ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αναπαράσταση Γνώσης και Εφαρμογή Ασαφούς Συλλογιστικής στο Ιατρικό Πλαίσιο

Δημήτριος Ν. Κωτσάκος

Επιβλέποντες: Ευστάθιος Χατζηευθυμιάδης, Επίκουρος Καθηγητής ΕΚΠΑ
Χρήστος Αναγνωστόπουλος, Υποψήφιος Διδάκτωρ ΕΚΠΑ

ΑΘΗΝΑ

ΔΕΚΕΜΒΡΙΟΣ 2007

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αναπαράσταση Γνώσης και Εφαρμογή Ασαφούς Συλλογιστικής στο Ιατρικό Πλαίσιο

Δημήτριος Ν. Κωτσάκος

Α.Μ.: 1115 2002 00041

ΕΠΙΒΛΕΠΟΝΤΕΣ:
Ευστάθιος Χατζηευθυμιάδης, Επίκουρος Καθηγητής ΕΚΠΑ
Χρήστος Αναγνωστόπουλος, Υποψήφιος Διδάκτωρ ΕΚΠΑ

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία πραγματεύεται το πρόβλημα της μοντελοποίησης ιατρικής γνώσης χρησιμοποιώντας δύο διαφορετικούς τρόπους αναπαράστασης. Εξετάζεται η προσέγγιση μέσω ασαφών μεταβλητών και των αντίστοιχων ασαφών συνόλων καθώς και η οντολογική προσέγγιση, ανάλογα με τη φύση της αναπαριστώμενης πληροφορίας.

Στα πλαίσια της εργασίας αναπτύχθηκε μια ενδεικτική ιατρική οντολογία, ή οποία μέσω ενός μηχανισμού συμπερασμού κατατάσσει έναν ασθενή σε κάποια από τις ορισμένες σε αυτήν κλάσεις. Η κατάταξη γίνεται βάσει του ιατρικού ιστορικού του ασθενούς και των καθημερινών συνηθειών του, αφού σύμφωνα με την ιατρική γνώση παίζουν καθοριστικό ρόλο στην κατάταξη ενός ασθενούς σε κλάση χαμηλού ή υψηλού κινδύνου.

Επίσης χρησιμοποιήθηκαν ασαφή σύνολα που αντιστοιχούν σε δείκτες του ανθρώπινου οργανισμού, καθώς και οι ασαφείς κανόνες στους οποίους οι δείκτες αυτοί εμπλέκονται. Οι τιμές των δεικτών αυτών λαμβάνονται μέσω αισθητήρων τοποθετημένων στο περιβάλλον του ασθενούς, δίνοντας κάθε στιγμή εικόνα για την τρέχουσα κατάσταση της υγείας του.

Για ασθενείς τους οποίους η οντολογική αναπαράσταση κατατάσσει σε ομάδες υψηλού κινδύνου, αν οι δείκτες που παρακολουθούν οι αισθητήρες κυμανθούν σε μη φυσιολογικά επίπεδα, το σύστημα είτε λαμβάνει τα κατάλληλα μέτρα, αν υπάρχει η δυνατότητα αυτή, είτε ειδοποιεί το ειδικό προσωπικό.

Συμπερασματικά, αποδεικνύεται μέσω της εφαρμογής στο ιατρικό πλαίσιο, πως ο συνδυασμός λογικής περιγραφών και ασαφούς λογικής στην αναπαράσταση γνώσης, μπορεί να δώσει ενδιαφέροντα αποτελέσματα και πιο ισχυρές εφαρμογές αξιοποίησης των μηχανισμών συμπερασμού που υποστηρίζουν οι δύο λογικές.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ιατρικές εφαρμογές διάχυτου υπολογισμού

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: επίγνωση πλαισίου, πληροφορία πλαισίου, ασαφής λογική, ασαφής συλλογιστική

ABSTRACT

The current BSc thesis deals with the problem of modelling medical knowledge using two different ways of representation. The two approaches that are examined are the one using fuzzy variables and the corresponding fuzzy sets as well as the ontological one. The choice about which one to follow each time depends on the nature of information to be represented.

In the context of work an indicative medical ontology was developed. It's use is to classify a patient instance in a predefined class in the ontology using a strong reasoning mechanism. The classification is based on the medical background of the patient and his daily habits, because according to the medical knowledge the latter play a decisive role in the classification of a patient in a high- or low-danger class.

We also used fuzzy sets that correspond to indicators of human organism, as well as the fuzzy rules in which these indicators are involved. The values of these indicators are taken from sensors placed in the environment of the patient, giving every moment a clear view of his current health situation.

For patients that were classified by the ontological representation in high danger classes, if the indicators that are watched by the sensors do not oscillate in natural levels the system takes the suitable measures. If such an ability is unavailable the system notifies the special personnel.

In conclusion, via this application in the medical context it is proved, that the combination of description logics and fuzzy logic in knowledge representation can produce interesting results and provide much more powerful applications that make use of the reasoning mechanisms that the two kinds of logic support.

SUBJECT AREA: Medical applications of pervasive computing

KEYWORDS: context awareness, fuzzy logic, fuzzy reasoning

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον κύριο Χατζηευθυμιάδη, που καθημερινά μου δείχνει πόσο ποιοτική σχέση μπορούν να έχουν οι φοιτητές με τους καθηγητές τους. Χωρίς τη συμβολή και τη στήριξή του δε θα ήταν δυνατή η ολοκλήρωση της εργασίας αυτής.
Θα ήθελα επίσης να ευχαριστήσω το Χρήστο Αναγνωστόπουλο, για την τεράστια υπομονή που έδειξε όλο αυτό το διάστημα της δουλειάς μας.
Το Μανόλη, τη Χρύσα, τον Αλέξη, τον Παύλο και το Γιώργο.
Τέλος, θέλω να ευχαριστήσω τους γονείς μου, που όλα αυτά τα χρόνια προσπαθούν για το καλύτερο για μένα. Ήταν σε κάθε μου προσπάθεια κοντά μου για να με στηρίζουν, να με συμβουλεύουν, να με αγαπούν. Σας αγαπώ πολύ.

Δημήτρης

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο γίνεται αρχική αναφορά στις βασικές έννοιες που θα μελετηθούν στη συνέχεια, παρουσιάζοντας αρχικά το σκοπό της εργασίας αυτής. Αρχικά, θα παρουσιαστεί το κύριο πρόβλημα που ώθησε στην ερεύνα και την αναζήτηση των προτάσεων που περιγράφονται ακολούθως καθώς και μία εισαγωγή στο διάχυτο υπολογισμό. Στο τέλος της ενότητας αυτής, θα γίνει αναφορά στα κεφάλαια της εργασίας, περιγράφοντας με λίγα λόγια τα κύρια σημεία τους.

Ο «διάχυτος» ή «πανταχού παρών υπολογισμός» αποτελεί μια επέκταση του χώρου της επικοινωνίας ανθρώπου-μηχανής, όπου η επεξεργασία της διαθέσιμης πληροφορίας ενσωματώνεται σε καθημερινά αντικείμενα και δραστηριότητες των οντοτήτων του περιβάλλοντος. Σύμφωνα με τον Mark Weiser [12], ο οποίος θεωρείται ο πρωτεργάτης του διάχυτου υπολογισμού:

«*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it. ».*

Σχεδόν μια εικοσαετία μετά, η πρόοδος του υλικού και του λογισμικού κατέστησαν καθημερινά τα στοιχεία που το 1991 έκαναν το διάχυτο υπολογισμό ανέφικτο. Οι υπολογιστές δε βρίσκονται πλέον μόνο στους χώρους εργασίας με τη μορφή επιτραπέζιων μηχανών, αλλά στην καθημερινή ζωή με τη μορφή ασύρματων δικτύων, φορητών υπολογιστών, αισθητήρων, ελεγκτών και μικροϋπολογιστών. Η πρόοδος αυτή στην τεχνολογία υλικού κατέστησε εφικτές ερευνητικές και όχι μόνο προσπάθειες σε πανεπιστήμια και βιομηχανία, με σκοπό να αναπτυχθεί ο χώρος που αρχικά ο Weiser ονόμασε «ubiquitous» και αργότερα μετονομάστηκε σε «pervasive» computing.

Η πρόκληση για το διάχυτο υπολογισμό είναι να μπορέσει δημιουργήσει έξυπνα περιβάλλοντα, με υπολογιστικές και επικοινωνιακές δυνατότητες, όσο γίνεται ενσωματωμένα στην καθημερινή ζωή και όσο γίνεται λιγότερο παρεμβατικά και ορατά.

Ο διάχυτος υπολογισμός τα τελευταία χρόνια βρίσκει εφαρμογή στο πλαίσιο της βελτίωσης των υπηρεσιών υγείας. Συνήθως τέτοιες προσπάθειες περιλαμβάνουν μηχανήματα παρακολούθησης δεικτών του ανθρώπινου οργανισμού όπως αυτόματα πιεσόμετρα, θερμόμετρα, όργανα μέτρησης παλμών κ.ά. Τέτοια μηχανήματα πλέον

μπορούν να επικοινωνήσουν με μεγαλύτερα υπολογιστικά συστήματα και να ανταλλάξουν χρήσιμες για τον ασθενή πληροφορίες. Μέσω αυτής της πολυδιάστατης συλλογής δεδομένων για τον ασθενή, οι υπηρεσίες υγείας αποχτούν τη δυνατότητα να αναλύσουν περισσότερο τους καθημερινούς παράγοντες που επηρεάζουν την υγεία του ασθενούς και ακολούθως να παραγάγουν εξατομικευμένες θεραπείες, με καλύτερα προσδοκώμενα αποτελέσματα.

Ένα άλλο πλεονέκτημα της εισόδου του διάχυτου υπολογισμού στα συστήματα υγείας είναι η άμεση αντίδραση που μπορεί να παρέχει σε περιπτώσεις έκτακτης ανάγκης. Γίνεται εφικτή η παρακολούθηση του ασθενή από απόσταση, καθώς και η άμεση προσπέλαση του ιατρικού ιστορικού του, όταν κάτι τέτοιο απαιτηθεί. Παρέχοντας στους ειδικούς της υγείας άμεση και πλήρη πληροφορία σχετικά με την εκάστοτε περίπτωση, αναμένεται οι διαγνώσεις να λαμβάνουν περισσότερο υπόψιν τους το ιστορικού του ασθενούς από όσο γινόταν παλιότερα.

Ο χειρουργικός τομέας επίσης κερδίζει πολλά από τη νέα προσέγγιση, αφού ως τώρα οι χειρουργοι και οι βοηθοί τους έπρεπε να παρακολουθούν και να ελέγχουν πολλές ζωτικές πληροφορίες κάτω από καταστάσεις άγχους και πίεσης. Τα συστήματα που αναπτύσσονται συλλέγουν και επεξεργάζονται τις πληροφορίες αυτές, επιδιώκοντας να εντοπίσουν πεδία ενδιαφέροντος και να ειδοποιήσουν τους ειδικούς αν αυτό κριθεί απαραίτητο.

Για να γίνουν τέτοιες προσεγγίσεις υλοποιήσιμες και εφικτές, είναι απαραίτητο να χρησιμοποιηθεί η έννοια της πληροφορίας πλαισίου, και να αξιοποιηθεί κατάλληλα. Τίθενται λοιπόν κάποια ζητήματα αναπαράστασης της πληροφορίας αυτής, καθώς και ανάπτυξης μηχανισμών συμπερασμού που θα οδηγήσουν στη λήψη αποφάσεων.

Στο κεφάλαιο 2 δίνονται ορισμοί σχετικοί με την πληροφορία πλαισίου και τα στοιχεία που την αποτελούν. Επίσης δίνονται κάποια παραδείματα εκμετάλλευσης της πληροφορίας αυτής και πιθανές εφαρμογές.

Στο κεφάλαιο 3 παρουσιάζονται δύο διαφορετικοί τρόποι αναπαράστασης της πληροφορίας πλαισίου και δίνονται οι σχετικοί ορισμοί. Για κάθέναν από τους δύο τρόπους αναπαράστασης παρουσιάζεται ο τρόπος με τον οποίο εντάχθηκε στο σύστημα που αναπτύχθηκε.

Στο κεφάλαιο 4 προτείνεται ένας διαφορετικός τρόπος προσέγγισης της αναπαράστασης της πληροφορίας πλαισίου, με χρήση αλγορίθμων συσταδοποίησης μεγάλων συνόλων εισόδου. Από τα σύνολα αυτά και τις συστάδες που προκύπτουν

μπορούν να εξαχθούν χρήσιμα συμπεράσματα σχετικά με την κατανομή των σημείων εισόδου, και έτσι να παραχθούν αυτόματα οι ασαφείς κανόνες που αναφέρονται στο κεφάλαιο 3.

Στο κεφάλαιο 5 γίνεται σύντομη παρουσίαση και αναφορά στο λογισμικό που χρησιμοποιήθηκε για την ανάπτυξη του συστήματος. Παρατίθενται επίσης σημαντικά τμήματα του κώδικα, μαζί με επεξηγήσεις για καθένα από αυτά.

Στο κεφάλαιο 6 παρουσιάζονται τα συμπεράσματα που προέκυψαν από τη μελέτη και την ανάπτυξη του συστήματος, καθώς και σκέψεις για μελλοντικές εφαρμογές αντίστοιχων ιδεών.

Τέλος, στο κεφάλαιο 7 παρατίθενται οι αναφορές που χρησιμοποιήθηκαν για τη συγγραφή της εργασίας.

ΚΕΦΑΛΑΙΟ 2

ΠΛΗΡΟΦΟΡΙΑ ΠΛΑΙΣΙΟΥ

Ένας από τους στόχους του Διάχυτου Υπολογισμού (Pervasive Computing) είναι να εκμεταλλευτεί την τεχνολογία των κατανεμημένων και κινητών τερματικών, ασύρματων δικτύων αισθητήρων (αισθητήρες διαφόρων τύπων όπως κίνησης, θερμοκρασίας, υγρασίας κ.ά.), για την ανάπτυξη εφαρμογών βασισμένων στην πληροφορία πλαισίου.

Σε πρώτο επίπεδο πρέπει να καθοριστεί η έννοια της πληροφορίας πλαισίου (context) και πώς μπορεί (1) να ανακτηθεί, (2) να επεξεργασθεί και να (3) χρησιμοποιηθεί, έτσι ώστε να βελτιστοποιείται κάθε φορά η επιλογή των παραμέτρων της που θα λαμβάνονται υπ' όψιν. Για παράδειγμα άλλες παράμετροι της πληροφορίας πλαισίου (context) χρησιμοποιούνται σε μια εφαρμογή γραφείου (π.χ. κανονισμένα meetings, incoming emails, προγραμματισμένες εργασίες κ.ά.) και άλλες σε μια εφαρμογή ιατρικής παρακολούθησης ενός ασθενή στο νοσοκομείο ή στο σπίτι (θερμοκρασία δωματίου, θερμοκρασία σώματος, αρτηριακή πίεση κ.ά.).

2.1. Ορισμός πληροφορίας πλαισίου

Πληροφορία πλαισίου *(context)* είναι οποιαδήποτε πληροφορία μπορεί να προσθέσει ένα στοιχείο στην περιγραφή της τρέχουσας κατάστασης ή / και του ιστορικού μιας οντότητας σε ένα περιβάλλον διάχυτου υπολογισμού. Παραδείγματα τέτοιας πληροφορίας είναι το αποτέλεσμα της ενέργειας μιας οντότητας, το περιβάλλον στο οποίο βρίσκεται και ενεργεί η οντότητα αυτή, οι μελλοντικές ενέργειες της οντότητας, το ιστορικό ενεργειών της οντότητας καθώς και οι τρέχουσες δραστηριότητες στις οποίες πιθανόν να εμπλέκεται η οντότητα.

2.2. Ορισμός οντότητας

Ως οντότητα *(contextual entity)* μπορούν να αναφερθούν τα εξής:

- ένας χρήστης (φυσικό πρόσωπο – user)

- μια ομάδα χρηστών
- είτε μια εφαρμογή / υπηρεσία (π.χ. Location Based Service)
- ένα υπολογιστικό αντικείμενο (π.χ. (Personal Digital Assistant, Shared Printer)
- μια πιο αφηρημένη έννοια (π.χ. Move).

Ένας πιο τυπικός ορισμός σχετικά με το «τι είναι πληροφορία πλαισίου» δεν είναι εύκολο να δοθεί, αφού κάτι ειδικότερο αυτού που προηγήθηκε ενδέχεται να περιοριστεί σε συγκεκριμένες εφαρμογές του διάχυτου υπολογισμού. Στα πλαίσια της πτυχιακής εργασίας θα χρησιμοποιηθεί αυτός ο ορισμός, ο οποίος θα συγκεκριμενοποιηθεί κατά τη διαδικασία μοντελοποίησης της πληροφορίας πλαισίου για τη συγκεκριμένη εφαρμογή.

2.3. Μοντέλα πληροφορίας πλαισίου

Για την περιγραφή / αναπαράσταση της πληροφορίας πλαισίου χρησιμοποιείται η έννοια των μοντέλων πληροφορίας πλαισίου (context models) τα οποία περιγράφουν τις κατάλληλες οντότητες / παραμέτρους στις οποίες πρέπει να εστιάσει η εκάστοτε εφαρμογή για την ανάκτηση της πληροφορίας πλαισίου (contextual information) που της είναι απαραίτητη, καθώς και τις συσχετίσεις / εξαρτήσεις που ορίζονται μεταξύ των οντοτήτων οι οποίες απασχολούν την εφαρμογή. Οι συσχετίσεις ερμηνεύουν τη δραστηριότητα των οντοτήτων και οι εξαρτήσεις «περιορίζουν» την πληροφορία που δύναται να διαχειριστεί μια context-aware εφαρμογή.

2.3.1. Εφαρμογή: Μοντέλο πληροφορίας πλαισίου φοιτητή

Για την αναπαράσταση της πληροφορίας πλαισίου που αφορά ένα φοιτητή μπορεί να χρησημοποιηθεί το ακόλουθο μοντέλο:

- $C = \{ c_1, c_2, \ldots c_n\}$ το σύνολο των μαθημάτων που παρακολουθεί,
- $E = \{ e_1, e_2, \ldots e_n\}$ οι ημερομηνίες που εξετάζονται τα μαθήματα αυτά,
- $P = \{ p_1, p_2, \ldots p_n\}$ οι ημερομηνίες παράδοσης των Projects που πιθανόν έχουν τα μαθήματά του,
- D = (d,m,y) η τρέχουσα ημερομηνία,
- L η τοποθεσία του

- A = { a_1, a_2,… a_m} το σύνολο των συναντήσεων που πιθανόν έχει κανονίσει με καθηγητές του

Έτσι κάθε στιγμή η πληροφορία πλαισίου του φοιτητή αποτελείται από κάποια στιγμιότυπα των παραπάνω παραμέτρων. Μια εφαρμογή βασισμένη στην πληροφορία πλαισίου γνωρίζει το μοντέλο που επιλέχθηκε και μπορεί να εκμεταλλευτεί τα στιγμιότυπα αυτά προκειμένου να προβεί στις κατάλληλες κάθε φορά ενέργειες, ή να ειδοποιήσει τον φοιτητή ώστε αυτός να κάνει κάποιες ενέργειες ανάλογα με την κατάσταση στην οποία βρίσκεται.

1. Θα μπορούσε να πρόκειται για μια εφαρμογή επίγνωσης θέσης που τρέχει στον εξυπηρετητή του πανεπιστημίου, η οποία αν η θέση του φοιτητή είναι κοντά στα εργαστήρια του τμήματος και η ημερομηνία είναι κοντά σε κάποια προγραμματισμένη συνάντηση, να συνδέεται με τον υπολογιστή του φοιτητή και να «κατεβάζει» το απαραίτητο υλικό για τη συνάντηση αυτή (αναλυτική βαθμολογία, βιογραφικό κτλ).

2. Όταν ο φοιτητής φεύγει από το σπίτι το Personal Digital Assistant (PDA) του συνδέεται στο υπολογιστή του σπιτιού και αντιγράφει τα τελευταία τροποποιημένα αρχεία σχετικά με τα projects του φοιτητή. Όταν φθάσει στο πανεπιστήμιο το PDA συνδέεται στο λογαριασμό του φοιτητή και μεταφέρει εκεί τα αρχεία, χωρίς παρέμβαση από το φοιτητή.

3. Κάποιες μέρες πριν την εξέταση ενός μαθήματος, η context-aware εφαρμογή του PDA μπορεί να συνδέεται με τα PDA όλων των άλλων φοιτητών που έχουν το ίδιο μάθημα όταν διαπιστωθεί ότι βρίσκονται σε κοντινή απόσταση, και να ανταλλάζουν όλο το υλικό που σχετίζεται με το συγκεκριμένο μάθημα (σημειώσεις, διαφάνειες κτλ) καθώς και να συνδέεται στο Διαδίκτυο και να κατεβάζει σχετικά άρθρα, δημοσιεύσεις, e-books κτλ.

2.3.2. Όρια παρέμβασης διάχυτου υπολογισμού

Υπάρχει πάντα η πιθανότητα η εφαρμογή να ξεπεράσει τα όρια του επιθυμητού από το φοιτητή, και να κατεβάσει άρθρα που δεν αντικατοπτρίζουν τις ανάγκες του. Έτσι, αν ο φοιτητής σβήσει σχετικά άμεσα τα αρχεία αυτά, η εφαρμογή τις επόμενες φορές θα πρέπει να είναι σε θέση να κατεβάσει λιγότερα ή περισσότερο σχετικά με το αντικείμενο άρθρα. Ένα βασικό στόχος του διάχυτου υπολογισμού είναι να μην επεμβαίνει στη ζωή του χρήστη παραπάνω από όσο θεωρεί εκείνος απαραίτητο. Έτσι μέσω αλγορίθμων εκμάθησης ένα σύστημα μπορεί να εκμεταλλεύεται την ανατροφοδότηση που παίρνει από τους χρήστες σχετικά με τις ενέργειες που εκτελούν, και ανάλογα να προσαρμόζεται στις συνήθειές τους.

ΚΕΦΑΛΑΙΟ 3

ΑΝΑΠΑΡΑΣΤΑΣΗ ΠΛΗΡΟΦΟΡΙΑΣ ΠΛΑΙΣΙΟΥ

Στο κεφάλαιο αυτό βρίσκεται το μεγαλύτερο κομμάτι της ανάλυσης που έγινε σε στην εργασία. Πιο ειδικά για τη συγκεκριμένη εργασία, και ακολουθώντας μια ιατρικοκεντρική προσέγγιση, η πληροφορία πλαισίου σε πρώτο επίπεδο διαχωρίζεται βάσει της συχνότητας μεταβολής των τιμών των μεταβλητών σε:

- *Δυναμική πληροφορία πλαισίου*: Αποτελείται από πληροφορίες που συλλέγονται από το περιβάλλον του χρήστη, και υφίστανται συχνές αλλαγές. Παραδείγματα τέτοιων πληροφοριών είναι η θερμοκρασία χώρου, η θερμοκρασία σώματος, η πίεση, το σήμα ECG (Electrocardiogram) κ.ά.

- *Στατική πληφορορία πλαισίου*: Αποτελείται από πληροφορίες που δεν αλλάζουν ή που αλλάζουν με σχετικά μικρούς ρυθμούς. Είναι ουσιαστικά το προφίλ (ιστορικό) του χρήστη. Παραδείγματα πληροφοριών που συνθέτουν τη στατική πληροφορία πλαισίου είναι οι συνήθειες του χρήστη (κάπνισμα, άθληση, διατροφή κ.ά.) ή το ιατρικό ιστορικό του (πιθανές επεμβάσεις στο παρελθόν, χρόνιες ασθένειες, έλλειψη ενζύμων κ.ά.).

Με βάση αυτό το διαχωρισμό προκύπτουν δύο είδη παραμέτρων πληροφορίας πλαισίου, οι δυναμικές και οι στατικές.
Η φύση της πληροφορίας πλαισίου που διαχειρίζεται η εκάστοτε εφαρμογή, οδηγεί και σε ένα δεύτερου επιπέδου διαχωρισμό των παραμέτρων που την καθορίζουν. Κάποιες μεταβλητές πληροφορίας πλαισίου παίρνουν πάντα ρητές, καθορισμένες διακριτές τιμές (crisp) ενώ κάποιες άλλες μεταβλητές είναι ασαφείς (fuzzy), με αποτέλεσμα να μη μπορεί να καθοριστεί εξαρχής η τιμή τους. Με βάση αυτό το δεύτερο διαχωρισμό προκύπτουν δύο ακόμα είδη παραμέτρων πληροφορίας πλαισίου, οι σαφείς και οι ασαφείς.

Για τις μεταβλητές που χρησιμοποιούνται από τις βασισμένες σε πληροφορία πλαισίου εφαρμογές προκύπτουν οι εξής συνδυασμοί:

a) Σαφής – Δυναμική μεταβλητή (crisp – dynamic)
b) Σαφής – Στατική μεταβλητή (crisp – static), όπως για παράδειγμα η ηλικία, το βάρος, το ύψος, ο αριθμός των έως τώρα επεμβάσεων ενός ασθενή κτλ.
c) Ασαφής – Στατική μεταβλητή (fuzzy – static), όπως για παράδειγμα ο χαρακτηρισμός της συχνότητας καπνίσματος ενός ασθενή ως «ΜΕΓΑΛΗΣ» ή «ΜΕΤΡΙΑΣ», ή της διατροφής του ως «ΚΑΚΗΣ» κτλ.
d) Ασαφής – Δυναμική μεταβλητή (fuzzy – dynamic), όπως για παράδειγμα οι μετρήσεις αισθητήρων στο περιβάλλον ενός ασθενή, οι οποίοι παίρνουν τιμές για τη θερμοκρασία του, την υγρασία του περιβάλλοντος κτλ και τις αναπαριστούν με ασαφή τρόπο όταν αυτό είναι επιθυμητό και αποδοτικό.

Οποιαδήποτε πληροφορία είναι στατική (b ή c) θεωρούμε ότι περιγράφει το προφίλ του ασθενή (Patient Profile), καθώς αυτό δεν είναι ευμετάβλητο και άρα θεωρείται στατικό. Για την αναπαράσταση του προφίλ του ασθενή θα χρησιμοποιηθούν κλάσεις και ιδιότητες από μία κατάλληλα διαμορφωμένη οντολογία, η οποία θα αναλυθεί παρακάτω (Patient Profile Ontology).
Οποιαδήποτε πληροφορία είναι δυναμική (a ή d) θεωρούμε ότι είναι πληροφορία πλαισίου και περιγράφει την τρέχουσα κατάσταση του ασθενή (Patient Context). Ως επί το πλείστον αποτελείται από μετρήσεις αισθητήρων.

3.1. Οντολογική Αναπαράσταση Πληροφορίας Πλαισίου

3.1.1. Εισαγωγή

Μία *οντολογία* (ontology) είναι ο ορισμός τυπικών προδιαγραφών σχετικά με μία *εννοιολογία* (conceptualization). Μία εννοιολογία είναι μία αφηρημένη, απλοποιημένη άποψη ενός κόσμου που μία εφαρμογή επιθυμεί να αναπαραστήσει για κάποιο συγκεκριμένο σκοπό. Κάθε βάση γνώσης, σύστημα ή πράκτορας βασισμένος σε γνώση ενσωματώνει κάποια εννοιολογία είτε ρητά είτε άρρητα. Συχνά η εννοιολογία εκφράζεται χρησιμοποιώντας έννοιες, αντικείμενα και άλλες οντότητες καθώς και τις μεταξύ τους σχέσεις που θεωρείται ότι υπάρχουν σε κάποια περιοχή ενδιαφέροντος.
Όταν η γνώση ενός πεδίου αναπαρίσταται με ένα δηλωτικό φορμαλισμό (π.χ. με μία γλώσσα βασισμένη σε λογική, όπως Λογική Πρώτης Τάξης ή Περιγραφική Λογική), το

σύνολο των αντικειμένων καλείται *σύμπαν*. Το σύμπαν και οι μεταξύ τους σχέσεις είναι γνωστά ως *λεξικό αναπαράστασης* σύμφωνα με το οποίο μία εφαρμογή αναπαριστά τη γνώση του πεδίου ενδιαφέροντός της. Έτσι η οντολογία μίας εφαρμογής είναι εφικτό να περιγραφεί ορίζοντας ένα σύνολο από όρους αναπαράστασης. Σε μία τέτοια οντολογία οι ορισμοί συσχετίζουν τα ονόματα των οντοτήτων του σύμπαντος (π.χ. κλάσεις, σχέσεις, συναρτήσεις ή άλλα αντικείμενα) με κείμενο που είναι ανθρώπινα αναγνώσιμο και το οποίο περιγράφει την έννοιά τους, δηλαδή τη σημασία που αυτά έχουν στον ανθρώπινο κόσμο, και τυπικά αξιώματα που περιορίζουν την ερμηνεία και τη σωστή χρήση αυτών των όρων.

Τα τελευταία χρόνια, οι οντολογίες χρησιμοποιούνται από επιχειρήσεις και επιστημονικές κοινότητες ως ένα πρότυπο διαμοιρασμού, επαναχρησιμοποίησης και επεξεργασίας γνώσης. Πλέον οι οντολογίες παίζουν κεντρικό ρόλο σε συστήματα πληροφοριών διοίκησης, σε συστήματα ηλεκτρονικού εμπορίου, καθώς και σε web services του Σημασιολογικού Ιστού.

3.1.2. Στοιχεία Οντολογιών

Οι οντολογίες όπως χρησιμοποιούνται από την επιστήμη των υπολογιστών ορίζονται από τον Studer (1998) ως εξής: «*An ontology is a formal, explicit specification of a shared conceptualization».* Για τον ορισμό μιας οντολογίας μπορούν να χρησιμοποιηθούν περισσότεροι από ένας φορμαλισμοί αναπαράστασης γνώσης και αντίστοιχα οι γλώσσες που τους εκφράζουν. Παρόλ' αυτά όλοι οι φορμαλισμοί που χρησιμοποιούνται για το σκοπό αυτό, έχουν ένα ελάχιστο σετ κοινών στοιχείων που προσφέρουν:

- *Κλάσεις,* οι οποίες αναπαριστούν *concepts*. Για παράδειγμα, αν πεδίο ενδιαφέροντος είναι τα ταξίδια, τα concepts προς μοντελοποίηση θα μπορούσαν να είναι οι τοποθεσίες (πόλεις, χωριά κτλ.) και τα μέσα μεταφοράς (αεροπλάνα, τρένα, αυτοκίνητα κτλ.). Οι κλάσεις σε μια οντολογία συνήθως οργανώνονται σε ιεραρχική δομή, η οποία προσφέρει τη δυνατότητα χρήσης μηχανισμών κληρονομικότητας.
- *Σχέσεις,* οι οποίες αναπαριστούν συσχετισμούς μεταξύ κλάσσεων ενός πεδίου ενδιαφέροντος. Οι σχέσεις ορίζονται τυπικά ως εξής: Για κάθε σχέση R ισχύει $R \subseteq C1 \times C2 \times ... \times Cn$, όπου C_i οι κλάσεις τις οποίες συσχετίζει. Συνήθως οι

οντολογίες περιλαμβάνουν δυαδικές σχέσεις, δηλαδή του τύπου $R \subseteq C1 \times C2$. Η κλάση C1 σε μία τέτοια σχέση αποτελεί το domain (πεδίο ορισμού) της σχέσης αυτής, ενώ η κλάση C2 αποτελεί το range (πεδίο τιμών) της σχέσης. Για παράδειγμα η σχέση hasDisease, έχει domain την κλάση Human και range την κλάση Disease. Για τη σχέση αυτή ισχύει $hasDisease \subseteq Human \times Disease$. Στιγμιότυπα των σχέσεων που είναι ορισμένες σε μία οντολογία προκύπτουν μέσω γνώσης από το πεδίο ενδιαφέροντος. Για παράδειγμα για να την αναπαράσταση της γνώσης ότι το στιγμιότυπο Jim της κλάσης Human, έχει την ασθένεια CAD, η οποία είναι στιγμιότυπο της κλάσης Disease, χρησιμοποιείται η τριπλέτα (hasDisease, Jim, CAD) γραμμένη έτσι ή σε κάποιον αντίστοιχα εκφραστικό φορμαλισμό. Οι δυαδικές σχέσεις των οντολογιών χρησιμοποιούνται για να εκφράσουν *γνωρίσματά* τους. Τα γνωρίσματα διαφέρουν από τις σχέσεις ως προς το ότι το πεδίο τιμών τους είναι ένας τύπος δεδομένων, όπως αλφαριθμητικό, αριθμός κτλ, ενώ το πεδίο τιμών μιας σχέσης είναι συνήθως μια άλλη κλάση. Για παράδειγμα το πεδίο τιμών του γνωρίσματος hasTemperature είναι ένας αριθμός κινητής υποδιαστολής.

Οι οντολογίες αποτελούνται από δύο βασικά στοιχεία: ***Ονόματα*** (για τις έννοιες που περιγράφουν) και ***Ιδιότητες*** (που συνδέουν τις έννοιες αυτές μεταξύ τους).

Παράδειγμα 1:

Ονόματα: Πορτοκάλι, Μήλο, Φαγώσιμο, Φρούτο, Λαχανικό,

Ιδιότητες:

- Κάθετί που είναι Πορτοκάλι είναι πορτοκαλί.
- Κάθετί που είναι Μήλο είναι ή κόκκινο ή πράσινο.
- Κάτι που είναι Μήλο ΔΕΝ είναι Πορτοκάλι, και το αντίστροφο.
- Κάθετί που είναι ή Μήλο ή Πορτοκάλι είναι Φρούτο
- Κάτι που είναι Φρούτο ΔΕΝ είναι Λαχανικό, και το αντίστροφο.
- Κάθετί που είναι ή Φρούτο ή Λαχανικό είναι Φαγώσιμο.

Οι ιδιότητες στις οντολογίες δίνουν δυνατότητες συμπερασμού καταστάσεων, όπως φαίνεται από το επόμενο παράδειγμα.

Παράδειγμα 2:

Ονόματα: Αμάξι, Κίνδυνος, Ταχύτητα, Ρόδα

Ιδιότητες:

- Κάθετί που έχει 4 Ρόδες είναι Αμάξι
- Κάθε Αμάξι που κινείται με Ταχύτητα > 100 χλμ/ώρα ΚΑΙ έχει ο οδηγός είναι ΥΕΑ είναι σε Κίνδυνο
- Κάθε οδηγός που έχει πιει > X ml αλκοόλ είναι υπό την επήρρεια αλκοόλ.
- Κάθε αμάξι έχει έναν Οδηγό

Έστω ένα στιγμιότυπο της κλάσης «Αμάξι», Α = { ΟΔ1, Ταχ=150χλμ/ώρα } και ένα στιγμιότυπο της κλάσης «Οδηγός», ΟΔ1 = { 70 ετών, X^2 ml alcohol, μειωμένη όραση }. Με βάση τις παραπάνω ιδιότητες είναι εύκολο να βγει το συμπέρασμα ότι «το αμάξι Α είναι σε Κίνδυνο». Εδώ υπεισέρχεται η έννοια της *ασαφούς συλλογιστικής*, η οποία εξηγείται παρακάτω.

Ερευνητική δραστηριότητα που συντελείται αυτόν τον καιρό σχετικά με τη χρήση των Οντολογιών, είναι η εφαρμογή τους στο Σημασιολογικό Ιστό, καθώς και σε κάθε εφαρμογή που επιχειρεί συμπερασμό καταστάσεων. Σήμερα είναι αδύνατο να αναζητήσει κανείς μια έννοια στο διαδίκτυο, καθώς και ό,τι είναι σχετικό με αυτή, ξεπερνώντας το όριο που η λεξικογραφική σύγκριση των λέξεων αναζήτησης με το περιεχόμενο των ιστοσελίδων επιβάλλει. Ο Σημασιολογικός Ιστός επιχειρεί να εισαγάγει σημασιολογία στο περιεχόμενο των ιστοσελίδων, κάνοντας χρήση της Οντολογικής Αναπαράστασης αυτού.

3.1.3. Η Γλώσσα Περιγραφής Οντολογιών OWL

Οι διαφορετικές γλώσσες περιγραφής και ορισμού οντολογιών προσφέρουν και διαφορετικά εργαλεία μοντελοποίησης αυτών. Η εκφραστικότητα των προηγούμενων προτύπων RDF και RDF Schema ήταν αρκετά περιορισμένη σε ιεραρχίες κλάσεων και ιδιοτήτων. Το Web Ontology Working Group του W3C υπέδειξε κάποιες περιπτώσεις οντολογιών για το Σημασιολογικό Ιστό, όπου η εκφραστικότητα του RDF(S) δεν ήταν επαρκής. Η επόμενη πρόταση που έγινε από ερευνητικά κέντρα σε Ευρώπη και Αμερική ήταν η DAML+OIL, η οποία υπερείχε σε εκφραστικά χαρακτηριστικά, και έγινε

το σημείο αφετηρίας για την πρόταση της OWL (Web Ontology Language) ή οποία στη συνέχεια έγινε πρότυπο του W3C και πλέον είναι η πιο κοινά αποδεκτή γλώσσα περιγραφής/ορισμού οντολογιών στο Σημασιολογικό Ιστό.

Όσο πιο πλούσια είναι μια γλώσσα τόσο λιγότερο αποδοτικοί είναι οι μηχανισμοί συμπερασμού που υποστηρίζει, και γι' αυτόν ακριβώς το λόγο χρειάζεται να γίνουν αμοιβαίες παραχωρήσεις έτσι ώστε η γλώσσα να εξυπηρετεί τους σκοπούς για τους οποίους σχεδιάστηκε: Ο συμπερασμός να μπορεί να γίνει σε ικανοποιητικό χρόνο, ενώ η γλώσσα να μπορεί να εκφράσει μεγάλες κλάσεις και πλούσια γνώση, απαιτήσεις αμοιβαία συγκρουόμενες. Έτσι το Web Ontology Working Group του W3C όρισε τρεις υπο-γλώσσες οι οποίες απαρτίζουν τη γλώσσα OWL, και κάθε μια από τις οποίες φιλοδοξεί να ικανοποιήσει διαφορετικές πλευρές των παραπάνω απαιτήσεων.

- *OWL Full*: Η γλώσσα OWL, μαζί με ολόκληρο το σετ των χαρακτηριστικών που υποστηρίζει. Επιτρέπει επίσης και συνδυασμό των χαρακτηριστικών αυτών με το RDF(S). Ένα πλεονέκτημα της OWL Full είναι ότι είναι πλήρως συμβατή με το RDF και σε συντακτικό και σε σημασιολογικό επίπεδο. Κάθε έγκυρο RDF αρχείο είναι ταυτόχρονα και ένα έγκυρο OWL Full αρχείο. Ένα σημαντικό μειονέκτημα της OWL Full είναι το γεγονός ότι είναι μη-αποφασίσιμη, καθιστώντας μη κάθε προσπάθεια δημιουργίας μηχανισμών πλήρους συμπερασμού. Συμπερασματικά η OWL Full απευθύνεται σε χρήστες που απαιτούν τη μέγιστη εκφραστικότητα, συντακτική ελευθερία και πλήρη συμβατότητα με το RDF Schema, χωρίς καμμία εγγύηση ως προς την αποφασισιμότητα.

- *OWL DL (Description Logic)*: Η OWL DL είναι μια υπογλώσσα της OWL Full, χρησιμοποιώντας τμήμα μόνο της εκφραστικής της δύναμης, υποστηρίζοντας όμως αποδοτικούς μηχανισμούς πλήρους συμπερασμού. Παρόλ' αυτά, η OWL DL δεν είναι πλήρως συμβατή με το RDF. Συνήθως ένα RDF αρχείο χρειάζεται να επεκταθεί σε κάποια επίπεδα και να περιοριστεί σε κάποια άλλα έτσι ώστε να προκύψει ένα έγκυρο OWL DL αρχείο. Αντίστροφα, κάθε έγκυρο OWL DL αρχείο είναι και έγκυρο RDF αρχείο. Η επιλογή ανάμεσα στην OWL DL και στην OWL Full σχετίζεται συνήθως με τις ανάγκες της εφαρμογής για χρήση των εργαλείων του RDFS Schema για μετα-μοντελοποίηση (για παράδειγμα αν υπάρχει η ανάγκη ορισμού μιας κλάσης που αποτελείται από κλάσεις κτλ.). Στην παρούσα

εργασία χρησιμοποιήθηκε αυτή η υπο-γλώσσα της OWL για τη μοντελοποίηση της ιατρικής γνώσης, αφού ήταν απαραίτητη η ύπαρξη ενός μηχανισμού εξαγωγής συμπερασμάτων και η εκφραστικότητά της κρίθηκε πλήρως επαρκής για τις ανάγκες της οντολογίας.

- *OWL Lite*: Η OWL Lite είναι η πιο απλη συντακτικά υπο-γλώσσα της OWL. Χρησιμοποιείται σε περιπτώσεις που χρειάζεται μόνο μια απλή ιεραρχία κλάσσεων και απλοί περιορισμοί, αφού η εκφραστικότητά της είναι περιορισμένη. Ένα πλεονέκτημα της OWL Lite είναι ότι είναι πιο εύκολη και στη χρήση της (όσον αφορά στους χρήστες της) αλλά και στην υλοποίησή της (όσον αφορά στους κατασκευαστές εργαλείων για τη γλώσσα).

Για τις τρεις παραπάνω υπογλώσσες της OWL και τις οντολογίες που παράγουν ισχύει η προς τα πάνω συμβατότητα ως εξής:

- Κάθε έγκυρη OWL Lite οντολογία είναι ταυτόχρονα μια έγκυρη OWL DL οντολογία.
- Κάθε έγκυρη OWL DL οντολογία είναι ταυτόχρονα μια έγκυρη OWL Full οντολογία

3.1.4. Μοντελοποίηση Ιατρικής Γνώσης – Κλάσεις Οντολογίας

Στην ενότητα αυτή θα γίνει η παρουσίαση της οντολογίας που αναπτύχθηκε στα πλαίσια της εργασίας, ως παράδειγμα χρήσης γνώσης και συμπερασμού από οντολογίες στο ιατρικό πλαίσιο.

Στην οντολογία που χρησιμοποιείται, βασικές κλάσεις κατηγοριοποίησης είναι οι εξής δύο:

- ***HumanHighRisk***
- ***HumanLowRisk***

Οδηγός στη μοντελοποίηση της ιατρικής γνώσης ήταν ο ακόλουθος πίνακας, ο οποίος καθορίζει σε φυσική γλώσσα τι συνήθειες ή γνωρίσματα απαιτείται να έχει ένας ασθενής ώστε να κατηγοριοποιηθεί σε μία από τις δύο αυτές κλάσεις.

	SmokingFrequency	NutritionQuality	Age
HumanHighRisk	High	Αδιάφορο	Old
“	High	Non-Healthy	Αδιάφορο
“	Medium	Non-Healthy	Old
HumanLowRisk	Low or Medium	Healthy	Αδιάφορο
“	Medium	Non-Healthy	Young

Πίνακας 1

Κρίθηκε απαραίτητο να δημιουργηθούν κλάσεις που να αναπαριστούν τις συνήθειες Smoking και Nutrition και την ιδιότητα Age. Επίσης ορίστηκαν οι ιδιότητες hasAge, hasNutrition και hasHabbit.

Ακολουθώντας την παραπάνω λογική προέκυψαν οι εξής κλάσεις και υπο-κλάσεις:

- Nutrition
- Habbits
    - Smoking
    - Sports
- Age

που είναι οι κλάσεις στις οποίες αντιστοιχούν οι πληροφορίες που υπάρχουν από το ιστορικό του ασθενή. Σύμφωνα με τους ειδικούς αυτές οι 4 κλάσεις παίζουν σημαντικό ρόλο σχετικά με το χαρακτηρισμό ενός ασθενή ως «Υψηλού» ή «Χαμηλού» κινδύνου.

Στη σχεδίαση της οντολογίας δημιουργήθηκαν τα απαραίτητα *Value Partitions* ώστε να χρησιμοποιηθούν ως πεδία τιμών σε ιδιότητες που εξειδικεύουν τις παραπάνω κλάσεις σε ενδιαφέροντα υποσύνολα για τη γνώση που έπρεπε να αναπαρασταθεί. Τα Value Partitions δεν είναι τμήμα της OWL και καμμίας άλλης γλώσσας σχεδίασης οντολογιών, αλλά είναι ένα πρότυπο σχεδίασης (Design Pattern). Τα πρότυπα σχεδίασης στη σχεδίαση οντολογιών είναι αντίστοιχα με τα πρότυπα σχεδίασης στον αντικειμενοστρεφή προγραμματισμό, αποτελούν δηλαδή λύσεις σε σχεδιαστικά προβλήματα που προκύπτουν αρκετά συχνά. Αναπτύχθηκαν από ειδικούς της σχεδίασης στον εκάστοτε φορμαλισμό και έχουν αναγνωριστεί ως δοκιμασμένες λύσεις στις αντίστοιχες σχεδιαστικές δυσκολίες. Τα Value Partitions στην OWL

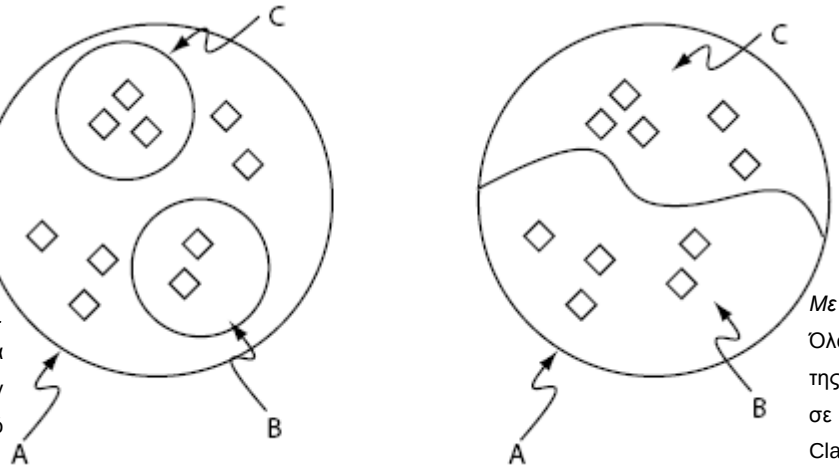
χρησιμοποιούνται τις πιο πολλές φορές ως range κάποιας ιδιότητας, για να περιορίσουν τις τιμές που μπορεί να πάρει, κάνοντας χρήση του «covering» αξιώματος. Το αξίωμα αναλύεται στα εξής δύο τμήματα:

- Την κλάση που «καλύπτεται»
- Τις κλάσεις που κάνουν την κάλυψη

Έστω η εξής ιεραρχία κλάσσεων:

- ClassA
    - ClassB
    - ClassC

Έστω ένα αξίωμα που «καλύπτει» την κλάση ClassA από τις κλάσεις ClassB και ClassC. Αυτό σημαίνει ότι ένα αντικείμενο που ανήκει στην ClassA, επιβάλλεται να ανήκει είτε στην ClassB είτε στην ClassC. Αυτό έρχεται σε αντίθεση με τη συνήθη περίπτωση, όπου ένα αντικείμενο μπορεί να ανήκει στην ClassA και σε καμμία από τις ClassB και ClassC. Έτσι, όταν εφαρμόζεται το αξίωμα, η ClassA θα έχει ως υπερκλάση της τη λογική ένωση ClassB U ClassC και οι κλάσεις ClassB και ClassC είναι μεταξύ τους ξένες (disjoint). Στην εικόνα 1 φαίνεται η διαφορά μεταξύ της εφαμογής και μη του covering αξιώματος.



Χωρίς covering αξίωμα. Υπάρχουν αντικείμενα της ClassA που δεν ανήκουν σε καμμία από τις ClassB/ClassC.

Με covering αξίωμα. Όλα τα αντικείμενα της ClassA ανήκουν σε μία από τις ClassB/ClassC.

Εικόνα 1: Διαφορά χρήσης και μη του covering αξιώματος

Τα Value Partitions, και οι αντίστοιχες υποκλάσεις που δημιουργήθηκαν στη συγκεκριμένη οντολογία είναι τα εξής:

QuantityVP	Disease	FoodQualityVP	AgeVP
High	HighRiskDisease	Healthy	Old
Medium	MediumRiskDisease	Non-Healthy	Young
Low	LowRiskDisease		

Πίνακας 2

Το QuantityVP είναι το γενικότερο Value Partition από τα τέσσερα που χρησιμοποιήθηκαν, παίζοντας το ρόλο του range για μια σειρά από ιδιότητες, ενώ τα άλλα τρία αποτελούν range για μία ιδιότητα το καθένα.

Τέλος, για πληρότητα και δοκιμαστικούς σκοπούς δημιουργήθηκαν οι κλάσεις:

- Patient
- HealthyFedHuman
- BadlyFedHuman
- ExcessiveSmoker
- YoungHuman

3.1.5. Μοντελοποίηση Ιατρικής Γνώσης – Ιδιότητες Οντολογίας

Για το συσχετισμό των κλάσσεων μεταξύ τους, και τη δημιουργία περιορισμών και γνωρισμάτων πάνω σε αυτές, κρίθηκε απαραίτητο να οριστούν οι εξής ιδιότητες:

- hasFoodQuality
- hasNutrition
- hasSmokingLevel
- hasHabbit
- hasAgeLevel
- hasAge
- hasDisease
- hasSportsLevel
- hasRiskValue

Στους περιορισμούς που χρησιμοποιήθηκαν για τους ορισμούς των κλάσεων HumanHighRisk και HumanLowRisk καθώς και των δοκιμαστικών κλάσσεων, οι ιδιότητες hasAgeLevel, hasFoodQuality και hasSmokingLevel έχουν πεδίο τιμών τα AgeVP, FoodQualityVP και QuantityVP αντίστοιχα.

3.2. Ασαφής Αναπαράσταση Πληροφορίας Πλαισίου

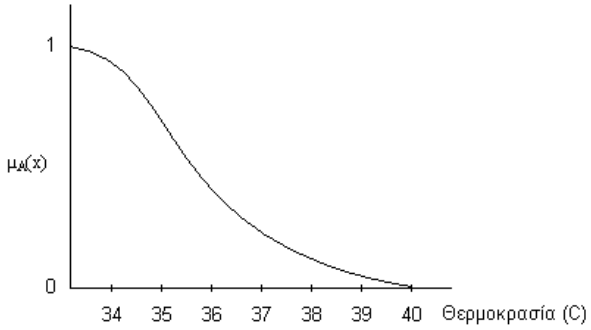
3.2.1. Εισαγωγή

Η αναπαράσταση πληροφορίας πλαισίου με τη χρήση ασαφών συνόλων διαφοροποιείται από την κλασική λογική, καθώς δεν αναπαριστά τη γνώση με δύο μόνο τιμές (αληθές-ψευδές) αλλά χειρίζεται και τιμές μεταξύ των δύο αυτών ακραίων καταστάσεων, υπό τη μορφή κάποιου βαθμού συγγένειας, όπως αυτός ορίζεται από την αντίστοιχη συνάρτηση συγγένειας.

3.2.2. Ασαφές Σύνολο

Ένα *ασαφές σύνολο* A ορίζεται μέσω της συνάρτησης συγγένειας $\mu_A(x)$: $X \rightarrow [0,1]$. Η τιμή της συνάρτησης για κάθε x αντιπροσωπεύει το βαθμό αληθείας ή συγγένειας του x

από το σύνολο X στο σύνολο A. Όσο μεγαλύτερη τιμή έχει η συνάρτηση για κάποιο x τόσο πιο «πολύ» ανήκει το x στο σύνολο A. Για παράδειγμα, έστω το ασαφές σύνολο Υποθερμία. Ένας τρόπος αναπαράστασής του είναι το διάγραμμα μιας συνάρτησης συγγένειας, στο οποίο αναπαρίστανται οι τιμές της $\mu_A(x)$ για κάθε x του πεδίου ενδιαφέροντος, το οποίο στην περίπτωσή μας είναι η Θερμοκρασία.



Εικόνα 2: Συνάρτηση συγγένειας της ασαφούς μεταβλητής Θερμοκρασία

Ένας άλλος τρόπος αναπαράστασης ενός ασαφούς συνόλου είναι η απαρίθμηση των ζευγών $(x, \mu_A(x))$ για τις διακριτές τιμές του x από το πεδίου ενδιαφέροντος, που είναι ένα υποσύνολο του πεδίου ορισμού της $\mu_A(x)$. Για το προηγούμενο παράδειγμα, το ασαφές σύνολο Υποθερμία μπορεί να αναπαρασταθεί ως εξής:

Υποθερμία = { 34/0.9, 35/0.8, 36/0.4, 37/0.3, 38/0.2, 39/0.1, 40/0 }

Η σημασία του οποίου είναι ότι η θερμοκρασία x = 34 ανήκει στο ασαφές σύνολο Υποθερμία με βαθμό συγγένειας 0.9η θερμοκρασία x = 35 ανήκει στο ασαφές σύνολο Υποθερμία με βαθμό συγγένειας 0.8 κ.ο.κ.

3.2.3. Ασαφής Μεταβλητή

Ασαφής μεταβλητή είναι μια μεταβλητή όπου οι τιμές τις είναι ασαφή σύνολά. Για παράδειγμα τα ασαφή σύνολα {Υποθερμία, Κανονική, Πυρετός} θα μπορούσαν να είναι το πεδίο τιμών της ασαφούς μεταβλητής «Θερμοκρασιακή Κατάσταση Ασθενούς» (ΘΚΑ).

3.2.4. Ασαφής Κανόνας

Ένας ασαφής κανόνας ***R*** είναι ένας συζευκτικός Horn κανόνας που σε modus ponens λογισμό συντάσσεται ως εξής:

$$\boldsymbol{R}\textbf{: If } (x_1 \textbf{ is } A^n{}_1) \textbf{ and } (x_2 \textbf{ is } A^k{}_2) \textbf{ and } \ldots \textbf{ and } (x_m \textbf{ is } A^l{}_m) \textbf{ Then } (y \textbf{ is } B^j)$$

Το πρώτο μέρος του κανόνα (antecedent-part) αποτελείται από τις **ασαφείς μεταβλητές** (fuzzy variables) x_i ,i = 1,…, m, που δέχονται ως τιμές ασαφή σύνολα $A^n{}_i$ π.χ. Υποθερμία, Υγρασία, και το αποτέλεσμα του κανόνα (consequent-part) είναι η ασαφής μεταβλητή ***y*** που δέχεται ως τιμή το ασαφές σύνολο B^j π.χ., βαθμός επικινδυνότητας κατάστασης του ασθενούς. Οι ασαφείς κανόνες αποτελούν την βάση για την ασαφή συλλογιστική και τον τελικό συμπερασμό της τιμής αποτελέσματος ***y*.** Κάθε ασαφής κανόνας "**If** x is A **Then** y is B" μπορεί να γραφτεί και μέσω της αναλυτικής περιγραφής του, χρησιμοποιώντας μια σχέση συνεπαγωγής ***R***(x,y), η οποία ορίζεται ως

$$\boldsymbol{R}(x,y) = u(x,y) = \varphi\,(\mu_A(x), \mu_B(y))$$

όπου φ ένας τελεστής συνεπαγωγής, σύμφωνα με τον οποίο συνδυάζονται οι τιμές των $\mu_A(x)$ και $\mu_B(y)$. Γνωστοί τελεστές συνεπαγωγής είναι οι:

Όνομα Τελεστή	Συμβολισμός	Αναλυτική Έκφραση Τελεστή
Zadeh Max-Min	φ_m	$(\mu_A(x) \wedge \mu_B(y)) \vee (1 - \mu_A(x))$
Mandani Min	φ_c	$\mu_A(x) \vee \mu_B(y)$
Larsen Product	φ_p	$\mu_A(x) \cdot \mu_B(y)$
Arithmetic	φ_a	$1 \wedge (1 - \mu_A(x) + \mu_B(y))$
Boolean	φ_b	$(1 - \mu_A(x)) \vee \mu_B(y)$

Πίνακας 3

Μια βάση γνώσης αποτελείται από ένα σύνολο ασαφών κανόνων που επιτρέπεται το αποτέλεσμα ενός ασαφούς κανόνα να εμφανίζεται στο πρώτο μέρος ενός άλλου ασαφούς κανόνα. Για παράδειγμα:

R_1: If (ΘΚΑ is Πυρετός) Then (Επικινδυνότητα is Μεγάλη).
R_2: If (Επικινδυνότητα is Μεγάλη) and (Διατροφή is Κακή) Then (ΠοσότηταΦαρμάκου is Μεγάλη).

Εδώ οι ασαφείς μεταβλητές ΘΚΑ, Επικινδυνότητα, Διατροφή και ΠοσοότηταΦαρμάκου παίρνουν τιμές στα ασαφή σύνολα Πυρετός, Μεγάλη, Κακή και Μεγάλη αντίστοιχα.

3.2.5. Ασαφής Συλλογιστική

Δοθέντος ενός συνόλου από ασαφείς κανόνες είναι εφικτό να εξαχθούν συμπεράσματα, ενδεχομένως ασαφή. Έστω ένα πρόβλημα ασαφούς συλλογιστικής με έναν ασαφή κανόνα της μορφής:

IF x is *A* ***THEN*** y is *B*

x is A’ y is B’ (?).

Το πρόβλημα αυτό λύνεται μέσω της διαδικασίας Generalized Modus Ponens (GMP), σύμφωνα με την οποία ισχύει:

$$B'=A' \circ R(x,y) \quad (1)$$

Η άγνωστη παράμετρος B’ υπολογίζεται συναρτήσει της γνωστής A’ και της συνάρτησης συνεπαγωγής του κανόνα “**If** x is A **Then** y is B”. Ο τελεστής ° είναι η συνάθροιση για κάθε ένα στοιχείο του A και του R. Εάν πρόκεται για πίνακες τότε είναι το γινόμενο των δύο αυτών πινάκων. Εάν πρόκειται για διανύσματα τότε είναι το εξωτερικό γινόμενο των διανυσμάτων αυτών, οπότε είναι κι αυτό ένα διάνυσμα.

Στη γενική περίπτωση τα προβλήματα ασαφούς συλλογιστικής περιλαμβάνουν περισσότερους του ενός κανόνες. Η επίλυση ενός τέτοιου προβλήματος προϋποθέτει τον υπολογισμό της συνάρτησης συνεπαγωγής ***R***(x,y) για κάθε κανόνα, την παραγωγή επιμέρους αποτελεσμάτων μέσω μιας διαδικασίας όπως η GMP που περιγράφτηκε παραπάνω, τη συνάθροιση των αποτελεσμάτων αυτών και τέλος την αποσαφήνιση κατά την οποία το ασαφές αποτέλεσμα μετατρέπεται σε μία συγκεκριμένη πραγματική τιμή.

Έστω το σύνολο ασαφών κανόνων:

IF ΘΚΑ is *Πυρετός* ***THEN*** ΕΠ is *Μεγάλη*
IF ΘΚΑ is *Κανονική* ***THEN*** ΕΠ is *Μικρή*

όπου η ασαφής μεταβλητή ΘΚΑ παίρνει τιμές στο πεδίο { Κανονική, Πυρετός }, η ασαφής μεταβλητή ΕΠ (Επικινδυνότητα) παίρνει τιμές στο πεδίο { Μικρή, Μεγάλη } και τα Κανονική, Πυρετός, Μικρή, Μεγάλη είναι ασαφή σύνολα. Επίσης δίνονται οι τιμές των συναρτήσεων συγγένειας για όλα τα ασαφή σύνολα του προβλήματος:

- Κανονική: { 34/0.2, 35/0.5, 36/0.8, 37/0.9, 38/0.4, 39/0.3, 40/0.2 }
- Πυρετός: { 34/0.1, 35/0.2, 36/0.2, 37/0.3, 38/0.8, 39/0.9, 40/1 }
- Μικρή: { 0/1, 2/0.8, 4/0.6, 6/0.4, 8/0.2, 10/0.0 }
- Μεγάλη: { 0/0.0, 2/0.2, 4/0.4, 6/0.6, 8/0.8, 10/1}

Δοθείσας μιας τιμής για την ασαφή μεταβλητή ΘΚΑ, είναι εφικτό να υπολογιστεί η τιμή της ασαφούς μεταβλητής ΕΠ, βάσει των δύο ασαφών κανόνων του προβλήματος, μέσω της διαδικασίας GMP. Αν ήταν γνωστή η τιμή της ΕΠ, θα μπορούσε να υπολογιστεί η τιμή της ΘΚΑ μέσω της διαδικασίας Generalized Modus Tollens (GMT). Έστω για τις ανάγκες του παραδείγματος ΘΚΑ=39.

3.2.5.1. Συνάρτηση συνεπαγωγής για κάθε ασαφή κανόνα του προβλήματος

Για τον υπολογισμό των συναρτήσεων συνεπαγωγής των κανόνων θα χρησιμοποιηθεί ο τελεστής συνεπαγωγής φ_p (Larsen Product) ως ο συχνότερα χρησιμοποιούμενος. Οι κανόνες Κ1 και Κ2 συσχετίζουν δύο ασαφείς μεταβλητές, επομένως η αναπαράσταση της συνάρτησης συνεπαγωγής μπορεί να γίνει με έναν πίνακα δύο διαστάσεων για κάθε κανόνα, έστω R_1 για τον K_1 και R_2 για τον K_2. Για τον K_1 προκύπτει ο ακόλουθος πίνακας, του οποίου κάθε κελί (i,j) ισούται με το γινόμενο:

$$\mu_{\text{Πυρετός}}(\text{ΘΚΑ}_i) \cdot \mu_{\text{Μεγάλη}}(\text{ΕΠ}_j)$$

IF ΘΚΑ is *Πυρετός* ***THEN*** ΕΠ is *Μεγάλη*

Πίνακας 4

R_{K1}	ΕΠ	0	2	4	6	8	10
ΘΚΑ		0	0.2	0.4	0.6	0.8	1
34	0.2	0	0.02	0.04	0.06	0.08	0.1
35	0.2	0	0.04	0.08	0.12	0.16	0.2
36	0.2	0	0.04	0.08	0.12	0.16	0.2
37	0.3	0	0.06	0.12	0.18	0.24	0.3
38	0.8	0	0.16	0.32	0.48	0.64	0.8
39	0.9	0	0.18	0.36	0.54	0.72	0.9
40	1	0	0.2	0.4	0.6	0.8	1

Αντίστοιχα προκύπτει ο πίνακας για τον K_2:

IF ΘΚΑ is *Κανονική* ***THEN*** ΕΠ is *Μικρή*

R_{K2}	ΕΠ	0	2	4	6	8	10
ΘΚΑ		1	0.8	0.6	0.4	0.2	0
34	0.1	0.1	0.08	0.06	0.04	0.02	0
35	0.5	0.5	0.40	0.30	0.20	0.10	0
36	0.8	0.8	0.64	0.48	0.32	0.16	0
37	0.9	0.9	0.72	0.54	0.36	0.18	0
38	0.4	0.4	0.32	0.24	0.16	0.08	0
39	0.2	0.2	0.16	0.12	0.08	0.04	0
40	0.1	0.1	0.08	0.06	0.04	0.02	0

3.2.5.2. Επιμέρους αποτελέσματα για κάθε ασαφή κανόνα του προβλήματος

Στο συγκεκριμένο πρόβλημα είναι γνωστή η τιμή μιας ασαφούς μεταβλητής που βρίσκεται στο antecedent-part των ασαφών κανόνων και ζητείται η τιμή της ασαφούς μεταβλητής που βρίσκεται στο consequent-part τους. Επομένως θα εφαρμοστεί η συλλογιστική διαδικασία GMP για την εξαγωγή των επιμέρους αποτελεσμάτων για κάθε κανόνα. Από τον τύπο (1) ισχύει ότι B'=A'°R(x,y). Στο παράδειγμα το B' αντιστοιχεί στην ασαφή μεταβλητή ΕΠ και το A' στην ασαφή μεταβλητή ΘΚΑ, της οποίας η τιμή είναι γνωστή. Έτσι, εφαρμόζοντας τον τύπο (1) στον πρώτο κανόνα του προβλήματος προκύπτει:

IF ΘΚΑ is *Πυρετός* ***THEN*** ΕΠ is *Μεγάλη*

ΘΚΑ' = 39 ΕΠ'$_{K1}$ = ?

$$ΕΠ'_{K1} = ΘΚΑ' \circ R_{K1}(ΘΚΑ_{Πυρετός}, ΕΠ_{Μεγάλη}) \quad (2)$$

Αν η ασαφής μεταβλητή ΘΚΑ' αναπαρασταθεί με τη μορφή ασαφούς συνόλου, προκύπτει το σύνολο

ΘΚΑ' = { 34/0, 35/0, 36/0, 37/0, 38/0, 39/1, 40/0 }, αφού η συνάρτηση συγγένειας για τη συγκεκριμένη τιμή της ΘΚΑ έχει μηδενική τιμή για οποιαδήποτε άλλη τιμή της ΘΚΑ. Για να εφαρμοστεί ο τύπος (2) πρέπει να εφαρμοστεί η μέθοδος σύνθεσης max-min.

$ΕΠ'_{K1}$ = [34/0, 35/0, 36/0, 37/0, 38/0, 39/1, 40/0]

ο

	0	2	4	6	8	10
34	0	0.02	0.04	0.06	0.08	0.1
35	0	0.04	0.08	0.12	0.16	0.2
36	0	0.04	0.08	0.12	0.16	0.2
37	0	0.06	0.12	0.18	0.24	0.3
38	0	0.16	0.32	0.48	0.64	0.8
39	0	0.18	0.36	0.54	0.72	0.9
40	0	0.2	0.4	0.6	0.8	1

= { 0/0, 2/0.18, 4/0.36, 6/0.54, 8/0.72, 10/0.9 }.

Το i-οστό στοιχείο του ασαφούς συνόλου $ΕΠ'_{K1}$ ισούται με το μεγαλύτερο από τα ανά ζεύγη μικρότερα στοιχεία μεταξύ του πρώτου πίνακα και της i-οστής στήλης του δεύτερου. Το τρίτο στοιχείο για παράδειγμα ισούται με:

max { min(0,0.04), min(0,0.08), min(0,0.08), min(0,0.12), min(0,0.32), min(1,0.36), min(0,0.4) } = max { 0,0,0,0,0,0.36,0 } = 0.36.

Αντίστοιχα για το δεύτερο κανόνα ισχύει:

IF ΘΚΑ is *Κανονική* ***THEN*** ΕΠ is *Μικρή*

ΘΚΑ' = 39 $ΕΠ'_{K2}$ = ?

$$ΕΠ'_{K2} = ΘΚΑ' \circ R_{K2}(ΘΚΑ_{Κανονική}, ΕΠ_{Μικρή}) \quad (3)$$

$$ΕΠ'_{K2} = [\ 34/0,\ 35/0,\ 36/0,\ 37/0,\ 38/0,\ 39/1,\ 40/0\ ]$$

ο

	0	2	4	6	8	10
34	0.1	0.08	0.06	0.04	0.02	0
35	0.5	0.40	0.30	0.20	0.10	0
36	0.8	0.64	0.48	0.32	0.16	0
37	0.9	0.72	0.54	0.36	0.18	0
38	0.4	0.32	0.24	0.16	0.08	0
39	0.2	0.16	0.12	0.08	0.04	0
40	0.1	0.08	0.06	0.04	0.02	0

$$= \{\ 0/0.2,\ 2/0.16,\ 4/0.12,\ 6/0.08,\ 8/0.04,\ 10/0\ \}.$$

3.2.5.3. Συνάθροιση επιμέρους αποτελεσμάτων

Στην ασαφή συλλογιστική ενεργοποιούνται πάντα περισσότεροι του ενός κανόνες με αποτέλεσμα να προκύπτουν πολλά αποτελέσματα, τα οποία πρέπει να συναθροιστούν προκειμένου να δώσουν ένα τελικό αποτέλεσμα για τη ζητούμενη τιμή της ασαφούς μεταβλητής. Οι συνηθέστερες μέθοδοι για τη συνάθροιση είναι η *MAX* και *SUM*.

1) *SUM*
   Η μέθοδος αυτή υπολογίζει το αποτέλεσμα ως το άθροισμα των επιμέρους αποτελεσμάτων σημείο προς σημείο. Είναι η συνηθέστερη μέθοδος όταν κατά τον υπολογισμό των συναρτήσεων συνεπαγωγής των κανόνων χρησιμοποιείται ο τελεστής συνεπαγωγής φ_p (Larsen Product) όπως στο παράδειγμα. Τα επιμέρους αποτελέσματα ήταν:

$$ΕΠ'_{K2} = \{\ 0/0.2,\ 2/0.16,\ 4/0.12,\ 6/0.08,\ 8/0.04,\ 10/0\ \}$$
$$ΕΠ'_{K1} = \{\ 0/0,\quad 2/0.18,\ 4/0.36,\ 6/0.54,\ 8/0.72,\ 10/0.9\ \}.$$

   Με τη μέθοδο συνάθροισης *SUM* προκύπτει:

ΕΠ' = { 0/0.2, 2/0.34, 4/0.48, 6/0.62, 8/0.76, 10/0.9 }.

2) *MAX*
   Η μέθοδος αυτή υπολογίζει το αποτέλεσμα ως το μέγιστο των επιμέρους αποτελεσμάτων για κάθε ζεύγος σημείων. Τα επιμέρους αποτελέσματα ήταν:

$$ΕΠ'_{K2}= \{ 0/0.2, 2/0.16, 4/0.12, 6/0.08, 8/0.04, 10/0 \}$$
$$ΕΠ'_{K1}= \{ 0/0, \quad 2/0.18, 4/0.36, 6/0.54, 8/0.72, 10/0.9 \}.$$

   Με τη μέθοδο συνάθροισης *MAX* προκύπτει:

ΕΠ' = { 0/0.2, 2/0.18, 4/0.36, 6/0.54, 8/0.72, 10/0.9 }.

Με τη μέθοδο *SUM* είναι πιθανό να προκύψουν τιμές συγγένειας μεγαλύτερες της μονάδας. Αν η μέθοδος αποσαφήνισης που θα υιοθετηθεί δεν είναι η μέθοδος *Centroid* πρέπει να γίνει κανονικοποίηση των τιμών αυτών.

3.2.5.4. Αποσαφήνιση

Από το στάδιο της συνάθροισης προέκυψε ένα ασαφές σύνολο για την τιμή της ζητούμενης ασαφούς μεταβλητής. Στο στάδιο της αποσαφήνισης το σύνολο αυτό μετατρέπεται σε μία πραγματική τιμή. Οι δύο συχνότερα χρησιμοποιούμενες μέθοδοι αποσαφήνισης είναι η *Centroid* και η *Maximum*.

1) *Centroid*
   Η μέθοδος αυτή υπολογίζει το αποτέλεσμα ως το κεντροειδές της συνάρτησης συγγένειας για την ασαφή μεταβλητή που ζητείται. Χρησιμοποιώντας το αποτέλεσμα της μεθόδου *SUM* προκύπτει:

$$ΕΠ_{Final}= \frac{0\cdot 0.2+2\cdot 0.34+4\cdot 0.48+6\cdot 0.62+8\cdot 0.76+10\cdot 0.9}{0.2+0.34+0.48+0.62+0.76+0.9} = \frac{21.4}{3.3} = 6.48$$

2) *Maximum*

Η μέθοδος αυτή υπολογίζει το αποτέλεσμα ως την τιμή που αντιστοιχεί στη μέγιστη τιμή συγγένειας στο ασαφές σύνολο του αποτελέσματος της συνάθροισης. Χρησιμοποιώντας το αποτέλεσμα της μεθόδου *MAX* προκύπτει:

$$\text{ΕΠ}_{Final}=10$$

Στο συγκεκριμένο παράδειγμα, και για τιμή ΘΚΑ'=39 η *Maximum* μέθοδος αποσαφήνισης έδωσε πιο ικανοποιητικό αποτέλεσμα από τη μέθοδο *Centroid*, αφού η τιμή της ΘΚΑ' έχει αρκετά έως πολύ μεγάλη τιμή συγγένειας στο ασαφές σύνολο εισόδου «Πυρετός» και έτσι αναμενόταν ο πρώτος κανόνας να επηρεάσει το αποτέλεσμα πολύ περισσότερο από ότι ο δεύτερος. Παρόλα αυτά, επειδή στα ασαφή σύνολα εισόδου υπάρχει επικάλυψη, η ΘΚΑ'=39 εκτός από «Πυρετός» με βαθμό πίστης 0.9, ανήκει και στο ασαφές σύνολο εισόδου «Κανονική» με βαθμό πίστης 0.3. Το γεγονός αυτό ενεργοποιεί το δεύτερο κανόνα περισσότερο από ότι θα περίμενε κανείς επηρεάζοντας έτσι το τελικό αποτέλεσμα.

3.2.6. Μοντελοποίηση Ιατρικής Γνώσης – Παραγωγή ασαφών κανόνων

Οι ασαφείς μεταβλητές που αποφασίστηκε να λάβουν μέρος στη συλλογιστική διαδικασία είναι μεταβλητές που αντιστοιχούν στη θερμοκρασία (έστω Τ) και στην τιμή του αιματοκρίτη του ασθενούς (έστω Η). Οι τιμές για τις δύο μεταβλητές λαμβάνονται μέσω κατάλληλων αισθητήρων όταν η διαδικασία κατηγοριοποίησης βάσει του ιστορικού του ασθενή, δώσει ως αποτέλεσμα ότι ο ασθενής ανήκει σε κλάση HighRisk. Η συλλογιστική διαδικασία, στο τέλος της αποσαφήνισης θα δώσει τιμή για μια ασαφή μεταβλητή που αντιστοιχεί στον τρέχοντα κίνδυνο του ασθενούς (έστω D). Το σύστημα θα αναλάβει να εκτελέσει την κατάλληλη ενέργεια σύμφωνα με την τιμή D.

Για κάθε μία από τις τρεις εμπλεκόμενες ασαφείς μεταβλητές (Τ, Η και D) ορίζονται τρία ασαφή σύνολα, *LOW, MEDIUM* και *HIGH*. Για τις μεταβλητές Τ και Η τα σύνολα αυτά καλύπτουν όσο φάσμα τιμών συναντάται σε ζώντες ανθρώπους. Για παράδειγμα, για τη θερμοκρασία Τ ενός ασθενή δεν έχει νόημα να καλυφθούν τιμές μικρότερες από τους 30 και μεγαλύτερες από τους 44 βαθμούς Κελσίου. Αντίστοιχες τιμές προκύπτουν από την ιατρική γνώση και για την ασαφή μεταβλητή που αντιστοιχεί στον αιματοκρίτη. Για την

ασαφή μεταβλητή που αντιστοιχεί στον κίνδύνο που διατρέχει ο ασθενής καλύπτεται ένα φάσμα από 0 έως 100.

Η ιατρική γνώση που υπάρχει σχετικά με τη σχέση των τριών ασαφών μεταβλητών είναι *μόνο* ότι οι τιμές των ασαφών μεταβλητών Τ και Η πρέπει να είναι σε *«φυσιολογικά»* επίπεδα προκειμένου ο κίνδυνος να είναι μικρός. Επομένως η γνώση αυτή μεταφράζεται στον κανόνα:

IF T is *MEDIUM* and H is *MEDIUM* ***THEN*** D is *LOW* (1)

αφού *μόνο* όταν *και* οι δύο ασαφείς μεταβλητές εισόδου βρίσκονται σε φυσιολογικά επίπεδα (όταν δηλαδή είναι *MEDIUM* σύμφωνα με την ιατρική γνώση) ο κίνδυνος που διατρέχει ο ασθενής υψηλού ρίσκου είναι μικρός.

Με αφετηρία τον κανόνα (1), και μέσω της ιατρικής εμπειρίας, δίνοντας την ίδια βαρύτητα στις δύο μεταβλητές εισόδου, προκύπτει το εξής συμπέρασμα: όταν *μόνο μία* από τις δύο μεταβλητές εισόδου πάρει *«μη φυσιολογική»* τιμή τότε ο κίνδυνος αυξάνει, αλλά δε φτάνει στα υψηλότερα επίπεδα. «Μη φυσιολογικά» ασαφή σύνολα τιμών για τις Τ και Η είναι τα σύνολα *HIGH* και *LOW*. Έτσι προκύπτουν οι ακόλουθοι ασαφείς κανόνες:

IF T is	*HIGH*	and H is	*MEDIUM*	***THEN***	D is	*MEDIUM*	(2)
IF T is	*LOW*	and H is	*MEDIUM*	***THEN***	D is	*MEDIUM*	(3)
IF T is	*MEDIUM*	and H is	*HIGH*	***THEN***	D is	*MEDIUM*	(4)
IF T is	*MEDIUM*	and H is	*LOW*	***THEN***	D is	*MEDIUM*	(5)

Πίνακας 5

Οι ασαφείς κανόνες (2) και (3) και οι κανόνες (4) και (5) μπορούν να συγχωνευτούν στους εξής κανόνες:

IF T is	*NOT MEDIUM*	and H is	*MEDIUM*	***THEN***	D is	*MEDIUM*	(6)
IF T is	*MEDIUM*	and H is	*NOT MEDIUM*	***THEN***	D is	*MEDIUM*	(7)

Πίνακας 6

Τέλος, όταν και οι δύο μεταβλητές εισόδου έχουν μη φυσιολογικές τιμές ο κίνδυνος που προκύπτει είναι ο μέγιστος δυνατός. Βάσει των δύο συμβάσεων προκύπτει ο κανόνας (8):

- Οι δύο μεταβλητές εισόδου είναι ισοβαρείς, πράγμα που σημαίνει ότι έχουν ίση επίδραση στην ενεργοποίηση των κανόνων και κατ' επέκταση στο τελικό αποτέλεσμα.
- Οι μη φυσιολογικές τιμές προς τις μεγαλύτερες τιμές του φάσματος έχουν την ίδια επίδραση με τις μη φυσιολογικές τιμές προς τις μικρότερες τιμές του φάσματος. Δηλαδή για έναν ασθενή Α1 δεν είναι «καλύτερο» να έχει θερμοκρασία Τ1 *μεγαλύτερη* από το φυσιολογικό σε σχέση με έναν ασθενή Α2 που έχει θερμοκρασία Τ2 *μικρότερη* από το φυσιολογικό. Αλλιώς, ενδιαφέρει μόνο αν οι τιμές εισόδου είναι φυσιολογικές ή όχι, και όχι το αν είναι μικρές ή μεγάλες.

IF T is *NOT MEDIUM* and H is *NOT MEDIUM* ***THEN*** D is *HIGH* (8)

Οι κανόνες που προέκυψαν και τροφοδότησαν την εφαρμογή είναι οι εξής τέσσερις:

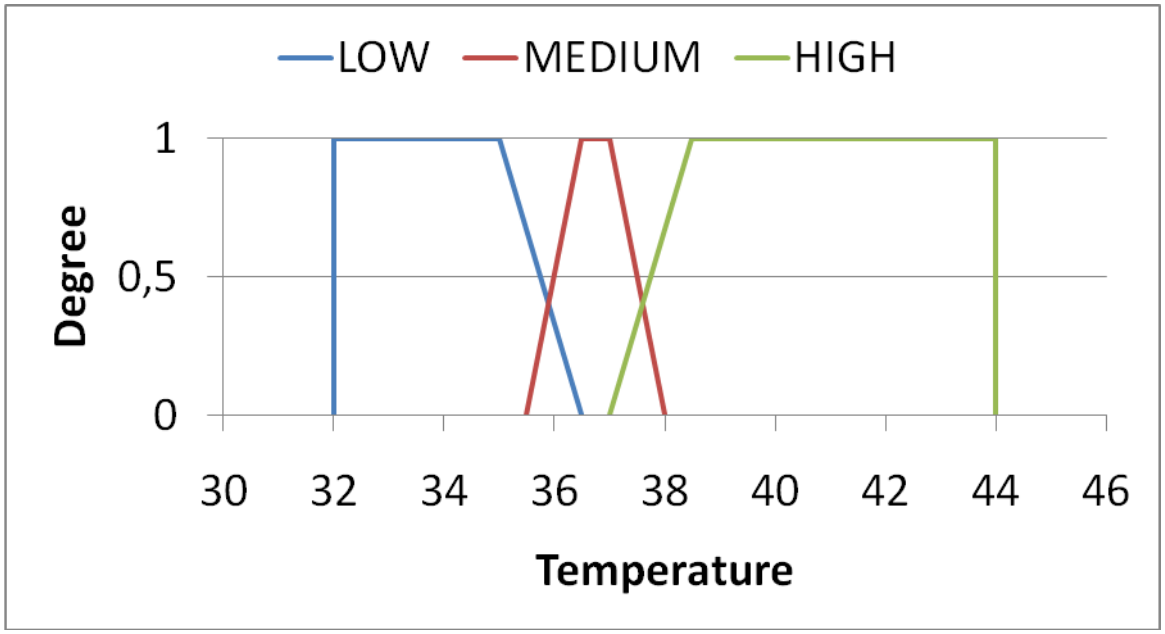
IF T is	*MEDIUM*	and H is	*MEDIUM*	***THEN***	D is	*LOW*
IF T is	*NOT MEDIUM*	and H is	*MEDIUM*	***THEN***	D is	*MEDIUM*
IF T is	*MEDIUM*	and H is	*NOT MEDIUM*	***THEN***	D is	*MEDIUM*
IF T is	*NOT MEDIUM*	and H is	*NOT MEDIUM*	***THEN***	D is	*HIGH*

Πίνακας 7

3.2.7. Μοντελοποίηση Ιατρικής Γνώσης – Δημιουργία ασαφών συνόλων

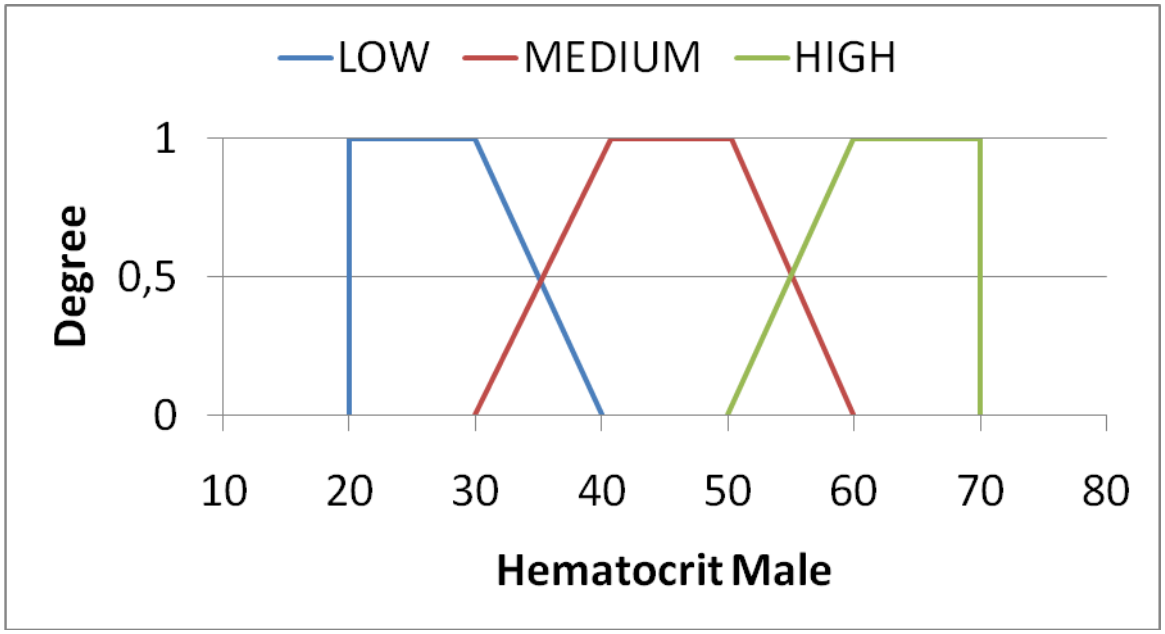
Προκειμένου να ενεργοποιηθούν οι κανόνες που περιγράφηκαν στην προηγούμενη ενότητα είναι αναγκαίο να οριστούν τα ασαφή σύνολα *HIGH, MEDIUM* και *LOW* για κάθε ασαφή μεταβλητή που εμπλέκεται στους κανόνες, επομένως για τις ασαφείς μεταβλητές T (Temperature), H (Hematocrit) και D (Danger). Τα ασαφή σύνολα προέκυψαν μέσω της ιατρικής εμπειρίας. Στο κεφάλαιο 4 προτείνεται μια άλλη μέθοδος αυτόματης κατασκευής των τραπεζίων που αντιστοιχούν στα ασαφή σύνολα, βασισμένη σε αλγορίθμους συσταδοποίησης μεγάλων συνόλων ιατρικών δεδομένων και αντίστοιχων βαθμών επικινδυνότητας.

Τα ασαφή σύνολα που αντιστοιχούν στην ασαφή μεταβλητή *Temperature*:

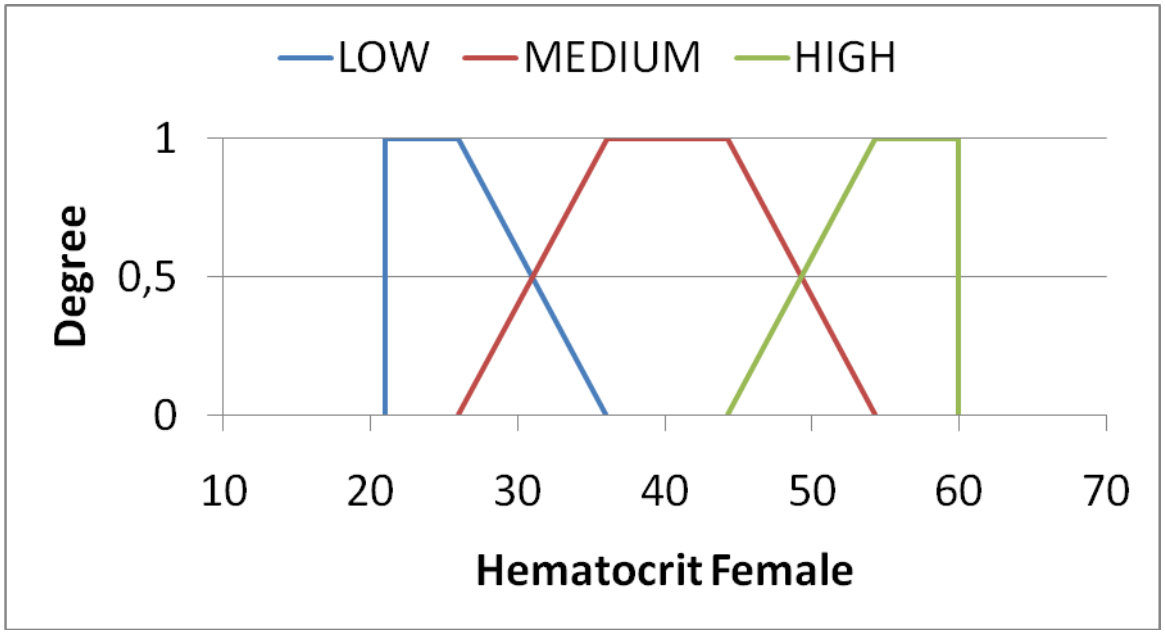


Εικόνα 3: Τα ασαφή σύνολα που αντιστοιχούν στην ασαφή μεταβλητή *Temperature*

Για την ασαφή μεταβλητή *Hematocrit* υπάρχει διαχωρισμός, ανάλογα με το αν ο ασθενής είναι άντρας ή γυναίκα. Έτσι τα τραπέζια που προκύπτουν είναι τα εξής:

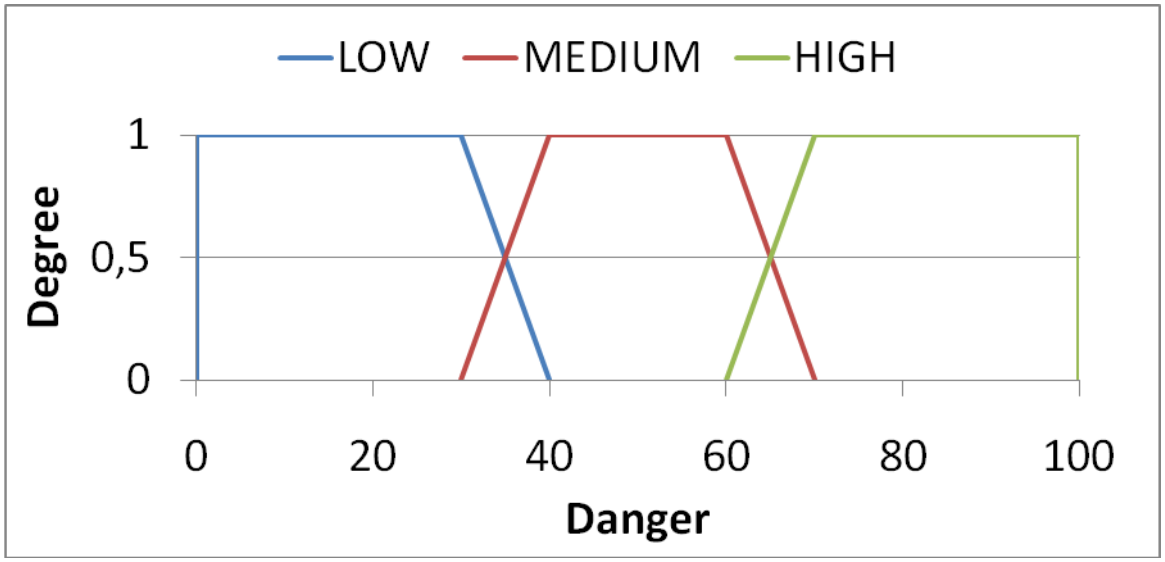


Εικόνα 4: Τα ασαφή σύνολα που αντιστοιχούν στην ασαφή μεταβλητή *Hematocrit Male*



Εικόνα 5: Τα ασαφή σύνολα που αντιστοιχούν στην ασαφή μεταβλητή *Hematocrit Female*

Τέλος, ορίζονται τα ασαφή σύνολα για την ασαφή μεταβλητή *Danger,* διαμερίζοντας το διάστημα [0,100] σε τρία τμήματα:



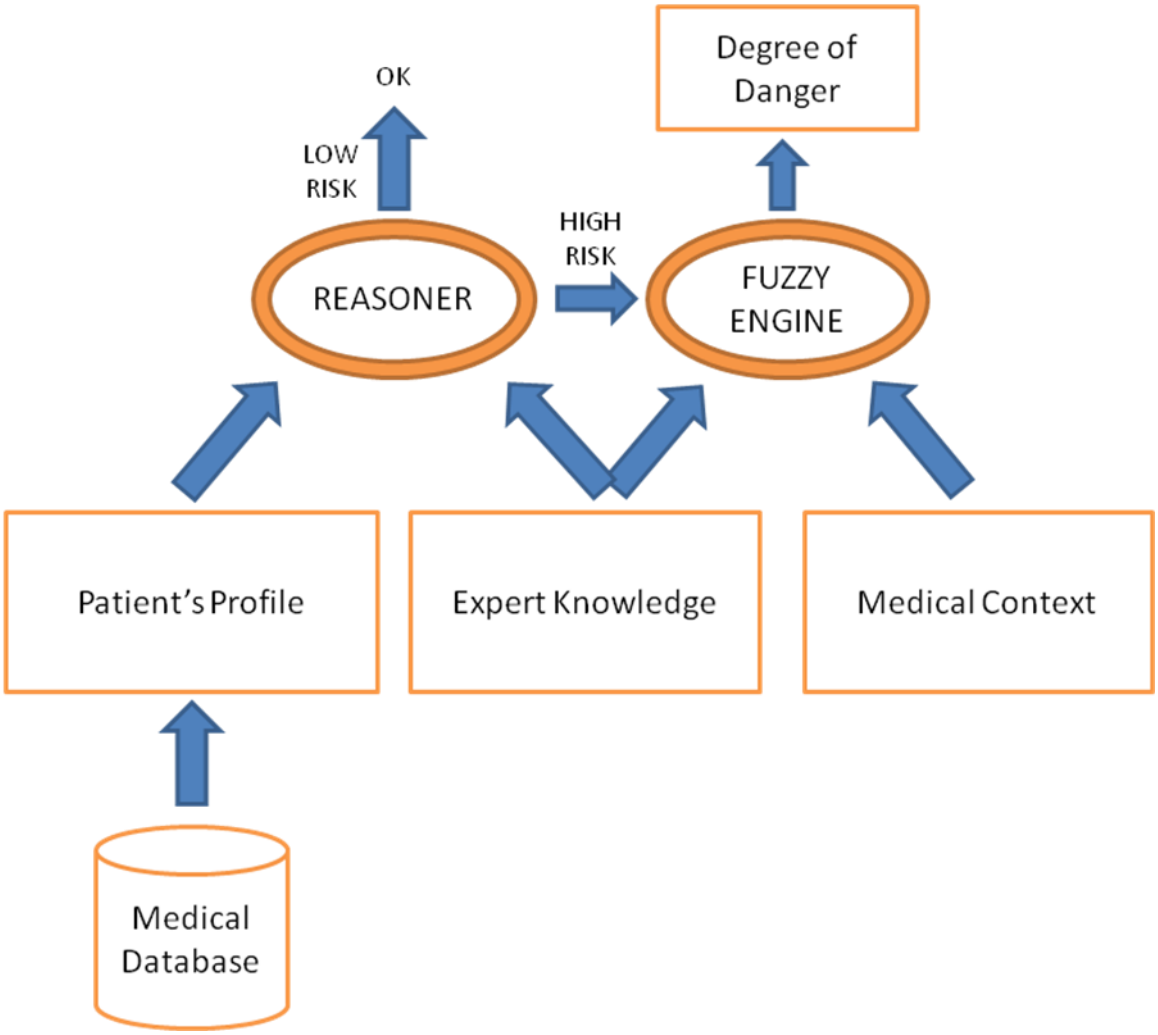
Εικόνα 6: Τα ασαφή σύνολα που αντιστοιχούν στην ασαφή μεταβλητή Danger

ΚΕΦΑΛΑΙΟ 4

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

4.1. Περιγραφή Συστήματος

Το σύστημα που αναπτύχθηκε βασίστηκε την αρχιτεκτονική της εικόνας 7:



Εικόνα 7: Αρχιτεκτονική Συστήματος

- Για κάθε ασθενή υπάρχουν καταχωρημένες σε βάσεις δεδομένων (Medical Database) πληροφορίες σχετικές με το ιατρικό προφίλ του (Patient's Profile), όπως οι διατροφικές του συνήθειες, χρόνιες παθήσεις, συνθήκες διαβίωσης, συχνότητα άθλησης κ.ά.

- Στο περιβάλλον του ασθενούς είναι εγκατεστημένοι αισθητήρες (Medical Context) που καταγράφουν την τρέχουσα θερμοκρασία χώρου και σώματος, το σήμα ECG, την τρέχουσα τιμή του αιματοκρίτη κ.ά.
- Με τη βοήθεια της ιατρικής εμπειρίας (Expert Knowledge) έχει αναπτυχθεί μια κατάλληλη ιατρική οντολογία που λαμβάνει μέρος στην κατηγοριοποίηση του ασθενούς βάσει του ιστορικού του προφίλ ως HighRiskHuman ή ως LowRiskHuman. Αν ο ασθενής κατηγοριοποιηθεί ως LowRiskHuman η διαδικασία σταματάει, μιας και η κατάσταση δεν είναι μεγάλου κινδύνου. Αν ο ασθενής κατηγοριοποιηθεί ως HighRiskHuman τότε το σύστημα μπαίνει σε κατάσταση κινδύνου (EmergencyMode).
- Με τη βοήθεια της ιατρικής εμπειρίας έχουν κατασκευαστεί κατάλληλοι ασαφείς κανόνες σχετικοί με τις τιμές που παίρνουν ζωτικοί δείκτες του ασθενούς όπως αυτοί που καταγράφονται από τους αισθητήρες. Οι κανόνες αυτοί τροφοδοτούν έναν ασαφή ελεγκτή (FuzzyEngine).
- Ο ασαφής ελεγκτής δεδομένων των τιμών που λαμβάνει από τους αισθητήρες, ενεργοποιεί τους κανόνες, και παράγει μέσω ασαφούς συλλογιστικής έναν τρέχοντα βαθμό επικινδυνότητας για τον ασθενή (Danger).

4.2. Παράδειγμα Εκτέλεσης

Για παράδειγμα, έστω ότι οι δείκτες που παρακολουθούνται στο περιβάλλον του ασθενούς είναι η θερμοκρασία σώματος (T) και ο αιματοκρίτης του (H). Οι πληροφορίες που αποθηκεύονται στη βάση δεδομένων σχετικά με το ιστορικό του προφίλ είναι η ποιότητα διατροφής του, η συχνότητα καπνίσματος, και η ηλικία του.
Έστω ότι για το στιγμιότυπο ασθενή “Jim” ισχύουν οι τιμές:

- Ποιότητα Διατροφής: Κακή (Non-healthy)
- Συχνότητα Καπνίσματος: Μεγάλη (High)
- Ηλικία: 85

Ο reasoner βάσει αυτών των στοιχείων και της οντολογίας που αναπτύχθηκε, θα δώσει ως αποτέλεσμα ότι ο ασθενής “Jim” ανήκει στην κλάση HighRiskHuman, και ως εκ τούτου το σύστημα θα μπει σε κατάσταση κινδύνου. Θα ζητηθούν από τους αισθητήρες

οι τρέχουσες τιμές θερμοκρασίας σώματος και αιματοκρίτη. Έστω ότι οι τιμές αυτές είναι:

- Θερμοκρασία σώματος: 40
- Αιματοκρίτης: 65

Η FuzzyEngine δεδομένων των τιμών αυτών, των ασαφών κανόνων, και των ασαφών συνόλων που αντιστοιχούν στις ασαφείς μεταβλητές «Θερμοκρασία Σώματος» και «Αιματοκρίτης» θα δώσει ως αποτέλεσμα, μετά από τη διαδικασία αποσαφήνισης μια τιμή για την ασαφή μεταβλητή Danger, η οποία παίρνει τιμές στο διάστημα [0,100]. Η τιμή που προκύπτει από το σύστημα για το συγκεκριμένο παράδειγμα είναι 82.1304, τιμή φυσιολογική, αφού και η θερμοκρασία σώματος και ο αιματοκρίτης έχουν τιμές πολύ μεγαλύτερες από το φυσιολογικό, που βάσει της ιατρικής γνώσης είναι το σύνολο *MEDIUM* και για τις δύο ασαφείς μεταβλητές.

ΚΕΦΑΛΑΙΟ 5

ΣΥΣΤΑΔΟΠΟΙΗΣΗ ΚΑΙ ΕΞΑΓΩΓΗ ΓΝΩΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΠΛΑΙΣΙΟΥ

Στο κεφάλαιο αυτό θα γίνει μια εισαγωγή σε έννοιες που αφορούν στις μεθόδους συσταδοποίησης που έχουν προταθεί, καθώς και στους τρόπους με τους οποίους κατηγοριοποιούνται οι μέθοδοι αυτές. Στη συνέχεια θα μελετηθούν δύο συγκεκριμένοι συχνά χρησιμοποιούμενοι αλγόριθμοι συσταδοποίησης καθώς και τα μειονεκτήματα και πλεονεκτήματα που αυτοί έχουν ως προς την εκμετάλλευση των αποτελεσμάτων τους στη συγκεκριμένη εφαρμογή. Τέλος, προτείνονται δύο μέτρα, με τη χρήση των οποίων μπορεί να γίνει εξαγωγή ασαφούς γνώσης από πειραματικά σύνολα εισόδου.

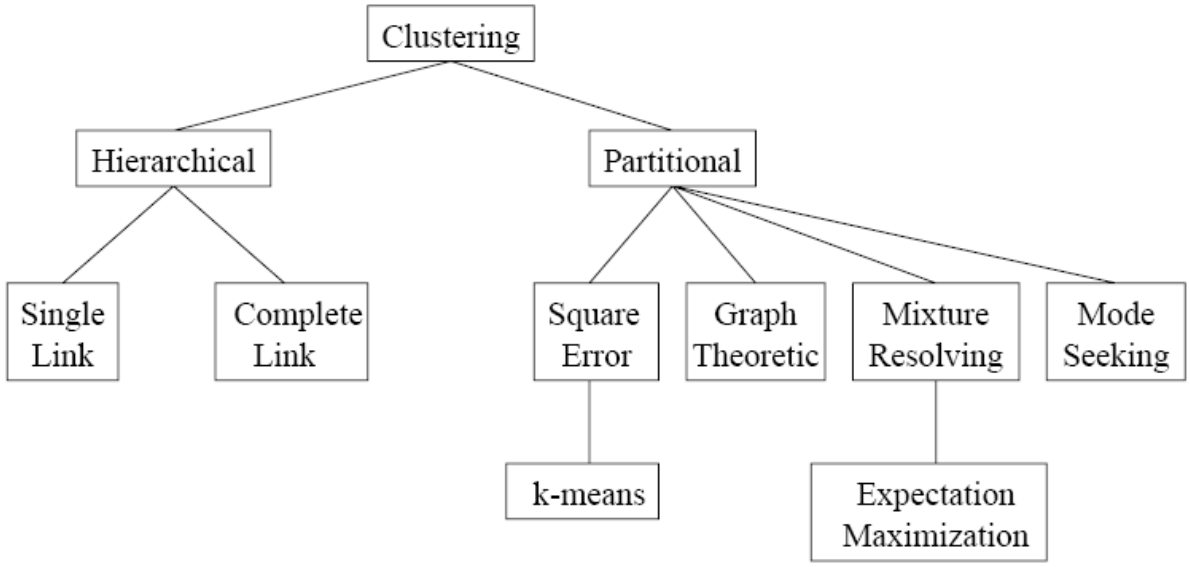
5.1. Συσταδοποίηση

Συσταδοποίηση ονομάζεται η διαδικασία της κατηγοριοποίησης αντικειμένων που ανήκουν σε ένα σύνολο εισόδου σε διαφορετικές κλάσεις, οι οποίες ονομάζονται συστάδες. Πιο συγκεκριμένα είναι η διαδικασία της διαμέρισης του συνόλου δεδομένων σε υποσύνολα (συστάδες), έτσι ώστε τα δεδομένα σε μία συστάδα να σχετίζονται σημασιολογικά. Η συσταδοποίηση δεδομένων είναι συνήθης πρακτική στατιστικής ανάλυσης και χρησιμοποιείται στους κλάδους της μηχανικής μάθησης, της εξόρυξης δεδομένων, της αναγνώρισης προτύπων, της βιοπληροφορικής και αλλού. Η κατηγοριοποίηση των δεδομένων σε k συστάδες, όπου το k είναι γνωστό εξαρχής, συχνά ονομάζεται *k-clustering*.

Οι εγγραφές συσταδοποιούνται με βάση την ομοιότητα (ή την απόσταση) που παρουσιάζουν μεταξύ τους, και δεν καταχωρούνται σε προκαθορισμένες συστάδες. Επαφίεται στον ειδικό του πεδίου ενδιαφέροντος να καθορίσει τη σημασιολογία που θα έχει κάθε μία από τις συστάδες που θα προκύψουν. Για παράδειγμα, συστάδες από παρόμοια συμπτώματα μπορεί να υποδεικνύουν διαφορετικές ασθένειες, συστάδες που περιλαμβάνουν τα προϊόντα που αγόρασαν πελάτες ενός καταστήματος μπορεί να υποδεικνύουν διαφορετικές αγοραστικές συνήθειες κτλ.

Η εικόνα 8 δίνει μία σχηματική αναπαράσταση της ιεραρχίας των μεθόδων συσταδοποίησης που έχουν προταθεί, χωρίς να αποτελεί το μόνο τρόπο ταξινόμησής

τους. Στο υψηλότερο επίπεδο οι μέθοδοι διαχωρίζονται σε *ιεραρχικές* (hierarchical) και *διαμεριστικές* (partitional) προσεγγίσεις.



Εικόνα 8: Διαχωρισμός μεθόδων συσταδοποίησης

- Οι *ιεραρχικές* μέθοδοι παράγουν σταδιακά το σύνολο των συστάδων, βασιζόμενες σε κάθε βήμα στις συστάδες που προέκυψαν από το προηγούμενο. Υλοποιούν είτε *συσσωρευτικούς* (“bottom-up”) είτε *διαιρετικούς* (“top-down”) αλγορίθμους. Οι συσσωρευτικοί αλγόριθμοι ξεκινούν με κάθε στοιχείο εισόδου να αποτελεί μία ξεχωριστή συστάδα και σε κάθε βήμα συγχωνεύουν τις δύο κοντινότερες συστάδες σε μία μεγαλύτερη. Οι διαιρετικοί αλγόριθμοι ξεκινούν με όλο το σύνολο εισόδου να αποτελεί μια συστάδα και συνεχίζουν διαιρώντας το σε συνεχώς μικρότερες συστάδες.

- Οι *διαμεριστικές* μέθοδοι δημιουργούν από την αρχή ένα σύνολο συστάδων με πλήθος ίσο με το ζητούμενο, και σε κάθε βήμα το τροποποιούν βελτιώνοντάς το. Κατ’ αυτό τον τρόπο δεν διαμορφώνουν ένα δέντρο συσταδοποίησης όπως οι ιεραρχικές μέθοδοι, και ενδείκνυνται για εφαρμογές που περιλαμβάνουν μεγάλα σύνολα εισόδου όπου η κατασκευή ενός τέτοιου δέντρου είναι υπολογιστικά απαγορευτική. Ένα μειονέκτημα των διαμεριστικών μεθόδων είναι ότι απαιτούν ως είσοδο το πλήθος των ζητούμενων συστάδων, που συνήθως δεν είναι εξαρχής γνωστό.

Ένας άλλος τρόπος διαχωρισμού των μεθόδων συσταδοποίησης είναι ως προς τη σαφήνεια, σύμφωνα με την οποία οι μέθοδοι χωρίζονται σε σαφείς (hard) και ασαφείς (fuzzy). Ένας σαφής αλγόριθμος συσταδοποίησης αναθέτει κάθε στοιχείο εισόδου σε μία και μόνο συστάδα κατα τη διάρκεια της εκτέλεσής του και στην έξοδο που παράγει. Αντίθετα, ένας ασαφής αλγόριθμος αναθέτει σε κάθε στοιχείο εισόδου βαθμούς συγγένειας ως προς περισσότερες από μία συστάδα. Η έξοδος ενός ασαφούς αλγορίθμου μπορεί να μετατραπεί εύκολα σε σαφή συσταδοποίηση, αναθέτοντας κάθε στοιχείο στη συστάδα για την οποία παρουσιάζει το μεγαλύτερο βαθμό συγγένειας.
Τέλος, ο πίνακας 8 παρουσιάζει μια συνολική συγκριτική εικόνα των πιο δημοφιλών αλγορίθμων συσταδοποίησης ως προς την πολυπλοκότητα χρόνου και χώρου του καθενός.

Αλγόριθμος Συσταδοποίησης	Πολυπλοκότητα Χρόνου	Πολυπλοκότητα Χώρου
leader	O(kn)	O(k)
k-means	O(nkl)	O(k)
ISODATA	O(nkl)	O(k)
Shortest spanning path	$O(n^2)$	O(n)
Single-line	$O(n^2 \log n)$	$O(n^2)$
Complete-line	$O(n^2 \log n)$	$O(n^2)$

Πίνακας 8

k: πλήθος των συστάδων

n: πλήθος σημείων εισόδου

l: πλήθος επαναλήψεων μέχρι τη σύγκλιση του αλγορίθμου

5.2. Αλγόριθμος Συσταδοποίησης K-means

Ο αλγόριθμος k-means ανήκει στην ομάδα των διαμεριστικών αλγορίθμων συσταδοποίησης, οι οποίοι δοθείσας μία βάσης n αντικειμένων και ενός αριθμού k, διαιρώντας επαναληπτικά το αρχικό σύνολο παράγουν k ομάδες αντικειμένων. Στόχος είναι η εύρεση της ομαδοποίησης που ελαχιστοποιεί ένα κριτήριο συσταδοποίησης. Η εύρεση του ολικού βέλτιστου απαιτεί την εξαντλητική διερεύνηση όλων των δυνατών τρόπων ομαδοποίησης, μέθοδος που για μεγάλα n είναι μη αποδοτική. Ο αλγόριθμος k-means (MacQueen 1967) χρησιμοποιώντας μια ευριστική μέθοδο επιτυγχάνει να προσεγγίσει ικανοποιητικά το ολικό βέλτιστο, σε γραμμικό χρόνο. Στον k-means κάθε cluster αντιπροσωπεύεται από το κέντρο του. Το κριτήριο συσταδοποίησης είναι η

ελαχιστοποίηση της μέσης τετραγωνικής απόστασης των αντικειμένων από το πλησιέστερο για κάθε αντικείμενο κέντρο συστάδας:

$$\min E = \sum_{i=1}^{k} \sum_{x \in C_i} (x - \mu_i)^2$$

όπου μ_i είναι το κέντρο της συστάδας i, το οποίο δεν είναι απαραίτητο να αντιστοιχεί σε ένα από τα αντικείμενα – σημεία εισόδου. Ο k-means αποτελείται από 5 βήματα, δοθέντος ενός αριθμού συστάδων k και ενός συνόλου n αντικειμένων – σημείων :

1. Εύρεση k τυχαίων αρχικών κέντρων για τις k συστάδες, μ_j, j=1,2,..,k.
2. Υπολογισμός της απόστασης κάθε σημείου του συνόλου εισόδου από το κέντρο κάθε συστάδας $d_{ij} = (x_i - \mu_j)^2$, i=1,2,..,n και j=1,2,..,k
3. Κάθε σημείο x_i αντιστοιχίζεται στη συστάδα για την οποία ισχύει:

   $\min_j (d_{ij})$, με j=1,2,..,k
4. Επαναπροσδιορισμός των κέντρων των συστάδων ως εξής:

   $\mu_j = \frac{\sum_{i=1}^{n_j} x_i}{n_j}$

   με j=1,2,..,k και n_j να ισούται με τον αριθμό των σημείων που έχουν ανατεθεί στη συστάδα j.
5. Αν τα κέντρα των συστάδων μετακινήθηκαν λιγότερο από ένα όριο ε μικρό, τέλος αλγορίθμου. Αλλιώς πήγαινε στο βήμα 2.

Ο αλγόριθμος k-means είναι σχετικά αποδοτικός, καθώς συγκλίνει σε λύση σε χρόνο O(tkn), όπου t το πλήθος των επαναλήψεων, k το πλήθος των συστάδων, και n το πλήθος των σημείων εισόδου με t, k $\ll$ n. Η λύση στην οποία καταλήγει ο αλγόριθμος είναι τις περισσότερες φορές ένα τοπικό βέλτιστο. Παρόλα αυτά μπορεί να εφαρμοστεί μόνο όπου έχει νόημα η ύπαρξη του μέσου, συνεπώς όχι για κατηγορικά δεδομένα. Επίσης το πλήθος των συστάδων k θα πρέπει να καθοριστεί εξ' αρχής, γεγονός που περιορίζει σημαντικά το πεδίο εφαρμογής του. Δεν χειρίζεται αποτελεσματικά το θόρυβο και παράγει μόνο κυρτές συστάδες.

5.3. Αλγόριθμος Εύρεσης Κλάσεων Συσταδοποίησης Leader Follower

Το κυριότερο μειονέκτημα του k-means, είναι ότι ο αριθμός των συστάδων πρέπει να είναι εκ των προτέρων γνωστός, με κίνδυνο να είναι ακατάλληλος για την κατανομή των δεδομένων εισόδου. Ο online αλγόριθμος Leader Follower επιλύει το πρόβλημα αυτό, παράγοντας έναν κατάλληλο αριθμό k για τις συστάδες εξόδου, εξετάζοντας για κάθε νέο σημείο εισόδου αν η απόστασή του από το κοντινότερο κέντρο συστάδας είναι μεγαλύτερη ή μικρότερη από ένα μικρό όριο το οποίο αντιστοιχεί στη μέγιστη ακτίνα που μπορεί να έχει μια συστάδα. Αν το σημείο είναι μακριά ακόμα και από το κοντινότερο κέντρο συστάδας τότε δημιουργείται μια νέα συστάδα που περιέχει προς το παρόν μόνο το σημείο αυτό, και αυξάνεται ο αριθμός k των συστάδων. Αν το σημείο είναι αρκετά κοντά σε ένα κέντρο συστάδας, τότε ανατίθεται στη συστάδα αυτή και ο αριθμός k των συστάδων μένει αμετάβλητος.

```
Input: X = {Xi, i=1,...,n}, e in (0,1)
Start
        C1 = {X1}
        For i=2:n Do
                Find cluster j such as: D = distance(Cj,Xi) is the minimum
                If (D < e)
                        Cj = Cj union {Xi}
                        Update center of Cj
                else
                        create new cluster Cj with center Xi
        end
end
End.
```

Στο συγκεκριμένο αλγόριθμο πρέπει να δοθεί ως είσοδος ένα όριο απόστασης σημείου-συστάδας, πάνω από το οποίο το σημείο δεν μπορεί να ανατεθεί στη συστάδα. Η διαφορά από τον k-means είναι ότι το όριο αυτό επηρεάζει έμμεσα το τελικό αποτέλεσμα, και όχι άμεσα όπως το k στον k-means. Ένα ακόμα μειονέκτημα του Leader Follower αλγορίθμου είναι ότι το σύνολο συστάδων εξόδου που θα προκύψει εξαρτάται σε μεγάλο βαθμό από τη σειρά που θα παρουσιαστούν τα σημεία εισόδου. Η σειρά παρουσίασης των σημείων εισόδου επηρεάζει λιγότερο τη συσταδοποίηση όσο αυξάνεται ο αριθμός των σημείων.

5.4. Εξαγωγή Ασαφών Κανόνων και Γνώσης από Συσταδοποίηση (Trapezoid Fuzzy Sets from Centers of Clusters using degrees of Separation and Compactness)

Οι μέθοδοι συσταδοποίησης που περιγράφηκαν παραπάνω έχουν ως στόχο να παραγάγουν συστάδες από σημεία εισόδου, έτσι ώστε τα σημεία που ανήκουν στην ίδια συστάδα να είναι πιο κοντά μεταξύ τους από ότι τα σημεία που ανήκουν σε διαφορετικές συστάδες. Η ιδιότητα αυτή μπορεί να εκφραστεί μέσω των εννοιών της διαχωρισιμότητας και της πυκνότητας των συστάδων που προκύπτουν.

5.4.1. Πυκνότητα Συστάδων (Cluster Compactness)

Η έννοια της πυκνότητας των συστάδων βασίζεται στην τυπική απόκλιση ενός συνόλου σημείων, που δίνεται από τον τύπο:

$$v(X) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} d^2(x_i, \bar{x})}$$

$d(x_i, x_j)$: μια μετρική απόστασης μεταξύ των σημείων x_i και x_j

N: το πλήθος των σημείων του συνόλου X

$\bar{x} = \frac{1}{N}\sum_i x_i$: ο μέσος του συνόλου X

Όσο μικρότερη είναι η τυπική απόκλιση ενός συνόλου, τόσο μεγαλύτερη είναι η ομοιότητα των σημείων που περιλαμβάνει, ως προς τη μετρική απόστασης d που χρησιμοποιήθηκε για τον υπολογισμό της τυπικής απόκλισης.

Η πυκνότητα Cmp για ένα σύνολο k συστάδων c_1, c_2, .. , c_k όπως αυτές προέκυψαν από κάποιον αλγόριθμο συσταδοποίησης με είσοδο ένα σύνολο σημείων X ορίζεται ως:

$$Cmp = \frac{1}{k}\sum_{i}^{k} \frac{v(c_i)}{v(X)}$$

όπου $v(c_i)$ είναι η τυπική απόκλιση της συστάδας i, και $v(X)$ η τυπική απόκλιση του συνόλου X. Η πυκνότητα των συστάδων Cmp εκφράζει το πόσο διάσπαρτες είναι οι συστάδες σε σχέση με το σύνολο εισόδου. Όσο μικρότερη είναι η τιμή Cmp για ένα σύνολο συστάδων, τόσο μεγαλύτερη είναι η μέση πυκνότητά τους. Το εύρος τιμών της Cmp είναι [0,1].

Α) Για k=1, όταν δηλαδή η έξοδος είναι μία μόνο συστάδα που περιλαμβάνει όλα τα σημεία εισόδου.

Τότε $v(c_i) = v(X)$. Οπότε $Cmp = \frac{v(c_i)}{v(x)} = 1$

Β) Για k = N = |X|, όταν δηλαδή κάθε συστάδα περιλαμβάνει ένα και μοναδικό σημείο εισόδου.

Τότε $d^2(x_i, \bar{x}) = 0$ για κάθε $x_i \in X$, αφού $x_i = \bar{x}$. Οπότε $v(c_i) = 0$ για κάθε i. Επομένως $Cmp = 0$

Παρατηρείται ότι μια οσοδήποτε μικρή τιμή για την πυκνότητα Cmp δεν εξασφαλίζει πάντα «καλύτερη» συσταδοποίηση. Όταν ο αριθμός των συστάδων είναι μεγάλος, ή το σύνολο εισόδου είναι διάσπαρτο, τότε η τιμή της πυκνότητας Cmp θα είναι μικρή, γεγονός που οδηγεί στο συμπέρασμα ότι η πυκνότητα της συσταδοποίησης δεν αρκεί για να χαρακτηρίσει την ποιότητα του αποτελέσματος.

5.4.2. Διαχωρισιμότητα Συστάδων (Cluster Separation)

Το μέτρο της διαχωρισιμότητας Sep για ένα σύνολο k συστάδων ορίζεται ως:

$$Sep \equiv \frac{1}{k(k-1)} \sum_{i=1}^{k} \sum_{j=1, j \neq i}^{k} exp\left(-\frac{d^2\left(x_{c_i}, x_{c_j}\right)}{2\sigma^2}\right)$$

σ: μια Gaussian σταθερά

x_{c_i} : το κέντρο της συστάδας c_i

d( ) : η μετρική απόστασης που χρησιμοποιήθηκε από τον αλγόριθμο συσταδοποίησης που παρήγαγε το σύνολο των k συστάδων

Το μέτρο της διαχωρισιμότητας των συστάδων βασίζεται στην απόσταση μεταξύ των κέντρων τους. Όσο μικρότερη είναι η τιμή της διαχωρισιμότητας Sep τόσο πιο ανόμοιες είναι μεταξύ τους οι συστάδες που προέκυψαν από τον αλγόριθμο συσταδοποίησης. Στην ειδική και μη επιθυμητή περίπτωση που η έξοδος αποτελείται από μία μόνο συστάδα η οποία περιλαμβάνει όλα τα σημεία εισόδου προκύπτει Sep = 0. Ούτε το μέτρο της διαχωρισιμότητας αρκεί για να χαρακτηρίσει την ποιότητα του αποτελέσματος. Απαιτείται ο συνδυασμός των δύο μέτρων για να προκύψει κάποιο συμπέρασμα σχετικά με την ποιότητα της συσταδοποίησης και την αποτελεσματικότητα του αλγορίθμου που χρησιμοποιήθηκε. Προκύπτει το μέτρο Ocq(β) με $\beta \in [0,1]$:

$$Ocq(\beta) \equiv \beta \times Cmp + (1 - \beta) \times Sep$$

Όταν είναι β=0.5 δίνονται ίδιες βαρύτητες στη διαχωρισιμότητα και στην πυκνότητα της συσταδοποίησης.

5.4.3. Συνδυασμός των δύο μέτρων

Τα δύο μέτρα που παρουσιάστηκαν μπορούν να χρησιμοποιηθούν για την προσεγγιστική κατασκευή των τραπεζίων των ασαφών συνόλων που αντιστοιχούν στις ασαφείς μεταβλητές T (Temperature) και H (Hematocrit).

Όσο μικρότερη τιμή έχει το μέτρο ***Cmp***:

- Τόσο μεγαλύτερη είναι η μέση πυκνότητα της κάθε συστάδας ξεχωριστά
- Επομένως τόσο καλύτερη είναι η προσέγγιση ότι τα σημεία γύρω από το κέντρο κάθε συστάδας έχουν με αυτό τον ίδιο βαθμό συγγένειας σε ένα ασαφές σύνολο που αντιπροσωπεύει τη συστάδα.
- Επομένως τόσο πιο πλατιές θα είναι οι *πάνω* έδρες των τραπεζίων αναπαράστασης των αντίστοιχων ασαφών συνόλων.

Όσο μικρότερη τιμή έχει το μέτρο ***Sep***:

- Τόσο πιο ανόμοιες είναι οι συστάδες μεταξύ τους.
- Επομένως τόσο πιο μακριά είναι η μία συστάδα από τις υπόλοιπες.

- Επομένως τόσο πιο στενές θα είναι οι *κάτω* έδρες των τραπεζίων αναπαράστασης των αντίστοιχων ασαφών συνόλων.

ΚΕΦΑΛΑΙΟ 6

ΛΟΓΙΣΜΙΚΟ ΑΝΑΠΤΥΞΗΣ ΣΥΣΤΗΜΑΤΟΣ

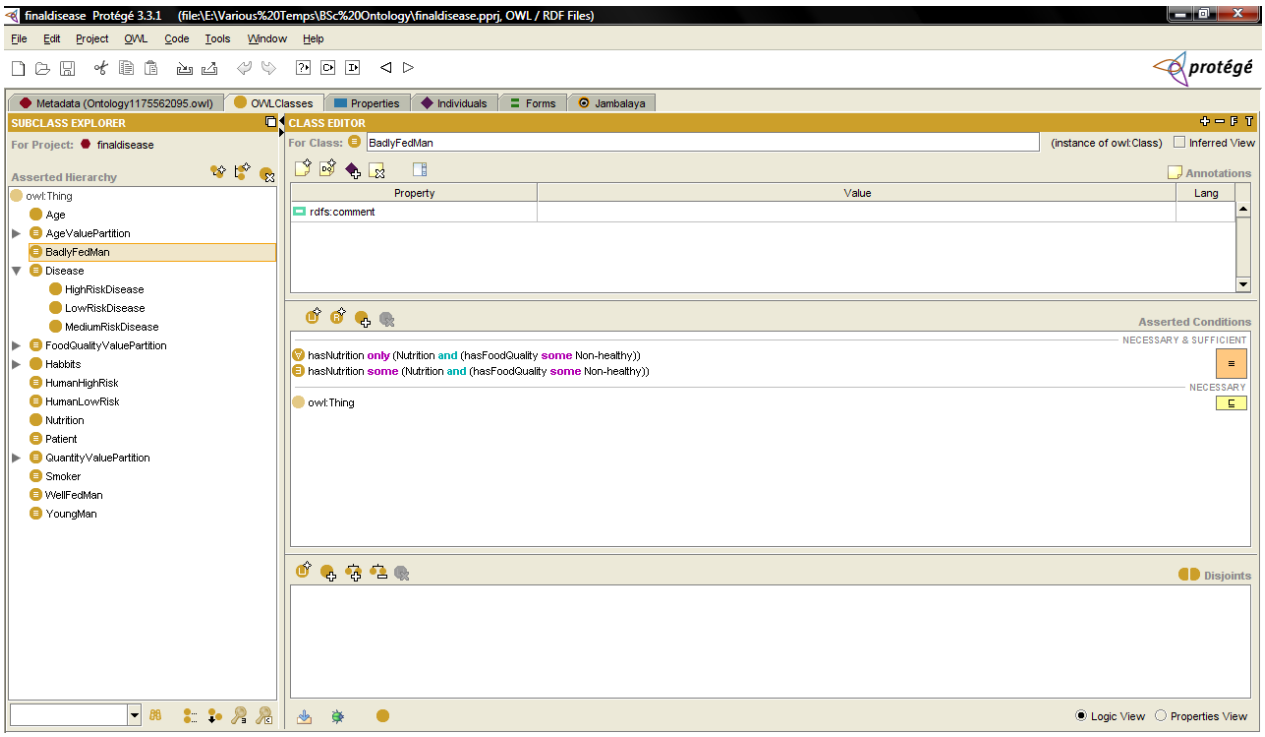
6.1. Protégé

Η ανάπτυξη της οντολογίας της εφαρμογής έγινε με τη χρήση του εργαλείου Protégé, το οποίο αποτελεί έναν ανοιχτού λογισμικού (open source) επεξεργαστή οντολογιών και ένα framework επεξεργασίας βάσεων γνώσης και αναπτύχθηκε στο πανεπιστήμιο Stanford. Πιο συγκεκριμένα χρησιμοποιήθηκε ο επεξεργαστής οντολογιών *Protégé-OWL*, ο οποίος παρέχεται ως επέκταση του εργαλείου. Το Protégé προσφέρει μια σειρά από εργαλεία και δομές που διευκολύνουν τη δημιουργία, την οπτικοποίηση και την επεξεργασία των οντολογιών σε περισσότερα από ένα πρότυπα, όπως RDF(S), OWL και XML Schema. Το Protégé έχει αναπτυχθεί σε Java και προσφέρει τη δυνατότητα εγκατάστασης διάφορων διαθέσιμων plugins για την περαιτέρω επεξεργασία και χρήση των οντολογιών που διαχειρίζεται. Επίσης δίνεται η δυνατότητα ανάπτυξης νέων plugins μέσω του Java-based API (Application Programming Interface) που διατίθεται. Το συγκεκριμένο API χρησιμοποιήθηκε στην εφαρμογή που αναπτύχθηκε για την κατηγοριοποίηση ενός νέου στιγμιοτύπου ασθενή.
Ο επεξεργαστής Protégé-OWL που χρησιμοποιήθηκε κάνει εφικτή τη δημιουργία OWL οντολογιών. Σύμφωνα με το W3C «μια OWL οντολογία περιέχει περιγραφές κλάσσεων, ιδιότητες και στιγμιότυπα αυτών. Η σημασιολογία της OWL, δεδομένης μιας οντολογίας γραμμένης στη γλώσσα, προσδιορίζει τις λογικές συνέπειες, τα γεγονότα δηλαδή που δεν είναι ρητά ορισμένα στην οντολογία, αλλά προκύπτουν σημασιολογικά.»
Ο Protégé-OWL editor (εικόνα 4) προσφέρει δυνατότητες:

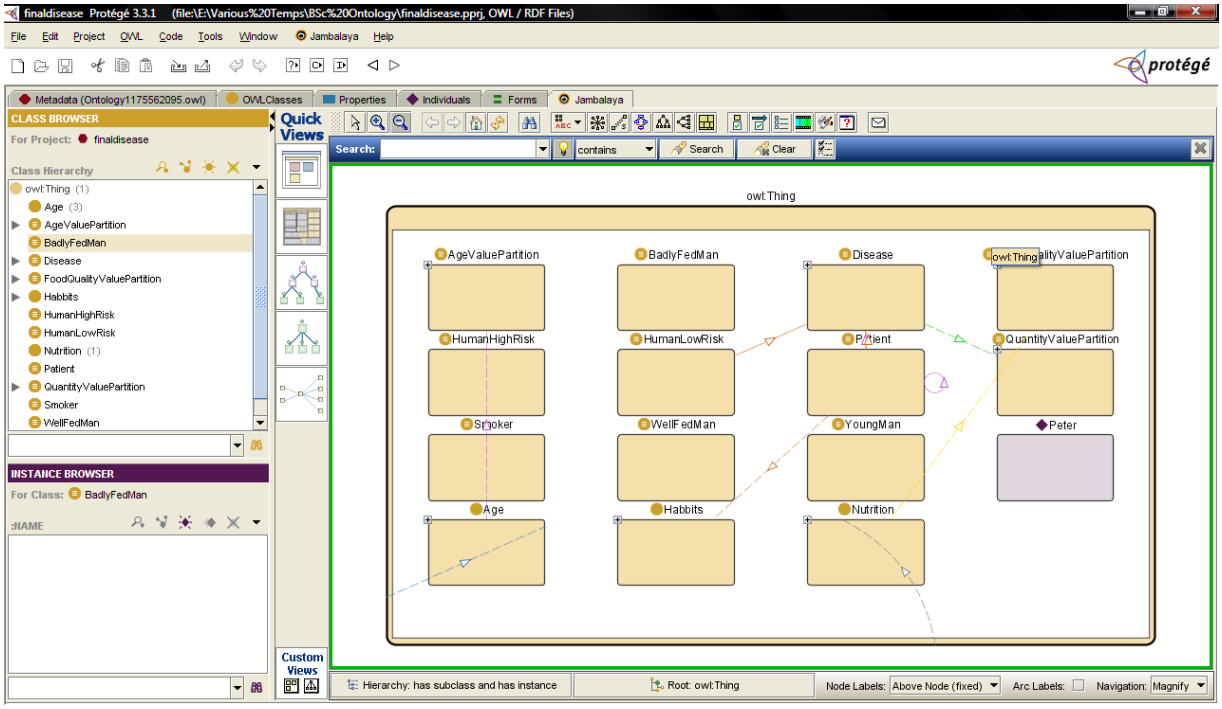
- Φόρτωσης και αποθήκευσης OWL και RDF οντολογιών
- Επεξεργασίας και οπτικής αναπαράστασης κλάσσεων και ιδιοτήτων
- Χρήσης OWL εκφράσεων για τον ορισμό λογικών χαρακτηριστικών κλάσσεων
- Επικοινωνίας με reasoners που λειτουργούν ως ταξινομητές για description logics

Μια άλλη χρήσιμη επέκταση του Protégé είναι η *Jambalaya* (εικόνα 5), η οποία αναπτύχθηκε στο πανεπιστήμιο της Victoria και επιτρέπει την οπτικοποίηση όλης της

βάσης γνώσης. Είναι βασισμένη σε SHriMP (**S**imple **H**ierarchical **M**ulti-**P**erspective), το οποίο αποτελεί μία τεχνική οπτικοποίησης, σχεδιασμένη για την υποστήριξη εμφάνισης και εξερεύνησης σύνθετων χώρων πληροφορίας, και στη βιβλιοθήκη Piccolo, η οποία αναπτύχθηκε από το πανεπιστήμιο του Maryland. Με την Jambalaya ο χρήστης μπορεί να εργαστεί σε έναν ιεραρχικό επεξεργαστή οντολογιών, ο οποίος επιτρέπει την εμφάνιση και επεξεργασία των ήδη υπαρχόντων δεδομένων. Η τεχνική SHriMP έχει τη δυνατότητα να κάνει zoom και animations της οντολογίας δίνοντας καλύτερη αίσθηση ως προς τις συσχετίσεις των κλάσεων και των ιδιοτήτων τους.



Εικόνα 9: 1ο Screenshot του Protégé



Εικόνα 10: 2ο Screenshot του Protégé

6.2. Reasoners Pellet/RACER

6.2.1. Pellet

Ο Pellet [14] είναι ένας ανοιχτού λογισμικού OWL DL reasoner, γραμμένος σε Java. Βασίζεται στους ταμπλώ αλγορίθμους για Description Logics (DL) και υποστηρίζει ολόκληρη την εκφραστικότητα της OWL DL, η οποία χρησιμοποιήθηκε για το σύστημα, περιλαμβάνοντας μηχανισμούς συμπερασμού και για απαριθμητές κλάσεις. Ο Pellet ήταν ο πρώτος reasoner που υποστήριζε ολόκληρη την OWL-DL, για παράδειγμα την DL λογική SHOIN(D) και από την 1.1. έκδοσή του υποστηρίζει όλα τα χαρακτηριστικά της OWL 1.1 (όπως τη DL λογική SROIQ(D)), εκτός από τους n-αδικούς τύπους δεδομένων. Περιλαμβάνει επίσης τεχνικές βελτιστοποίησης που έχουν προταθεί στη σχετική με DL βιβλιογραφία, όπως επίσης και πολλές καινοτόμες βελτιστοποιήσεις σχετικές με ιεραρχικό συμπερασμό και απαντήσεις επερωτήσεων με ζεύξεις [13]. Η ανάπτυξη λογισμικού βασισμένου στον Pellet είναι εφικτή μέσω των APIs (Application Programming Interfaces) που παρέχει για τα εργαλεία επεξεργασίας RDF/OWL Jena και Manchester OWL-API.

6.2.2. RACER(Pro)

Ο RACER (Renamed Abox and Concept Expression Reasoner) αναπτύχθηκε το 1997 από τα πανεπιστήμια Concordia University και Harburg και είναι ένας δεύτερος reasoned που χρησιμοποιήθηκε για λόγους πληρότητας. Ο Racer χειρίζεται μεγάλα Aboxes [16] σε συνδιασμό με μεγάλα και εκφραστικά Tboxes [17]. Παρέχει βελτιστοποιημένες υπηρεσίες συμπερασμού για σύνθετες εφαρμογές βασισμένες σε οντολογική αναπαράσταση γνώσης. Ο Racer, σύμφωνα με τους κατασκευαστές του, παρέχει πολύ περισσότερα στοιχεία από την OWL, υποστηρίζοντας κανόνες, συμπερασμό περιορισμών και απαντήσεις σε εκφραστικές επερωτήσεις για παράδειγμα με το συνακτικό της SPARQL. Η γλώσσα επερωτήσεων που χρησιμοποιεί είναι η nRQL (new Racer Query Language). Τέλος, η ακεραιότητα των δεδομένων εξασφαλίζεται μέσω του AllegroGraph, μια πολύ γρήγορη και σταθερή μηχανή αποθήκευσης δισεκατομμυρίων τριπλετών.

ΚΕΦΑΛΑΙΟ 7

ΣΥΝΟΨΗ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ

Συνοψίζοντας,υπάρχουν περισσότεροι από ένας τρόποι αναπαράστασης της πληροφορίας πλαισίου. Ανάλογα με τη φύση της, επιλέχθηκε είτε οντολογική είτε μέσω ασαφών συνόλων αναπαράσταση. Ειδικότερα, κύριος σκοπός της εργασίας αυτής ήταν η ανάπτυξη μιας εφαρμογής, η οποία βάσει του ιστορικού ενός ασθενή και μετρήσεων από αισθητήρες, να μπορεί να αποφανθεί σχετικά με την επικινδυνότητα της κατάστασής του.

Το ιατρικό ιστορικό ενός ασθενή μοντελοποιείται κατηγοριοποιώντας το προφίλ του σε μια ή περισσότερες κλάσεις μιας κατάλληλα διαμορφωμένης οντολογίας. Η οντολογία διαμορφώνεται με τη βοήθεια των ειδικών βάσει προηγούμενης εμπειρίας, και περιλαμβάνει στοιχεία από τις διατροφικές συνήθειες, προηγούμενες επεμβάσεις, ασθένειες, χρόνιες παθήσεις, συχνότητα άθλησης, καπνίσματος κτλ. του ασθενή.

Οι μετρήσεις από τους αισθητήρες δίνουν πληροφορία σχετική με την τρέχουσα κατάσταση του ασθενή, καταγράφοντας ανά πάσα στιγμή τους παλμούς, τη θερμοκρασία, το σήμα ECG, την πίεση κ.ά. και τροφοδοτώντας το σύστημα με τις αριθμητικές τιμές που προκύπτουν. Η πληροφορία που συλλέγεται από τους αισθητήρες αυτούς εμπεριέχει ασάφεια, η οποία μπορεί να χρησιμοποιηθεί, για να βγουν συμπεράσματα μέσω διαδικασιών ασαφούς συλλογιστικής.

Ο συνδυασμός των δύο προσεγγίσεων, της λογικής περιγραφών και της ασαφούς λογικής, δίνει ενδιαφέροντα αποτελέσματα ως προς την ανάπτυξη συστημάτων διάχυτου υπολογισμού στο χώρο της υγείας, μιας και η διαδικασία λήψης αποφάσεων ενισχύεται σημαντικά. Η φύση της πληροφορίας στο συγκεκριμένο πλαίσιο επιτρέπει και κάποιες φορές επιβάλλει την χρήση ασαφών μεταβλητών για την περιγραφή της. Για παράδειγμα, είναι γενικώς ανεπιθύμητο η θερμοκρασία ενός ασθενούς να είναι «*μεγάλη*». Είναι όμως δύσκολο, ακόμα και για τους ειδικούς του ιατρικού κλάδου, να καθοριστούν με ακρίβεια και ασφάλεια οι τιμές για τις οποίες η θερμοκρασία ενός ασθενούς είναι «*μεγάλη*». Αντίθετα, η ασαφής προσέγγιση προσφέρει τη δυνατότητα

καλύτερης εκμετάλλευσης της ιατρικής γνώσης και της ενσωμάτωσής της σε συστήματα διάχυτου υπολογισμού, αφού η γνώση αυτή βασίζεται σε μεγάλο βαθμό στην εμπειρία και στο ιστορικό και δεν είναι πάντα εφικτό και ασφαλές η μοντελοποίησή της να γίνει μέσω της κλασσικής λογικής. Επίσης, η κλασσική προσέγγιση δεν είναι δυνατόν να εγκαταληφθεί, αφού υπάρχουν μεταβλητές του πλαισίου των οποίων η φύση δεν εισάγει ασάφεια, όπως για παράδειγμα η ηλικία ενός ασθενούς, ή ο αριθμός των επεμβάσεων στις οποίες έχει υποβληθεί.

Τέλος, η εκμετάλλευση των μεγάλων σε ποσότητα ιατρικών δεδομένων για την αυτόματη παραγωγή ασαφών κανόνων και συνόλων, θα ενισχύσει ακόμα περισσότερο την ορθότητα της μοντελοποίησης της γνώσης που οι ειδικοί εκμεταλλεύονται με σκοπό να παράγουν διαγνώσεις και θεραπείες για κάθε περίπτωση ασθενούς.

ΠΑΡΑΡΤΗΜΑ

Για την ανάπτυξη της εφαρμογής έγινε χρήση του Java Fuzzy Engine, του Edward S. Sazonov. Δημιουργήθηκε η κλάση MyFuzzyClass, η οποία έχει ως μέλη της μεθόδους δημιουργίας και διαχείρισης ασαφών συνόλων, των αντίστοιχων ασαφών κανόνων, καθώς και την αποτίμηση αυτών για συγκεκριμένες τιμές εισόδου των ασαφών μεταβλητών που εμπλέκονται. Η κλάση αυτή υλοποιεί τη λειτουργικότητα του υπολογισμού του βαθμού επικινδυνότητας, βάσει των τιμών των αισθητήρων.

```
public class MyFuzzyClass {

	private Involvement temperature;
	private Involvement hematocrit;
	private Involvement danger;
	private Sex sex;
	private FuzzyEngine fuzzyEngine;
	private double dangerResult;
	private String[] rules;
	/**
	 * @param args
	 */
	public void resetFengine()
	{
		fuzzyEngine.reset();
	}
	public MyFuzzyClass(Sex s) {
		System.out.println("Contructing a fuzzy evaluator...");
		try {
			fuzzyEngine = new FuzzyEngine();
			createCurves();
			createRules();
			registerVariables();
			dangerResult=-1;
sex=s;
		}
		catch (Exception e)
		{
			e.printStackTrace();
```

```
		}
	}

	public void setTemperatureValue(double value) {
		temperature.setInputValue(value);
	}

	public void setHematocritValue(double value) {
		hematocrit.setInputValue(value);
	}

	public double getDangerResultValue() {
		return dangerResult;
	}

	public void evaluateAll()
	{
		try {
			FuzzyBlockOfRules fuzzyBlock = new FuzzyBlockOfRules(rules);
			fuzzyBlock.setFuzzyEngine(fuzzyEngine);
			fuzzyBlock.parseBlock();
			fuzzyBlock.evaluateBlock();
			String result = fuzzyBlock.evaluateBlockText();
			//System.out.println(result);
			dangerResult = danger.defuzzify();
			//System.out.println(dangerResult);
		} catch (Exception e) {
			e.printStackTrace();
		}
	}


	private void createCurves()	{
		temperature = new Involvement("temperature");
		temperature.add(Involvement.LOW, 32.0, 32.0, 35.0, 36.5);
		temperature.add(Involvement.MEDIUM,35.5, 36.5, 37.0, 38.0);
		temperature.add(Involvement.HIGH, 37.0, 38.5, 44.0, 44.0);

		hematocrit = new Involvement("hematocrit");
		if (s==Sex.MALE) {
```

```
                        hematocrit.add(Involvement.MEDIUM, 30.0, 40.7, 50.3, 60.0);
                        hematocrit.add(Involvement.LOW, 20.0, 20.0, 30.0, 40.0);
                        hematocrit.add(Involvement.HIGH, 50.0, 60.0, 70.0, 70.0);
                }
                else {
                        hematocrit.add(Involvement.MEDIUM, 26.0, 36.1, 44.3, 54.3);
                        hematocrit.add(Involvement.LOW, 21.0, 21.0, 26.0, 36.0);
                        hematocrit.add(Involvement.HIGH, 44.3, 54.3, 60.0, 60.0);
                }

                danger = new Involvement("danger");
                danger.add(Involvement.LOW, 0.0, 0.0, 30.0, 40.0);
                danger.add(Involvement.MEDIUM, 30.0, 40.0, 60.0, 70.0);
                danger.add(Involvement.HIGH, 60.0, 70.0, 100.0, 100.0);
}

        private void registerVariables() {
                fuzzyEngine.register(temperature);
                fuzzyEngine.register(danger);
                fuzzyEngine.register(hematocrit);
        }

        private void createRules() {

                rules = new String[4];
                rules[0]="if temperature is " +Involvement.MEDIUM+ " and hematocrit is "
+Involvement.MEDIUM+
                " then danger is "+Involvement.LOW;
                rules[1]="if temperature is not " +Involvement.MEDIUM+ " and hematocrit is not "
+Involvement.MEDIUM+
                " then danger is "+Involvement.HIGH;
                rules[2]="if temperature is not " +Involvement.MEDIUM+ " and hematocrit is "
+Involvement.MEDIUM+
                " then danger is "+Involvement.MEDIUM;
                rules[3]="if temperature is " +Involvement.MEDIUM+ " and hematocrit is not "
+Involvement.MEDIUM+
                " then danger is "+Involvement.MEDIUM;
        }
}
```

Για την ταξινόμηση ενός ασθενή σε κλάση χαμηλού ή υψηλού κινδύνου, δημιουργήθηκε η κλάση OntoProcesser, η οποία κάνοντας χρήση του Protégé Java API (Application Program Interface) αποχτά πρόσβαση στην οντολογία του συστήματος, δημιουργεί μια νέα κλάση βάσει των δεδομένων εσόδου, και την ταξινομεί μέσω του reasoner του συστήματος.

```
public class OntoProcesser {

        private static String PELLET="http://localhost:8081";
        private static String RACER="http://localhost:8080";

        private String owlPath;
        private OWLModel owlModel;
        private Collection<?> classes;
        private Collection<?> properties;
        private int classesAdded;

        private HashMap<String, OWLProperty> propertiesMap;
        private HashMap<String, OWLNamedClass> classesMap;

        private OWLProperty hasAge;
        private OWLProperty hasAgeLevel;
        private OWLProperty hasHabbit;
        private OWLProperty hasSmokingLevel;
        private OWLProperty hasNutrition;
        private OWLProperty hasFoodQuality;

        private OWLNamedClass old;
        private OWLNamedClass young;
        private OWLNamedClass medium;
        private OWLNamedClass healthy;
        private OWLNamedClass nonhealthy;
        private OWLNamedClass high;
        private OWLNamedClass low;
        private OWLNamedClass age;
        private OWLNamedClass smoking;
        private OWLNamedClass nutrition;
```

```
	public OntoProcesser(String path)	{
		classesAdded=0;
		owlPath=path;
		loadOntology();
	}

	private void loadOntology()	{
		try {
			//load the ontology
			owlModel = ProtegeOWL.createJenaOWLModelFromInputStream(
					new FileInputStream(new File(owlPath)));

			classes = owlModel.getUserDefinedOWLNamedClasses();
			classesMap = new HashMap<String, OWLNamedClass>();
			for (Iterator<?> it=classes.iterator(); it.hasNext();) {
				OWLNamedClass cls = (OWLNamedClass) it.next();
				classesMap.put(cls.getBrowserText(), cls);
			}

			properties = owlModel.getUserDefinedOWLProperties();
			propertiesMap = new HashMap<String, OWLProperty>();
			for (Iterator<?> it=properties.iterator(); it.hasNext();) {
				OWLProperty prt = (OWLProperty) it.next();
				propertiesMap.put(prt.getBrowserText(), prt);
			}

			setDesiredClasses();
			setDesiredProperties();

		}
		catch(Exception e) {
	   e.printStackTrace();
	}
	}

	public void printAllConcepts()	{
		//print all user defined concepts

		//take an Iterator in order to list all concepts in the collection object
		for (Iterator it = classes.iterator(); it.hasNext();) {
```

```
                    //catch a concept and then print its name
                    OWLNamedClass cls = (OWLNamedClass) it.next();
                    System.out.println("Defined Concept is "+cls.getBrowserText().toString());
                }

                int i=0;
                for (Iterator it = properties.iterator(); it.hasNext();) {

                        i++;
                        //catch a concept and then print its name
                    OWLProperty prt = (OWLProperty) it.next();
                        System.out.println("Property "+i+" is "+prt.getBrowserText().toString());
                    }
        }

        public void printFathers(String className) {
                //Now let's get the father(s) of the concept: className
                //first I load the concept
                OWLNamedClass namedClass = owlModel.getOWLNamedClass(className);

                //secondly I iterate all fathers before the reasoning!
                Collection fathers = namedClass.getSuperclasses(true);
                for (Iterator it = fathers.iterator(); it.hasNext();) {

                        RDFResource cls = (RDFResource) it.next();
                        //I want only concepts, thus, I type casting to OWLNamedClass...
                        if(cls.canAs(OWLNamedClass.class)){
                                OWLNamedClass fatherClass =
(OWLNamedClass)cls.as(OWLNamedClass.class);
                                System.out.println("Father of " + className + " is " +
fatherClass.getBrowserText().toString());
                        }
                }
        }

        private Collection getInferredFathers(String className) {

                Collection inferredSuperClasses = null;
                try {
```

```
                //Now I reason the ontology and I will take the inferred fathers from
                //the className concept. Hence, you will see that we take want we want!
                ReasonerManager reasonerManager = ReasonerManager.getInstance();
                ProtegeOWLReasoner reasoner = reasonerManager.getReasoner(owlModel);

// Set the reasoner URL and test the connection
reasoner.setURL(PELLET);

if(reasoner.isConnected()) {

    DIGReasonerIdentity reasonerIdentity = reasoner.getIdentity();
   System.out.println("Connected to " + reasonerIdentity.getName());

   // Get the className OWLNamedClass from the OWLModel
   OWLNamedClass namedClass = owlModel.getOWLNamedClass(className);
   if(namedClass != null) {
      System.out.println("Classifying taxonomy...");
      reasoner.classifyTaxonomy(null);
      System.out.println("...Classified taxonomy!");

      // System.out.println("Inferred superclasses of "+className+":");
       inferredSuperClasses = namedClass.getInferredSuperclasses();


       /*for(Iterator it = inferredSuperClasses.iterator(); it.hasNext();) {
          OWLNamedClass curClass = (OWLNamedClass) it.next();

          System.out.println("Inferred Father of "+className+" is "+curClass.getName());
       } */
       //THE END
   }
   else {
       System.out.println("Could not find "+className);
   }
}
else {
   System.out.println("Reasoner not connected!");
}

    }
```

```
                catch(Exception e) {
            e.printStackTrace();
        }
                return inferredSuperClasses;

            }


            public String createPrimitiveClass(SmokingFrequency sf, NutritionQuality nq, Age a)
            {
                classesAdded++;
                String className="tempClass."+classesAdded;
                 OWLNamedClass newClass = owlModel.createOWLNamedClass(className);

                if (sf==SmokingFrequency.HIGH)
                        addSmokingFreq(newClass,high);
                else if (sf==SmokingFrequency.MEDIUM)
                        addSmokingFreq(newClass,medium);
                else
                        addSmokingFreq(newClass,low);

                 if (a==Age.OLD)
                        addAgeLevel(newClass, old);
                 else
                        addAgeLevel(newClass, young);

                 if (nq==NutritionQuality.HEALTHY)
                        addFoodQuality(newClass, healthy);
                 else
                        addFoodQuality(newClass, nonhealthy);

                 return className;
            }

            private void addSmokingFreq (OWLNamedClass newClass, OWLNamedClass smokingLevel)
            {
                 // (hasSmokingLevel some high)
                 OWLSomeValuesFrom smokingSomeValuesFrom =
owlModel.createOWLSomeValuesFrom(hasSmokingLevel, smokingLevel);
```

```
                // Smoking AND (hasSmokingLevel some high)
                OWLIntersectionClass smokingIntersectionClass =
owlModel.createOWLIntersectionClass();
        smokingIntersectionClass.addOperand(smokingSomeValuesFrom);
        smokingIntersectionClass.addOperand(smoking);

        // hasHabbit some (Smoking AND (hasSmokingLevel some high))
        OWLSomeValuesFrom habbitSomeValuesFrom =
owlModel.createOWLSomeValuesFrom(hasHabbit, smokingIntersectionClass);
        newClass.addSuperclass(habbitSomeValuesFrom);

        // hasHabbit only (Smoking AND (hasSmokingLevel some high))
        OWLAllValuesFrom habbitAllValuesFrom = owlModel.createOWLAllValuesFrom(hasHabbit,
smokingIntersectionClass);
                newClass.addSuperclass(habbitAllValuesFrom);
    }


    private void addFoodQuality (OWLNamedClass newClass,OWLNamedClass foodQuality)
    {
                // (hasFoodQuality some Healthy)
                OWLSomeValuesFrom qualitySomeValuesFrom =
owlModel.createOWLSomeValuesFrom(hasFoodQuality, foodQuality);

                // Nutrition AND (hasFoodQuality some Healthy)
                OWLIntersectionClass intersectionClass = owlModel.createOWLIntersectionClass();
                intersectionClass.addOperand(qualitySomeValuesFrom);
        intersectionClass.addOperand(nutrition);

        // hasNutrition some (Nutrition AND (hasFoodQuality some Healthy))
        OWLSomeValuesFrom nutritionSomeValuesFrom =
owlModel.createOWLSomeValuesFrom(hasNutrition, intersectionClass);
        newClass.addSuperclass(nutritionSomeValuesFrom);

        // hasNutrition only (Nutrition AND (hasFoodQuality some Healthy))
        OWLAllValuesFrom nutritionAllValuesFrom =
owlModel.createOWLAllValuesFrom(hasNutrition, intersectionClass);
                newClass.addSuperclass(nutritionAllValuesFrom);
    }
```

```
	private void addAgeLevel (OWLNamedClass newClass,OWLNamedClass ageLevel)
	{
		// (hasAgeLevel some Old)
		OWLSomeValuesFrom ageLevelSomeValuesFrom =
owlModel.createOWLSomeValuesFrom(hasAgeLevel, ageLevel);

		// Age AND (hasAgeLevel some Old)
		OWLIntersectionClass intersectionClass = owlModel.createOWLIntersectionClass();
		intersectionClass.addOperand(ageLevelSomeValuesFrom);
	    intersectionClass.addOperand(this.age);

	    // hasNutrition some  (Age AND (hasAgeLevel some Old))
	    OWLSomeValuesFrom ageSomeValuesFrom =
owlModel.createOWLSomeValuesFrom(hasAge, intersectionClass);
	    newClass.addSuperclass(ageSomeValuesFrom);

	    // hasNutrition only (Nutrition AND (hasFoodQuality some Healthy))
	    OWLAllValuesFrom ageAllValuesFrom = owlModel.createOWLAllValuesFrom(hasAge,
intersectionClass);
		newClass.addSuperclass(ageAllValuesFrom);
	}

	private void setDesiredClasses()
	{
		young = classesMap.get("Young");
		old = classesMap.get("Old");
		high = classesMap.get("High");
		medium = classesMap.get("Medium");
		low = classesMap.get("Low");
		nonhealthy = classesMap.get("Non-healthy");
		healthy = classesMap.get("Healthy");
		age = classesMap.get("Age");
		smoking = classesMap.get("Smoking");
		nutrition = classesMap.get("Nutrition");
	}

	private void setDesiredProperties() //TODO throw exception if any is not listed
	{
		hasAge = propertiesMap.get("hasAge");
		hasAgeLevel = propertiesMap.get("hasAgeLevel");
```

```
            hasHabbit = propertiesMap.get("hasHabbit");
            hasSmokingLevel = propertiesMap.get("hasSmokingLevel");
            hasNutrition = propertiesMap.get("hasNutrition");
            hasFoodQuality = propertiesMap.get("hasFoodQuality");
    }

    public boolean isHighRiskHuman(String className)
    {
            return this.getInferredFathers(className).contains(classesMap.get("HumanHighRisk"));
    }

    public boolean isLowRiskHuman(String className)
    {
            return this.getInferredFathers(className).contains(classesMap.get("HumanLowRisk"));
    }

                                                    }
```

ΑΝΑΦΟΡΕΣ

1. Ιωάννης Βλαχάβας, Πέτρος Κεφαλάς, Νικόλαος Βασιλειάδης, Ιωάννης Ρεφανίδης, Φώτιος Κόκκορας, Ηλίας Σακελλαρίου, *«Τεχνητή Νοημοσύνη»,* 1η έκδοση, 2002.
2. Ji He, Ah-Hwee Tan, Chew-Lim Tan, Sam-Yuan Sung, *«On Quantitative Evaluation of Clustering Systems»,* Information Retrieval and Clustering, 2002.
3. Anind K. Day, Gregory D. Abowd, *«Towards a Better Understanding of Context and Context-Awareness».*
4. M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe, *«A Practical Guide to Building OWL Ontologies Using the Protégé –OWL Plugin»*, 2004.
5. A. K. Jain, M. N. Murty, and P. J. Flynn, *«Data Clustering: A Review»*, ACM Computing Surveys, Vol. 31, 1999.
6. J. A. Hartigan and M. A. Wong, *«A K-Means Clustering Algorithm»*, 1979.
7. J. M. Mendel, *«Fuzzy Logic Systems for Engineering: A Tutorial»*, 1995.
8. J. Jantzen, *«Tutorial on Fuzzy Logic»*, 1998.
9. Google, www.google.com
10. Wikipedia, www.wikipedia.com
11. Πάνος Πασσιάς, *«Fuzzy Reasoning about Situational Contexts»*, October, 2006.
12. M. Weiser, *«The Computer for the 21st Century»*, Sci. Amer., Sept., 1991.
13. E. Sirin , B. Parsia, B. C. Grau , A. Kalyanpur, Y. Katz, «*Pellet: A Practical OWL-DL Reasoner»*, 2007.
14. Pellet, http://pellet.owldl.com/
15. Racer, http://www.sts.tu-harburg.de/~r.f.moeller/racer/
16. Abox, http://en.wikipedia.org/wiki/Abox
17. Tbox, http://en.wikipedia.org/wiki/Tbox