστη τις γραφικές παραστάσεις των εκάστοτε MF, ενώ διαθέτει functionalities τόσο για την επεξεργασία τους όσο και για την παραγωγή SR.

6.1.3.1 Πεδία

- `private JLabel label1,label2,label3,label4,label5,label6;`

Πρόκειται για αντικείμενα της κλάσης JLabel τα οποία υποδεικνύουν στο χρήστη ποια είναι η χρησιμότητα των υπόλοιπων γραφικών components.

- `private JButton button1,button2,button3,button4,button5;`

Πρόκειται για αντικείμενα της κλάσης JButton τα οποία χρησιμεύουν στην αλληλεπίδραση χρήστη συστήματος.

- `private JComboBox combo;`

Πρόκειται για την υλοποίηση μιας droplist η οποία καλεί το χρήστη να επιλέξει ποιο σχήμα θέλει να έχει η MF που θα ορίσει.

- `private JTextField textField1,textField2,textField3;`

Πρόκειται για textfields στα οποία ο χρήστης εισάγει τις παραμέτρους της MF, το κάτω όριο και το πάνω όριο του παραθύρου προβολής των MF αντίστοιχα.

- `private DrawingPanel dPane;`

Πρόκειται για το panel πάνω στο οποίο θα σχεδιάζονται όλες οι MF.

- `private JPanel pane1,pane2,pane3,pane4,pane5,pane6,pane7;`

Πρόκειται για JPanels τα οποία σχετίζονται με τη γραφική παρουσίαση του interface.

- `private LeftFrame frameLeft;`

Πρόκειται για ένα αντικείμενο της κλάσης LeftFrame το οποίο χρησιμεύει ως αναφορά στο αριστερό frame και επιτρέπει την κλήση πολλών μεθόδων του.

- `private SemanticRulesManager semanticRulesManager;`

Προκειται για το διαχειριστή SR της εφαρμογής.

- `private char currentDragParameter='z';`

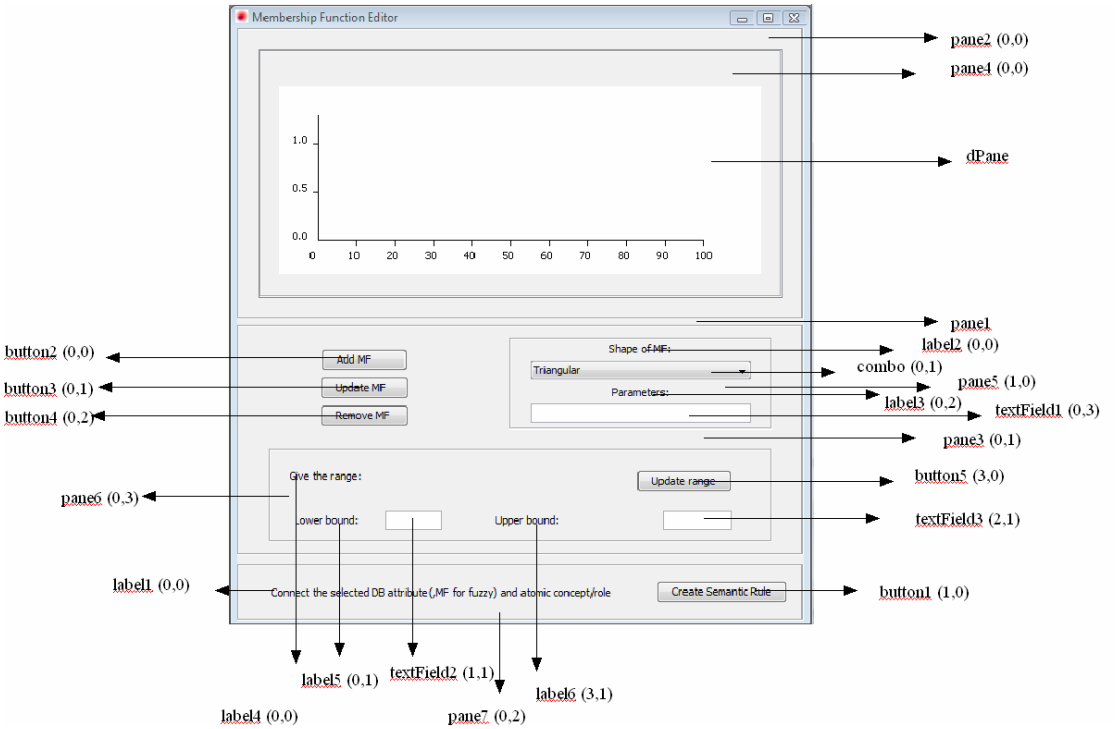
Πρόκειται για μία παράμετρο η οποία δείχνει ανάλογα με το τι σχήμα MF απεικονίζεται ποια παράμετρο της ο χρήστης τροποποιεί με το ποντίκι του.

6.1.3.2 Μέθοδοι

- `public RightFrame()`

Είναι ο κατασκευαστής της κλάσης. Εφόσον πρόκειται για subclass της JFrame χρησιμοποιεί τον πατρικό constructor και θέτει τον τίτλο. Ακόμη δίνει στο παρόν αντικείμενο την ικανότητα να ελαχιστοποιείται, να μεγιστοποιείται, να κλείνει καθώς και να γίνεται iconify. Θέτει το εικονίδιο το οποίο βρίσκεται σε σχετικό path, αφού πρώτα το έχει δημιουργήσει με τη βοήθεια της αντίστοιχης μεθόδου. Επιπλέον ορίζει να μη γίνεται τίποτα κατά το κλείσιμο

του παραθύρου μια και δε θέλουμε να κλείσει ο χρήστης το frame έστω και ηθελημένα. Δημιουργεί 7 panel εκ των οποίων τα 6 με ευδιάκριτα όρια. Δημιουργείται ένας GridBagLayoutManager για κάθε panel, προκειμένου να ρυθμίσει τη διάταξη των συστατικών που θα επικολληθούν σε αυτό. Το σχήμα 158 δείχνει τη θέση και τη χρησιμότητα του κάθε panel. Επιπλέον δείχνει που έχει τοποθετηθεί κάθε component και τις συντεταγμένες αυτού. Η διάταξη των συστατικών έχει γίνει με το γνωστό τρόπο. Αξίζει μόνο να αναφέρουμε μερικά πράγματα για την κατασκευή του combo. Το combo κατασκευάζεται χρησιμοποιώντας έναν πίνακα από Strings ο οποίος περιέχει τις επιλογές που θέλουμε να εμφανίζονται, στην περίπτωση μας τα σχήματα των MF.



Σχήμα 158. Γραφική διάταξη δεξιού frame.

- `protected ImageIcon createImageIcon(String path,String description)`

Πρόκειται για μία μέθοδο η οποία με όρισμα ένα path και μία ασήμαντη περιγραφή επιστρέφει ένα αντικείμενο της κλάσης ImageIcon. Χρησιμοποιείται για να θέσει το εικονίδιο του frame ενώ ουσιαστικά αυτό που κάνει είναι να αναζητήσει το εικονίδιο στο path που δίνεται και εκτός απροόπτου να κατασκευάσει το αντικείμενο της ImageIcon.

- `public void deactivateAllComponents(String selection)`

Απενεργοποιεί τα components label2, label3, label4, label5, label6, button2, button3, button4, button5, combo, textField1, textField2 και textField3. Επιπλέον καλεί τις μεθόδους resetBounds(), resetParameters() και resetPanel().

- `public void activateAllComponents(String selection)`

Ενεργοποιεί τα components που η προηγούμενη μέθοδος απενεργοποίησε. Επιπλέον σε περίπτωση που το selection περιέχεται στο hashBounds του frameLeft ανακτά το αντικείμενο b της κλάσης MyBounds από το hashBounds του frameLeft, καλεί τη setBothBounds με όρισμα το b, θέτει το επιλεγμένο DB Attribute στο dPane , θέτει το b στο dPane και καλεί τις μεθόδους showParameters() και refresh().Σε διαφορετική περίπτωση καλεί τις μεθόδους resetBounds(), resetParameters() και resetPanel().

- `public boolean isValidDouble(String entry)`

Πρόκειται για μια μέθοδο που αποκρίνεται εάν το entry είναι έγκυρος double αριθμός.

- `public boolean areValidParameters(String entry)`

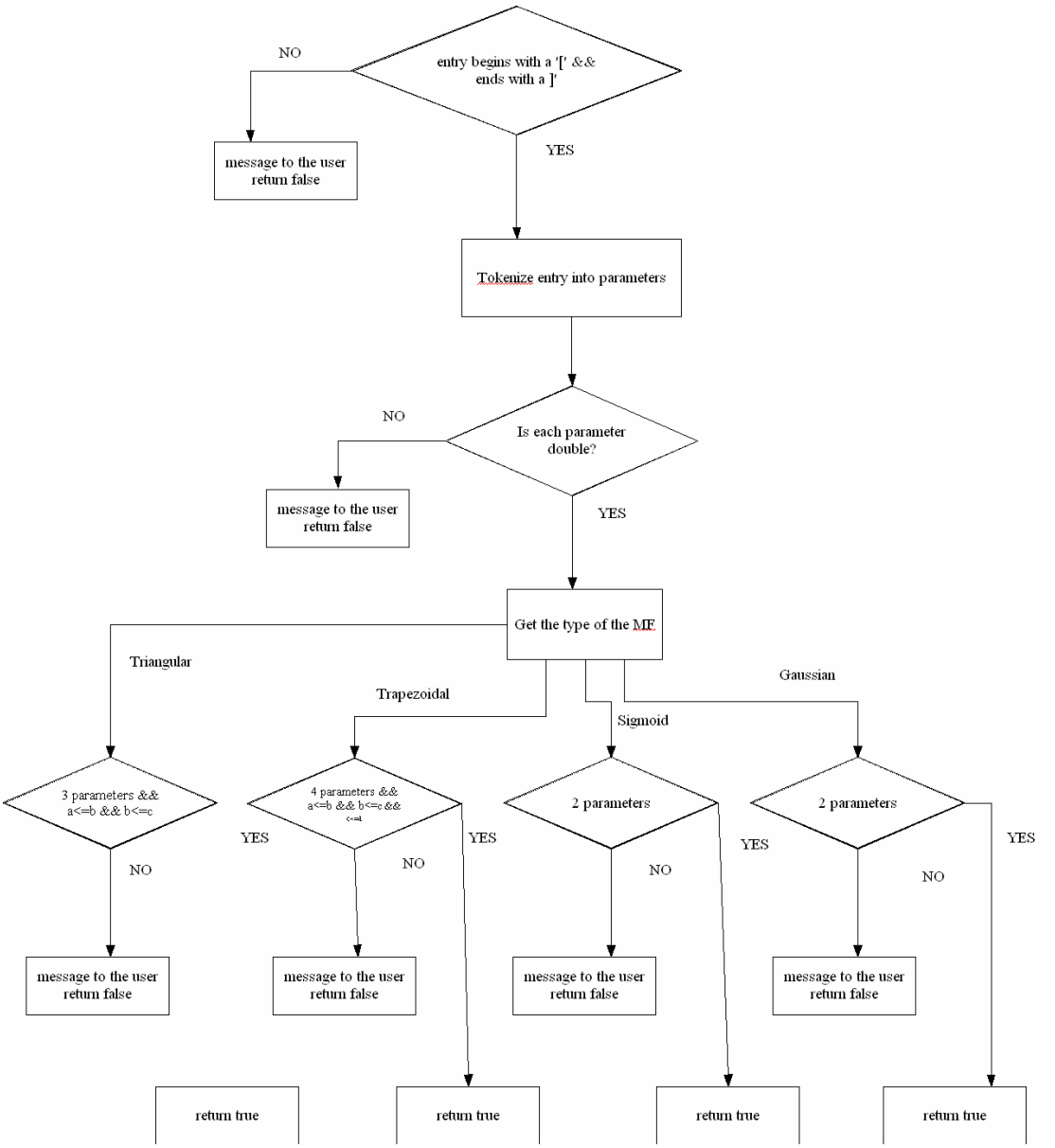
Πρόκειται για μια μέθοδο που αποκρίνεται για το αν το entry είναι έγκυρες παράμετροι πραγματοποιώντας μια σειρά από ελέγχους. Το σχήμα 159 περιγράφει τη λειτουργία της.

- `public ArrayList retrieveParameters(String entry)`

Πρόκειται για μία μέθοδο που λαμβάνει ως είσοδο ένα entry (το οποίο θα είναι της μορφής $[p_1 \ldots p_n]$) και θα επιστρέφει ένα ArrayList σε κάθε θέση του οποίου θα βρίσκεται μία από τις παραμέτρους.

- `public void actionPerformed(ActionEvent e)`

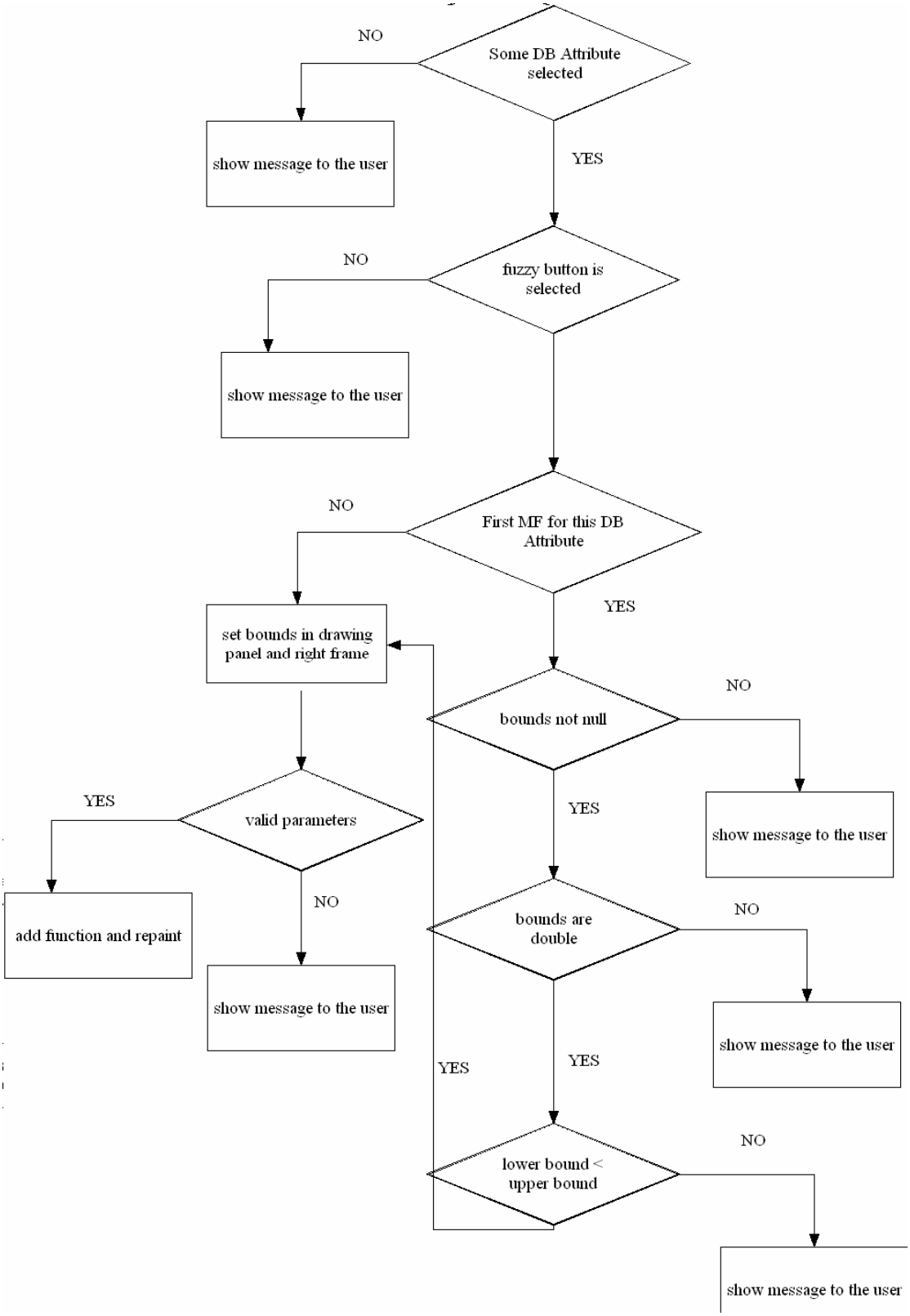
Ανιχνεύει ποιο component προκάλεσε το γεγονός και εκτελεί την αντίστοιχη ενέργεια καλώντας μία ξεχωριστή μέθοδο για κάθε component. Η υλοποίηση αυτής της μεθόδου είναι υποχρεωτική εφόσον η κλάση RightFrame υλοποιεί το interface ActionListener.



Σχήμα 159. Η μέθοδος areValidParameters(String entry)

- `public void manageAddMF()`

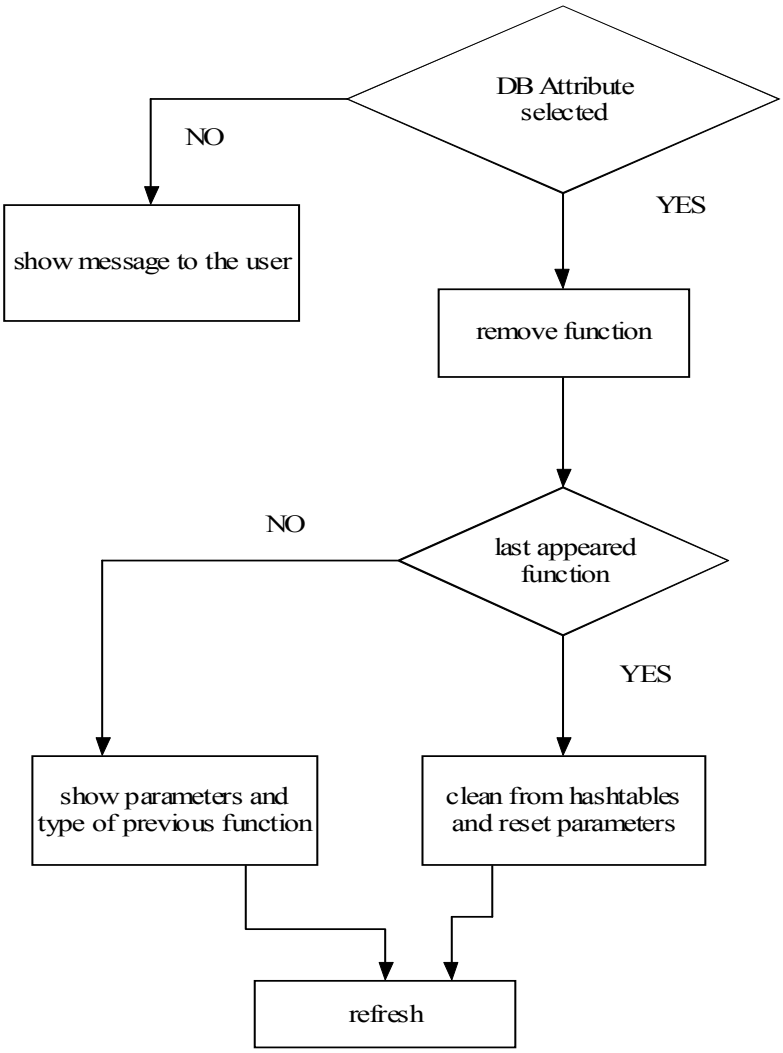
Πρόκειται για τη μέθοδο που εκτελείται όταν ο χρήστης πατήσει το κουμπί “Add MF”. Περιγράφεται από το σχήμα 160.



Σχήμα 160. Μέθοδος manageAddMF.

- `public void manageRemoveMF()`

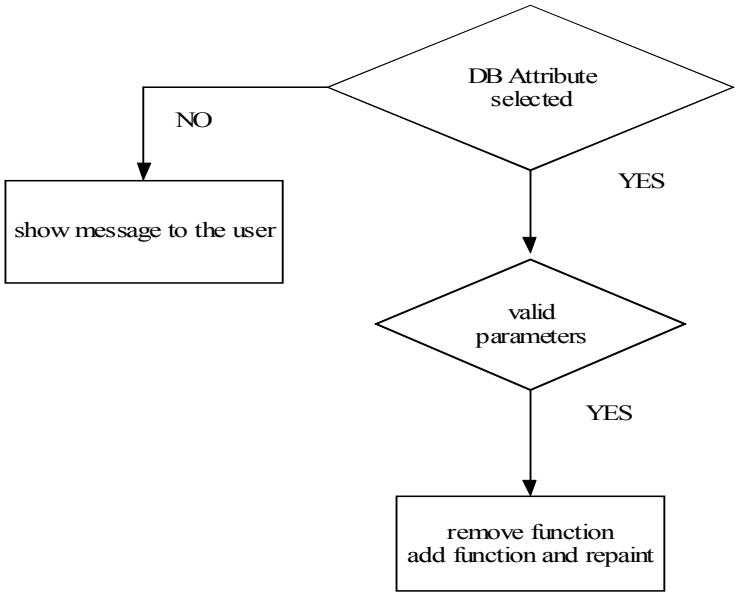
Πρόκειται για τη μέθοδο που εκτελείται όταν ο χρήστης πατήσει το κουμπί “Remove MF”. Περιγράφεται από το σχήμα 161.



Σχήμα 161. Μέθοδος manageRemoveMF.

- `public void manageUpdateMF()`

Πρόκειται για τη μέθοδο που εκτελείται όταν ο χρήστης πατήσει το κουμπί “Update MF”. Περιγράφεται από το σχήμα 162.



Σχήμα 162. Μέθοδος manageUpdateMF.

- `public void manageUpdateRange()`

Πρόκειται για τη μέθοδο που εκτελείται όταν ο χρήστης πατήσει το κουμπί “Update range”. Δημιουργεί ένα αντικείμενο της κλάσης UpdateRangeFrame και το καθιστά ορατό.

- `public void manageCreateSemanticRule()`

Πρόκειται για τη μέθοδο που εκτελείται όταν ο χρήστης πατήσει το κουμπί “Create Semantic Rule”. Περιλαμβάνει πλήθος ελέγχων προκειμένου να βεβαιωθεί ότι πληρούνται όλοι οι περιορισμοί για τη δημιουργία του SR, γι’ αυτό και περιγράφεται καλύτερα από το σχήμα 163.

Σχήμα 163. Μέθοδος manageAddSemanticRule().

- `public void addFunctionAndRepaint(String selection)`

Πρόκειται για τη μέθοδο που ακολουθεί μια επιτυχημένη προσθήκη MF. Εάν για το συγκεκριμένο DB Attribute selection είναι η πρώτη συνάρτηση που προστίθεται τότε εκτελείται η μέθοδος putIntoHashFunctions(selection). Στη συνέχεια ανάλογα με το σχήμα

της συνάρτησης δημιουργείται το αντίστοιχο αντικείμενο της αντίστοιχης κλάσης και προστίθεται στο σύνολο των συναρτήσεων για το DB Attribute selection του dPane. Επιπλέον καλείται η refresh().

- `public void setBothBounds(MyBounds b)`

Η μέθοδος αυτή θέτει τα όρια τα οποία λαμβάνει από το b στα αντίστοιχα textfields και επιπλέον στο dPane. Τα textFields κατόπιν τίθενται μη επεξεργάσιμα από το χρήστη για να μην μπορεί αυτός να τροποποιήσει τα όρια.

- `public void resetBounds()`

Θέτει το άδειο String “” στα αντίστοιχα textfields των ορίων και τα καθιστά εκ νέου επεξεργάσιμα.

- `public void resetParameters()`

Θέτει το άδειο String “” στο αντίστοιχο textfield των παραμέτρων.

- `public void resetPanel()`

Θέτει το άδειο DB Attribute στο dPane (έτσι ώστε να μην απεικονίζει MF κανενός DB Attribute) και ορίζει τα όρια 0-100. Επίσης καλεί τη refresh().

- `public void refresh()`

Ζωγραφίζει ξανά (σύμφωνα με δεδομένα που πιθανόν έχουν αλλάξει) το dPane αλλά και το pane4 που το περιέχει προκειμένου να επανασχεδιαστεί σωστά το σύστημα των αξόνων με τις γραφικές συναρτήσεις.

- `public void showParameters()`

Εμφανίζει στο textField1 τις παραμέτρους που αντιστοιχούν στην επιλεγμένη συνάρτηση του dPane. Χρησιμοποιεί DecimalFormat για να διασφαλίσει ότι κάθε παράμετρος θα εμφανίζεται με ακρίβεια το πολύ 4 δεκαδικών ψηφίων και με το χαρακτήρα ‘.’ στη θέση της

υποδιαστολής. Αφού πρώτα ανακτηθούν οι παράμετροι από το dPane (πρόκειται για τις παραμέτρους της επιλεγμένης συνάρτησης δηλαδή αυτής που βρίσκεται στην τελευταία θέση του ArrayList Functions του dPane), τοποθετούνται μέσα σε brackets και εμφανίζονται στο textField1. Εάν δεν υπάρχει καμία επιλεγμένη συνάρτηση μία εξαίρεση θα εγερθεί και το textField1 θα γεμίσει με το άδειο String.

- `public void showType()`

Εμφανίζει στο combo τον τύπο της επιλεγμένης συνάρτησης dPane. Αφού πρώτα ανακτηθεί ο τύπος από το dPane (πρόκειται για τον τύπο της επιλεγμένης συνάρτησης δηλαδή αυτής που βρίσκεται στην τελευταία θέση του ArrayList Functions του dPane) εμφανίζεται στο combo. Εάν δεν υπάρχει καμία επιλεγμένη συνάρτηση μία εξαίρεση θα εγερθεί και το combo θα εμφανίζει τη default επιλογή “Triangular”.

- `public void mouseClicked(MouseEvent e)`

Πρόκειται για τη μέθοδο που εκτελείται κάθε φορά που ο χρήστης κάνει κλικ στο dPane. Λαμβάνει τις συντεταγμένες του κλικ και καλεί με όρισμα αυτές τη manageDPaneClick(xClicked,yClicked).

- `public void manageDPaneClick(int x,int y)`

Σκοπός αυτής της μεθόδου είναι να διαπιστώσει εάν το κλικ έγινε σε κάποια από τις απεικονιζόμενες συναρτήσεις και αν ναι να σχεδιάσει την κλικαρισμένη συνάρτηση με κόκκινο και τις υπόλοιπες με μπλε. Για το σκοπό αυτό, ανακτά από το dPane όλες τις συναρτήσεις και για κάθε μία από αυτές υπολογίζει την τετμημένη της χρησιμοποιώντας ως τεταγμένη τη x. Εάν η διαφορά μεταξύ της υπολογισθείσας τετμημένης και της y είναι μικρότερη από 10 pixels τότε σημαίνει ότι η συνάρτηση όντως επιλεχτηκε από το χρήστη. Τότε καλείται η μέθοδος bringOnTop(i).

- `public void mouseEntered(MouseEvent e)`
- `public void mouseExited(MouseEvent e)`

Η υλοποίηση των 2 παραπάνω μεθόδων είναι υποχρεωτική εφόσον η κλάση RightFrame υλοποιεί το interface MouseListener. Δεν υλοποιούν κάποια λειτουργία.

- `public void mousePressed(MouseEvent e)`

Η μέθοδος αυτή συνεισφέρει κατά το ήμισυ στην υλοποίηση της λειτουργικότητας κατά την οποία ο χρήστης αλλάζει τις παραμέτρους των MF που έχει ορίσει με τη βοήθεια του ποντικιού. Εκτελείται κάθε φορά που ο χρήστης πιέζει το ποντίκι στο dPane. Λαμβάνει τις συντεταγμένες του σημείου και καλεί με όρισμα αυτές τη manageDPanePress(xPressed,yPressed).

- `public void manageDPanePress(int x, int y)`

Σκοπός αυτής της μεθόδου είναι να διαπιστώσει εάν το σημείο (x,y) βρίσκεται στο εσωτερικό κάποιου από τους σταυρούς που αντιπροσωπεύουν τις παραμέτρους και αν ναι για ποια από τις παραμέτρους πρόκειται. Κάθε παράμετρος συμβολίζεται με ένα χαρακτήρα ο οποίος έχει διαφορετικό νόημα για κάθε σχήμα MF και του οποίου η τιμή ανατίθεται στη μεταβλητή currentDragParameter. Εάν δεν πρόκειται για καμία από τις παραμέτρους των συναρτήσεων τίθεται στη μεταβλητή αυτή η τιμή ‘z’.

- `public void mouseReleased(MouseEvent e)`

Η μέθοδος αυτή συνεισφέρει κατά το έτερο ήμισυ στην υλοποίηση της λειτουργικότητας κατά την οποία ο χρήστης αλλάζει τις παραμέτρους των MF που έχει ορίσει με τη βοήθεια του ποντικιού. Εκτελείται κάθε φορά που ο χρήστης αφήνει το ποντίκι στο dPane. Λαμβάνει τις συντεταγμένες του σημείου και καλεί με όρισμα αυτές τη manageDPaneRelease(xReleased,yReleased).

- `public void manageDPaneRelease(int x, int y)`

Η μέθοδος αυτή σκοπό έχει να επανασχεδιάσει τη συνάρτηση της οποίας κάποια παράμετρο ο χρήστης άλλαξε με τη βοήθεια του ποντικιού του. Αρχικά η μέθοδος διαπιστώνει για ποιο τύπο συνάρτησης πρόκειται και στη συνέχεια για ποια παράμετρο αυτού του τύπου. Κατόποιν υπολογίζει τη νέα αλλαγμένη παράμετρο με βάση τις x,y συντεταγμένες και έαν η νέα παράμετρος ικανοποιεί κάποιους περιορισμούς (να είναι π.χ. εντός των ορίων) δημιουργείται ένα νέο ArrayList παραμέτρων με την αλλαγμένη καινούρια και τις απείραχτες παλιές οι οποίες τίθενται στην επιλεγμένη συνάρτηση. Εάν η currentDragParameter έχει την τιμή ‘z’ τίποτα από τα παραπάνω δε γίνεται. Επιπλέον εμφανίζονται οι ανανεωμένες παράμετροι στο textField1 και επανασχεδιάζεται η ανανεωμένη γραφική παράσταση.

6.1.4 DrawingPanel.java

```
public class DrawingPanel extends JPanel
```

Πρόκειται για την κλάση που σχεδιάζει τις γραφικές παραστάσεις των MF που ορίζει ο χρήστης σε ένα σύστημα αξόνων.

6.1.4.1 Πεδία

- `private int width;`

Πρόκειται για το πλάτος του panel.

- `private int height;`

Πρόκειται για το ύψος του panel.

- `private MyBounds bounds=new MyBounds(0,100);`

Πρόκειται για τα όρια που θα απεικονίζονται στον οριζόντιο άξονα. Το πεδίο αυτό αρχικοποιείται με ένα αντικείμενο της κλάσης MyBounds που έχει κάτω όριο 0 και άνω 100.

- `private String dbAttribute="";`

Πρόκειται για το DB Attribute εκείνο το οποίο είναι επιλεγμένο από την attributesList του αριστερού frame την τρέχουσα χρονική στιγμή από το χρήστη.

- `private Hashtable<String, ArrayList> hashFunctions= new Hashtable<String, ArrayList>();`

Είναι μία δομή HashTable η οποία συσχετίζει μοναδικά ένα String (που συγκεκριμένα πρόκειται για το DB Attribute) με μία ArrayList από αντικείμενα της κλάσης Function.

Πρόκειται για τη βασική δομή της κλάσης εφόσον επιτρέπει τη διατήρηση ενός συνόλου από DB Attributes κάθε ένα από τα οποία είναι συνδεδεμένο με ένα σύνολο από MF.

6.1.4.2 Μέθοδοι

- `protected DrawingPanel(final int w,final int h)`

Είναι ο κατασκευαστής της κλάσης. Δίνει στο σχεδιαζόμενο panel συγκεκριμένο ύψος και πλάτος.

- `public void addFunction(Function function)`

Η μέθοδος αυτή ανακτά το τρέχον ArrayList από συναρτήσεις που χρησιμοποιείται και προσθέτει σε αυτό μία ακόμη MF, τη function.

- `public void putIntoHashFunctions(String selection)`

Η μέθοδος αυτή αναλαμβάνει να προσθέσει στο hashFunctions ένα νέο ζεύγος κλειδιού-τιμής με κλειδί το selection (στην ουσία πρόκειται για κάποιο DB Attribute) και τιμή ένα (προς το παρόν) άδειο ArrayList.

- `public void removeFromHashFunctions(String selection)`

Απομακρύνει από το hashFunctions το ζεύγος κλειδιού-τιμής με κλειδί το selection (το οποίο πρόκειται για κάποιο DB Attribute).

- `public boolean isContained(String selection)`

Αποκρίνεται εάν το hashFunctions περιέχει ένα ζεύγος κλειδιού-τιμής με κλειδί το selection (το οποίο πρόκειται για κάποιο DB Attribute).

- `public void removeFunctionFromTop()`

Για το τρέχον ArrayList από Functions αφαιρεί τη Function που βρίσκεται στην τελευταία θέση του ArrayList και στη συνέχεια εισάγει στο hashFunctions το νέο μειωμένο ArrayList ως τιμή του κλειδιού dbAttribute.

- `public void drawFunctions(Graphics2D g2d)`

Πρόκειται για τη μέθοδο που αναλαμβάνει το σχεδιασμό των τρέχουσων συναρτήσεων. Κάθε Function σχεδιάζεται με παχιά μπλε γραμμή εκτός από την τελευταία η οποία σχεδιάζεται με κόκκινη γραμμή για να δείξει ότι είναι η επιλεγμένη. Ακόμη για κάθε Function σχεδιάζονται σημάδια με λεπτή μαύρη γραμμή για να δείξουν στο χρήστη ποια σημεία των σχεδιασμένων MF παριστάνουν παραμέτρους που μπορεί να αλλάξει με το ποντίκι του.

- `public ArrayList getFunctions()`

Επιστρέφει ένα ArrayList από τις Functions του τρέχοντος DB Attribute. Σε περίπτωση που το DB Attribute είναι το άδειο String “” ή σε περίπτωση που με το τρέχον DB Attribute δεν έχει συσχετιστεί καμία MF επιστρέφεται ένα άδειο ArrayList.

- `void drawXaxis(final Graphics2D g2d)`

Η μέθοδος αυτή σχεδιάζει τον οριζόντιο άξονα του συστήματος αξόνων. Κάθε φορά σχεδιάζεται διαφορετικός άξονας ανάλογα με τις τρέχουσες τιμές των ορίων. Ο οριζόντιος άξονας περιλαμάνει 10 μικρές κάθετες γραμμές σχεδιασμένες σε ισομεγέθη διαστήματα προκειμένου να έχει ο χρήστης εποπτεία των τετμημένων. Κάτω από κάθε γραμμούλα αναγράφεται η τιμή του x, η οποία εξαρτάται από τα όρια. Χρησιμοποιείται ένα DecimalFormat για να επιτύχουμε δεκαδική ακρίβεια 4 δεκαδικών ψηφίων και σημάδι υποδιαστολής το χαρακτήρα ‘.’.

- `void drawYaxis(final Graphics2D g2d)`

Η σχεδίαση του κάθετου άξονα είναι πιο απλή και περιλαμβάνει μόνο δύο σημάδια: ένα για το 0.5 και ένα για το 1. Αυτό οφείλεται στο ότι πρόκειται για MF κανονικών ασαφών συνόλων οι οποίες παίρνουν τιμές απόκλειστικά από το [0,1].

- `public int translateX(double x)`

Η μέθοδος αυτή μετατρέπει μία τετμημένη του απεικονιζόμενου συστήματος συντεταγμένων σε τετμημένη του συστήματος συντεταγμένων του panel. Στο σημείο αυτόν να επισημάνουμε ότι ενώ στα καρτεσιανά συστήματα συντεταγμένων που συνηθίζονται στα μαθηματικά οι τετμημένες αυξάνονται από τα αριστερά προς τα δεξιά και οι τεταγμένες από κάτω προς τα πάνω, στα JPanel της java τα χ αυξάνοται μεν από τα αριστερά προς τα δεξιά αλλά τα y από τα πάνω προς τα κάτω. Στη μέθοδο αυτή επιτυγχάνουμε αυτή τη μετατροπή αρχικά πολλαπλασιάζοντας το x με ένα scale factor το οποίο έχει προέλθει από τον υπολογισμό της διαφοράς των ορίων. Στη συνέχεια προσθέτουμε στο x την τετμημένη που το σημείο (0,0) των αξόνων έχει για το σύστημα συντεταγμένων του panel.Τέλος αφαιρούμε το κάτω όριο αφού πρώτα το έχουμε πολλαπλασιάσει με τον παραπάνω scale factor.

- `public double inverseTranslateX(int x)`

Πρόκειται για την αντίστροφη διαδικασία της μεθόδου translateX(double x).

- `public int translateY(double y)`

Η μέθοδος αυτή μετατρέπει μία τεταγμένη του απεικονιζόμενου συστήματος συντεταγμένων σε τεταγμένη του συστήματος συντεταγμένων του panel.. Στη μέθοδο αυτή επιτυγχάνουμε αυτή τη μετατροπή αφαιρώντας από την τεταγμένη του σημείου (0,0) των αξόνων το y αφού πρώτα το έχουμε πολλαπλασιάσει με το scale factor 100 ο οποίος είναι προκαθορισμένος αφού τα όρια του κάθετου άξονα δε μεταβάλλονται.

- `public int[] translateArrayX(double[] inputX)`

Δέχεται ως είσοδο έναν πίνακα από double και επιστρέφει ως έξοδο τον παραπάνω πίνακα τα μέλη του οποίου έχουν προηγουμένως περάσει από τη μέθοδο translateX(double x).

- `public int[] translateArrayY(double[] inputY)`

Δέχεται ως είσοδο έναν πίνακα από double και επιστρέφει ως έξοδο τον παραπάνω πίνακα τα μέλη του οποίου έχουν προηγουμένως περάσει από τη μέθοδο translateY(double y).

- `public void setMyBounds(MyBounds b)`

Πρόκειται για το setter του πεδίου bounds. Δεν έχει χρησιμοποιηθεί το όνομα setBounds γι'αυτή τη μέθοδο για να αποφύγουμε το overwrite με την ήδη υλοποιημένη μέθοδο setBounds της κλάσης JPanel.

- `public MyBounds getMyBounds()`

Πρόκειται για τον getter του πεδίου bounds. Δεν έχει χρησιμοποιηθεί το όνομα getBounds γι'αυτή τη μέθοδο για να αποφύγουμε το overwrite με την ήδη υλοποιημένη μέθοδο getBounds της κλάσης JPanel.

- `public void bringOnTop(int index)`

Η μέθοδος αυτή εντοπίζει τη Function του τρέχοντος ArrayList από Functions που βρίσκεται στη θέση i και την τοποθετεί στην τελευταία θέση του ArrayList.

- `public Function getFromTop()`

Επιστρέφει τη Function που βρίσκεται στην τελευταία θέση του τρέχοντος ArrayList από Functions.

- `public void paint(final Graphics g)`

Η μέθοδος αυτή καλείται κάθε φορά που σχεδιάζεται ένα αντικείμενο τύπου DrawingPanel. Αρχικά καθαρίζει ένα ορθογώνιο διαστάσεων 500 x 300 pixels βάφοντας το άσπρο, σχεδιάζει τον οριζόντιο άξονα με τις τιμές του, τον κάθετο άξονα με τις τιμές του και τέλος τις συναρτήσεις που έχουν οριστεί με την προϋπόθεση το dbAttribute να μην είναι άδειο όπως και το τρέχον ArrayList από Functions.

6.1.5 Function.java

```
public abstract class Function
```

Πρόκειται για μία αφηρημένη κλάση η οποία αντιπροσωπεύει μία membership function. Την κλάση αυτή θα επεκτείνουν οι MF που θα ακολουθήσουν και θα έχουν συγκεκριμένο σχήμα όπως τριγωνικό, τραπεζοειδές κλπ.

6.1.5.1 Πεδία

- `private String type;`

Πρόκειται για το πεδίο που περιέχει τον τύπο της συνάρτησης. Στη παρούσα υλοποίηση αυτό μπορεί να λαμβάνει μία από τις ακόλουθες τιμές: “Triangular”, “Trapezoidal”, “Sigmoid” και “Gaussian”.

- `private ArrayList parameters;`

Το πεδίο αυτό περιέχει τις παραμέτρους της κάθε συνάρτησης. Πρόκειται για ένα ArrayList από double αριθμούς.

- `private MyBounds bounds;`

Πρόκειται για ένα αντικείμενο της κλάσης MyBounds που αντιπροσωπεύει τα όρια προβολής της κάθε συνάρτησης.

6.1.5.2 Μέθοδοι

- `public double[] computeY(double[] arrayX,ArrayList parameters)`

Η παρούσα μέθοδος λαμβάνει ως είσοδο έναν πίνακα από double καθώς και ένα ArrayList από παραμέτρους και επιστρέφει έναν πίνακα από double που έχει προέλθει από τον υπολογισμό κάθε στοιχείου του πίνακα εισόδου. Για τον υπολογισμό χρησιμοποιείται η μέθοδος graph(arrayX[i], parameters).

- `public void draw(Graphics2D g2D,DrawingPanel dPane,int density)`

Η παρούσα μέθοδος αναλαμβάνει το σχεδιασμό της γραφικής παράστασης της συνάρτησης στο dPane. Επειδή η java δε διαθέτει μεθόδους απευθείας σχεδιασμού πολύπλοκων συναρτήσεων η μέθοδος αυτή σχεδιάζει ένα σύνολο από σημεία πυκνοτοποθετημένα μεταξύ τους και στη συνέχεια ενώνει κάθε δύο γειτονικά τέτοια σημεία με ένα ευθύγραμμο τμήμα. Με τον τρόπο αυτό φαίνεται σα να έχει σχεδιαστεί η συνάρτηση βάσει του τύπου της. Αρχικά λαμβάνεται ένα σύνολο από σημεία βάσει των ορίων του dPane, τα initialXs. Κατόπιν, με τη βοήθεια της computeY παραπάνω υπολογίζεται οι τετμημένες της συνάρτησης και τοποθετούνται στον πίνακα computedYs. Επειδή όμως αυτές είναι οι συντεταγμένες της συνάρτησης και όχι του panel απαιτείται πριν το σχεδιασμό η μετάφραση των συντεταγμένων στο σύστημα συντεταγμένων του panel. Η παράμετρος density δείχνει το πόσο πυκνά θέλουμε να είναι τα σημεία. Στην παρούσα υλοποίηση η κλήση της συνάρτησης draw γίνεται απο την drawFunctions της DrawingPanel με παράμετρο density 400. Σε κάθε for loop σχεδιάζεται ένα σημείο (ουσιαστικά ένα ορθογώνιο με μήκος και πλάτος ένα pixel) και μία ευθεία που συνδέει το παρόν σημείο με το τελευταίο σημείο. Τέλος αποθηκεύεται ως τελευταίο σημείο το παρόν σημείο.

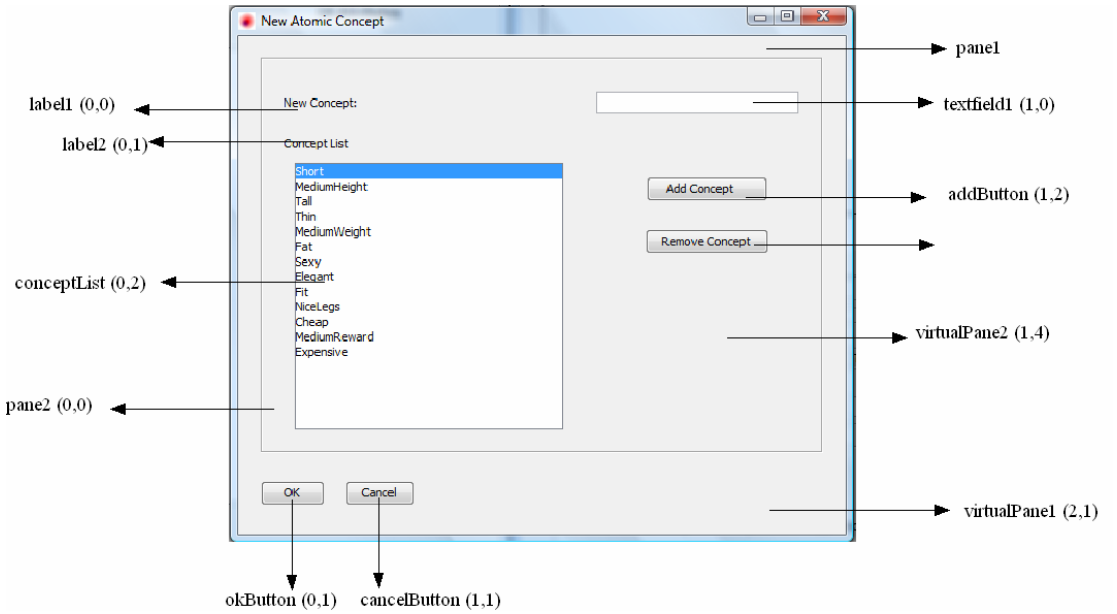
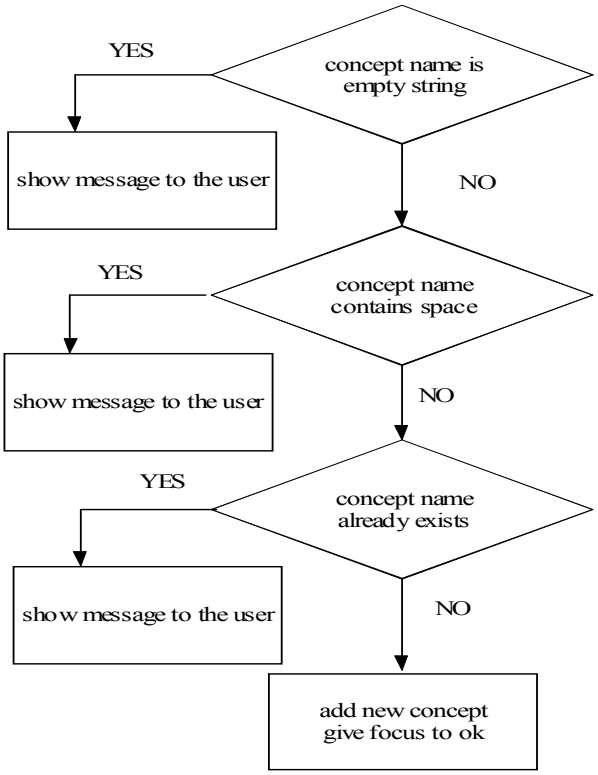
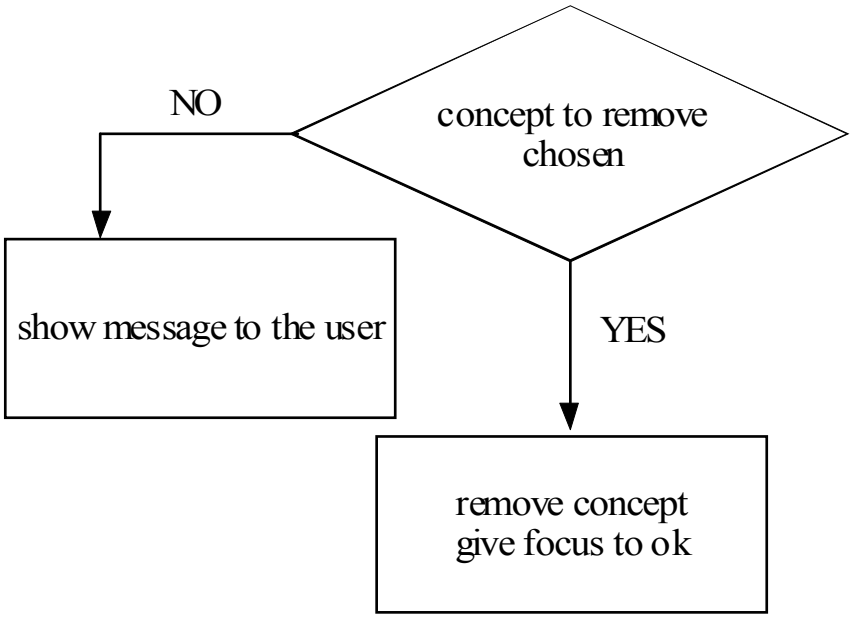
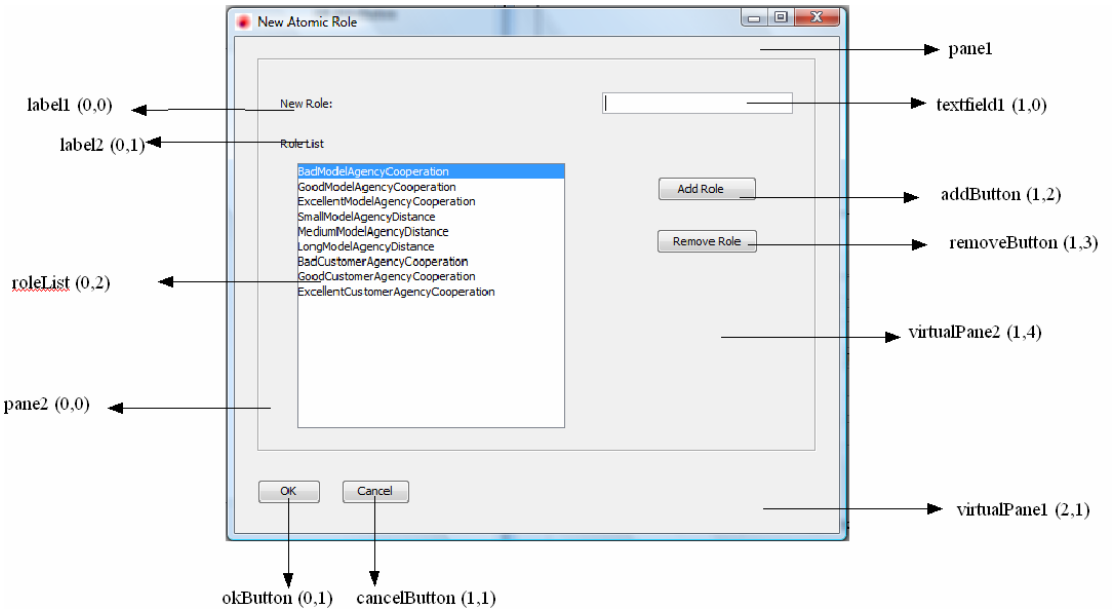
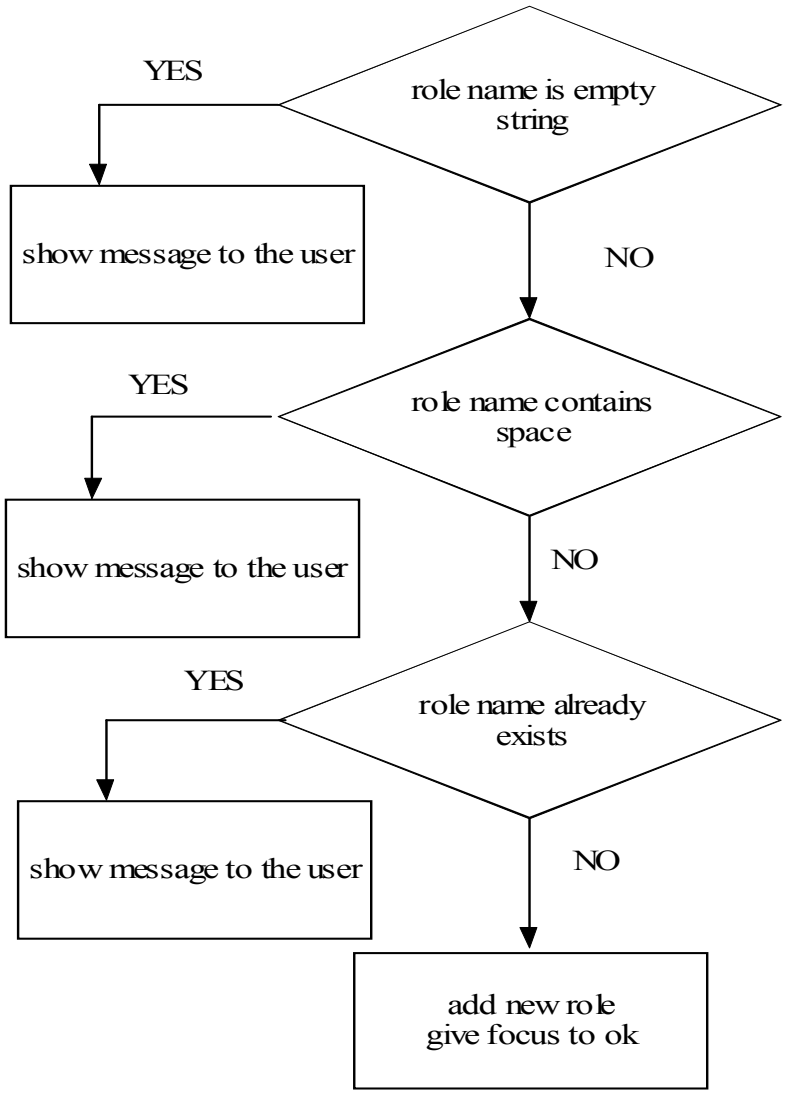
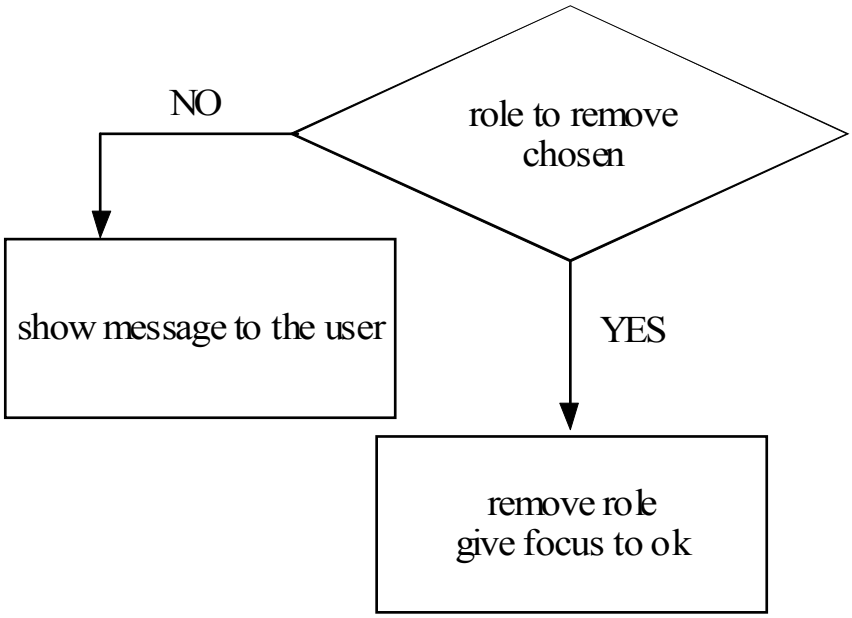
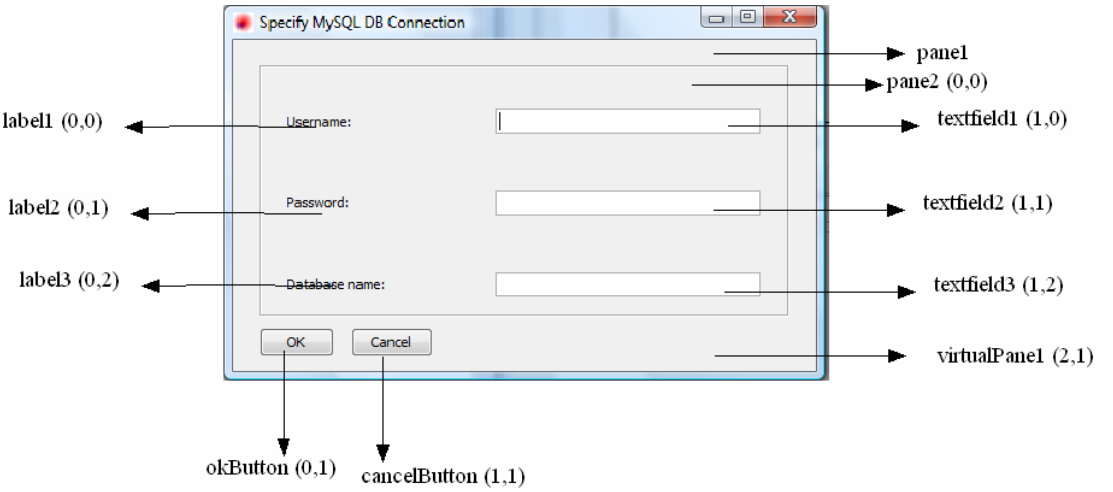
- ```
  public abstract double graph(double x,ArrayList parameters);
  ```

Η παρούσα αφηρημένη μέθοδος δέχεται ως είσοδο έναν double αριθμό x και ένα ArrayList από παραμέτρους και επιστρέφει την τιμή της συνάρτησης γι' αυτό το x. Η υλοποίηση της εξαρτάται από τον τύπο της συνάρτησης και είναι απαραίτητη εάν μία κλάση θέλει να κληρονομήσει από τη Function.

- ```
  public abstract void drawCrosses(Graphics2D g2D,ArrayList
  parameters,DrawingPanel dPane);
  ```

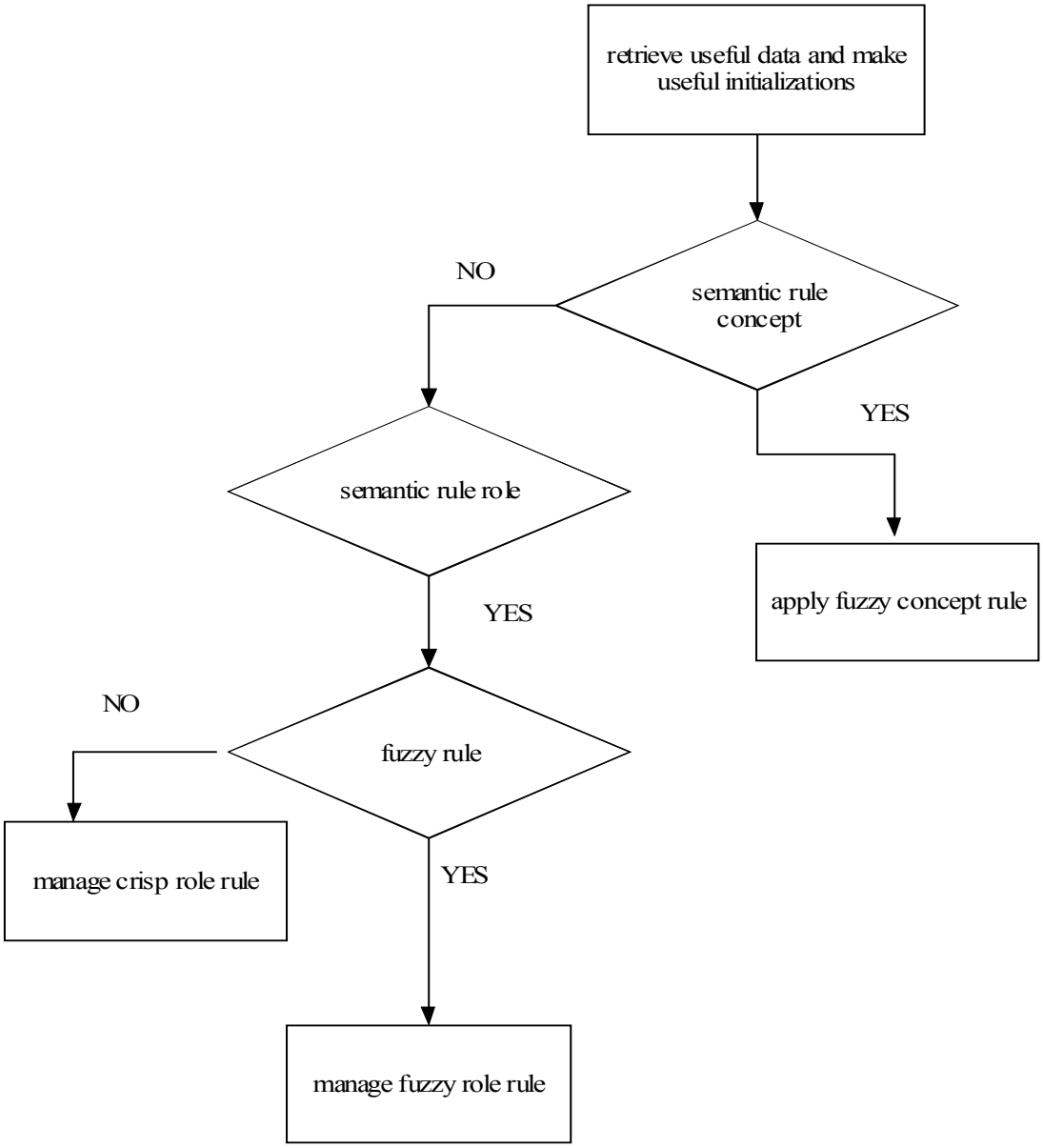
Η παρούσα αφηρημένη μέθοδος σχεδιάζει για κάθε συνάρτηση κάποια σημάδια τα οποίο βοηθούν το χρήστη να τροποποιήσει τη MF με τη βοήθεια του ποντικιού του. Η υλοποίηση της εξαρτάται από τον τύπο της συνάρτησης και είναι απαραίτητη εάν μία κλάση θέλει να κληρονομήσει από τη Function.

- ```
  public abstract boolean inCrosses(int x,int y,DrawingPanel
  dPane);
  ```

Η παρούσα αφηρημένη μέθοδος αποκρίνεται κατά πόσο το σημείο (x,y) βρίσκεται εντός των σταυρών που έχει σχεδιάσει η drawCrosses. Η υλοποίηση της εξαρτάται από τον τύπο της συνάρτησης και είναι απαραίτητη εάν μία κλάση θέλει να κληρονομήσει από τη Function.
  ```

- `public abstract char whichPar(int x,int y,DrawingPanel dPane);`

Η παρούσα αφηρημένη μέθοδος υπολογίζει στην αλλαγή ποιας παραμέτρου της συνάρτησης αναφέρεται το σημείο (x,y). Η υλοποίηση της εξαρτάται από τον τύπο της συνάρτησης και είναι απαραίτητη εάν μία κλάση θέλει να κληρονομήσει από τη Function.

6.1.6 TriangFunction.java

```
public class TriangFunction  extends Function
```

Πρόκειται για μία επέκταση της κλάσης Function που περιγράφηκε πιο πάνω.

6.1.6.1 Πεδία

Κανένα πρόσθετο πεδίο.

6.1.6.2 Μέθοδοι

- `public TriangFunction(ArrayList pars,double lBound,double uBound)`

Πρόκειται για τον κατασκευαστή της κλάσης ο οποίος θέτει τις παραμέτρους καθώς επίσης και τα δύο όρια.

- `public double graph(double x,ArrayList parameters);`

Πρόκειται ουσιαστικά για τη μέθοδο που υλοποιεί το μαθηματικό τύπο της συνάρτησης ο οποίος είναι ο εξής:

$$f_{triangular}(x) = \begin{cases} 1, \text{ εάν } x = a = c = b \\ \dfrac{b-x}{b-a}, \text{ εάν } a = c, a <= x, x <= b \\ \dfrac{x-a}{c-a}, \text{ εάν b} = c, a <= x, x <= c \\ \dfrac{x-a}{c-a}, \text{ εάν } a <= x, x <= c \\ \dfrac{b-x}{b-c}, \text{ εάν } c <= x, x <= b \\ 0 \quad , \text{ διαφορετικά} \end{cases}$$

Να σημειώσουμε ότι στον παραπάνω τύπο οι παράμετροι a,c,b αντιστοιχούν στην αριστερή, μεσαία και δεξιά τετμημένη των τριων γωνιών του τριγώνου.

- `public void drawCrosses(Graphics2D g2D,ArrayList parameters,DrawingPanel dPane);`

Η μέθοδος αυτή σχεδιάζει σταυρούς στις τρεις γωνίες του τριγώνου αφού πρώτα επαληθεύσει ότι αυτές βρίσκονται εντός των απεικονιζόμενων ορίων.

- `public boolean inCrosses(int x,int y,DrawingPanel dPane);`

Η μέθοδος αυτή αποκρίνεται για το αν το σημείο x,y βρίσκεται εντός των τριων σταυρών που η drawCrosses σχεδίασε.

- `public char whichPar(int x,int y,DrawingPanel dPane);`

Η μέθοδος αυτή διαπιστώνει σε ποιόν από τους τρεις σταυρούς (δηλαδή για ποια από τις τρεις παραμέτρους πρόκειται) βρίσκεται το σημείο (x,y) και αν βρίσκεται σε κάποιον από τους σταυρούς. Επιστρέφει ανάλογα ‘a’, ‘c’ ή ‘b’. Εάν δε βρίσκεται στο εσωτερικό κανενός από τους σταυρούς επιστρέφει ‘z’.

6.1.7 TrapezFunction.java

```
public class TrapezFunction  extends Function
```

Πρόκειται για μία επέκταση της κλάσης Function που περιγράφηκε πιο πάνω.

6.1.7.1 Πεδία

Κανένα πρόσθετο πεδίο.

6.1.7.2 Μέθοδοι

- `public TrapezFunction(ArrayList pars,double lBound,double uBound)`

Πρόκειται για τον κατασκευαστή της κλάσης ο οποίος θέτει τις παραμέτρους καθώς επίσης και τα δύο όρια.

- `public double graph(double x,ArrayList parameters);`

Πρόκειται ουσιαστικά για τη μέθοδο που υλοποιεί το μαθηματικό τύπο της συνάρτησης ο οποίος είναι ο εξής:

$$f_{trapezoidal}(x) = \begin{cases} 1, \text{ εάν } x = a = c = b = d \\ f_{triangular}(x), \text{ εάν } b = c \\ \frac{d-x}{d-c}, \text{ εάν } a = b = c \\ \frac{x-a}{b-a}, \text{ εάν } b = c = d \\ \frac{x-a}{b-a}, \text{ εάν } a <= x, x <= b \\ 1, \text{ εάν } b <= x, x <= c \\ \frac{d-x}{d-c}, \text{ εάν } c <= x, x <= d \\ 0 \quad , \text{ διαφορετικά} \end{cases}$$

Να σημειώσουμε ότι στον παραπάνω τύπο οι παράμετροι a,b,c,d αντιστοιχούν στην αριστερή, αριστερή μεσαία, δεξιά μεσαία και δεξιά τετμημένη των τεσσάρων γωνιών του τραπεζίου.

- `public void drawCrosses(Graphics2D g2D,ArrayList parameters,DrawingPanel dPane);`

Η μέθοδος αυτή σχεδιάζει σταυρούς στις τέσσερις γωνίες του τραπεζίου αφού πρώτα επαληθεύσει ότι αυτές βρίσκονται εντός των απεικονιζόμενων ορίων.

- `public boolean inCrosses(int x,int y,DrawingPanel dPane);`

Η μέθοδος αυτή αποκρίνεται για το αν το σημείο x,y βρίσκεται εντός των τεσσάρων σταυρών που η drawCrosses σχεδίασε.

- `public char whichPar(int x,int y,DrawingPanel dPane);`

Η μέθοδος αυτή διαπιστώνει σε ποιόν από τους τέσσερις σταυρούς (δηλαδή για ποια από τις τέσσερις παραμέτρους πρόκειται) βρίσκεται το σημείο (x,y) και αν βρίσκεται σε κάποιον από τους σταυρούς. Επιστρέφει ανάλογα ‘a’, ‘b’ , ‘c’ ή ‘d’. Εάν δε βρίσκεται στο εσωτερικό κανενός από τους σταυρούς επιστρέφει ‘z’.

6.1.8 GaussianFunction.java

```
public class GaussianFunction  extends Function
```

Πρόκειται για μία επέκταση της κλάσης Function που περιγράφηκε πιο πάνω.

6.1.8.1 Πεδία

Κανένα πρόσθετο πεδίο.

6.1.8.2 Μέθοδοι

- `public GaussianFunction(ArrayList pars,double lBound,double uBound)`

Πρόκειται για τον κατασκευαστή της κλάσης ο οποίος θέτει τις παραμέτρους καθώς επίσης και τα δύο όρια.

- `public double graph(double x,ArrayList parameters);`

Πρόκειται ουσιαστικά για τη μέθοδο που υλοποιεί το μαθηματικό τύπο της συνάρτησης ο οποίος είναι ο εξής:

$$f_{gaussian}(x) = e^{\frac{(x-c)^2}{-2\sigma^2}}$$

Να σημειώσουμε ότι στον παραπάνω τύπο η παράμετρος c αντιστοιχεί στην τετμημένη της κορυφής της καμπύλης, ενώ η παράμετρος σ στο «άνοιγμα» της καμπύλης.

- `public void drawCrosses(Graphics2D g2D,ArrayList parameters,DrawingPanel dPane);`

Η μέθοδος αυτή σχεδιάζει έναν σταυρό στην κορυφή της καμπύλης και δύο σταυρούς στα δύο σημεία για τα οποία ισχύει $f_{gaussian}(x) = 0.5$. Πριν τη σχεδίαση κάθε σταυρού επαληθεύει ότι αυτός βρίσκεται εντός των απεικονιζόμενων ορίων.

- `public boolean inCrosses(int x,int y,DrawingPanel dPane);`

Η μέθοδος αυτή αποκρίνεται για το αν το σημείο x,y βρίσκεται εντός των τριων σταυρών που η drawCrosses σχεδίασε.

- `public char whichPar(int x,int y,DrawingPanel dPane);`

Η μέθοδος αυτή διαπιστώνει σε ποιόν από τους τρεις σταυρούς (δηλαδή για ποια από τις δύο παραμέτρους πρόκειται) βρίσκεται το σημείο (x,y) και αν βρίσκεται σε κάποιον από τους σταυρούς. Επιστρέφει ανάλογα ‘c’ ή ‘s’. Εάν δε βρίσκεται στο εσωτερικό κανενός από τους σταυρούς επιστρέφει ‘z’.

6.1.9 SigmoidFunction.java

```
public class SigmoidFunction  extends Function
```

Πρόκειται για μία επέκταση της κλάσης Function που περιγράφηκε πιο πάνω.

6.1.9.1 Πεδία

Κανένα πρόσθετο πεδίο.

6.1.9.2 Μέθοδοι

- `public SigmoidFunction(ArrayList pars,double lBound,double uBound)`

Πρόκειται για τον κατασκευαστή της κλάσης ο οποίος θέτει τις παραμέτρους καθώς επίσης και τα δύο όρια.

- `public double graph(double x,ArrayList parameters);`

Πρόκειται ουσιαστικά για τη μέθοδο που υλοποιεί το μαθηματικό τύπο της συνάρτησης ο οποίος είναι ο εξής:

$$f_{sigmoid}(x) = \frac{1}{1 + e^{\frac{-a}{x-c}}}$$

Να σημειώσουμε ότι στον παραπάνω τύπο η παράμετρος c αντιστοιχεί στην τετμημένη της καμπύλης με τεταγμένη 0.5, ενώ η παράμετρος α στο «άνοιγμα» της καμπύλης.

- `public void drawCrosses(Graphics2D g2D,ArrayList parameters,DrawingPanel dPane);`

Η μέθοδος αυτή σχεδιάζει τρεις σταυρούς στα σημεία για τα οποία ισχύει $f_{gaussian}(x) = 0.5$, $f_{sigmoid}(x) = 0.2$ και $f_{sigmoid}(x) = 0.8$. Πριν τη σχεδίαση κάθε σταυρού επαληθεύει ότι αυτός βρίσκεται εντός των απεικονιζόμενων ορίων.

- `public boolean inCrosses(int x,int y,DrawingPanel dPane);`

Η μέθοδος αυτή αποκρίνεται για το αν το σημείο x,y βρίσκεται εντός των τριων σταυρών που η drawCrosses σχεδίασε.

- `public char whichPar(int x,int y,DrawingPanel dPane);`

Η μέθοδος αυτή διαπιστώνει σε ποιόν από τους τρεις σταυρούς (δηλαδή για ποια από τις δύο παραμέτρους πρόκειται) βρίσκεται το σημείο (x,y) και αν βρίσκεται σε κάποιον από τους σταυρούς. Επιστρέφει ανάλογα ‘c’ , ’l’ ή ‘r’. Εάν δε βρίσκεται στο εσωτερικό κανενός από τους σταυρούς επιστρέφει ‘z’.

6.1.10 OneFunction.java

```
public class OneFunction  extends Function
```

Πρόκειται για μία επέκταση της κλάσης Function που περιγράφηκε πιο πάνω. Παρά το γεγονός ότι αποτελεί επέκταση της Function δεν είναι γνήσια MF αλλά χρησιμοποιείται στην περίπτωση των crisp assertions και σε κάθε περίπτωση επιστρέφει 1. Ο λόγος για τον οποίο προστέθηκε είναι επειδή είναι απαραίτητο για τη δημιουργία κάθε SR να προστίθεται και μία συνάρτηση, και επειδή καμία από τις παραπάνω δεν είναι κατάλληλη αποφασίσαμε να προσθέσουμε μία ειδικά γι’αυτό το σκοπό.

6.1.10.1 Πεδία

Κανένα πρόσθετο πεδίο.

6.1.10.2 Μέθοδοι

- `public OneFunction(ArrayList pars,double lBound,double uBound)`

Πρόκειται για τον κατασκευαστή της κλάσης που σε αυτή την περίπτωση δεν κάνει κάτι.

- `public double graph(double x,ArrayList parameters);`

Η μέθοδος αυτή επιστρέφει πάντοτε 1.

- `public void drawCrosses(Graphics2D g2D,ArrayList parameters,DrawingPanel dPane);`

- `public boolean inCrosses(int x,int y,DrawingPanel dPane);`

- `public char whichPar(int x,int y,DrawingPanel dPane);`

Η υλοποίηση των 3 παραπάνω μεθόδων είναι υποχρεωτική εφόσον η κλάση OneFunction επεκτείνει την abstract Function. Δεν υλοποιούν κάποια λειτουργία.

6.1.11 SemanticRule.java

```
public abstract class SemanticRule
```

Πρόκειται για μία αφηρημένη κλάση η οποία αντιπροσωπεύει τη δομή εκείνη δεδομένων η οποία κωδικοποιεί τις επιθυμίες του χρήστη ως προς την παραγωγή των ασαφών υποθέσεων και την οποία το πρόγραμμα με τη σειρά του χρησιμοποιεί κατά την εξαγωγή του ABox. Την κλάση αυτή θα επεκτείνουν τα δύο είδη SR που θα ακολουθήσουν και θα αφορούν ατομικές έννοιες και ατομικούς ρόλους.

6.1.11.1 Πεδία

- `private String dbAttribute;`

Το πεδίο αυτό περιέχει το ιδιοχαρακτηριστικό της βάσης δεδομένων στο οποίο αναφέρεται ο SR.

- `private Function function;`

Το πεδίο αυτό περιέχει τη MF στην οποία αναφέρεται ο SR.

- `private String type;`

Το πεδίο αυτό παίρνει μία από τις τιμές “concept” ή “role” και δείχνει εάν ο SR αναφέρεται σε ατομική έννοια ή ατομικό ρόλο.

- `private boolean fuzzy=true;`

Το πεδίο αυτό δείχνει εάν ο SR αναφέρεται σε παραγωγή ασαφούς υποθέσεως ή όχι.

6.1.11.2 Μέθοδοι

- `public String retrieveRelation()`

Η παρούσα μέθοδος επιστρέφει τη σχέση από την οποία προέρχεται το DB Attribute του SR.

- `public String retrieveAttribute()`

Η παρούσα μέθοδος επιστρέφει το όνομα του DB Attribute του σημασιολογικού κανόνα χώρις το πρόθεμα που περιλαμβάνει το όνομα της σχέσης και την τελεία.

- `public abstract void printMe();`

Η παρούσα αφηρημένη μέθοδος τυπώνει το SR στο standard output και έχει δημιουργηθεί για λόγους debugging. Η υλοποίηση της είναι απαραίτητη εάν μία κλάση θέλει να κληρονομήσει από τη Function.

6.1.12 SemanticRuleConcept.java

```
public class SemanticRuleConcept extends SemanticRule
```

Πρόκειται για μία επέκταση της κλάσης SemanticRule που περιγράφηκε πιο πάνω. Καλύπτει την περίπτωση κάτα την οποία ο κανόνας αναφέρεται σε ατομική έννοια.

6.1.12.1 Πεδία

- `private String atomicConcept;`

Πρόκειται για το όνομα της ατομικής έννοιας στην οποία αναφέρεται ο παρών SR.

- `private String key;`

Πρόκειται για το DB Attribute εκείνο στο οποίο αναφέρεται η ατομική έννοια αυτού του SR. Συνήθως αποτελεί το πρωτεύον κλειδί κάποιας σχέσης ή μέρος του πρωτεύοντος κλειδιού αυτής.

6.1.12.2 Μέθοδοι

- `public SemanticRuleConcept(String dbAttribute,String ac,Function function)`

Πρόκειται για τον κατασκευαστή της κλάσης ο οποίος συνθέτει τον κανόνα θέτοντας στα κατάλληλα πεδία τα dbAttribute, ac και Function. Επιπλέον θέτει τον τύπο “concept” δηλώνοντας ότι πρόκειται για SR που εμπλέκει έννοια.

- `public void printMe()`

Η μέθοδος αυτή τυπώνει στη standard έξοδο τα δομικά στοιχεία από τα οποία αποτελείται ο παρών SR. Χρησιμοποιείται για debugging.

6.1.13 SemanticRuleRole.java

```
public class SemanticRuleRole extends SemanticRule
```

Πρόκειται για μία επέκταση της κλάσης SemanticRule που περιγράφηκε πιο πάνω. Καλύπτει την περίπτωση κάτα την οποία ο κανόνας αναφέρεται σε ατομικό ρόλο.

6.1.13.1 Πεδία

- `private String atomicRole;`

Πρόκειται για το όνομα του ατομικού ρόλου στον οποίο αναφέρεται ο παρών SR.

- `private String domain;`

Πρόκειται για το DB Attribute εκείνο στο οποίο αναφέρεται το domain του ατομικού ρόλου αυτού του σημασιολογικού κανόνα. Συνήθως αποτελεί μία από τις στήλες που συγκροτούν το πρωτεύον κλειδί κάποιας σχέσης.

- `private String domain;`

Πρόκειται για το DB Attribute εκείνο στο οποίο αναφέρεται το range του ατομικού ρόλου αυτού του SR. Συνήθως αποτελεί μία από τις στήλες που συγκροτούν το πρωτεύον κλειδί κάποιας σχέσης.

6.1.13.2 Μέθοδοι

- `public SemanticRuleRole(String dbAttribute,String ar,Function function)`

Πρόκειται για τον κατασκευαστή της κλάσης ο οποίος συνθέτει τον κανόνα θέτοντας στα κατάλληλα πεδία τα dbAttribute, ar και Function. Επιπλέον θέτει τον τύπο “role” δηλώνοντας ότι πρόκειται για SR που εμπλέκει ρόλο.

- `public void printMe()`

Η μέθοδος αυτή τυπώνει στη standard έξοδο τα δομικά στοιχεία από τα οποία αποτελείται ο παρών SR. Χρησιμοποιείται για debugging.

6.1.14 SemanticRulesManager.java

```
public class SemanticRulesManager
```

Η κλάση αυτή είναι υπεύθυνη για τη διαχείριση των σημασιολογικών κανόνων που δημιουργούνται και αφαιρούνται κατά τη λειτουργία του συστήματος.

6.1.14.1 Πεδία

- `private ArrayList semanticRules;`

Πρόκειται για το σύνολο των SR που έχει δημιουργήσει ο χρήστης μέχρι στιγμής.

- `private DBManager dbManager;`

Είναι μια αναφορά στο αντικείμενο της κλάσης DBManager η οποία διαχειρίζεται τις διασυνδέσεις με τη βάση δεδομένων.

- `private DBConnectionFrame dbcf;`

Είναι μια αναφορά στο αντικείμενο της κλάσης DBConnectionFrame η οποία κρατά ορισμένα στοιχεία για τη διασύνδεση με τη ΒΔ.

6.1.14.2 Μέθοδοι

- `public SemanticRulesManager(DBConnectionFrame dbcframe)`

Πρόκειται για τον κατασκευαστή της κλάσης. Δημιουργεί ένα (προς το παρόν άδειο) ArrayList για το πεδίο semanticRules και επιπλέον θέτει το DBConnectionFrame με τη βοήθεια το dbcframe.

- `public void addSemanticRule(SemanticRule sr)`

Προσθέτει στο ArrayList semanticRules τον κανόνα sr.

- `public boolean checkUniqueConcept(SemanticRuleConcept src)`

Η παρούσα μέθοδος ελέγχει εάν υπάρχει ήδη κάποιος SR για την ατομική έννοια του src. Για το σκοπό αυτό εξετάζει έναν έναν τους υπάρχοντες SR και εάν κάποιος από αυτούς εμπλέκει την ατομική έννοια του src επιστρέφεται false, αλλιώς true.

- `public boolean checkUniqueRole(SemanticRuleRole srr)`

Η παρούσα μέθοδος ελέγχει εάν υπάρχει ήδη κάποιος SR για τον ατομικό ρόλο του srr. Για το σκοπό αυτό εξετάζει έναν έναν τους υπάρχοντες SR και εάν κάποιος από αυτούς εμπλέκει τον ατομικό ρόλο του srr επιστρέφεται false, αλλιώς true.

- `public void removeSemanticRule(int index)`

Αφαιρεί από το ArrayList semanticRules τον κανόνα που βρίσκεται στη θέση index.

- `public void printMe()`

Πρόκειται για μία μέθοδο που χρησιμοποιείται για debugging και τυπώνει στη standard έξοδο το σύνολο των SR που έχουν δημιουργηθεί μέχρι στιγμής.

6.1.15 NewAtomicConceptFrame.java

public class NewAtomicConceptFrame **extends** JFrame **implements** ActionListener

6.1.15.1 Πεδία

- **private** JButton okButton,cancelButton,addButton,removeButton;

Πρόκειται για 4 κουμπιά που χρησιμοποιούνται κατά την αλληλεπίδραση με το χρήστη.

- **private** JLabel label1,label2;

Πρόκειται για δύο ετικέτες που συμμετέχουν στο γραφικό περιβάλλον.

- **private** JTextField textField1;

Πρόκειται για το πεδίο κειμένου που χρησιμοποιεί ο χρήστης για να εισάγει το όνομα της νέας ατομικής έννοιας.

- **private** JList conceptList;

Πρόκειται για τη λίστα των ατομικών εννοιών που εμφανίζεται στο χρήστη.

- **private** DefaultListModel conceptListModel;

Πρόκειται για τα περιεχόμενα της λίστας ατομικών εννοιων της παρούσας κλάσης.

- **private** DefaultListModel newConceptListModel;

Πρόκειται για μία αναφορά στα περιεχόμενα της λίστας ατομικών εννοιών του FuzzyFrame.

6.1.15.2 Μέθοδοι

- **public** NewAtomicConceptFrame(DefaultListModel existingConceptListModel)

Είναι ο κατασκευαστής της κλάσης. Εφόσον πρόκειται για subclass της JFrame χρησιμοποιεί τον πατρικό constructor και θέτει τον τίτλο. Στη συνέχεια τοποθετεί το JFrame σε συγκεκριμένη θέση ως προς τα όρια της οθόνης. Θέτει το εικονίδιο το οποίο βρίσκεται σε σχετικό path του σκληρού. Δημιουργεί 2 panel εκ των οποίων το 1 με ευδιάκριτα όρια. Δημιουργείται ένας GridBagLayoutManager για κάθε panel, προκειμένου να ρυθμίσει τη διάταξη των συστατικών που θα επικολληθούν σε αυτό. Το σχήμα 164 δείχνει τη θέση και τη χρησιμότητα του κάθε panel. Επιπλέον δείχνει που έχει τοποθετηθεί κάθε component και τις συντεταγμένες αυτού. Η διάταξη των συστατικών έχει γίνει με το γνωστό τρόπο.



Σχήμα 164. Γραφική διάταξη του "New Atomic Concept" frame.

Επιπλέον αξίζει να αναφέρουμε ότι για λόγους αισθητικής προσθέσαμε δύο JPanel τα οποία δεν έχουν καμία πρακτική σημασία αλλά βοηθούν στην καλύτερη τακτοποίηση των γραφικών components. Όσον αφορά τη λίστα των ατομικών εννοιών έχουμε δύο αντικείμενα της κλάσης DefaultListModel. Το conceptListModel περιέχει τα στοιχεία της λίστας που απεικονίζεται στο "New Atomic Concept" frame και αρχικοποιείται με τα ήδη υπάρχοντα στοιχεία της λίστας στο κύριο frame. Αυτά τα λαμβάνει από το όρισμα existingConceptListModel του κατασκευαστή. Το newConceptListModel αναφέρεται στα περιεχόμενα της λίστας του frame όπως αυτά θα διαμορφωθούν μετά τις (πιθανές) τροποποιήσεις που θα επιφέρει ο χρήστης.

- `public void actionPerformed(ActionEvent e)`

Ανιχνεύει ποιο component προκάλεσε το γεγονός και εκτελεί την αντίστοιχη ενέργεια καλώντας μία ξεχωριστή μέθοδο για κάθε component. Η υλοποίηση αυτής της μεθόδου είναι υποχρεωτική εφόσον η κλάση NewAtomicConceptFrame υλοποιεί το interface ActionListener.

- `protected void manageAddConcept()`

Το σχήμα 165 συνοψίζει την παραπάνω μέθοδο. Να διευκρινίσουμε ότι η παραχώρηση του focus στο okButton γίνεται για να έχει ο χρήστης τη δυνατότητα με το πάτημα του πλήκτρου enter να πατάει το okButton μετά την προσθήκη ενός atomic concept.



Σχήμα 165. Μέθοδος manageAddConcept().

- `protected void manageRemoveConcept()`

Το σχήμα 166 συνοψίζει την παραπάνω μέθοδο. Και πάλι η παραχώρηση του focus στο okButton γίνεται για τους λόγους της προηγούμενης μεθόδου.



Σχήμα 166. Μέθοδος manageRemoveConcept().

- `protected void manageOk()`

Η μέθοδος αυτή ουσιαστικά κατασκευάζει εκ νέου τη newConceptListModel μεταφέροντας της τα περιεχόμενα της conceptListModel και κλείνει το frame.

- `protected void manageCancel()`

Κλείνει το παρόν frame χωρίς να πραγματοποιηθεί καμία αλλαγή στη λίστα των ατομικών εννοιών ανεξάρτητα από προσθαφαιρέσεις που ο χρήστης μπορεί να έχει κάνει.

6.1.16 NewAtomicRoleFrame.java

```
public class NewAtomicRoleFrame extends JFrame implements ActionListener
```

6.1.16.1 Πεδία

- `private JButton okButton,cancelButton,addButton,removeButton;`

Πρόκειται για 4 κουμπιά που χρησιμοποιούνται κατά την αλληλεπίδραση με το χρήστη.

- `private JLabel label1,label2;`

Πρόκειται για δύο ετικέτες που συμμετέχουν στο γραφικό περιβάλλον.

- `private JTextField textField1;`
-

Πρόκειται για το πεδίο κειμένου που χρησιμοποιεί ο χρήστης για να εισάγει το όνομα της νέας ατομικής έννοιας.

- `private JList roleList;`

Πρόκειται για τη λίστα των ατομικών ρόλων που εμφανίζεται στο χρήστη.

- `private DefaultListModel roleListModel;`

Πρόκειται για τα περιεχόμενα της λίστας ατομικών ρόλων της παρούσας κλάσης.

- `private DefaultListModel newRoleListModel;`

Πρόκειται για μία αναφορά στα περιεχόμενα της λίστας ατομικών ρόλων του FuzzyFrame.

6.1.16.2 Μέθοδοι

- `public NewAtomicRoleFrame(DefaultListModel existingConceptListModel)`

Είναι ο κατασκευαστής της κλάσης. Εφόσον πρόκειται για subclass της JFrame χρησιμοποιεί τον πατρικό constructor και θέτει τον τίτλο. Στη συνέχεια τοποθετεί το JFrame σε συγκεκριμένη θέση ως προς τα όρια της οθόνης. Θέτει το εικονίδιο το οποίο βρίσκεται σε σχετικό path του σκληρού. Δημιουργεί 2 panel εκ των οποίων το 1 με ευδιάκριτα όρια. Δημιουργείται ένας GridBagLayoutManager για κάθε panel, προκειμένου να ρυθμίσει τη διάταξη των συστατικών που θα επικολληθούν σε αυτό. Το σχήμα 167 δείχνει τη θέση και τη

χρησιμότητα του κάθε panel. Επιπλέον δείχνει που έχει τοποθετηθεί κάθε component και τις συντεταγμένες αυτού. Η διάταξη των συστατικών έχει γίνει με το γνωστό τρόπο.



Σχήμα 167. Γραφική διάταξη του "New Atomic Concept" frame.

Επιπλέον αξίζει να αναφέρουμε ότι για λόγους αισθητικής προσθέσαμε δύο JPanel τα οποία δεν έχουν καμία πρακτική σημασία αλλά βοηθούν στην καλύτερη τακτοποίηση των γραφικών components. Όσον αφορά τη λίστα των ατομικών ρόλων έχουμε δύο αντικείμενα της κλάσης DefaultListModel. Το roleListModel περιέχει τα στοιχεία της λίστας που απεικονίζεται στο "New Atomic Role" frame και αρχικοποιείται με τα ήδη υπάρχοντα στοιχεία της λίστας στο κύριο frame. Αυτά τα λαμβάνει από το όρισμα existingRoleListModel του κατασκευαστή. Το newRoleListModel αναφέρεται στα περιεχόμενα της λίστας του frame όπως αυτά θα διαμορφωθούν μετά τις (πιθανές) τροποποιήσεις που θα επιφέρει ο χρήστης.

- `public void actionPerformed(ActionEvent e)`

Ανιχνεύει ποιο component προκάλεσε το γεγονός και εκτελεί την αντίστοιχη ενέργεια καλώντας μία ξεχωριστή μέθοδο για κάθε component. Η υλοποίηση αυτής της μεθόδου είναι υποχρεωτική εφόσον η κλάση NewAtomicRoleFrame υλοποιεί το interface ActionListener.

- `protected void manageAddRole()`

Το σχήμα 168 συνοψίζει την παραπάνω μέθοδο. Να διευκρινίσουμε ότι η παραχώρηση του focus στο okButton γίνεται για να έχει ο χρήστης τη δυνατότητα με το πάτημα του πλήκτρου enter να πατάει το okButton μετά την προσθήκη ενός atomic role.



Σχήμα 168. Μέθοδος manageAddRole().

- `protected void manageRemoveRole()`

Το σχήμα 169 συνοψίζει την παραπάνω μέθοδο. Και πάλι η παραχώρηση του focus στο okButton γίνεται για τους λόγους της προηγούμενης μεθόδου.



Σχήμα 169. Μέθοδος manageRemoveRole().

- `protected void manageOk()`

Η μέθοδος αυτή ουσιαστικά κατασκευάζει εκ νέου τη newRoleListModel μεταφέροντας της τα περιεχόμενα της roleListModel και κλείνει το frame.

- `protected void manageCancel()`

Κλείνει το παρόν frame χωρίς να πραγματοποιηθεί καμία αλλαγή στη λίστα των ατομικών ρόλων ανεξάρτητα από προσθαφαιρέσεις που ο χρήστης μπορεί να έχει κάνει.

6.1.17 MyOntologyManager.java

`public class MyOntologyManager`

Η παρούσα κλάση αναλαμβάνει τη διαχείριση της οντολογίας από την οποία το σύστημα λαμβάνει τις ατομικές έννοιες και τους ατομικούς ρόλους που εμφανίζει στο χρήστη. Χρησιμοποιεί εκτενώς το OWL API προκειμένου να επεξεργαστεί το .owl αρχείο που έχει υποδείξει ο χρήστης.

6.1.17.1 Πεδία

- `private String physicalURI="";`

Πρόκειται για το φυσικό URI (δηλαδή την ακριβή θέση στο σκληρό δίσκο) του .owl αρχείου το οποίο η παρούσα κλάση θα επεξεργαστεί.

- `private OWLOntologyManager man;`

Πρόκειται για το διαχειριστή της οντολογίας, ένα αντικείμενο η αρχικοποίηση του οποίου μέσω των κατασκευαστών που το OWL Api προσφέρει είναι απαραίτητη προκειμένου να γίνει η επεξεργασία της οντολογίας.

- `private OWLOntology ont;`

Πρόκειται για την οντολογία την οποία το παρόν αντικείμενο επεξεργάζεται. Είναι ουσιαστικά το .owl αρχείο που έχει εισάγει ο χρήστης.

6.1.17.2 Μέθοδοι

- `public MyOntologyManager(String physURI)`

Είναι ο κατασκευαστής της κλάσης. Δέχεται ως όρισμα το path του αρχείου το οποίο περιέχει την οντολογία προς επεξεργασία. Επιπλέον κατασκευάζει με τη βοήθεια των κατάλληλων μεθόδων του OWL Api το διαχειριστή της οντολογίας και τέλος την ίδια την οντολογία η οποία φορτώνεται από το physURI που έχει δοθεί ως όρισμα.

- `public ArrayList retrieveAtomicConcepts()`

Η παρούσα μέθοδος είναι υπεύθυνη για την «εξόρυξη» των ατομικών εννοιών από την οντολογία που έχει προσδιορισθεί. Αρχικά λαμβάνει όλες τις έννοιες (κλάσεις) της οντολογίας και στη συνέχεια ελέγχει εάν κάποια από αυτές είναι ορισμένη. Εάν αυτό ισχύει τότε δεν την συμπεριλαμβάνει στη λίστα των εννοιών που θα επιστραφούν επειδή πρόκειται

για σύνθετη και όχι γι'ατομική έννοια. Επιπλέον δεν περιλαμβάνει τις κλάσεις Thing και Nothing. Επιστρέφει ένα ArrayList στο οποίο έχουν συμπεριληφθεί όλες οι κατάλληλες κλάσεις.

- `public ArrayList retrieveAtomicRoles()`

Η παρούσα μέθοδος είναι υπεύθυνη για την «εξόρυξη» των ατομικών ρόλων από την οντολογία που έχει προσδιορισθεί. Αρχικά λαμβάνει όλους τους ρόλους (ιδιότητες) της οντολογίας και στη συνέχεια ελέγχει εάν κάποιος από αυτούς είναι ισοδύναμος με κάποιον άλλο ρόλο. Σε περίπτωση που κάτι τέτοιο ισχύει δεν πρόκειται για ένα ατομικό ρόλο επειδή έχει οριστεί με βάση κάποιους άλλους ρόλους. Σε αυτή την περίπτωση δεν συμπεριλαμβάνεται στη λίστα των ρόλων που θα επιστραφούν επειδή πρόκειται για σύνθετο και όχι γι'ατομικό ρόλο. Η διαδικασία πραγματοποιείται τόσο για τα data όσο και για τα object properties. Επιστρέφει ένα ArrayList στο οποίο έχουν συμπεριληφθεί όλες οι κατάλληλες ιδιότητες (data και object properties).

6.1.18 DBConnectionFrame.java

`public class` `DBConnectionFrame` **`extends`** `JFrame` **`implements`** `ActionListener,KeyListener`

Πρόκειται για την κλάση που υλοποιεί το interface μέσα από το οποίο ο χρήστης δίνει τα στοιχεία της βάσης δεδομένων με την οποία θα δουλέψει.

6.1.18.1 Πεδία

- `private JButton okButton,cancelButton;`

Είναι τα δύο κουμπιά που ο χρήστης χρησιμοποιεί για να επιβεβαιώσει το συνδυασμό username/password/database ή να εγκαταλείψει το interface χωρίς να συμβεί κάποια ενέργεια.

- `private JLabel label1,label2,label3;`

Οι τρεις αυτές ετικέτες είναι οι “Username:”, “Password:” και “Database:” του γραφικού περιβάλλοντος.

- `private JTextField textField1,textField2,textField3;`

Αντιστοιχούν στις 3 παραπάνω ετικέτες και προορίζονται προκειμένου ο χρήστης να συμπληρώσει τα κατάλληλα στοιχεία.

- `private DefaultListModel attributesListModel;`

Πρόκειται για μία αναφορά στα περιεχόμενα της λίστας των DB Attributes τα οποία εμφανίζονται στο χρήστη, στο κύριο interface.

- `private DBManager dbMan;`

Είναι ένα αντικείμενο της κλάσης DBManager το οποίο είναι υπεύθυνο για τη διαχείριση της ΒΔ.

- `private String username,password,database;`

Πρόκειται για τα 3 στοιχεία που δίνει ο χρήστης σχετικά με τη βάση δεδομένων με την οποία θα δουλέψει.

- `private FuzzyFrame frameFuzzy;`

Πρόκειται για μία αναφορά στην πατρική κλάση του συστήματος.

6.1.18.2 Μέθοδοι

- `public DBConnectionFrame(DefaultListModel existingAttributesListModel,FuzzyFrame ff)`

Είναι ο κατασκευαστής της κλάσης. Εφόσον πρόκειται για subclass της JFrame χρησιμοποιεί τον πατρικό constructor και θέτει τον τίτλο. Στη συνέχεια τοποθετεί το JFrame σε συγκεκριμένη θέση ως προς τα όρια της οθόνης. Θέτει το εικονίδιο το οποίο βρίσκεται σε σχετικό path του σκληρού. Κατόπιν θέτει τα πεδία του αντικειμένου χρησιμοποιώντας τα

existingAttributesListModel και ff τα οποία έχουν περαστεί σαν ορίσματα. Δημιουργεί 2 panel εκ των οποίων το 1 με ευδιάκριτα όρια. Δημιουργείται ένας GridBagLayoutManager για κάθε panel, προκειμένου να ρυθμίσει τη διάταξη των συστατικών που θα επικολληθούν σε αυτό. Το σχήμα 170 δείχνει τη θέση και τη χρησιμότητα του κάθε panel. Επιπλέον δείχνει που έχει τοποθετηθεί κάθε component και τις συντεταγμένες αυτού. Η διάταξη των συστατικών έχει γίνει με το γνωστό τρόπο.



Σχήμα 170. Γραφική διάταξη του "Specify MySQL DB Connection" frame.

- `public void actionPerformed(ActionEvent e)`

Ανιχνεύει ποιο component προκάλεσε το γεγονός και εκτελεί την αντίστοιχη ενέργεια καλώντας μία ξεχωριστή μέθοδο για κάθε component. Η υλοποίηση αυτής της μεθόδου είναι υποχρεωτική εφόσον η κλάση DBConnectionFrame υλοποιεί το interface ActionListener.

- `public void keyPressed(KeyEvent e)`

Θέλουμε το πάτημα του πλήκτρου enter να εκλαμβάνεται ως το πάτημα του κουμπιού "OK". Γι'αυτό και κάθε φορά που ο χρήστης πατά το enter εκτελείται η αντίστοιχη μέθοδος ενώ παράλληλα δίνεται το focus στο button "OK".

- `public void keyReleased(KeyEvent e)`
- `public void keyTyped(KeyEvent e)`

Η υλοποίηση των 2 παραπάνω μεθόδων είναι υποχρεωτική εφόσον η κλάση DBConnectionFrame υλοποιεί το interface KeyListener. Δεν υλοποιούν κάποια λειτουργία.

- `protected void manageOk()`

Η παρούσα μέθοδος επιχειρεί μια διασύνδεση με τη ΒΔ και ανάκτηση όλων των ιδιοχαρακτηριστικών της. Αρχικά λαμβάνονται τα username, password και database όπως αυτά έχουν προσδιοριστεί από το χρήστη και με βάση αυτά κατασκευάζεται ένα αντικείμενο της κλάσης DBManager. Στη συνέχεια καλείται η κατάλληλη μέθοδος του dbMan η οποία με τη σειρά της επιστρέφει μία ArrayList με τα DB Attributes. Τα DB Attributes προστίθενται ένα ένα στην attributesListModel και με τον τρόπο αυτό εμφανίζονται στο χρήστη. Επιπλέον επειδή το σύστημα πλέον διαθέτει ένα αντικείμενο τύπου DBManager παραχωρείται στο χρήστη η δυνατότητα να χρησιμοποιήσει κάποια γραφικά components τα οποία πριν ήταν απενεργοποιημένα, όπως τα fuzzy και crisp radio buttons του αριστερού frame ή οι επιλογές των μενού Export και View. Οι εξαιρέσεις διαχειρίζονται την περίπτωση κατά την οποία δεν μπορεί να εντοπιστεί το .jar αρχείο το οποίο περιλαμβάνει τις κλάσεις με τη βοήθεια των οποίων γίνεται η επικοινωνία με τη ΒΔ ή εάν εγερθεί κάποια sql εξαίρεση, στέλνοντας το κατάλληλο μήνυμα στο χρήστη. Τέλος κλείνει το frame.

- `protected void manageCancel()`

Κλείνει το frame χωρίς να γίνεται καμία διασύνδεση με τη ΒΔ.

6.1.19 DBManager.java

`public class DBManager`

Η κλάση αυτή είναι υπεύθυνη για ένα σύνολο από λειτουργίες που πραγματοποιούνται αναφερόμενες σε μια ΒΔ.

6.1.19.1 Πεδία

- `private String username;`

Το πεδίο αυτό σχετίζεται με το username της σύνδεσης sql που θα πραγματοποιηθεί.

- `private String password;`

Το πεδίο αυτό σχετίζεται με το password της σύνδεσης sql που θα πραγματοποιηθεί.

- `private String database;`

Το πεδίο αυτό σχετίζεται με το όνομα της βάσης δεδομένων της σύνδεσης sql που θα πραγματοποιηθεί.

- `protected Connection con=null;`

Το πεδίο αυτό θα περιέχει τη βασική sql σύνδεση που θα χρησιμοποιούν οι μέθοδοι.

6.1.19.2 Μέθοδοι

- `public ArrayList retrieveAttributes() throws Exception`

Η παρούσα μέθοδος επιστρέφει ένα ArrayList που περιέχει τα ιδιοχαρακτηριστικά της βάσης που ο χρήστης έχει προσδιορίσει. Αφού πραγματοποιήσει με επιτυχία τη σύνδεση λαμβάνει ένα αντικείμενο dbschema το οποίο ανήκει στην κλάση DatabaseMetaData. Πρόκειται για τα μεταδεδομένα της σύνδεσης con. Από το dbschema λαμβάνει ένα ResultSet, το columns το οποίο περιέχει πληροφορίες σχετικά με τη συγκεκριμένη ΒΔ. Τα πεδία της μεθόδου getColumns έχουν στα τρια τελευταία ορίσματα το String “%”, επειδή δε μας ενδιαφέρει κάποιο συγκεκριμένο pattern αλλά επιθυμούμε να ανακτήσουμε το σύνολο των πλειάδων. Κάθε πλειάδα του columns αφορά ένα DB Attribute της ΒΔ που έχει προσδιορίσει ο χρήστης. Έτσι στη συνέχεια λαμβάνοντας για κάθε μία από αυτές τις πλειάδες την τιμή της στήλης με το όνομα “COLUMN_NAME” και “TABLE_NAME” έχουμε ανακτήσει το αντίστοιχο DB Attribute. Όσον αφορά τη μορφή με την οποία εμφανίζονται στο χρήστη έχουμε ορίσει να

είναι: όνομα πίνακα + '.' + όνομα ιδιοχαρακτηριστικού, π.χ. Model.Height. Σε περίπτωση που εγερθεί κάποια εξαίρεση αυτή όχι μόνο πιάνεται αλλά και εγείρεται εκ νέου ώστε να πιαστεί από την κλάση που κάλεσε την παρούσα μέθοδο. Με τον τρόπο αυτό φτάνει εως το χρήστη με τη μορφή ενός μηνύματος, διαφορετικού για κάθε περίπτωση.

- `public boolean checkOneColumnPrimaryKey(String dbAttribute)`

Η παρούσα μέθοδος αποκρίνεται εάν το dbAttribute προέρχεται από πίνακα ο οποίος έχει πρωτεύον κλειδί μιας στήλης. Αφού πραγματοποιήσει τη σύνδεση λαμβάνει το όνομα του πίνακα και στη συνέχεια με βάση αυτό το όνομα του πίνακα λαμβάνει ένα αντικείμενο της κλάσης ResultSet το οποίο περιέχει όλα τα primary keys. Εάν αυτά είναι περισσότερα από 1 επιστρέφει false, αλλιώς true.

- `public ArrayList retrieveColumnKeys(String dbAttribute)`

Η μέθοδος αυτή επιστρέφει ένα ArrayList το οποίο περιέχει όλα τα DB Attributes τα οποία συγκροτούν το πρωτεύον κλειδί του πίνακα στον οποίο ανήκει το όρισμα dbAttribute. Αφού πραγματοποιήσει τη σύνδεση λαμβάνει το όνομα του πίνακα και στη συνέχεια με βάση αυτό το όνομα του πίνακα λαμβάνει ένα αντικείμενο rs της κλάσης ResultSet το οποίο περιέχει όλα τα primary keys. Διατρέχει το rs και για κάθε στήλη του πρωτεύοντος κλειδιού κάνει την αντίστοιχη προσθήκη στο ArrayList που θα επιστραφεί.

- `public String retrieveRelation(String dbAttribute)`

Για το dbAttribute, επιστρέφει τον πίνακα από τον οποίο προέρχεται.

- `public String retrieveAttribute(String dbAttribute)`

Για το dbAttribute, επιστρέφει το όνομα του ιδιοχαρακτηριστικού, δίχως το όνομα του πίνακα.

6.1.20 ABoxManager.java

```
public class AboxManager
```

Πρόκειται για μία κλάση η οποία είναι υπεύθυνη για τις λειτουργίες που αφορούν την παραγωγή του Abox, που αποτελεί και τη βασική λειτουργία του συστήματος.

6.1.20.1 Πεδία

- `private SemanticRulesManager srm;`

Πρόκειται για μία αναφορά στο διαχειριστή των SR του συστήματος.

- `private ArrayList semanticRules;`

Πρόκειται για το σύνολο των SR βάσει των οποίων θα γίνει η παραγωγή του σώματος των ισχυρισμών.

- `private DBManager dbm;`

Πρόκειται για μία αναφορά στο διαχειριστή της βάσης δεδομένων του συστήματος.

- `private OWLOntologyManager omanager;`

Πρόκειται για ένα αντικείμενο της κλάσης OWLOntologyManager με τη βοήθεια του οποίου γίνεται η αλληλεπίδραση με τις παραγόμενες οντολογίες.

6.1.20.2 Μέθοδοι

- `public ABoxManager(SemanticRulesManager semanticRulesManager, DBManager dbMan)`

Είναι ο κατασκευαστής της κλάσης. Θέτει χρησιμοποιώντας τα ορίσματα τους διαχειριστές SR και ΒΔ, ενώ ακόμη θέτει με τη βοήθεια του διαχειριστή κανόνων το πεδίο semanticRules.

- `public void setSemanticRulesFromManager()`

Η παρούσα μέθοδος θέτει το πεδίο semanticRules χρησιμοποιώντας όμως το άλλο πεδίο srm προκειμένου να λάβει αυτούς τους κανόνες.

- `public URI buildAssertionAxioms() throws Exception`

Η παρούσα μέθοδος αρχικά δημιουργεί ένα φυσικό URI στο οποίο θα αποθηκεύσει την οντολογία που θα δημιουργήσει. Κατόπιν δημιουργεί ένα URI για την οντολογία (σε http μορφή) και αντιστοιχίζει τα δύο παραπάνω URI το ένα με το άλλο. Στη συνέχεια δημιουργεί μία οντολογία με βάση το URI και λαμβάνει ένα εργοστάσιο παραγωγής OWLData από τον omanager. Μετά για κάθε SR καλεί τη μέθοδο που είναι υπεύθυνη για την παραγωγή των κατάλληλων ισχυρισμών από το συγκεκριμένο κανόνα. Τέλος σώζει την (τροποποιημένη πλέον) οντολογία σε OWLXMLOntologyFormat και στη θέση του δίσκου που υποδεικνύει το physicalURI. Επιπλέον επιστρέφει αυτό το physicalURI.

- `public void printAssertionAxioms(URI physicalURI,StyledDocument doc)`

Η παρούσα μέθοδος καλείται με δύο ορίσματα: το physicalURI και το doc. Σκοπό έχει για τα αξιώματα που περιέχει η οντολογία η οποία βρίσκεται στη θέση του physicalURI να τα μετατρέψει σε φιλική προς το χρήστη γραπτή μορφή χρησιμοποιώντας τα κατάλληλα χρώματα και ακολουθώντας το πρότυπο που ο racer έχει καθιερώσει. Αρχικά δημιουργεί τον κατάλληλο διαχειριστή οντολογίας και την οντολογία και κατόπιν δημιουργεί ένα αντικείμενο της κλάσης Style, το default. Κατόπιν δημιουργεί τρια διαφορετικά αντικείμενα της κλάσης Style, ένα για μπλε γράμματα, ένα για κόκκινα και ένα για μαύρα. Κατόπιν εξετάζει ένα ένα τα αξιώματα και διακρίνει τρεις περιπτώσεις:

Εάν πρόκειται για ισχυρισμό κλάσης λαμβάνονται όλα τα αξιώματα που σχετίζονται με το συγκεκριμένο αξίωμα και περιλαμβάνουν annotations. Για κάθε ένα από αυτά εισάγεται ένα σύνολο απο strings τα οποία περιλαμβάνουν το όνομα της κλάσης, το όνομα του ατόμου, το σύμβολο του ασαφούς ισχυρισμού καθώς επίσης και το βαθμό αυτού. Τα strings αυτά τυπώνονται κάθε ένα με το κατάλληλο χρώμα. Έτσι η λέξη instance (η οποία δηλώνει ότι πρόκειται για αξίωμα κλάσης) γράφεται με μπλε, το σύμβολο και ο βαθμός της ασάφειας με κόκκινο ενώ όλα τα υπόλοιπα με μαύρο. Εάν δεν υπάρχουν annotations τυπώνεται μόνο το άτομο και η κλάση στην οποία αυτό ανήκει.

Εάν πρόκειται για ισχυρισμό Object Property λαμβάνονται όλα τα αξιώματα που σχετίζονται με το συγκεκριμένο αξίωμα και περιλαμβάνουν annotations. Για κάθε ένα από αυτά εισάγεται ένα σύνολο απο strings τα οποία περιλαμβάνουν το όνομα της object property, τα ονόματα

των ατόμων που εμπλέκονται, το σύμβολο του ασαφούς ισχυρισμού καθώς επίσης και το βαθμό αυτού. Τα strings αυτά τυπώνονται κάθε ένα με το κατάλληλο χρώμα. Έτσι η λέξη related (η οποία δηλώνει ότι πρόκειται για αξίωμα ιδιότητας) γράφεται με μπλε, το σύμβολο και ο βαθμός της ασάφειας με κόκκινο ενώ όλα τα υπόλοιπα με μαύρο. Εάν δεν υπάρχουν annotations τυπώνονται μόνο τα άτομα και η ιδιότητα αντικειμένων που τα συνδέει.

Εάν τέλος πρόκειται για ισχυρισμό Data Property τυπώνεται το όνομα της data property, το όνομα του ατόμου που αφορά και η τιμή της data property. Τα strings αυτά τυπώνονται κάθε ένα με το κατάλληλο χρώμα. Έτσι η λέξη related (η οποία δηλώνει ότι πρόκειται για αξίωμα ιδιότητας) γράφεται με μπλε, ενώ όλα τα υπόλοιπα με μαύρο.

- ```
  public void applySemanticRuleToOntology(SemanticRule
  sr,OWLOntology ont,OWLDataFactory fac) throws Exception
  ```

Η παρούσα μέθοδος με βάση ένα SR προσθέτει ένα σύνολο από ασαφείς ισχυρισμούς στην οντολογία. Το σχήμα 171 δείχνει τις αποφάσεις που παίρνονται κατά την εκτέλεση της μεθόδου και τις διαφορετικές ομάδες εντολών που εκτελούνται με βάση αυτές. Είναι τρία τα δυνατά σενάρια:

- SR που αναφέρεται σε ατομική έννοια. Αρχικά ανακτώνται από τη βάση δεδομένων τα ζευγάρια πρωτεύον κλειδί – DB Attribute . Για κάθε ένα από αυτά τα ζευγάρια τώρα θα προστεθούν μια σειρά από αξιώματα. Δημιουργείται ένα OWL άτομο του οποίου το όνομα (τμήμα του URI) προέρχεται από το concatenation του ονόματος της στήλης του πρωτεύοντος κλειδιού και της αριθμητικής τιμής του πρωτεύοντος κλειδιού (π.χ. ModelID647). Επίσης, δημιουργείται μία OWL κλάση με όνομα (τμήμα του URI) το όνομα της ατομικής έννοιας. Με βάση αυτά τα δύο δημιουργείται ένα αντικείμενο axiom της κλάσης OWLClassAssertionAxiom. Στη συνέχεια υπολογίζεται η ασάφεια, λαμβάνοντας τη συνάρτηση από τον κανόνα και υπολογίζοντας το y δίνοντας ως x την τιμή της ΒΔ για το DB Attribute. Εάν η ασάφεια δεν είναι 0, προστίθενται συνολικά 3 αξιώματα. Το πρώτο είναι το axiom το οποίο προστίθεται με τη βοήθεια της μεθόδου applyChange του omanager. Το δεύτερο και τρίτο απαιτούν κάποια περαιτέρω δομικά στοιχεία. Χρησιμοποιώντας την ασάφεια που υπολογίστηκε από τη MF δημιουργείται ένα αντικείμενο constant της κλάσης OWLConstant και το “leq” ή το “geq” για το δεύτερο και τρίτο αξίωμα αντίστοιχα. Κατόπιν κατασκευάζεται ένα αντικείμενο fuzzyDegree της κλάσης OWLConstantAnnotation χρησιμοποιώντας το hasFuzzinessURI και το constant. Με τη χρήση του fuzzyDegree αλλά και του axiom κατασκευάζεται το αντικείμενο annAx της κλάσης OWLAxiomAnnotationAxiom. Στην
  ```

τελευταία κατασκευή το axiom είναι απαραίτητο προκειμένου να διευκρινίζεται σε ποιο αξίωμα αναφέρεται το συγκεκριμένο annotation.Τέλος τα δύο αξιώματα που μόλις κατασκευάστηκαν προστίθενται στην οντολογία με το γνωστό Visitor design pattern.

- SR που αναφέρεται σε ατομικό ρόλο αλλά για τη fuzzy περίπτωση. Αυτό σημαίνει ότι έχουμε να κάνουμε με Object Property. Αρχικά ανακτώνται από τη βάση δεδομένων οι τριπλέτες domain-range – DB Attribute . Για κάθε μία από αυτές τις τριπλέτες τώρα θα προστεθούν μια σειρά από αξιώματα. Δημιουργούνται δύο OWL άτομα των οποίων τα ονόματα (τμήματα του URI) προέρχονται από το concatenation του ονόματος της στήλης που αντιστοιχεί στο domain ή range και της αριθμητικής τιμής αυτής της στήλης (π.χ. AgencyID566, CustomerID89). Επίσης, δημιουργείται μία OWL Object Property με όνομα (τμήμα του URI) το όνομα του ρόλου. Με βάση αυτά τα δύο δημιουργείται ένα αντικείμενο axiom της κλάσης OWLObjectPropertyAssertionAxiom. Στη συνέχεια υπολογίζεται η ασάφεια, λαμβάνοντας τη συνάρτηση από τον κανόνα και υπολογίζοντας το y δίνοντας ως x την τιμή της ΒΔ για το DB Attribute. Εάν η ασάφεια δεν είναι 0, προστίθενται συνολικά 3 αξιώματα. Το πρώτο είναι το axiom το οποίο προστίθεται με τη βοήθεια της μεθόδου applyChange του omanager. Το δεύτερο και τρίτο απαιτούν κάποια περαιτέρω δομικά στοιχεία. Χρησιμοποιώντας την ασάφεια που υπολογίστηκε από τη MF δημιουργείται ένα αντικείμενο constant της κλάσης OWLConstant και το “leq” ή το “geq” για το δεύτερο και τρίτο αξίωμα αντίστοιχα. Κατόπιν κατασκευάζεται ένα αντικείμενο fuzzyDegree της κλάσης OWLConstantAnnotation χρησιμοποιώντας το hasFuzzinessURI και το constant. Με τη χρήση του fuzzyDegree αλλά και του axiom κατασκευάζεται το αντικείμενο annAx της κλάσης OWLAxiomAnnotationAxiom. Στην τελευταία κατασκευή το axiom είναι απαραίτητο προκειμένου να διευκρινίζεται σε ποιο αξίωμα αναφέρεται το συγκεκριμένο annotation.Τέλος τα δύο αξιώματα που μόλις κατασκευάστηκαν προστίθενται στην οντολογία με το γνωστό Visitor design pattern.
- SR που αναφέρεται σε ατομικό ρόλο αλλά για την crisp περίπτωση. Αυτό σημαίνει ότι έχουμε να κάνουμε με Data Property. Αρχικά ανακτώνται από τη βάση δεδομένων οι τριπλέτες domain-data value– DB Attribute . Για κάθε μία από αυτές τις τριπλέτες τώρα θα προστεθεί ένα αξίωμα. Δημιουργείτα ένα OWL άτομο του οποίου το όνομα (τμήμα του URI) προέρχεται από το concatenation του ονόματος της στήλης που αντιστοιχεί στο domain και της αριθμητικής τιμής αυτής της στήλης (π.χ. ModelID789). Επίσης, δημιουργείται μία OWL Data Property με όνομα (τμήμα του URI) το όνομα του ρόλου. Με βάση αυτά τα δύο δημιουργείται ένα αντικείμενο axiom της κλάσης OWLDataPropertyAssertionAxiom.



Σχήμα 171. Μέθοδος applySemanticRuleToOntology

Κατά την εκτέλεση της παρούσας μεθόδου είναι δυνατό να εγερθούν ένα σύνολο από διαφορετικές εξαιρέσεις οι οποίες πιάνονται και εγείρονται εκ νέου στην κλάση που καλεί αυτή τη μέθοδο.

- `public String retrieveDegreeFromAnnotation(String note)`

Η παρούσα μέθοδος απομονώνει και επιστρέφει από το note (το οποίο ουσιαστικά πρόκειται ένα αντικείμενο της κλάσης OWLConstant στο οποίο έχει εφαρμοσθεί η toString()) το βαθμό ασάφειας.

- `public String retrieveSymbolFromAnnotation(String note)`

Η παρούσα μέθοδος απομονώνει και επιστρέφει από το note (το οποίο ουσιαστικά πρόκειται ένα αντικείμενο της κλάσης OWLConstant στο οποίο έχει εφαρμοσθεί η toString()) το σύμβολο, δηλαδή επιστρέφει είτε "<=" είτε ">=".

- `public String retrieveValueFromObject(String value)`

Η μέθοδος αυτή χρησιμοποιείται κατά την «ανάγνωση» των αντικειμένων της κλάσης OWLDataPropertyAssertionAxiom και χρησιμεύει στο να εξάγεται από το object της data property η τιμή.

6.1.21 ABoxFrame.java

```
public class ABoxFrame extends JFrame implements ActionListener
```

Η παρούσα κλάση είναι υπεύθυνη για το frame το οποίο παρουσιάζει τους ισχυρισμούς του Abox σε εύληπτη και φιλική για το χρήστη μορφή.

6.1.21.1 Πεδία

- `private JButton button1;`

Είναι το "OK" button του frame.

- `private JLabel label1;`

Είναι το "Abox" label του frame.

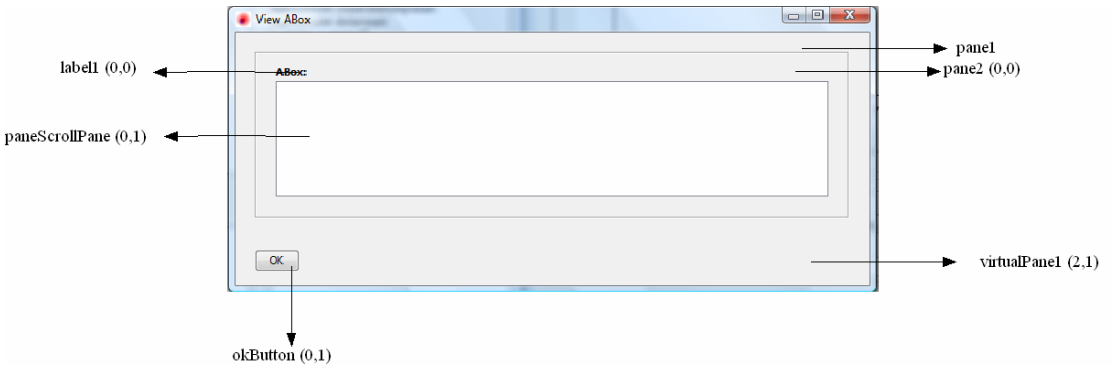
- `private StyledDocument doc;`

Πρόκειται για ένα αντικείμενο το οποίο περιλαμβάνει όλο το κείμενο που απεικονίζει το textPane. Επειδή ανήκει στην κλάση StyledDocument αυτό το κείμενο είναι δυνατόν να απεικονίζεται με διαφορετικό τρόπο κατά βούληση, όπως π.χ. στην περίπτωση μας με διαφορετικά χρώματα.

6.1.21.2 Μέθοδοι

- `public ABoxFrame()`

Είναι ο κατασκευαστής της κλάσης. Εφόσον πρόκειται για subclass της JFrame χρησιμοποιεί τον πατρικό constructor και θέτει τον τίτλο. Στη συνέχεια τοποθετεί το JFrame σε συγκεκριμένη θέση ως προς τα όρια της οθόνης. Θέτει το εικονίδιο το οποίο βρίσκεται σε σχετικό path του σκληρού. Δημιουργεί 2 panel εκ των οποίων το 1 με ευδιάκριτα όρια. Δημιουργείται ένας GridBagLayoutManager για κάθε panel, προκειμένου να ρυθμίσει τη διάταξη των συστατικών που θα επικολληθούν σε αυτό. Το σχήμα 172 δείχνει τη θέση και τη χρησιμότητα του κάθε panel. Επιπλέον δείχνει που έχει τοποθετηθεί κάθε component και τις συντεταγμένες αυτού. Η διάταξη των συστατικών έχει γίνει με το γνωστό τρόπο.



Σχήμα 172. Γραφική διάταξη του "View ABox" frame.

Από τον υπόλοιπο κώδικα αξίζει να σχολιάσουμε λίγο την προσθήκη του text pane. Αρχικά δημιουργούμε ένα αντικείμενο textPane της κλάσης JTextPane. Επιπλέον το θέτουμε μη επεξεργάσιμο, εφόσον δε θέλουμε να τροποποιείται από το χρήστη. Λαμβάνουμε από αυτό το αντικείμενο doc της κλάσης StyledDocument μέσω του οποίου θα «τυπωθεί» το ABox στο

“View ABox” frame. Τέλος το textPane εισάγετα στο paneScrollPane της κλάσης JScrollPane, έτσι ώστε ο χρήστης να έχει τη δυνατότητα να δει όλα τα περιεχόμενα του ABox και όχι μόνο οσα χωρά το frame.

- `public void actionPerformed(ActionEvent e)`

Κλείνει το frame όταν πατηθεί το κουμπί “OK”. Η υλοποίηση αυτής της μεθόδου είναι υποχρεωτική εφόσον η κλάση ABoxFrame υλοποιεί το interface ActionListener.

6.1.22 ViewSemanticRulesFrame.java

`public class ViewSemanticRulesFrame extends JFrame implements ActionListener`

Η παρούσα κλάση είναι υπεύθυνη για την προβολή των SR στο χρήστη και τη δυνατότητα αφαίρεσης κάποιων από αυτούς.

6.1.22.1 Πεδία

- `private JButton removeButton;`

Πρόκειται για το κουμπί που επιτρέπει στο χρήστη να αφαιρεί κανόνες.

- `private JLabel label1;`

Είναι η ετικέτα “Semantic Rules”.

- `private SemanticRulesManager srm;`

Μία αναφορά στο αντικείμενο του συστήματος που διαχειρίζεται τους SR.

- `private JTable table;`

Ο πίνακας που περιλαμβάνει όλους τους SR.

- `private MyTableModel tableModel;`

Είναι υπεύθυνο για τα περιεχόμενα του πίνακα καθώς επίσης και για την ορθή προβολή τους.

- `private JPanel pane1,pane2;`

Συμμετέχουν στο γραφικό περιβάλλον του frame.

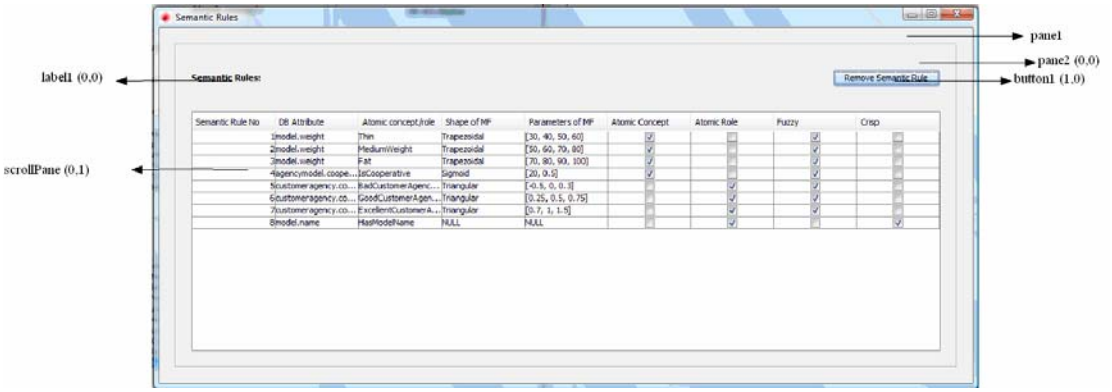
- `private int currentSize;`

Αντιπροσωπεύει το τρέχον πλήθος των SR.

6.1.22.2 Μέθοδοι

- `public ViewSemanticRulesFrame(SemanticRulesManager srm)`

Είναι ο κατασκευαστής της κλάσης. Εφόσον πρόκειται για subclass της JFrame χρησιμοποιεί τον πατρικό constructor και θέτει τον τίτλο. Στη συνέχεια τοποθετεί το JFrame σε συγκεκριμένη θέση ως προς τα όρια της οθόνης. Θέτει το εικονίδιο το οποίο βρίσκεται σε σχετικό path του σκληρού. Δημιουργεί 2 panel εκ των οποίων το 1 με ευδιάκριτα όρια. Δημιουργείται ένας GridBagLayoutManager για κάθε panel, προκειμένου να ρυθμίσει τη διάταξη των συστατικών που θα επικολληθούν σε αυτό. Το σχήμα 173 δείχνει τη θέση και τη χρησιμότητα του κάθε panel. Επιπλέον δείχνει που έχει τοποθετηθεί κάθε component και τις συντεταγμένες αυτού. Η διάταξη των συστατικών έχει γίνει με το γνωστό τρόπο.



Σχήμα 173. Γραφική διάταξη του "Semantic Rules" frame.

Στο σημείο αυτό αξίζει να σχολιάσουμε ορισμένες εντολές σχετικά με την προσθήκη του αντικειμένου της κλάσης JTable. Αφού ληφθούν οι σημασιολογικοί κανόνες και υπολογιστεί το πλήθος τους κατασκευάζεται με όρισμα αυτό το πλήθος ένα αντικείμενο tableModel της κλάσης MyTableModel. Πρόκειται για μία κλάση που εμείς έχουμε κατασκευάσει και περιγράφεται σε άλλο σημείο. Μετά τη μέθοδο updateTableData() κατασκευάζεται με όρισμα το tableModel ένα αντικείμενο table της κλάσης JTable. Για το table τίθενται ορισμένες ιδιότητες όπως το μέγεθος που επιθυμούμε να φαίνεται στο χρήστη και το ότι κάθε φορά που αλλάζει το πλάτος μίας στήλης η αλλαγή που συνεπάγεται για τις υπόλοιπες στήλες θα ισοκατανέμεται σε αυτές. Επίσης επιτρέπεται η επιλογή γραμμών και απαγορεύεται η επιλογή στηλών, ενώ τέλος επιτρέπεται κάθε φορά η επιλογή μίας μόνο γραμμής. Τέλος το table ενθυλακώνεται σε ένα scrollPane προκειμένου να μπορεί ο χρήστης να βλέπει το σύνολο των κανόνων εάν αυτοί είναι πολλοί.

- `public void actionPerformed(ActionEvent e)`

Ανιχνεύει ότι το component που προκάλεσε το γεγονός είναι το button1 και εκτελεί την αντίστοιχη ενέργεια καλώντας την αντίστοιχη μέθοδο. Η υλοποίηση αυτής της μεθόδου είναι υποχρεωτική εφόσον η κλάση ViewSemanticRulesFrame υλοποιεί το interface ActionListener.

- `protected void manageRemoveSemanticRule()`

Αρχικά ελέγχεται εάν ο χρήστης έχει επιλέξει κάποιον κανόνα προς απομάκρυνση και εάν όχι εμφανίζεται ανάλογο μήνυμα στο χρήστη. Κατά την αφαίρεση του κανόνα αρχικά αφαιρείται ο κανόνας από το σύνολο των κανόνων που ο semantic rules manager κρατά με την

αντίστοιχη μέθοδο και μειώνεται το μέγεθος των κανόνων κατά 1. Τίθεται το νέο μέγεθος στο tableModel και ενημερώνονται τα δεδομένα τόσο μέσω του tableModel, όσο και μέσω του table.

- `public void updateTableData()`

Η παρούσα μέθοδος είναι υπεύθυνη για την αρχικοποίηση των δεδομένων καθώς επίσης και για την ενημέρωση των δεδομένων κάθε φορά που αυτά αλλάζουν. Αρχικά ελέγχει εάν πρόκειται για ασαφή κανόνα και εάν ναι διακρίνει δύο περιπτώσεις: εάν εμπλέκεται έννοια ή ρόλος. Για κάθε μία από τις δύο αυτές περιπτώσεις κάνει το αντίστοιχο cast και στη συνέχεια θέτει τις αντίστοιχες τιμές στο tableModel. Κάθε μία από αυτές τις εντολές λαμβάνει την κατάλληλη τιμή με την κατάλληλη μέθοδο του SR και την τοποθετεί στη σωστή γραμμή και στήλη.

6.1.23 MyTableModel.java

```
public class MyTableModel extends AbstractTableModel
```

Η παρούσα κλάση είναι υπεύθυνη για τη διαχείριση των περιεχομένων του πίνακα ο οποίος απεικονίζει στο χρήστη τους SR, καθώς επίσης και για την προβολή τους με διαφορετικό τρόπο ανάλογα με τον τύπο των δεδομένων που απεικονίζονται. Είναι παιδί της κλάσης AbstractTableModel.

6.1.23.1 Πεδία

- `private int currentSize;`

Πρόκειται για το τρέχον πλήθος των SR.

- `private String[] columnNames = {"Semantic Rule No","DB Attribute","Atomic concept/role","Shape of MF","Parameters of MF","Atomic Concept","Atomic Role","Fuzzy","Crisp"};`

Πρόκειται για έναν πίνακα ο οποίος περιλαμβάνει τα ονόματα των στηλών του πίνακα.

- `private Object[][] data;`

Πρόκειται για τον διδιάστατο πίνακα ο οποίος σε κάθε θέση που προσδιορίζεται από το δείκτη της γραμμής και της στήλης περιέχει το δεδομένο εκείνο που θα απεικονιστεί στην ίδια γραμμή και στήλη του αντικειμένου κλάσης JTable.

6.1.23.2 Μέθοδοι

- `public MyTableModel(int cs)`

Πρόκειται για τον κατασκευαστή της κλάσης. Εκτός από την κλήση της μεθόδου κατασκευής της πατρικής κλάσης θέτει και το τρέχον πλήθος των SR.

- `public int getCurrentSize()`

Επιστρέφει το τρέχον πλήθος των SR.

- `public void setCurrentSize(int cs)`

Η παρούσα μέθοδος θέτει το τρέχον πλήθος των SR και παράλληλα αρχικοποιεί το αντικείμενο data με ένα διδιάστατο πίνακα του οποίου η μία διάσταση ισούται με το πλήθος των κανόνων.

- `public int getColumnCount()`

Επιστρέφει το πλήθος των στηλών του πίνακα.

- `public int getRowCount()`

Επιστρέφει το πλήθος των γραμμών του πίνακα που ουσιαστικά πρόκειται για το πλήθος των κανόνων.

- `public String getColumnName(int col)`

Επιστρέφει το όνομα της στήλης που βρίσκεται στη θέση col.

- `public Object getValueAt(int row, int col)`

Επιστρέφει την τιμή του πίνακα που βρίσκεται στη γραμμή row και στη στήλη col.

- `public Class getColumnClass(int c)`

Επιστρέφει την κλάση του δεδομένου του πίνακα που βρίσκεται στη γραμμή 0 και στη στήλη c.

- `public boolean isCellEditable(int row, int col)`

Υλοποιώντας αυτή τη μέθοδο ώστε να επιστρέφει false επιτυγχάνουμε να καθιστούμε όλα τα κελιά του πίνακα μη επεξεργάσιμα.

- `public void setValueAt(Object value, int row, int col)`

Η μέθοδος αυτή θέτει μία νέα τιμή στον πίνακα στη προσδιοριζόμενη θέση και παράλληλα καλεί τη μέθοδο η οποία εξασφαλίζει ότι θα αλλάξουν και τα απεικονιζόμενα στοιχεία του πίνακα.

- `public void updateData()`

Κάθε φορά που καλείται γίνεται ενημέρωση των δεδομένων που το tableModel περιέχει.

6.1.24 ClarifyConcept.java

```
public class ClarifyConcept extends JFrame implements ActionListener
```

Η παρούσα κλάση παρέχει αντικείμενα τα οποία διαχειρίζονται μία ειδική περίπτωση: εάν ο χρήστης επιχειρήσει τη δημιουργία ενός SR ο οποίος εμπλέκει ένα DB Attribute το οποίο με

τη σειρά του προέρχεται από ένα πίνακα που έχει πρωτεύον κλειδί που αποτελείται από περισσότερες από μία στήλες θα δημιουργήθεί πρόβλημα κατά τη δημιουργία του ABox. Το πρόβλημα αυτό προέρχεται από το γεγονός ότι δε γνωρίζουμε σε ποια από τις στήλες του πρωτεύοντος κλειδιού αναφέρεται το άτομο που θα εμφανιστεί στο ABox. Ωστόσο αυτό επιλύεται ρωτώντας το χρήστη σε ποιο ακριβώς ιδιοχαρακτηριστικό της ΒΔ αναφέρεται το άτομο που θα παραχθεί κατά τη δημιουργία του ισχυρισμού. Αυτή η δυνατότητα διασαφήνισης εκ μέρους του χρήστη είναι ο σκοπός αυτής της κλάσης.

6.1.24.1 Πεδία

- `private JButton button1,button2;`

Αντιστοιχούν στα δύο κουμπιά “Create Semantic Rule” και “Cancel” αντίστοιχα.

- `private JLabel label1,label2;`

Αντιστοιχεί στις δύο ετικέτες που συνθέτουν την οδήγία προς το χρήστη (βλ. και σχήμα 174).

- `private DefaultListModel keysListModel;`

Αντιστοιχεί στα περιεχόμενα της λίστας που εμφανίζεται στο χρήστη.

- `private SemanticRulesManager srm;`

Είναι μια αναφορά στο διαχειριστή SR του συστήματος.

- `private SemanticRuleConcept src;`

Είναι μια αναφορά στο SR έννοιας για τον οποίο γίνεται η διασαφήνιση.

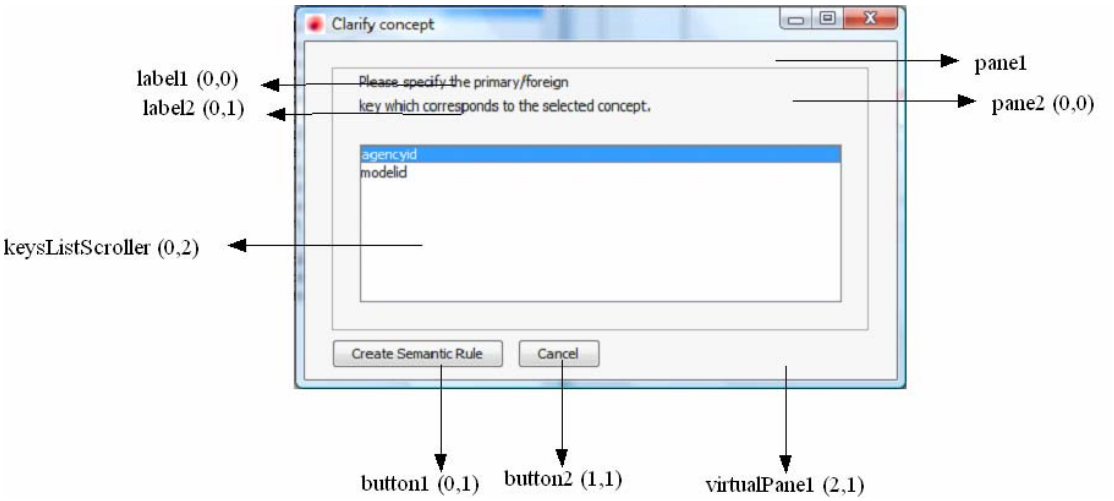
- `private JList keysList;`

Αντιστοιχεί στη λίστα που εμφανίζεται στο χρήστη.

6.1.24.2 Μέθοδοι

- `public ClarifyConcept(SemanticRulesManager srman, SemanticRuleConcept srconcept)`

Είναι ο κατασκευαστής της κλάσης. Εφόσον πρόκειται για subclass της JFrame χρησιμοποιεί τον πατρικό constructor και θέτει τον τίτλο. Στη συνέχεια τοποθετεί το JFrame σε συγκεκριμένη θέση ως προς τα όρια της οθόνης. Θέτει το εικονίδιο το οποίο βρίσκεται σε σχετικό path του σκληρού. Δημιουργεί 2 panel εκ των οποίων το 1 με ευδιάκριτα όρια. Δημιουργείται ένας GridBagLayoutManager για κάθε panel, προκειμένου να ρυθμίσει τη διάταξη των συστατικών που θα επικολληθούν σε αυτό. Το σχήμα 174 δείχνει τη θέση και τη χρησιμότητα του κάθε panel. Επιπλέον δείχνει που έχει τοποθετηθεί κάθε component και τις συντεταγμένες αυτού. Η διάταξη των συστατικών έχει γίνει με το γνωστό τρόπο.



Σχήμα 174. Γραφική διάταξη του "Clarify Concept" frame.

- `public void actionPerformed(ActionEvent e)`

Ανιχνεύει ότι το component που προκάλεσε το γεγονός είναι το button1 και εκτελεί την αντίστοιχη ενέργεια καλώντας την αντίστοιχη μέθοδο. Σε περίπτωση που πρόκειται για το button2 απλά κλείνει το παράθυρο χωρίς να κάνει κάποια ενέργεια. Η υλοποίηση αυτής της μεθόδου είναι υποχρεωτική εφόσον η κλάση ClarifyConcept υλοποιεί το interface ActionListener.

- `protected void manageButton()`

Η μέθοδος αυτη αρχικά ελέγχει εάν είναι επιλεγμένο κάποιο key από τη λίστα. Εάν όχι εμφανίζει το αντίστοιχο μήνυμα στο χρήστη. Εάν ναι θέτει το κατάλληλο πεδίο του src, προσθέτει το SR και δείχνει στο χρήστη ένα μήνυμα που επιβεβαιώνει την επιτυχημένη δημιουργία του κανόνα.

6.1.25 ClarifyRole.java

```
public class ClarifyRole extends JFrame implements ActionListener
```

Η παρούσα κλάση παρέχει αντικείμενα τα οποία διαχειρίζονται την περίπτωση κατά την οποία ο χρήστης επιχειρήσει τη δημιουργία ενός SR ο οποίος εμπλέκει ατομικό ρόλο, αλλά με επιλεγμένο το fuzzy radio button. Τότε ο χρήστης θα πρέπει να διευκρινίσει ποια από τις στήλες του πίνακα από τον οποίο προέρχεται το DB Attribute αναφέρεται στο domain του ρόλου και ποια στο range. Αυτή η δυνατότητα διασαφήνισης εκ μέρους του χρήστη είναι ο σκοπός αυτής της κλάσης.

6.1.25.1 Πεδία

- `private JButton button1,button2;`

Αντιστοιχούν στα δύο κουμπιά “Create Semantic Rule” και “Cancel” αντίστοιχα.

- `private JLabel label1,label2,label3;`

Αντιστοιχεί στην ετικέτα της οδηγίας προς το χρήστη, τη “Domain:” και τη “Range:”.

- `private DefaultListModel domainListModel,rangeListModel;`

Αντιστοιχεί στα περιεχόμενα της λίστας που εμφανίζεται στο χρήστη τόσο για το domain όσο και για το range του ρόλου.

- `private SemanticRulesManager srm;`

Είναι μια αναφορά στο διαχειριστή SR του συστήματος.

- `private SemanticRuleConcept srr;`

Είναι μια αναφορά στο SR ρόλου για τον οποίο γίνεται η διασαφήνιση.

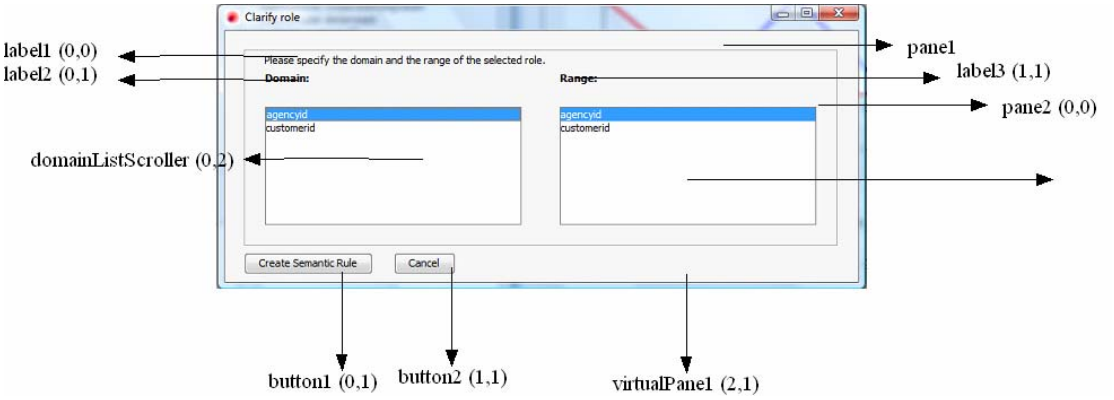
- `private JList domainList,rangeList;`

Αντιστοιχεί στις λίστες που εμφανίζονται στο χρήστη τόσο για το domain όσο και για το range του ρόλου.

6.1.25.2 Μέθοδοι

- `public ClarifyRole(SemanticRulesManager srman, SemanticRuleRole srrole)`

Είναι ο κατασκευαστής της κλάσης. Εφόσον πρόκειται για subclass της JFrame χρησιμοποιεί τον πατρικό constructor και θέτει τον τίτλο. Στη συνέχεια τοποθετεί το JFrame σε συγκεκριμένη θέση ως προς τα όρια της οθόνης. Θέτει το εικονίδιο το οποίο βρίσκεται σε σχετικό path του σκληρού. Δημιουργεί 2 panel εκ των οποίων το 1 με ευδιάκριτα όρια. Δημιουργείται ένας GridBagLayoutManager για κάθε panel, προκειμένου να ρυθμίσει τη διάταξη των συστατικών που θα επικολληθούν σε αυτό. Το σχήμα 175 δείχνει τη θέση και τη χρησιμότητα του κάθε panel. Επιπλέον δείχνει που έχει τοποθετηθεί κάθε component και τις συντεταγμένες αυτού. Η διάταξη των συστατικών έχει γίνει με το γνωστό τρόπο.



Σχήμα 175. Γραφική διάταξη του "Clarify Role" frame.

- `public void actionPerformed(ActionEvent e)`

Ανιχνεύει ότι το component που προκάλεσε το γεγονός είναι το button1 και εκτελεί την αντίστοιχη ενέργεια καλώντας την αντίστοιχη μέθοδο. Σε περίπτωση που πρόκειται για το button2 απλά κλείνει το παράθυρο χωρίς να κάνει κάποια ενέργεια. Η υλοποίηση αυτής της μεθόδου είναι υποχρεωτική εφόσον η κλάση ClarifyConcept υλοποιεί το interface ActionListener.

- `protected void manageButton()`

Η μέθοδος αυτη αρχικά ελέγχει εάν είναι επιλεγμένο κάποιο domain και κάποιο range από τις δύο αντίστοιχες λίστες. Εάν όχι εμφανίζει το αντίστοιχο μήνυμα στο χρήστη. Επιπλέον τσεκάρει ότι δεν έχει επιλεγεί το ίδιο DB Attribute τόσο για domain όσο και για range. Αυτό οφείλεται είτε στο ότι ο χρήστης έχει όντως επιλέξει το ίδιο domain και range είτε στο ότι υπάρχει μόνο ένα DB Attribute για να επιλέξει ο χρήστης τόσο στη domain list όσο και στη range list. Σε αυτή την περίπτωση εμφανίζεται το κατάλληλο επεξηγηματικό μήνυμα στο χρήστη. Εάν τέλος ο χρήστης έχει κάνει σωστά τις επιλογές του, προστίθεται ο SR και εμφανίζεται στο χρήστη ένα μήνυμα που επιβεβαιώνει την επιτυχημένη δημιουργία του κανόνα.

6.1.26 ClarifyCrispRole.java

public class ClarifyCrispRole **extends** JFrame **implements** ActionListener

Η παρούσα κλάση παρέχει αντικείμενα τα οποία διαχειρίζονται μία ειδική περίπτωση: εάν ο χρήστης επιχειρήσει τη δημιουργία ενός crisp SR ο οποίος εμπλέκει ένα DB Attribute το οποίο με τη σειρά του προέρχεται από ένα πίνακα που έχει πρωτεύον κλειδί που αποτελείται από περισσότερες από μία στήλες θα δημιουργήθεί πρόβλημα κατά τη δημιουργία του ABox. Το πρόβλημα αυτό προέρχεται από το γεγονός ότι δε γνωρίζουμε σε ποια από τις στήλες του πρωτεύοντος κλειδιού αναφέρεται το άτομο που θα εμφανιστεί στο ABox. Ωστόσο αυτό επιλύεται ρωτώντας το χρήστη σε ποιο ακριβώς ιδιοχαρακτηριστικό της ΒΔ αναφέρεται το άτομο που θα παραχθεί κατά τη δημιουργία του ισχυρισμού. Αυτή η δυνατότητα διασαφήνισης εκ μέρους του χρήστη είναι ο σκοπός αυτής της κλάσης.

6.1.26.1 Πεδία

- **private** JButton button1,button2;

Αντιστοιχούν στα δύο κουμπιά “Create Semantic Rule” και “Cancel” αντίστοιχα.

- **private** JLabel label1,label2;

Αντιστοιχεί στις δύο ετικέτες που συνθέτουν την οδήγία προς το χρήστη (βλ. και σχήμα 176).

- **private** DefaultListModel keysListModel;

Αντιστοιχεί στα περιεχόμενα της λίστας που εμφανίζεται στο χρήστη.

- **private** SemanticRulesManager srm;

Είναι μια αναφορά στο διαχειριστή SR του συστήματος.

- **private** SemanticRuleRole srr;

Είναι μια αναφορά στο SR ρόλου για τον οποίο γίνεται η διασαφήνιση.

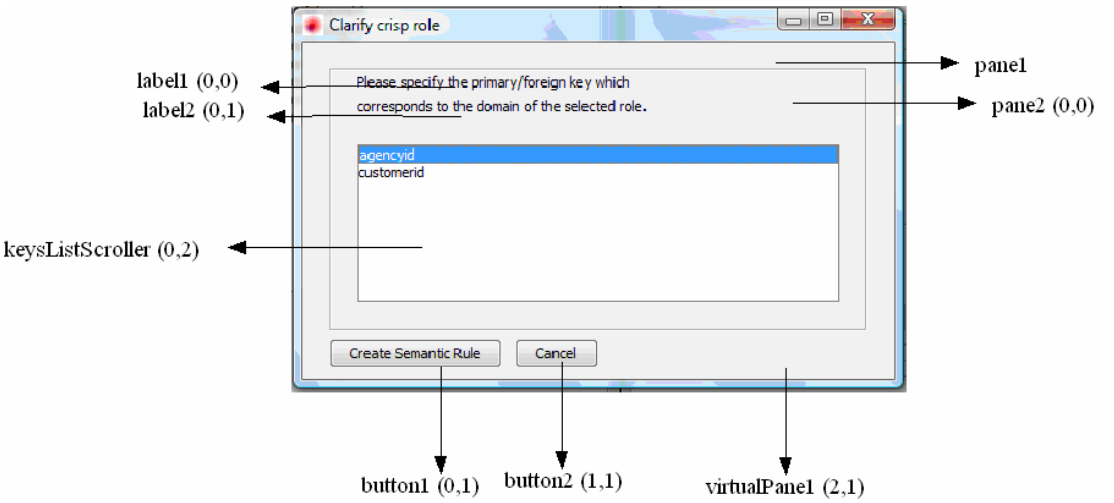
- `private JList keysList;`

Αντιστοιχεί στη λίστα που εμφανίζεται στο χρήστη.

6.1.26.2 Μέθοδοι

- `public ClarifyCrispRole(SemanticRulesManager srman, SemanticRuleRole srrole)`

Είναι ο κατασκευαστής της κλάσης. Εφόσον πρόκειται για subclass της JFrame χρησιμοποιεί τον πατρικό constructor και θέτει τον τίτλο. Στη συνέχεια τοποθετεί το JFrame σε συγκεκριμένη θέση ως προς τα όρια της οθόνης. Θέτει το εικονίδιο το οποίο βρίσκεται σε σχετικό path του σκληρού. Δημιουργεί 2 panel εκ των οποίων το 1 με ευδιάκριτα όρια. Δημιουργείται ένας GridBagLayoutManager για κάθε panel, προκειμένου να ρυθμίσει τη διάταξη των συστατικών που θα επικολληθούν σε αυτό. Το σχήμα 176 δείχνει τη θέση και τη χρησιμότητα του κάθε panel. Επιπλέον δείχνει που έχει τοποθετηθεί κάθε component και τις συντεταγμένες αυτού. Η διάταξη των συστατικών έχει γίνει με το γνωστό τρόπο.



Σχήμα 176. Γραφική διάταξη του "Clarify crisp role" frame.

- `public void actionPerformed(ActionEvent e)`

Ανιχνεύει ότι το component που προκάλεσε το γεγονός είναι το button1 και εκτελεί την αντίστοιχη ενέργεια καλώντας την αντίστοιχη μέθοδο. Σε περίπτωση που πρόκειται για το button2 απλά κλείνει το παράθυρο χωρίς να κάνει κάποια ενέργεια. Η υλοποίηση αυτής της μεθόδου είναι υποχρεωτική εφόσον η κλάση ClarifyConcept υλοποιεί το interface ActionListener.

- `protected void manageButton()`

Η μέθοδος αυτη αρχικά ελέγχει εάν είναι επιλεγμένο κάποιο key από τη λίστα. Εάν όχι εμφανίζει το αντίστοιχο μήνυμα στο χρήστη. Εάν ναι θέτει το κατάλληλο πεδίο του src, προσθέτει το SR και δείχνει στο χρήστη ένα μήνυμα που επιβεβαιώνει την επιτυχημένη δημιουργία του κανόνα.

6.1.27 MyBounds.java

```
public class MyBounds
```

Η παρούσα κλάση είναι μία δομή δεδομένων που αντιπροσωπεύει ένα ζεύγος από όρια, το κάτω όριο και το πάνω όριο.

6.1.27.1 Πεδία

- `private double lowerBound,upperBound;`

Πρόκειται για τα άνω και κάτω όρια.

6.1.28 OwlFilter.java

```
public class OwlFilter extends FileFilter
```

Η παρούσα κλάση χρησιμοποιείται από τον file chooser κατά τη φόρτωση του .owl αρχείου και σκοπό έχει να απεικονίζονται από αυτόν μόνο τα .owl αρχεία και τα directories. Κληρονομεί από τη FileFilter.

6.1.28.1 Πεδία

Κανένα πεδίο.

6.1.28.2 Μέθοδοι

- `public boolean accept(File f)`

Η μέθοδος αυτή λαμβάνει ως όρισμα ένα αρχείο και εάν έχει κατάληξη .owl ή είναι directory αποκρίνεται true αλλιώς false.

- `public String getDescription()`

Η μέθοδος αυτή είναι υπεύθυνη για το τι γράφεται στη droplist με την ετικέτα “Files of type:” του file chooser.

6.1.29 UpdateRangeFrame.java

```
public class UpdateRangeFrame extends JFrame implements ActionListener,KeyListener
```

Η παρούσα κλάση παρέχει αντικείμενα τα οποία κατασκευάζονται κάθε φορά που το κουμπί “Update range” πατιέται. Είναι υπεύθυνη για την αλλαγή των ορίων προβολής των γραφικών παραστάσεων των MF.

6.1.29.1 Πεδία

- `private JButton okButton,cancelButton;`

Αντιστοιχούν στα δύο κουμπιά “OK” και “Cancel” αντίστοιχα.

- `private JLabel label1,label2,label3;`

Αντιστοιχούν στις ετικέτες “Give the new:” , “Lower bound” και “Upper bound”.

- `private JTextField textField1,textField2;`

Σε αυτά τα πεδία κειμένου ο χρήστης εισάγει το κάτω και το άνω όριο αντίστοιχα.

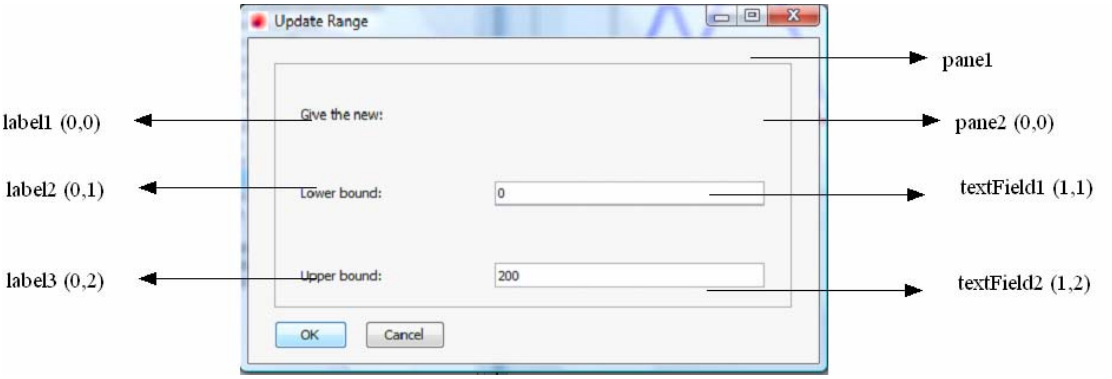
- `private RightFrame frameRight;`

Είναι μια αναφορά στο δεξί frame του συστήματος.

6.1.29.2 Μέθοδοι

- `public UpdateRangeFrame(RightFrame fr)`

Είναι ο κατασκευαστής της κλάσης. Εφόσον πρόκειται για subclass της JFrame χρησιμοποιεί τον πατρικό constructor και θέτει τον τίτλο. Στη συνέχεια τοποθετεί το JFrame σε συγκεκριμένη θέση ως προς τα όρια της οθόνης. Θέτει το εικονίδιο το οποίο βρίσκεται σε σχετικό path του σκληρού. Δημιουργεί 2 panel εκ των οποίων το 1 με ευδιάκριτα όρια. Δημιουργείται ένας GridBagLayoutManager για κάθε panel, προκειμένου να ρυθμίσει τη διάταξη των συστατικών που θα επικολληθούν σε αυτό. Το σχήμα 177 δείχνει τη θέση και τη χρησιμότητα του κάθε panel. Επιπλέον δείχνει που έχει τοποθετηθεί κάθε component και τις συντεταγμένες αυτού. Η διάταξη των συστατικών έχει γίνει με το γνωστό τρόπο.



Σχήμα 177. Γραφική διάταξη του "Update range" frame.

- `public void actionPerformed(ActionEvent e)`

Ανιχνεύει ποιο component προκάλεσε το γεγονός και εκτελεί την αντίστοιχη ενέργεια καλώντας μία ξεχωριστή μέθοδο για κάθε component. Η υλοποίηση αυτής της μεθόδου είναι υποχρεωτική εφόσον η κλάση UpdateRangeFrame υλοποιεί το interface ActionListener.

- `public void keyPressed(KeyEvent e)`

Θέλουμε το πάτημα του πλήκτρου enter να εκλαμβάνεται ως το πάτημα του κουμπιού "OK". Γι'αυτό και κάθε φορά που ο χρήστης πατά το enter εκτελείται η αντίστοιχη μέθοδος ενώ παράλληλα δίνεται το focus στο button "OK".

- `public void keyReleased(KeyEvent e)`
- `public void keyTyped(KeyEvent e)`

Η υλοποίηση των 2 παραπάνω μεθόδων είναι υποχρεωτική εφόσον η κλάση UpdateRangeFrame υλοποιεί το interface KeyListener. Δεν υλοποιούν κάποια λειτουργία.

- `protected void manageOk()`

Πρόκειται για τη μέθοδο που εκτελείται με το πάτημα του κουμπιού "OK". Εκτελεί ένα σύνολο ελέγχων για να βεβαιωθεί ότι τα νέα όρια είναι κατάλληλα (ότι είναι αριθμοί, ότι είναι το κάτω όριο μικρότερο από το πάνω όριο, ότι έχουν δοθεί και τα δύο νέα όρια). Εάν κάποιος έλεγχος αποτύχει εμφανίζεται το αντίστοιχο μήνυμα στο χρήστη. Εάν είναι

επιτυχημένοι όλοι οι έλεγχοι εκτελείται η μέθοδος setBothBounds του δεξιού frame με ορίσματα τα όρια που έχει εισάγει ο χρήστης στα πεδία κειμένου.

- `protected void manageCancel()`

Κλείνει το παράθυρο χωρίς να επιτελεί κάποια αλλαγή ορίων.

6.1.30 Utils.java

```
public class Utils
```

Η κλάση αυτή δεν είναι προσανατολισμένη σε κάποιο κοινό σκοπό όπως οι υπόλοιπες 29 του συστήματος. Σε αυτή έχουμε συμπεριλάβει κάποιες μεθόδους τις οποίες χρησιμοποιούμε σε περισσότερες από μία κλάσεις και θεωρήσαμε χρήσιμο να τις μετατρέψουμε σε static και να τις εντάξουμε σε μία κλάση από όπου και θα καλούνται, παρά να τις γράφουμε σε κάθε κλάση που τις χρειάζεται ξεχωριστά.

6.1.30.1 Πεδία

Κανένα πεδίο.

6.1.30.2 Μέθοδοι

- `public static String getExtension(File f)`

Η μέθοδος αυτή δέχεται ως όρισμα ένα αρχείο και αφού επεξεργαστεί το όνομα του επιστρέφει την κατάληξη αυτού.

- `public static double[] getInitialX(int density,MyBounds b)`

Η μέθοδος αυτή δέχεται ως ορίσματα ένα ζεύγος ορίων και την επιθυμητή πυκνότητα και επιστρέφει έναν πίνακα από αριθμούς double, οι οποίοι κατανέμονται σε ισομεγέθη διαστήματα μεταξύ των ορίων.

- `public static void buildConstraints(GridBagConstraints gbc, int gx, int gy,int gw, int gh, int wx, int wy)`

Η μέθοδος αυτή χρησιμοποιείται από όλες τις κλάσεις που χρησιμοποιούν αντικείμενα της κλάσης GridBagLayout και διευκολύνει την ανάθεση τιμών στα πεδία αντικειμένων της κλάσης GridBagConstraints.

- `public static String[] token(String str)`

Η μέθοδος αυτή δέχεται ως όρισμα ένα String str και επιστρέφει έναν πίνακα από strings τα οποία έχουν προέλθει από τον επιμερισμό του str σε υποσυμβολοσειρές οι οποίες διαχωρίζονται με κενό στο αρχικό str.

- `public static void plotCross(Graphics2D g2d,int x,int y)`

Η μέθοδος αυτή για το σημείο (x,y) και για το g2d σχεδιάζει ένα διαγώνιο σταυρό στο σημείο αυτό.

7 Επίλογος

Ακολουθεί μία σύνοψη της διπλωματικής εργασίας καθώς επίσης και κάποιες κατευθύνσεις για επέκταση του συστήματος που παρουσιάστηκε.

7.1 Σύνοψη

Η παρούσα διπλωματική εργασία ασχολήθηκε με τη δυνατότητα διασύνδεσης βάσεων δεδομένων και οντολογικής γνώσης με σκοπό την παραγωγή ασαφών υποθέσεων, ύστερα από αλληλεπίδραση με το χρήστη. Ιδιαίτερη έμφαση δόθηκε στην ανάπτυξη ενός συστήματος λογισμικού που επιτυγχάνει αυτή τη διασύνδεση.

Αρχικά παρουσιάσαμε το θεωρητικό υπόβαθρο που καλύπτει τις βασικές έννοιες οι οποίες εμπλέκονται στο σύστημα όπως σχεσιακές βάσεις δεδομένων, οντολογίες, γλώσσες περιγραφικής λογικής, η γλώσσα OWL, η fuzzy επέκταση της καθώς επίσης και κομμάτια της ασαφούς συνολοθεωρίας, όπως ασαφή σύνολα και ασαφείς αριθμοί.

Στη συνέχεια προτείναμε με τυπικό τρόπο τη μεθοδολογία που ακολουθήθηκε κατά την ανάπτυξη του συστήματος.

Κατόπιν δώσαμε την περιγραφή της αρχιτεκτονικής του συστήματος, περιγράφοντας τη δομή των επιμέρους υποσυστημάτων και τη λειτουργία τους.

Ύστερα προσθέσαμε την περιγραφή της διαδικασίας ελέγχου του συστήματος που αποτελεί ταυτόχρονα και το εγχειρίδιο χρήσης του συστήματος καθώς καλύπτει το σύνολο των δυνατοτήτων του συστήματος και δίνει παραστατικά μέσω των σχημάτων τον τρόπο χρήσης του.

Το τελευταίο κεφάλαιο περιλαμβάνει μια ενδελεχή περιγραφή του λογισμικού του συστήματος, παραθέτοντας για κάθε κλάση τις μεταβλητές τις και τις μέθοδους τις, συνοδευόμενες από σχόλια..

7.2 Μελλοντικές επεκτάσεις

Πιθανές μελλοντικές επεκτάσεις του παρόντος συστήματος είναι:

1. Προσθήκη περαιτέρω MF στο σύστημα έτσι ώστε να παρέχεται πιο πλούσια εκφραστικότητα στο χρήστη. Αναφέρονται κάποιες τέτοιες συναρτήσεις:

- Η *διπλή γκαουσιανή συνάρτηση.* Η συνάρτηση αυτή είναι μια ασύμμετρη γκαουσιανή συνάρτηση της οποίας το αριστερό και δεξιό σκέλος προέρχονται από δύο διαφορετικές γκαουσιανές συναρτήσεις (με διαφορετικές παραμέτρους). Εάν για τις δύο παραμέτρους c_1 και c_2 ισχύει $c_1 < c_2$ τότε ένα τμήμα της συνάρτηση θα λαμβάνει την τιμή1, αλλιώς η συνάρτηση μόνο θα προσεγγίζει την τιμή 1. Η συνάρτηση αυτή προσφέρει τα πλεονεκτήματα των γκαουσιανών συναρτήσεων δίνοντας ταυτόχρονα τη δυνατότητα για μια ασύμμετρη προσέγγιση.
- Η *γενικευμένη συνάρτηση σχήματος καμπάνας.* Υπακούει στον εξής τύπο:

$$f(x) = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2b}}$$

  Η παράμετρος c αντιστοιχεί στο κέντρο της καμπύλης.

- Η συνάρτηση που αντιστοιχεί σε *διαφορά σιγμοειδών συναρτήσεων.*Η καμπύλη ορίζεται ως η διαφορά των δύο διαφορετικών σιγμοειδών συναρτήσεων. Με τον τρόπο αυτό εξαλείφεται η ασυμπτωτική συμπεριφορά των σιγμοειδών καμπυλών στο 1.
- Η συνάρτηση που αντιστοιχεί σε *γινόμενο σιγμοειδών συναρτήσεων.*Πρόκειται για αντίστοιχη περίπτωση με την από πάνω.
- Η συνάρτηση *σχήματος-Z* . Ονομάζεται έτσι εξαιτίας του σχήματος της καμπύλης ενώ πρόκειται για μία sp-γραμμή με δύο ακριανά σημεία που αποτελούν και τις παραμέτρους της.
- Η συνάρτηση *σχήματος-Π* . Ονομάζεται έτσι εξαιτίας του σχήματος της καμπύλης ενώ πρόκειται για μία sp-γραμμή οριζόμενη από 4 σημεία που αποτελούν και τις παραμέτρους της. Τα δύο από τα 4 σημεία αντιστοιχούν στα «πόδια» της συνάρτησης ενώ τα άλλα δύο στους «ώμους» της.
- Η συνάρτηση *σχήματος-S* . Ονομάζεται έτσι εξαιτίας του σχήματος της καμπύλης ενώ πρόκειται για μία sp-γραμμή με δύο ακριανά σημεία που αποτελούν και τις παραμέτρους της.

2. Κωδικοποίηση αρχείων με ειδική κατάληξη τα οποία ο χρήστης θα είναι σε θέσει να σώσει και να ανοίξει μέσω επιλογών File→Save και File→Open αντίστοιχα. Κάθε τέτοιο αρχείο θα πρέπει να περιέχει ένα TBox, μία διασύνδεση σε μία ΒΔ και το σύνολο των SR που έχουν δημιουργηθεί με τις MF τους.
3. Μία ακόμη επέκταση θα μπορούσε να αφορά τη δυνατότητα εφαρμογής των SR σε ένα TBox το οποίο θα περιείχε επίσης σύνθετες έννοιες και ρόλους, με αποτέλεσμα να απαιτείται συλλογιστική για την παραγωγή των ασαφών υποθέσεων.

8 *Βιβλιογραφία*

[1] http://www.w3.org/2000/Talks/1206-xml2k-tbl/Overview.html

[2] http://www.w3.org/2007/OWL/wiki/Syntax#Ontologies

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

[4] http://www.w3.org/2007/OWL/wiki/Primer

[5] http://en.wikipedia.org/wiki/Relational_database

[6] http://en.wikipedia.org/wiki/Symbol_grounding

[7] http://en.wikipedia.org/wiki/Ontology_%28computer_science%29

[8] http://en.wikipedia.org/wiki/Gang_of_Four_%28software%29

[9] http://www.ifomis.org/bfo/, a formal upper ontology designed to support scientific research

[10] http://obi.sourceforge.net/, an open access, integrated ontology for the description of biological and clinical investigations

[11] http://sig.biostr.washington.edu/projects/fm/AboutFM.html, for human anatomy

[12] http://cidoc.ics.forth.gr/, an ontology for "cultural heritage information"

[13] Fuzzy Logic Toolbox, Matlab, User's Guide.

[14] Franz Baader, Ian Horrocks, Ulrike Sattler, " Description Logics," Handbook of Knowledge Representation, Elsevier, 2007 .

[15] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P.F. (2002) The Description Logic Handbook: Theory, Implementation and Applications. Cambridge Univeristy Press.

[16] Baader, F., Horrocks, I., Sattler, U. (2002) Description Logics for the Semantic Web. KI – Kunstliche Intelligenz, Vol. 16, No. 4, pp. 57-59.

[17] Baader, F., (1991) Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles. Proceedings of the 12th International Joint Conference on Artificial Intelligence.

[18] Sean Bechhofer, Raphael Volz, Phillip Lord, " Cooking the Semantic Web with the OWL API".

[19] Bechhofer, S., van Harmelen F., Hendler, J., Horrocks, I., McGuinness., D.L., Patel-Schneider P.F., Stein, A.L., eds. (2004) OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/.

[20] Tim Berners-Lee, James Hendler, Ora Lassila, " The Semantic Web," Scientific American Magazine, May 17, 2001.

[21] Ronald J. Brachman, Hector J. Levesque, Knowledge representation and reasoning, 1st ed. , Morgan Kaufmann, May 19, 2004.

[22] Bremerman, H. J. [1962], "Optimization through evolution and recombination." Yovits, M. C , G.T. Jacobi and G. D. Goldstein, eds., Self-Organizing Systems. Spartan Books, Washington, pp. 93-106.

[23] Brickey, D., Guha, R.V. (2000) RDF Vocabulary Description Language 1.0: RDF Schema, W3C. http://www.w3.org/TR/2000/CR-rdf-schema-20000327.

[24] Rogers Cadenhead, Laura Lemay, Πλήρες εγχειρίδιο της java 2, 3η έκδ. , Μόσχος Γκιούρδας.

[25] Codd, E.F. (1970). "A Relational Model of Data for Large Shared Data Banks". Communications of the ACM 13 (6): 377–387.

[26] H. M. Deitel, P. J. Deitel, S. E. Santry, Advanced Java 2 platform : how to program, 2002.

[27] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, November 10, 1994.

[28] Harnad, S. (1990) The Symbol Grounding Problem. *Physica D* 42: 335-346.

[29] Mathew Horridge, Sean Bechhofer, Olaf Noppens, " Igniting the OWL 1.1 Touch Paper: The OWL API".

[30] Horrocks, I., Patel-Schneider, P.: Reducing owl entailment to description logic satisfiability. Web Semantics (2004) 345–357

[31] George J. Klir, Bo Yuan, Fuzzy Sets and Fuzzy Logic Theory and Applications, Prentice Hall PTR, 1995.

[32] Lassila, O., Swick, R. (1999) W3C Resource Description Framework model and syntax specification. http://www.w3.org/TR/RECrdf-syntax/

[33] Mendelson, E. (1987) Introduction to Mathematical Logic, 3rd ed.

[34] Peter F. Patel-Schneider, Ian Horrocks, and Boris Motik, OWL 1.1 Web Ontology Language: Structural Specification and Functional-Style Syntax.

[35] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. Technical report, W3C, Feb. 2004. W3C Recommendation, URL http://www.w3.org/TR/2004/REC-owl-semantics-20040210/.

[36] Silberschatz, Korth, Sudarshan, Database System Concepts, 4th ed. , 2002.

[37] Nick Simou, Giorgos Stoilos, Kostas Pardalis, Vassilis Tzouvaras, Giorgos Stamou, Stefanos Kollias, " Storing and Querying Fuzzy Knowledge in the Semantic Web".

[38] Giorgos Stoilos, Nikos Simou, Giorgos Stamou, Stefanos Kollias, " Uncertainty and the Semantic Web".

[39] Giorgos Stoilos, Giorgos Stamou, Vassilis Tzouvaras, Jeff Z. Pan, Ian Horrocks, " Fuzzy OWL: Uncertainty and the Semantic Web".

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

[41] G. Stoilos et al., “The Fuzzy Description Logic f-SHIN,” Proc. Int’l Workshop Uncertainty Reasoning for the Semantic Web, 2005, CEUR Electronic Workshop Proc., pp. 67–76; www.image.ntua.gr/papers/385.pdf.

[42] U. Straccia. Towards a fuzzy description logic for the semantic web. In Proceedings of the 2nd European Semantic Web Conference, 2005.

[43] U. Straccia, “Reasoning within Fuzzy Description Logics,” J. Artificial Intelligence Research, vol. 14, Apr. 2001, pp. 137–166.

[44] Weaver, W. ,1948, "Science and compiexity."American Scientist, 36(4), pp. 536-544.

[45] Russel Winder and Graham Roberts, Developing Java software.

[46] Turing, A.M. (1950) Computing Machinery and Intelligence. Mind 49 433-460

[47] Zadeh, L. A. [1965a], "Fuzzy sets and systems." In: Fox, J., ed., System Theory. Polytechnic Press, Brooklyn, NY, pp. 29-37.

[48] Zadeh, L. A. [1965b], "Fuzzy sets." Information and Control, 8(3), pp. 338-353.

[49] Εισαγωγή στις περιγραφικές λογικές. Σύνταξη, Σημασιολογία και Αλγόριθμοι Συλλογιστικής. Γ.Στοϊλος, Σημειώσεις, 2007.

[50] Γλώσσες Αναπαράστασης Γνώσης στο Σημασιολογικό Ιστό, Γ. Στοΐλος, Σημειώσεις, 2007.

[51] Ιωάννης Βλαχάβας, Πέτρος Κεφαλας, Νικόλαος Βασιλειάδης, Φώτης Κόκκορας, Ηλίας Σακελλαρίου, Τεχνητή νοημοσύνη, 3η έκδ. , Β. Γκιούρδας Εκδοτική, Φεβρουάριος 2006.