ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΠΟΛΥΜΕΣΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΥΣΤΗΜΑ ΣΥΛΛΟΓΙΣΜΟΥ ΓΙΑ ΧΡΟΝΙΚΕΣ ΟΝΤΟΛΟΓΙΕΣ OWL

Σπουδαστές:
ΚΕΜΕΣΙΔΗΣ ΘΟΔΩΡΗΣ ΑΜ 955
ΠΑΙΔΑΡΑΚΗΣ ΕΥΤΥΧΗΣ-ΙΩΑΝΝΗΣ ΑΜ 1116

Επιβλέπων καθηγητής:
δρ ΠΑΠΑΔΑΚΗΣ ΝΙΚΟΣ

ΗΡΑΚΛΕΙΟ, ΟΚΤΩΒΡΙΟΣ 2008

Ευχαριστίες….

Θα ήθελα να ευχαριστήσω καταρχάς τον δρ Παπαδάκη Νίκο πρώτον για την ανάθεση της εργασίας , την ηθική και ουσιαστική στήριξη , την συνεργασία και φυσικά την πορεία που μας έδειξε που έπρεπε να ακολουθήσουμε ώστε να τελειώσουμε με επιτυχία την εργασία αυτή!

Επίσης δεν θα παρέλειπα να ευχαριστήσω θερμά τους γονείς που με στηρίζουν πάντα,έτσι και τώρα με στήριξαν όπως μπορούσαν ακόμα και σ'αυτή τη δύσκολη προσπάθεια που έκανα τόσο καιρό.

Τέλος ευχαριστώ όλους όσους συνέβαλαν στην ολοκλήρωση αυτής της σημαντικής για μένα και για την πορεία της επαγγελματικής μου σταδιοδρομίας εργασίας!

ΠΕΡΙΕΧΟΜΕΝΑ

Σελίδες

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία παρουσιάζουμε το PROTON, ένα πρόγραμμα συλλογιστικής στην Prolog για τη διαχείριση της χρονικής πληροφορίας στις οντολογίες OWL. Πρώτα παρουσιάζομαι το πρόβλημα πλαισίου για το οποίο θα υλοποιήσομε μια λύση στην πτυχιακή μας εργασία και κάνουμε μια ανασκόπηση των προτεινομένων λύση. Ειδικότερα, μετατρέπουμε τις οντολογίες σε ισότιμες τριπλέτες με το εργαλείο SWI-Prolog και έπειτα τις μετατρέπουμε σε κατηγορήματα στην Prolog. Σχεδιάσαμε ένα πρόγραμμα συλλογιστικής το οποίο έχει τη δυνατότητα χειρισμού των σχέσεων ανάμεσα σε χρονικές στιγμές ή χρονικά διαστήματα καθώς και του υπολογισμού συμπερασμάτων που βασίζονται σε αυτές τις σχέσεις.

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

Ο Σημασιολογικός Ιστός δεν αποτελεί ξεχωριστό Ιστό αλλά διεύρυνση του σημερινού, στα πλαίσια του οποίου η πληροφορία λαμβάνει μια σαφώς ορισμένη σημασία, στοχεύοντας έτσι στη βελτίωση της επικοινωνίας μεταξύ ανθρώπων και υπολογιστών. Ήδη δρομολογούνται τα πρώτα βήματα για την δημιουργία του Σημασιολογικού Ιστού εντός της δομής του ήδη υπάρχοντος Ιστού . Στο κοντινό μέλλον, οι σχεδιαστές θα χρησιμοποιήσουν αυτή τη νέα δυνατότητα λειτουργιών καθώς οι μηχανές θα είναι ικανές να επεξεργαστούν και να "καταλάβουν" τα δεδομένα τα οποία προς το παρόν απλά εμφανίζουν.

Ήδη εφαρμόζονται δύο σημαντικές τεχνολογίες για την ανάπτυξη του Σημασιολογικού Ιστού: οι XML και RDF. Η XML επιτρέπει σε όλους να δημιουργήσουν τις δικές τους κρυφές και φανερές ετικέτες ή να σχολιάσουν ιστοσελίδες. Τα σενάρια (scripts) μπορούν να χρησιμοποιήσουν αυτές τις ετικέτες με περίπλοκους τρόπους, αλλά οι σχεδιαστές των σεναρίων πρέπει να ξέρουν για ποιο σκοπό κάθε δημιουργός ιστοσελίδας χρησιμοποιεί την συγκεκριμένη ετικέτα.

Η κύρια λειτουργία του σημασιολογικού Ιστού είναι η έκφραση της σημασίας των δεδομένων. Για να γίνει αυτό εφικτό χρειάζονται αρκετά επίπεδα. Η πληροφορία δομείται σε επίπεδα:

1) Το επίπεδο XML το οποίο αντιπροσωπεύει τα δεδομένα

2) Το επίπεδο RDF το οποίο αντιπροσωπεύει τη σημασία των δεδομένων

3) Το επίπεδο της Οντολογίας, το οποίο αντιπροσωπεύει την τυπική κοινή συναίνεση σχετικά με την σημασία των δεδομένων (OWL)

4) Το λογικό επίπεδο, το οποίο δίνει την δυνατότητα για έξυπνη λογική συλλογιστική με σημαίνοντα δεδομένα

Σε αυτή την έκθεση θα περιγράψουμε ένα σύστημα το οποίο εκτελείται στο τέταρτο επίπεδο (Λογικό Επίπεδο) σε περίπτωση που τα δεδομένα υπάρχουν στα πρώτα τρία επίπεδα κατά τη χρονική διάρκεια. Το Λογικό (τέταρτο) επίπεδο βασίζεται σε πληροφορία που αντλείται από το Οντολογικό επίπεδο. Στην παρούσα εργασία, η έμφαση δίνεται στη χρονική πληροφορία και στη μεταβαλλόμενη πληροφορία σε σχέση με το χρόνο. Η ενασχόληση με το χρόνο και με τον τρόπο με τον οποίο η πληροφορία μεταβάλλεται προοδευτικά σε σχέση με το χρόνο αποτελεί ένα πολύ γνωστό πρόβλημα στην αναπαράσταση της γνώσης. Η αναπαράσταση των οντολογικών γλωσσών όπως οι OWL και RDF βασίζονται τυπικά σε δυαδικές σχέσεις (πχ. το να είσαι υπάλληλος μιας εταιρείας) οι οποίες δεν μπορούν να αναπαρασταθούν χρονικά. Συνεπώς η οντολογική πληροφορία του λογικού συλλογισμού που αναπαρίσταται στις OWL ή RDF δεν μπορεί να λάβει υπόψη της τη χρονική πληροφορία. Ας υποθέσουμε για παράδειγμα ότι έχουμε μια εταιρική μετοχή που έχει μια

τιμή. Το κύριο πρόβλημα είναι η αναπαράσταση του χρόνου και της πληροφορίας που μεταβάλλεται σε συνάρτηση με το χρόνο. Για παράδειγμα, πρέπει να αναπαραστήσουμε το χρόνο και την αξία της μετοχής σε όλες τις χρονικές στιγμές. Κατόπιν, πρέπει να αναπαραστήσουμε τις μεταβολές της πληροφορίας σε σχέση με τον χρόνο και να συλλογιστούμε λογικά αυτές τις μεταβολές. Για παράδειγμα, θα πρέπει να πουλήσουμε τις μετοχές αν η αξία τους αυξηθεί κατά 10% ή περισσότερο κατά τη διάρκεια των τελευταίων ωρών. Αυτή η εργασία αναφέρεται και στα δύο αυτά προβλήματα και προτείνει λύσεις. Το πρώτο πρόβλημα επιλύεται στο οντολογικό επίπεδο χρησιμοποιώντας την γνωστή

«διαχρονιστική» προσέγγιση[1].[12]. Το δεύτερο πρόβλημα επιλύεται στο λογικό επίπεδο χρησιμοποιώντας το πρόγραμμα συλλογιστικής.

Στην παρούσα εργασία παρουσιάζουμε το PROTON, ένα σύστημα λογικού συλλογισμού χρονικών οντολογιών στην OWL. Η αρχιτεκτονική του συστήματος παρουσιάζεται στο σχήμα 2. Το σύστημα δέχεται ως είσοδο μια χρονική οντολογία σε owl και την μετατρέπει σε τριπλέτα (υποκείμενο, κατηγόρημα, αντικείμενο) χρησιμοποιώντας το SWI-Prolog[2]. Έπειτα τα τρία μέρη μετατρέπονται σε κατηγορηματικές προτάσεις στην Prolog. Ο χρονικός λογικός συλλογισμός εφαρμόζεται πάνω σε αυτές τις πληροφορίες βασιζόμενος σε χρονικό λογισμό της κατάστασης. Ο προτεινόμενος λογικός συλλογισμός υποστηρίζει τις παρακάτω λειτουργίες:

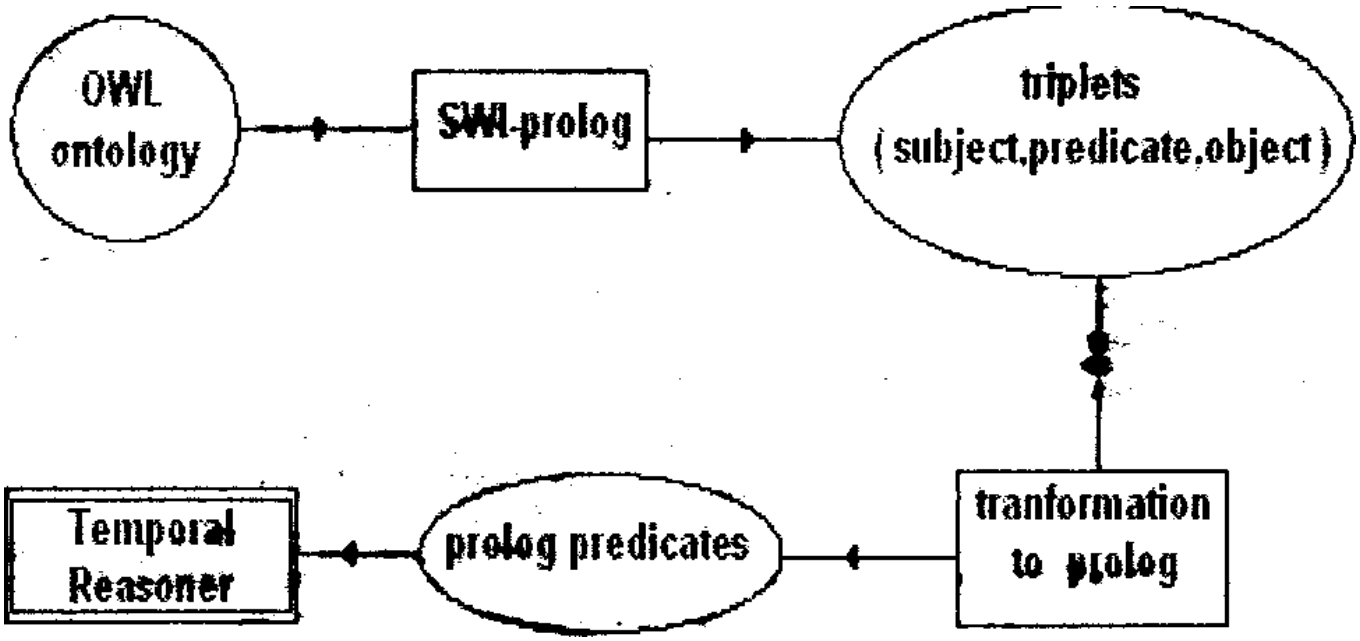
1)Χειρισμός του χρόνου για τις χρονικές στιγμές καθώς και για τα χρονικά διαστήματα και τις μεταξύ τους σχέσεις (συμπεριλαμβανομένων και των σχέσεων του Allen [2]).

2) Καθορισμός των χρονικών ιδιοτήτων για την πληροφορία σε κάθε χρονική στιγμή (πχ. η εύρεση της αξίας μιας μετοχής στο τέλος της ημέρας).

3) Αξιολόγηση των κανόνων για να εξαχθούν συμπεράσματα (πχ. πότε θα πρέπει να πουληθεί η μετοχή).

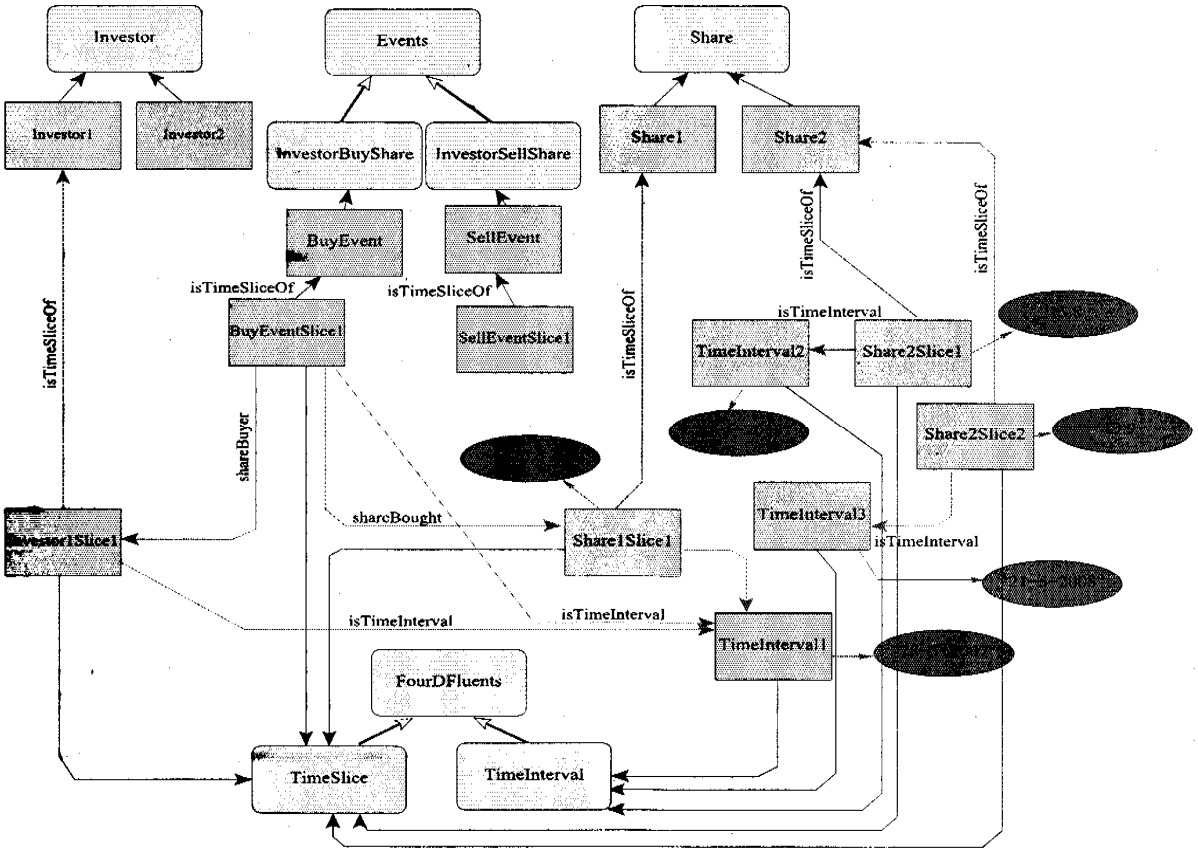
4)Προσομοίωση της εξέλιξης του κόσμου. Ειδικότερα: Εύρεση όλων των μεταβολών που συμβαίνουν με την πάροδο του χρόνου και αξιολόγηση των κανόνων (σε κάθε χρονικό σημείο) που παράγουν το συμπέρασμα.

[1] ΣτΜ: perdurandist approach
[2] http://www.swi-prolog.org/



Σχήμα 1. Η Αρχιτεκτονική του Συστήματος

Στη συνέχεια, εκμεταλλευόμαστε τους μηχανισμούς που προσφέρει η Prolog για να εφαρμόσουμε το πρόγραμμα συλλογιστικής: στη Prolog, πρέπει μόνο να καθορίσουμε τι πρέπει να γίνει αλλά όχι και τον τρόπο εφαρμογής όπως συμβαίνει σε λειτουργικές γλώσσες όπως η Java ή η C++.



Σχήμα 2. Το Παράδειγμα Διαχείρισης Μετοχών

Η παρούσα έκθεση διαρθρώνεται ως εξής: Στο κεφάλαιο 2 θα περιγράψουμε πώς αναπαρίσταται ο χρόνος στο οντολογικό επίπεδο στην OWL. Επίσης παρουσιάζουμε τις τυποποιήσεις για το συλλογισμό με τις προτεινόμενες δικές μας προεκτάσεις για την υποστήριξη του χρονικού συλλογισμού. Το κεφάλαιο 3 περιγράφει τα κύρια συστατικά μέρη του προτεινόμενου προγράμματος συλλογιστικής και την υποστηριζόμενη λειτουργικότητα συνοδευόμενη από συμπεράσματα και άλλα θέματα για περεταίρω έρευνα στο Κεφάλαιο 4.

ΚΕΦΑΛΑΙΟ 2

Ανασκόπηση προτεινόμενων λύσεων στο πρόβλημα Πλαισίου

2.1 Το πρόβλημα του πλαισίου μέσα από τη Μονότονη Λογική

Η πιο εύκολη προσέγγιση του προβλήματος του πλαισίου προέρχεται από την μονοτονική λογική (monotonic 10gic). Ως μονοτονική χαρακτηρίζεται η κατηγορηματική λογική (predicate 10gic), δηλαδή η λογική πρώτης τάξης η οποία αντιμετωπίζει το πρόβλημα της μη προσπελασιμότητας των στοιχείων των γεγονότων της προτασιακής λογικής. Η κατηγορηματική λογική η οποία επεκτείνει την προτασιακή λογική εισάγοντας όρους (terms), κατηγορήματα (predicates), και ποσοδείκτες (quantifiers), αντιστοιχίζει με τη φυσική γλώσσα, την ικανοποιητική έκφραση ποσοτικοποίησης των εννοιών με τους κατάλληλους ποσοδείκτες και την ικανότητά της να συλλάβει τη γενικότητα, αλλά δεν δίνει τη δυνατότητα έκφρασης ασαφών τιμών, όπως επίσης και αναθεώρησης του συμπεράσματος που έχει προστεθεί στη γνώση, αν αποδειχθεί λανθασμένο. Δηλαδή, με απουσία συγκεκριμένων στοιχείων δεν μπορεί να εξαχθεί κάποιο γενικό συμπέρασμα. [2]

Η μονοτονική προσέγγιση στο πρόβλημα του πλαισίου προτάθηκε αρχικά από τον John McCarthy [5]. Όπως αναφέρεται από τον Παπαδάκη [3], υπάρχουν ρητά δηλωμένα αξιώματα από τα οποία μπορούν να βγουν συμπεράσματα, των οποίων

τα κατηγορήματα (predicates) και οι συναρτήσεις επηρεάζονται από κάθε πιθανή ενέργεια (action). Αυτά τα αξιώματα χωρίζονται σε δύο κατηγορίες:

1. Τα αξιώματα ενέργειας (action axioms), τα οποία δηλώνουν ποιες προτάσεις επηρεάζονται, όταν συμβαίνει κάποια ενέργεια.
2. Τα αξιώματα πλαισίου (frame axioms), τα οποία δηλώνουν τα πράγματα τα οποία παραμένουν ανεπηρέαστα, όταν συμβαίνει κάποια ενέργεια.

Όλα τα αξιώματα του πλαισίου (frame axioms) πρέπει να καθοριστούν ρητά για κάθε μία ξεχωριστή ενέργεια. Επιπλέον, κάθε κατηγόρημα πρέπει να καθοριστεί αν αλλάζει μετά από την εκτέλεση μιας ενέργειας. Τότε μπορούμε να συμπεράνουμε ότι ο αριθμός των αξιωμάτων του πλαισίου θα είναι: ε χ r, όπου ε ο αριθμός των ενεργειών και κ ο αριθμός των κατηγορημάτων.
Προσπαθώντας να δώσουμε έναν ορισμό για την πολυπλοκότητα θα λέγαμε ότι τόσο πιο πολύπλοκο είναι ένα πρόβλημα, όσο περισσότερα γεγονότα εξετάζονται, διότι για να καθοριστεί τι παραμένει αληθές μετά την εκτέλεση μιας ενέργειας, όλα τα κατηγορήματα πρέπει να εξεταστούν ένα προς ένα. Για κάθε ενέργεια, δηλαδή, πρέπει να γραφούν πάρα πολλοί κανόνες. Επομένως, όσο μεγαλύτερο είναι το πλαίσιο το προβλήματος, τόσο μεγαλύτερη είναι η πολυπλοκότητά του, με αποτέλεσμα τεράστια δυσκολία στην κωδικοποίηση του προβλήματος. Με βάση όλα τα παραπάνω συνεπάγεται ότι κάθε κατηγόρημα (ιδιότητα του κόσμου) εκφράζει με σαφήνεια ότι μια συγκεκριμένη ενέργεια δεν το επηρεάζει σε δύο συνεχόμενες καταστάσεις. Έτσι, χρειάζεται ένα σύνολο αξιωμάτων για κάθε ενέργεια. Δηλαδή, για κάθε αξίωμα το οποίο περιγράφει πως αλλάζει ο κόσμος υπάρχει ένα άλλο σύνολο αξιωμάτων που περιγράφει πως δεν αλλάζει.

2.2 Από τη μονοτονική στη μη-μονοτονική προσέγγιση

Όταν συμβαίνει μία ενέργεια, ένα μικρό σύνολο κατηγορημάτων-αντικειμένων επηρεάζεται, δηλαδή, το μεγαλύτερο μέρος των κατηγορημάτων παραμένουν αμετάβλητα. Η default προσέγγιση υποδεικνύει ότι είναι απαραίτητο να οριστούν αξιώματα μόνο για εκείνα τα κατηγορήματα τα οποία αλλάζουν ως αποτέλεσμα κάποιας συγκεκριμένης ενέργειας, ενώ για τα υπόλοιπα ισχύει ένα δεδομένο αξίωμα (default axiom) το οποίο δηλώνει ότι: κάθε αντικείμενο το οποίο δεν έχει δηλωθεί ότι θα αλλάξει μετά από μία ενέργεια, παραμένει το ίδιο. [3]

Το δεδομένο αξίωμα (default axiom) μπορεί να έχει τη μορφή:

Ps : Pdo(a,s) / Pdo(a,s)

αν δηλαδή το κατηγόρημα P είναι αληθές στην κατάσταση s και μετά την εκτέλεση της ενέργειας a, τότε μπορούμε να συμπεράνουμε το p.

Η Συγκεκριμένη μέθοδος πάσχει από μεγάλη πολυπλοκότητα, επειδή το δεδομένο αξίωμα (default axiom) θα αποτιμηθεί για όλα τα κατηγορήματα μετά από την εκτέλεση κάθε μίας ενέργειας, έτσι ώστε να εξακριβωθεί ποια από αυτά τα κατηγορήματα παραμένουν αληθή [3].

Όπως προαναφέρθηκε, η κατηγορηματική λογική δεν παρέχει την δυνατότητα έκφρασης ασαφών τιμών και παραγωγής γενικών συμπερασμάτων. Δηλαδή, δεν επιτρέπει την αναθεώρηση κάποιου συμπεράσματος, το οποίο αποδεικνύεται λανθασμένο. Η λογική που επιτρέπει την αναθεώρηση συμπερασμάτων λέγεται μημονότονη λογική. Επιπλέον σε μια μη-μονότονη λογική προσθέτοντας νέα αξιώματα, είναι δυνατό να μειωθεί ο αριθμός των συμπερασμάτων, αφαιρώντας αυτά που αποδεικνύονται εσφαλμένα μετά την προσθήκη. Έτσι έχουμε τη μη-μονότονη λογική που είναι κατάλληλη για καταστάσεις στις οποίες η γνώση

μεταβάλλεται, λόγω μεταβολών που συμβαίνουν στον κόσμο, για καταστάσεις, για τις οποίες δεν έχουμε πλήρη γνώση (incomplete knowledge) ή η γνώση δημιουργείται κατά τη διάρκεια της εκτέλεσης ενεργειών, για της οποίες δεν είμαστε βέβαιοι για την αναγκαιότητα ή την ορθότητά τους και τέλος για καταστάσεις στις οποίες το σύστημα χρησιμοποιεί υποθέσεις (assumptions) στα πλαίσια της στρατηγικής επίλυσης καταστάσεων.

Περίπτωση της μη-μονότονης συλλογιστικής είναι και η συλλογιστική εύλογων υποθέσεων (default reasoning) που περιγράφτηκε παραπάνω. Χρησιμοποιείται στην περίπτωση που ένα γεγονός συνάγεται από ένα δοσμένο σύνολο γεγονότων, τουλάχιστον ενός, διότι έτσι συμβαίνει συνήθως και διότι δεν υπάρχει ένδειξη για το αντίθετο. Υπάρχει βέβαια και η δυνατότητα κατάργησης αυτού του γεγονότος, εφόσον αποδειχθεί η μη ορθότητά του.

Θέλοντας να δώσουμε έναν ορισμό για τη μη-μονοτονικότητα θα μπορούσαμε να πούμε ότι το φαινόμενο της αναίρεσης κάποιων συμπερασμάτων, τα οποία έχουν προκύψει από προηγούμενους υπολογισμούς, λέγεται μη-μονοτονικότητα (υοτιmonotonicity). Εάν μια δήλωση φ ακολουθείται από ένα σύνολο από premises Μ ς Μ', φ δεν ακολουθείται απαραίτητα από το Μ' [1].

Έτσι η default logic είναι ίσως η κατάλληλη μέθοδος για μη-μονοτονικό λογισμό (non-monotonic reasoning), λόγω της απλότητας της έννοιας της εύλογης υπόθεσης (default) [1].

2.3 Λογισμός Καταστάσεων (SITUATION CALCULUS)

Το πρόβλημα του πλαισίου εμφανίστηκε στα πλαίσια του Λογισμού Καταστάσεων (situation calculus), για να αναπαρασταθεί ένας κόσμος ο οποίος υφίστατο αλλαγές. Όπως αναφέρεται από τον Παπαδάκη [3], ο Λογισμός Καταστάσεων προτάθηκε από τον John Mc Carthy το 1969 για τη λύση του προβλήματος του πλαισίου, παρέχοντας έναν καθορισμό για τις ενέργειες και τις επιπτώσεις (έμμεσες και άμεσες).

Γνωρίζουμε ότι κάθε ενέργεια πρέπει να ορίζει με σαφήνεια ολόκληρη την κατάσταση που θα έχει ο κόσμος μετά την εκτέλεσή της. Για κάθε ενέργεια πρέπει να καθοριστούν τα αξιώματα του πλαισίου. Έτσι ο Λογισμός Καταστάσεων (Situation Calculus) είναι μια γλώσσα δεύτερης τάξης, η οποία σχεδιάστηκε για την αναπαράσταση των αλλαγών, οι οποίες συμβαίνουν στον κόσμο που μας ενδιαφέρει.

Όλες οι αλλαγές που συμβαίνουν σε έναν κόσμο είναι αποτέλεσμα εκτέλεσης κάποιων ενεργειών. Μια πιθανή εξέλιξη του κόσμου είναι μια πιθανή ακολουθία ενεργειών και αναπαρίστανται από όρους πρώτης τάξης, που καλείται κατάσταση (situation).

Στο λογισμό καταστάσεων χρησιμοποιούμε φυσικά καταστάσεις, και μάλιστα η πρώτη κατάσταση από την οποία ξεκινούν τα πάντα ονομάζεται αρχική. Έτσι, λοιπόν, η Αρχική Κατάσταση συμβολίζεται με So και θεωρείται η κατάσταση στην οποία καμιά ενέργεια δεν έχει συμβεί. Επιπλέον, ορίζεται η δυαδική συνάρτηση do, έτσι ώστε με do(a,s) δηλώνεται η κατάσταση- αποτέλεσμα, μετά την εκτέλεση της ενέργειας a στην κατάσταση s. Μια ενέργεια μπορεί να είναι on(x, Y), το οποίο σημαίνει ότι το αντικείμενο χ, βρίσκεται πάνω στο αντικείμενο Υ.

Σε ένα δυναμικό κόσμο πολλά πράγματα αλλάζουν, έτσι οι τιμές των κατηγορημάτων και των συναρτήσεων σε ένα δυναμικό κόσμο διαφέρουν από μια κατάσταση σε μια άλλη. Τα κατηγορήματα και οι συναρτήσεις των οποίων οι τιμές αλλάζουν από μια κατάσταση σε μια άλλη καλούνται αντιστοίχως κατηγορήματα με εύρος (fluent predicates) και συναρτησιακό εύρος (functional fluents). Ακολουθώντας τη σύμβαση του Παπαδάκη ως fluent θα αναφέρονται τόσο τα κατηγορήματα με εύρος, όσο και το συναρτησιακό εύρος.

Άρα πραγματικά η σχέση που προκύπτει μεταξύ καταστάσεων είναι η «μικρότερη ή ίση».[3]

Για την εκτέλεση κάποιας ενέργειας πρέπει να υπάρχουν κάποιες συνθήκες, οι οποίες καλούνται προαπαιτούμενες συνθήκες ή προϋποθέσεις (preconditions). Η δυαδική συνάρτηση Poss δηλώνει αν μια προϋπόθεση (precondition) ισχύει. Δηλαδή, αν ένα κατηγόρημα Poss(a,s) είναι αληθές, αυτό σημαίνει ότι η ενέργεια a μπορεί να εκτελεσθεί στην κατάσταση s.

Αυτό μπορούμε να το δούμε στο παρακάτω παράδειγμα. Υποθέτουμε ότι ένας σκύλος σ θέλει να δαγκώσει bite το κόκκαλό του k. Για να μπορέσει να συμβεί αυτό, δεν πρέπει να έχει τίποτα μέσα στο στόμα του (Π.χ. φαγητό) ¬haveInMouth , να υπάρχει ένα κόκκαλο το οποίο να μην είναι μεγάλο για ¬big(k) για να μπορέσει να το πιάσει με το στόμα του ο σκύλος και να βρίσκεται δίπλα του (nextTo) το κόκκαλο. Έτσι έχουμε:

Poss(bite(σ, k), ε)⊃ [(∀,z 'dz) -,haveInMouth(σ, X, Z)] Λ ¬big(k) Λ nextTo(σ, k, s) Παρατηρούμε ότι το κατηγόρημα Poss εκφράζει τους απαραίτητους κανόνες που πρέπει να ισχύουν, για να είναι δυνατή η εκτέλεση της ενέργειας του δαγκώματος (bite).

Αιτιολογικοί νόμοι είναι αυτοί που μας δείχνουν πως οι ενέργειας επηρεάζονται από τις τιμές των fluents, και ονομάζονται κανόνες ή αποτελέσματα.

Στη συνέχεια θα παρατεθούν κάποια αξιώματα, έτσι ώστε να γίνει κατανοητό ότι η μικρότερη σχέση που υπάρχει μεταξύ δύο καταστάσεων είναι <= (μικρότερη ή ίση).

Ένα βασικό αξίωμα στο Λογισμό Καταστάσεων είναι το αξίωμα της επαγωγής (axiom ofinduction), το οποίο ικανοποιεί το ότι η αρχική κατάσταση ανήκει στο σύνολο των καταστάσεων και η δυαδική συνάρτηση do(a,s) ανήκει στο σύνολο των καταστάσεων, εάν η κατάσταση s ανήκει και αυτή στο σύνολο των καταστάσεων και η ενέργεια a ανήκει στο σύνολο των ενεργειών:

$(\forall P).P(So) \Lambda (\forall a, s) [P(s) \supset P(do(a, s))] \supset (\forall s) P(s).$

Επιπλέον, μπορούν να συνταχθούν οι καταστάσεις σε κάποια σειρά:

$(\forall s.s 7\neq So) So < s$ και $\neg s < So$

όπου $s < s'$ σημαίνει ότι η κατάσταση s' είναι επόμενη κατάσταση της s.

Εφόσον το P είναι μια δυαδική σχέση μεταξύ των καταστάσεων, συμπεραίνουμε ότι:

$(\forall s, s').s \neq s' = (\forall P). \{[(\forall a, S_1).Poss(a, S_1) \supset P(S_1, do(a, S_1))]$

$\Lambda (\forall a, S_1. S_2).POSS(a, S_2) \Lambda P(S_1, S_2) \supset P(S_1, do(a, S_2))]\} \supset P(s, s').$

Effect Axioms

Η δυναμική του Κόσμου καθορίζεται λεπτομερώς από τα αξιώματα επιρροής (effect αχίοms):Ετσι, καθορίζουν τα αποτελέσματα μιας δοσμένης ενέργειας σε μια αληθή τιμή ενός δοσμένου fluent. Επομένως, μπορούν να αναφερθούν και ως οι αιτιολογικοί κανόνες του κόσμου. Διακρίνονται σε θετικά (positive) και αρνητικά (negative) αξιώματα επιρροής. Ως παράδειγμα έχουμε:

Positive effect axioms

fragile(x, s) ⊃ broken (X, do(drop(r,x), s)

Negative effect axioms

¬broken(x, do(repair(r,x), s).

2.4 Η λύση μέσα από τα αξιώματα επιρροής (effect axioms)

Το πλήθος των αναμενόμενων αξιωμάτων του πλαισίου (frame axioms) σε συνδυασμό με την παρατήρηση αυτών ταυτόχρονα, ώθησε τον Pednault [7] το 1989 να τα συστηματικοποιήσει σε μία μέθοδο, η οποία θα τα εξασφαλίζει μέσα από τα αξιώματα επιρροής (effect axioms), θετικά (positive) και αρνητικά (negative).

Έχοντας ένα σύνολο θετικών και αρνητικών αξιωμάτων του πλαισίου, ένα για κάθε ενέργεια, για κάθε fluent F(x, s) αξιωματικά έχουμε:

$(E^{+})F(X, Y, s) \supset F(x, do(A(y), s))$

και $(E^{-})_F(x,y, s) \supset \neg F(x, do(A(y),))$.

Το $(E^{+})_F(x, y, s)$ δείχνει τις προϋποθέσεις (preconditions) που πρέπει να ισχύουν, έτσι ώστε, εφόσον η ενέργεια A(Y) εκτελεσθεί, το fluent F να γίνει αληθές στην επόμενη κατάσταση της ενέργειας A.

Αντίστοιχα, το $(E-)_F(X, Y, ε)$ δείχνει τις προϋποθέσεις (preconditions) που πρέπει να ισχύουν, έτσι ώστε, εφόσον η ενέργεια A(y) εκτελεσθεί, το fluent F να γίνει ψευδές στην επόμενη κατάσταση της ενέργειας A.

Υποθέτουμε ότι ισχύει το F(x, s) και το ¬F(x, do(A(y), s)). Αυτό σημαίνει ότι το F είναι αληθές στην κατάσταση s και μετά την εκτέλεση της ενέργειας A γίνεται ψευδές. Ο μόνος τρόπος το F να γίνει ψευδές είναι αν και μόνο εάν το (E-)F(X, Y, s) ήταν αληθές. Γράφοντάς το με αξιώματα έχουμε:

$F(x, s) \Lambda \neg F(x, do(A(y), s)) \supset (E^{-})F(x,y, s)$

Το οποίο είναι ισοδύναμο με:

$F(x, s) \Lambda \neg (E^{-})_F(X, Y, s) \supset {}_F(x, do(A(y), s))$.

Από παρόμοια επιχειρήματα προκύπτει το:

$\neg F(x, s) \Lambda \neg (E^{-})_F(x,y, s) \supset \neg F(x, do(A(y), s))$.

Οι δύο τελευταίες προτάσεις παίζουν ακριβώς το ρόλο των αξιωμάτων του πλαισίου, αφού έχουν ακριβώς τη συντακτική μορφή των θετικών και αρνητικών αξιωμάτων του πλαισίου και ακολουθούν λογική επιχειρηματολογία.

Παρακάτω παραθέτω ένα παράδειγμα, έτσι ώστε να γίνει εμφανής η παραπάνω θέση του Pednault. Κάποιες σύγχρονες βρύσες με το πάτημα (push) ενός κουμπιού αφήνουν το νερό να τρέξει (flowWater) και με επανάληψη του πατήματος του κουμπιού σταματούν τη ροή του νερού. Περιγράφοντας προκύπτουν οι προτάσεις:

$\neg flowWater(x, s) \supset \subset flowWater(x, do(push(x), s))$

και $flowWater(x, s) \supset \neg flowWater(x, do(push(x), s))$,

από τα οποία μπορεί να προκύψουν τα ισοδύναμα:

$\neg flowWater(x, s) \Lambda y = \chi \supset flowWater(x, do(push(y), s))$

και $flowWater(x, s) \Lambda y = X \supset flowWater(x, do(push(y), s))$.

Έτσι, μπορούμε να φτάσουμε στο συμπέρασμα:

flowWater(x, s) Λ ¬flowWater(x, do(push(y), s))

⊃ flowWater(x, s) Λ y = X.

Από αυτό προκύπτει ένα θετικό αξίωμα του πλαισίου:

flowWater(x, s) Λ Y 7≠ χ ⊃ flowWater(x, do(push(y), s))

το οποίο σημαίνει ότι η ενέργεια του πατήματος του κουμπιού (push) δεν επηρεάζει το fluent jlowWater, εάν το Y είναι διαφορετικό του χ, δηλαδή, αν πρόκειται για κουμπί κάποιας άλλης βρύσης. Αντίστοιχα μπορούμε να γράψουμε και το αρνητικό αξίωμα του πλαισίου:

¬flowWater(x, s) Λ y 7 X ⊃ ¬flowWater(x, do(push(y), s)).

Ολοκληρώνοντας μπορούμε από τα αξιώματα επιρροής (effect axioms) να οδηγηθούμε στα αξιώματα του πλαισίου (frame axioms) μέσω ενός συστηματικοποιημένου τρόπου.

Αυτή η λύση του προβλήματος του πλαισίου απαιτεί τον καθορισμό 2 x A x F αξιωμάτων του πλαισίου(όπου A οι ενέργειας και F τα fluents).

Θα πρέπει επιπλέον να αναφερθεί ότι το κατηγόρημα Poss(a, s), το οποίο, όπως προαναφέρθηκε, εκφράζει τις προϋποθέσεις που πρέπει να ισχύουν, για να είναι δυνατή η εκτέλεση της ενέργειας α στην κατάσταση s, δεν είναι μέρος αυτού του συστηματικοποιημένου τρόπου, αλλά σε πολλές περιπτώσεις θα μπορούσε να προστεθεί.

2.5 Η λύση μέσα από καθολικά ποσοτικοποιημένες (universally quantified) ενέργειες

Η προσέγγιση του Hass[4] βασίζεται στην ίδια κεντρική ιδέα. Κάποιες ενέργειες μόνο επηρεάζουν το fluent F. Για να αλλάξει, δηλαδή, ένα fluent πρέπει κάποιες συγκεκριμένες ενέργειας να αλλάξουν, αλλιώς αν δεν συμβεί κάποια συγκεκριμένη ενέργεια, τα fluents παραμένουν αμετάβλητα. Έτσι τα αξιώματα του πλαισίου έχουν τις εξής μορφές:

$F(x, s) \Lambda F(x, do(a, s)) \supset a_F(X, a, s)$

$\neg F(x, s) \Lambda F(x, do(a, s)) \supset \beta_F(X, a, s)$.

όπου η ενέργεια (action) α είναι αναμφίβολα καθολικά ποσοτικοποιημένη (universally quantified). Στα παραπάνω αξιώματα του πλαισίου, εάν αλλάξει η τιμή ενός fluent P, τότε τα α-και βF παρέχουν εξαντλητική εξήγηση γι' αυτή την αλλαγή.

Τα αξιώματα αυτά μπορούν να γραφούν ισοδύναμα ως:

$\neg F(x, s) \Lambda aF(X, a, S) \supset \neg F(x, do(a, s))$

$\neg F(x, s) \Lambda -,\beta F(X, a, \varepsilon) \supset \neg F(x, do(a, s))$.

Οι δύο τελευταίες προτάσεις παίζουν ακριβώς το ρόλο των αξιωμάτων του πλαισίου, αφού έχουν ακριβώς τη συντακτική μορφή των θετικών και αρνητικών αξιωμάτων του πλαισίου και ακολουθούν λογική επιχειρηματολογία. Επιπλέον, η ενέργεια α είναι καθολικά ποσοτικοποιημένη (universally quantified). Οπότε ο αριθμός των αξιωμάτων που είναι αναγκαίος, ώστε να περιγραφούν όλες οι πιθανές αλλαγές στις τιμές των fluent που είναι αληθείς είναι ίσος με 2 χ F.

Παρόμοια με την προσέγγιση του Hass είναι και η προσέγγιση του Schubert [9] που βασίζεται σε αυτή του Hass και έχει μια ευνοϊκότερη προσέγγιση σε ότι ονομάζεται επεξηγηματικά κλειστά αξιώματα (explanation closure axioms),

επειδή δίνουν μια ολοκληρωμένη εξήγηση πως αλλάζουν τα fluents, για την αναπαράσταση των γνωστών αξιωμάτων του πλαισίου.

Παρακάτω παραθέτω ένα παράδειγμα, έτσι ώστε να γίνει εμφανής η παραπάνω θέση του Hass.

Υποθέτουμε το παράδειγμα που αναφέραμε με το σκύλο. Για να έχει το κόκκαλο στο στόμα του σε αυτή την κατάσταση και να μην το έχει στην κατάσταση που μας οδηγεί η ενέργεια a μετά την εκτέλεση της (do(a, s)), θεωρούμε ότι ο μόνος τρόπος να συμβεί αυτό είναι ο σκύλος είτε να έχει αφήσει το κόκκαλο κάτω (putdown), είτε να το έχει ρίξει (drop), είτε να το έχει φάει (eat). Επομένως, εφόσον τα haveInMouth(σ, k, s) και
¬haveInMouth(σ, k, do(a,s)) είναι αληθή, έχουμε:

$\supset$ haveInMouth(σ, k, s) Λ ¬haveInMouth(σ, k, do(a,s))

$\supset$ a = PutDown(σ, k) v a = drop(σ, k) v a = eat(σ, k)

Πρέπει να επισημανθεί ότι η παραπάνω πρόταση είναι καθολικά ποσοτικοποιημένη (universally quantified) πάνω στην ενέργεια a. Με λογικές ισοδυναμίες προκύπτει το παρακάτω αξίωμα πλαισίου:

haveInMouth(σ, k, s) Λ a ≠ PutDown(σ, k) Λ a 1≠ drop(σ, k) Λ a ≠ eat(σ, k)

$\supset$ haveInMouth(σ, k, do(a,s))

Το παραπάνω αξίωμα μας λέει ότι το fluent haveInMouth επηρεάζεται μόνο από τις ενέργειες putDown, drop και eat, και από καμία άλλη. Έτσι, φτάσαμε στο αξίωμα του πλαισίου το οποίο περιγράφει τι δεν θα αλλάξει.

2.6 Η απλή λύση του Reiter

Το 1991 ο Reiter[8] πρότεινε μια απλή λύση στο πρόβλημα του πλαισίου και πολύ αποδοτική μάλιστα. Ο Reiter προσεγγίζει το πρόβλημα του πλαισίου προτείνοντας συντακτικούς μετασχηματισμούς αξιωμάτων, οι οποίοι συμπληρώνουν τη θεωρία ενεργειών (action theory) σε μια συγκεκριμένη συντακτική μορφή. Οι μετασχηματισμοί αυτοί περιγράφουν τις προϋποθέσεις και τα άμεσα αποτελέσματα για κάθε ενέργεια σε μια συγκεκριμένη περιγραφή για κάθε fluent σε κάθε κατάσταση, το οποίο προκύπτει από την εκτέλεση μιας ενέργειας Η προσέγγισή του βασίζεται στις προσεγγίσεις των Hass [4], Pednault [7] και Schubert [9].

Έστω ότι έχουμε, για κάθε fluent F, αξιώματα της μορφής (γενική μορφή αξιωμάτων επιρροής θετική και αρνητική(generaI effect axioms)):

$Poss(a, s) \wedge (\gamma^{+})F(a, s) \supset F(do(a, s))$

και $Poss(a, s) \wedge (\gamma^{-})F(a, s) \supset \neg F(do(a, s))$.

Αυτά τα δύο αξιώματα μπορούν να παραχθούν για κάθε fluent F μέσω των κατάλληλων μετασχηματισμών, και για κατάλληλα ορισμένο κατηγόρημα Poss. Περιγράφουν όλες τις συνθήκες κάτω από τις οποίες η ενέργεια α μπορεί να θέσει το fluent f να είναι αληθές ή αντίστοιχα ψευδές. Με τον όρο συνθήκες εννοούνται οι προϋποθέσεις (Poss) και οι δυνατές τιμές που μπορεί να πάρει η ενέργεια α $((\gamma^{+})F(a, s))$, ώστε να ισχύει το fluent F. Αντίστοιχα, βέβαια, και για τις τιμές που μπορεί να πάρει η ενέργεια α , δηλαδή το $((\gamma^{-})F(a, s))$ να γίνει ψευδές, ώστε το fluent F να γίνει στην επόμενη κατάσταση της s ψευδές.

Έτσι, μπορούμε να καταλήξουμε συμπερασματικά, ότι για κάθε fluent F μπορούμε να έχουμε ένα αξίωμα της μορφής:

$$Poss(a, s) \supset [F(do(a, s)) = (\gamma^{+})_F(a, s) \vee F(s) \wedge (\gamma^{-})_F(a, s)].$$

Επιπλέον, λόγω των γενικών αξιωμάτων επιρροής (general effect axioms)) μπορούμε να διασφαλίσουμε ότι δεν υπάρχει περίπτωση να ισχύουν οι προϋποθέσεις για την ενέργεια a (Poss) και ταυτόχρονα να ισχύουν και οι δυνατές τιμές που θέτουν το fluent F αληθές και οι δυνατές τιμές που το θέτουν ψευδές. Δηλαδή, διασφαλίζεται ότι το ,∃(Poss(a, s) Λ $(\gamma^+)_F$(a, s) Λ $(\gamma^-)_F$(a, s)) είναι πάντα αληθές. Επομένως είναι αδύνατο να ισχύουν ταυτόχρονα το F(a, s) και το ,F(do(a, s)).

Η παραπάνω προσέγγιση θα γίνει καλύτερα κατανοητή μέσα από ένα παράδειγμα, όπως παρατίθεται από τον Reiter και επεξηγείται και από τον Παπαδάκη. Θεωρούμε ότι έχουμε ένα ρομπότ r, το οποίο κρατά (holding) ένα αντικείμενο X, που μπορεί να είναι και εύθραυστο (fragile), το οποίο μπορεί να το αφήσει (putDown) στο πάτωμα ή να του πέσει(droP). Υπάρχει και ένα fluent βόμβα (bomb), η οποία μπορεί να εκραγεί (explode).

Θεωρούμε ότι το αντικείμενο X το οποίο το ρομπότ το κρατάει στην κατάσταση s θα σπάσει στην επόμενη κατάσταση, αν το ρομπότ το ρίξει(drop) και ότι το αντικείμενο χ θα σπάσει εάν βρίσκεται δίπλα σε μια βόμβα, και στην επόμενη κατάσταση αυτή εκρήγνυται. Έτσι προκύπτουν τα παρακάτω δύο θετικά αξιώματα το πλαισίου:

fragile(x, s) ⊃ broken(x, do(drop(r, χ), s)

nextTo(b, χ, s) ⊃ broken(x, explode(b), s)

Γράφοντάς τα σε μία πρόταση, έτσι ώστε οι ενέργειες drop και explode να έχουν ως αποτέλεσμα το fluent broken, προκύπτει:

[(∃r, x){a = drop(r, χ) Λ fragile(x, s)}

v (∃b){ a= explode(b) Λ nextTo(b, χ, s)}]

⊃ broken(x, do(a, s)).

Έτσι περιγράψαμε ότι αν υπάρχει ρομπότ και ρίξει το αντικείμενο X και είναι εύθραυστο, ή υπάρχει μια βόμβα η οποία εκραγεί και βρίσκεται δίπλα στο αντικείμενο X, τότε θα σπάσει το αντικείμενο X.

Φυσικά, για να είναι δυνατή η εκτέλεση της ενέργειας a στην κατάσταση s, πρέπει να ισχύει το κατηγόρημα Poss(a, ε), δηλαδή, να ισχύουν οι προϋποθέσεις για την εκτέλεση της ενέργειας a στην παρούσα κατάσταση s. Επομένως έχουμε:

$$Poss(a, \varepsilon) \wedge [(\exists r, x)\{a = drop(r, x) \wedge fragile(x, S)\}$$
$$\vee (\exists b)\{a = explode(b) \wedge nextTo(b, x, s)\}]$$
$$\supset broken(x, do(a, s)).$$

Αυτό και στην ουσία είναι το θετικό αξίωμα του πλαισίου, το οποίο σε μια πιο συμπυκνωμένη και γενική μορφή θα μπορούσε να γραφτεί:

$$Poss(a, s) \wedge (\gamma+)_{broken}(a, S) \supset broken(x, do(a,s))$$

Αντίστοιχα, μπορεί να παραχθεί το αρνητικό αξίωμα του πλαισίου:

$$Poss(a, s) \wedge [(\exists r, x)\{a = repair(x, s)\}] \supset \neg broken(x, do(a,s)).$$

Και η γενική μορφή:

$$Poss(a, s) \wedge \vee [(\exists b)(\Upsilon-)broken(a,s) \supset \neg broken(x, do(a,s))$$

Οπότε μπορούμε να συμπεράνουμε ότι:

$$Poss(a, s) \supset [broken(do(a, s)) \equiv (\gamma^{+})_{broken}(a, s) \vee broken(s) \wedge \neg (\gamma^{-})_{broken}(a, s)].$$

Η συνέπεια και η διαύγεια της παραπάνω προσέγγισης έγκειται στην ποσοτικοποίηση των ενεργειών διαμέσου μετασχηματισμών και στη γενική μορφή της πληρότητας της υπόθεσης (Generalized Completeness Assumption), δηλαδή, στην περιγραφή όλων των συνθηκών κάτω από τις οποίες η ενέργεια a οδηγεί σε αληθές fluent στην επόμενη κατάσταση.

2.7 Σύγκριση των λύσεων

Μπορεί εύκολα να παρατηρηθεί ότι από τις λύσεις που παρουσιάστηκαν η λύση του Reiter στο πρόβλημα του πλαισίου απαιτεί τα λιγότερα αξιώματα για να περιγραφεί ένα πρόβλημα. Αυτά ανέρχονται στον αριθμό των A +F αξιωμάτων σε αντίθεση με το πλήθος των αξιωμάτων που απαιτούνται από τον Hass και είναι ίσα με 2 χ F και του Pednault που ανέρχονται σε 2 x A x F.

Αυτό αυτόματα κάνει και την προσέγγιση του Reiter ποιο απλή. Η πολυπλοκότητα που προκύπτει από την περιγραφή ενός προβλήματος λόγω του πλήθους των αξιωμάτων του πλαισίου είναι αρκετά μεγαλύτερη στις προσεγγίσεις των Pednault και Hass.

Ο Pednault για τον συστηματικό καθορισμό των αξιωμάτων του πλαισίου (frame axioms), για όλα τα ζεύγη fluent-ενέργεια από αξιώματα επιρροής (effect axioms), πρέπει να μετρηθεί το σύνολο των αξιωμάτων επιρροής (efect axioms), συμπεριλαμβανομένων και των ανενεργών.

Συγκεκριμένα θα πρέπει να μετρηθούν όλα εκείνα τα ζεύγη fluent-ενεργειών τα οποία δεν επηρεάζουν την τιμή αυτών των fluents, τα οποία είναι αληθή. Με άλλα λόγια θα πρέπει να απαριθμηθούν όλα τα αξιώματα του πλαισίου (frame axioms).

Επιπροσθέτως, ο αριθμός των αξιωμάτων του πλαισίου που πρέπει να καθοριστούν ανέρχεται σε 2 x A x P, όπου A ο αριθμός των ενεργειών και F ο αριθμός των fluents. Αξιοσημείωτο είναι το γεγονός ότι πολλά από αυτά παραμένουν ανενεργά, και έτσι αντιμετωπίζουμε πάλι τη δυσκολία του Προβλήματος του πλαισίου, ότι χρειάζονται, δηλαδή, πάρα πολλά αξιώματα του πλαισίου, για να μπορέσει να γίνει η περιγραφή.

Ο Pednault που χρησιμοποίησε την πληρότητα της υπόθεσης (completeneess assumption), για να αποδείξει τους συλλογισμούς του, αφήνει μια σημαντική απορία που είναι εάν μπορεί να αποτύχει η πληρότητα της υπόθεσης (completeneess assumption). Η απάντηση είναι σχεδόν αυτονόητη, μπορεί να συμβεί οποτεδήποτε υπάρχει ελλιπής γνώση για αυτό που επηρεάζει μια ενέργεια σε ένα fluent. Αυτό φαίνεται από ένα παράδειγμα που αναφέρει ο Reiter στο κεφάλαιο που περιγράφει την προσέγγιση του Pednault στο Πρόβλημα του Πλαισίου. Αν προσπαθήσουμε να σκεφτούμε τα αποτελέσματα του να τραβήξουμε την σκανδάλη από ένα πιθανώς γεμάτο (loaded) όπλο πάνω στο fluent loaded, μπορούμε εύκολα να παρατηρήσουμε ότι είναι ακαθόριστο αν το όπλο θα είναι γεμισμένο μετά την ενέργεια του να τραβήξουμε τη σκανδάλη (pulltrigger). Πράγμα που καθιστά ανενεργά τα θετικά (positive) και αρνητικά (negative) αξιώματα επιρροής (effect axioms).

Δηλαδή αντίστοιχα έχουμε:

false $\supset$ loaded(do(pulltrigger,s))

και false$\supset$ ¬loaded(do(pulltrigger,s))

Έτσι, σύμφωνα με την πληρότητα της υπόθεσης (completeneess assumption) προκύπτουν τα αντίστοιχα αξιώματα του πλαισίου, από τα οποία το πρώτο διαισθητικά και μόνο είναι λανθασμένο. Έτσι έχουμε:

loaded(s) $\supset$ loaded(do(pulltrigger,s))

και ¬loaded(s) $\supset$ ¬loaded(do(pultrigger,s))

Αν θέλαμε να διορθώσουμε τα δύο αξιώματα αποτελέσματος για αυτή τη συγκεκριμένη περίπτωση, έτσι ώστε να καθοριστούν το τράβηγμα της σκανδάλης (pulltriger) και το γέμισμα του όπλου (loaded), θα πρέπει να εισάγουμε και τον αριθμό σφαιρών (που περιέχονται) στο όπλο (containbullets(n,s) : n σφαίρες

περιέχονται στην κατάποση s), άλλα αυτός ο τρόπος διόρθωσης του προβλήματος είναι ξεκάθαρο ότι δεν μπορεί να λύσει γενικά όλες τις περιπτώσεις ασαφών ζευγών fluents-ενεργειών. Έτσι γι' αυτή την περίπτωση έχουμε την εξής λύση:

containbullets(n, s) Λ n ≥2 ⊃ loaded(do(pulltrigger, s))

και containbullets(n, s) Λ n < 2 ⊃ ¬loaded(do(pulltrigger, s)).

Όπου n θετικό ή μηδέν.

Στην Προσέγγιση του Hass παρατηρούμε ότι ο αριθμός αξιωμάτων που χρησιμοποιούνται για τη λύση του προβλήματος του πλαισίου είναι 2 x F, όπου F ο αριθμός των fluent. Αυτό είναι εφικτό λόγω των καθολικών ποσοδεικτών που χρησιμοποιεί. Σε σύγκριση με τη λύση του Pednault μπορούμε να πούμε ότι ακολουθούν την ίδια βασική ιδέα. Η αλλαγή ενός fluent έγκειται στην αλλαγή κάποιων (λίγων) συγκεκριμένων ενεργειών. Αν δεν αλλάξουν αυτές, δεν συμβαίνει καμία αλλαγή στα fluents. Η πολυπλοκότητά του είναι αυξημένη, αλλά όχι τόσο μεγάλη, όσο του Pednault.

Η Morgenstem[6] παρατηρεί κάποια αρνητικά στην προσέγγιση του Reiter και γράφει χαρακτηριστικά ότι η προσέγγιση του Reiter έχει ένα λογικό αριθμό απαιτούμενων αξιωμάτων και επιτρέπει να συμβούν "παράλληλα" γεγονότα. Παρ' όλα αυτά, υπάρχουν και μειονεκτήματα. Αφού επιτρέπει να συμβούν παράλληλες ενέργειες ,εφόσον αυτές είναι γνωστές, δεν μπορεί να διευθετήσει συστήματα τα οποία περιέχουν άγνωστες ενέργειες που δρουν παράλληλα. Στην προσέγγιση του Reiter σύμφωνα με την Morgenstem[6] υπάρχει η υπόθεση ότι τίποτε στον κόσμο δεν μπορεί να κριθεί σωστά, όταν δεν είναι όλα γνωστά. Έτσι, η γνώση του συστήματος για τα γεγονότα πρέπει να είναι ολοκληρωμένη.

Επιπλέον, υποστηρίζει ότι δεν είναι απαραίτητη μια τυπική της πλειοψηφούσας γνώμης λογική. Ένα δεύτερο μειονέκτημα είναι ότι χρειάζεται να προστεθούν πολύ ισχυρές υποθέσεις, για να γίνει μια ακριβή εξαγωγή συμπεράσματος [6]. Φυσικά, αυτή η άποψη επεκτείνεται εύκολα και στις προσεγγίσεις του Hass και του Pednault.

Δεν πρέπει όμως να ξεχνάμε ότι η προσέγγιση του Reiter απαιτεί μόνο F + A αξιώματα, όπου F fluents και A actions. Απαιτεί ,δηλαδή, ένα αξίωμα για κάθε fluent και ένα για κάθε ενέργεια (action). Επιπλέον, θεωρείται μια απλή λύση, διότι η πολυπλοκότητα που απαιτείται για την περιγραφή κάθε αξιώματος πλαισίου έχει μειωθεί δραστικά.

Όλα αυτά ίσως μπορούν να γίνουν πιο κατανοητά, αν περιγράψουμε το ίδιο παράδειγμα και με τις τρεις διαφορετικές προσεγγίσεις που παρουσιάστηκαν.

Υποθέτουμε ότι ένας άνθρωπος έχει την δυνατότητα να τραφεί, οι οποία περιγράφεται με το fluent eat, εφόσον παραγγείλει το φαγητό του (food), που περιγράφεται με την ενέργεια (action) order ή εφόσον αγοράσει το φαγητό του, το οποίο περιγράφεται με την ενέργεια (action) buy. Ο άνθρωπος παύει να τρέφεται, εφόσον το φαγητό χαλάσει, που περιγράφεται με την ενέργεια stale.

Η προσέγγιση του Pednault περιγράφει αυτό το τμήμα του κόσμου με 2xAxF αξιώματα του πλαισίου. Έχουμε 3 ενέργειες (Actions) και ένα fluent, επομένως περιγράφεται με 2χ3χΙ = 6 αξιώματα του πλαισίου.

Οπότε έχουμε το θετικό αξίωμα του πλαισίου:

$$eat(food, s) \wedge y \neq food \supset eat(food, do(order(y), s)), \qquad (1)$$

το οποίο σημαίνει ότι η ενέργεια του να παραγγελθεί το Υ το οποίο είναι διαφορετικό του food, order(y), δεν επηρεάζει το fluent eat(/Ood, s), όταν το Υ είναι διαφορετικό από το food. Αντίστοιχα, το αρνητικό αξίωμα του πλαισίου μας λέει ότι εάν δεν υπάρχει η δυνατότητα να τραφεί κάποιος με food, θα

συνεχίσει να μην υπάρχει η δυνατότητα να τραφεί με food, εφόσον παραγγελθεί το Υ το οποίο είναι διαφορετικό του Jood. Σε αξιωματική μορφή, η παραπάνω πρόταση γράφεται:

¬eat(food, s) Λ y≠food ⊃ ¬eat(food, do(order(y), s)) (2)

Αντίστοιχα, για κάθε ενέργεια (action), λοιπόν έχουμε το θετικό και το αρνητικό αξίωμα του πλαισίου. Για την ενέργεια bUY,σε σχέση με το fluent eat, έχουμε:

eat(food, s) Λ y≠ food ⊃ ¬eat(food, do(buy(y), s)) (3)

και ¬eat(food, s) Λ y≠food ⊃ ¬eat(food, do(buy(y), s)). (4)

Και για την ενέργεια stale, σε σχέση με το fluent eat έχουμε:

eat(food, s) Λ y≠food ⊃ eat(food, do(stale(y), s)) (5)

και ¬eat(food, s) Λ y≠food ⊃ ¬eat(food, do(stale(y), s)) (6)

Επομένως, έχουμε 3 θετικά αξιώματα του πλαισίου (possitive frame axioms) ((1), (3), (5)) και 3 αρνητικά αξιώματα του πλαισίου (negative frame axioms) ((2), (4), (6)), τα οποία μας λένε ότι το fluent eat ή το ¬eat παραμένει αμετάβλητο, εφόσον εκτελεστεί κάποια από τις ενέργειες πάνω στο Υ το οποίο είναι διαφορετικό του food.

Η προσέγγιση του Hass χρησιμοποιεί 2xF αξιώματα του πλαισίου, αφού χρησιμοποιεί καθολικά ποσοτικοποιημένες (universally quantified) ενέργειες. Οπότε για την περιγραφή του παραπάνω παραδείγματος απαιτούνται 2x1 = 2

αξιώματα πλαισίου, αλλά και ο ορισμός των καθολικά ποσοτικοποιημένων ενεργειών.

Οπότε έχουμε το θετικό αξίωμα του πλαισίου, το οποίο μας περιγράφει ότι η δυνατότητα τροφής με food στην κατάσταση s και η μη δυνατότητα τροφής στην επόμενη κατάσταση θα είναι αληθείς, εφόσον και όταν εκτελεσθεί η ενέργεια του να χαλάσει(staIe) η τροφή (food), δηλαδή αξιωματικά:

$eat(food, s) \wedge \neg eat(food, do(a, s)) \supset a = stale(food)$ (1)

ή αλλιώς μπορεί να γραφτεί:

$eat(food, s) \wedge \neg eat(food, do(a, s)) \supset a_F(food, a, s)$

όπου a_F(food, a, ε) είναι η εξήγηση του γιατί και του πότε θα αλλάξει το fluent eat, δηλαδή το fluent eat αλλάζει όταν και γιατί θα εκτελεσθεί η ενέργεια του να χαλάσει το τρόφιμο (stale).

Αντίστοιχα, το αρνητικό αξίωμα του πλαισίου, μας περιγράφει ότι η μη δυνατότητα τροφής ¬eat, η οποία είναι αληθής τώρα, θα αλλάξει εφόσον εκτελεσθούν οι ενέργειας order(food) ή buy(food), δηλαδή θα είναι αληθής και το fluent eat(food, do(a,s)), όταν εκτελεσθεί η ενέργεια a. Αξιωματικά περιγράφεται:

$\neg eat(food, s) \wedge eat(food, do(a, s))$

$\supset b = order(food) \vee b = buy(food)$ (2)

ή με τη χρήση καθολικού ποσοδείκτη:

$\neg eat(food, s) \wedge eat(food, do(a, s)) \supset \beta_F(food, a, s)$

όπου β_F(food, a, s) είναι η εξήγηση του γιατί και του πότε θα αλλάξει το fluent ¬eat, δηλαδή το fluent ¬eat θα γίνει eat όταν και γιατί εκτελέσθηκε είτε η ενέργεια order, είτε η ενέργεια buy.

Επομένως, έχουμε 2 αξιώματα πλαισίου, αλλά θεωρούμε ότι οι καθολικοί ποσοδείκτες θεωρούνται γνωστοί.

Η τελευταία προσέγγιση του Reiter χρειάζεται A +F αξιώματα για να περιγραφεί. Ένα αξίωμα για κάθε fluent και ένα για κάθε ενέργεια. Δηλαδή, στο παράδειγμά μας χρειαζόμαστε 4 αξιώματα για να περιγράψουμε αυτόν τον κόσμο.

Το fluent eat θα περιγραφεί γενικά ως:

$Poss(a, s) \supset [eat(do(a, s)) \equiv (\gamma^{+})_{eat}(a, s) \vee eat(s) \wedge \neg (\gamma^{-})eat(a, s)]$ (1)

Δηλαδή,

$Poss(a, s) \supset [eat(do(a, s)) \equiv [(\exists food)\, a = stale(food)] \vee eat(s) \wedge$

$[(\exists food)\{a = order(food) \vee a = buy(food)\}]]$, (1)

το οποίο σημαίνει ότι το fluent eat θα ισχύει, δηλαδή θα είναι αληθής, μόνο εάν η προϋπόθεση $(\gamma+)eat(a, s)$ είναι αληθής, δηλαδή μόνο αν εκτελεσθεί είτε η ενέργεια order, είτε η ενέργεια buy, αλλιώς, αν εκτελεσθεί η ενέργεια stale, δηλαδή αν προϋπόθεση $(\gamma-)eat(a, s)$ γίνει αληθής από ψευδής στην επόμενη κατάσταση, το fluent eat δεν θα ισχύει.

Επιπλέον, πρέπει να ισχύουν οι παρακάτω προϋποθέσεις ενέργειας:

$Have(telephone, s) \supset Poss(order, s)$ (2)

$Have(money, s) \supset Poss(buy, s)$ (3)

και $bad(condtions, \varepsilon) \supset Poss(stale, s)$,(4)

οι οποίες αντίστοιχα σημαίνουν, πως για να γίνει παραγγελία, θα πρέπει να υπάρχει τηλέφωνο, για να γίνει η εκτέλεση της αγοράς θα πρέπει να υπάρχουν χρήματα, και για να χαλάσει (το φαγητό) θα πρέπει να υπάρχουν κακές συνθήκες διατήρησης.

Επομένως με ένα αξίωμα για το fluent και τρία αξιώματα για τις προϋποθέσεις εκτέλεσης των ενεργειών, περιγράφηκε αυτό το κομμάτι του κόσμου που υποθέσαμε ότι έχουμε. Χαρακτηρίζεται ως απλή λύση, διότι είναι αρκετά εύκολο να περιγραφούν τα αξιώματα για τις προϋποθέσεις εκτέλεσης ενεργειών και

χρειάζεται μόνο ένα αξίωμα για το fluent, έτσι ώστε να περιγραφεί η κατάσταση που θέλουμε.

Ολοκληρώνοντας, σωστό θα ήταν να αναφερθεί ότι το πρόβλημα του πλαισίου που ζητά λύσεις τις τελευταίες δεκαετίες θα συνεχίσει να ταλανίζει και να απασχολεί την επιστημονική κοινότητα.

ΚΕΦΑΛΑΙΟ 3

Σχετικές Εργασίες και Ιστορική Αναδρομή

3.1 Η Χρονική Αναπαράσταση στις Οντολογίες

Παραδοσιακά στις οντολογίες υπάρχουν δύο προσεγγίσεις για την αναπαράσταση των μεταβολών στην πληροφορία σε σχέση με το χρόνο: η προσέγγιση με βάση τις εκάστοτε εκδόσεις ή δείγματα[3] [1] (η κλασική προσέγγιση) και η πρόσφατη «περντιουρανιστική» προσέγγιση [12].

Η πρώτη προσέγγιση έχει το μειονέκτημα του πλεονασμού των πληροφοριών καθώς βασίζεται στην επανάληψη της πληροφορίας. Επίσης, οι αλλαγές στο χρόνο μπορούν να εξαχθούν ως συμπέρασμα μέσω της σύγκρισης του παροντικών και των παρελθόντων καταστάσεων και δεν μπορούν να

[3] ΣτΜ: Versioning approach

αναπαρασταθούν άμεσα στην οντολογία.. Η δεύτερη προσέγγιση επιλύει αμφότερα τα ζητήματα. Σε αυτό το έργο υιοθετούμε την λύση που ονομάζεται επίσης τετραδιάστατη προσέγγιση. Η τετραδιάστατη προσέγγιση (περντιουρανιστική) βασίζεται στην τρισδιάστατη προσέγγιση (εντιουρανιστική) για την αναπαράσταση του τρισδιάστατου (στατικού) κόσμου. Αυτή η προσέγγιση διακρίνει τον κόσμο σε δύο βασικές κατηγορίες οντοτήτων: Τα αντικείμενα που χαρακτηρίζονται από διάρκεια (φυσικά αντικείμενα όπως αυτοκίνητα, εταιρίες και ανθρώπους) και τα συμβάντα (γεγονότα όπως η αγορά ενός αυτοκινήτου). Τα πρώτα αναπαριστούν τη χρονική ανεξάρτητη πληροφορία (πληροφορία που υπάρχει σε όλες τις στιγμές) ενώ τα δεύτερα έχουν χρονικά μέρη. Το κύριο θέμα με αυτή την προσέγγιση είναι ότι η διαχρονική ταυτότητα των διαρκών προσεγγίζεται με την αναγνώριση μιας σειράς ιδιοτήτων που μένουν αμετάβλητες στο χρόνο. Τα διαρκή αναπαρίστανται με μια σειρά ιδιοτήτων που δεν μεταβάλλονται ανά το χρόνο (πχ. το DNA του Τζον) καθώς και από μια σειρά ιδιοτήτων των οποίων οι τιμές αποτελούν συνάρτηση του χρόνου.

Στην τετραδιάστατη (περντιουρανιστική) προσέγγιση όλες οι οντότητες υπάρχουν συνεχώς, χωρίς να γίνεται διάκριση μεταξύ των διαρκών και των συμβάντων. Κάθε οντότητα έχει ένα αρχικό κι ένα τελικό σημείο και είναι ένα απλό γεγονός (πχ. ο χρόνος ζωής του ήλιου). Μια οντότητα μπορεί να περιγραφεί ως «χωροχρονικό σκουλήκι», καθώς τα κομμάτια του σκουληκιού αποτελούν τα χρονικά μέρη της οντότητας. Με αυτή την προσέγγιση το πρόβλημα της διαχρονικής οντότητας δεν υπάρχει πλέον καθώς μια οντότητα έχει χρονικά κομμάτια και οι μεταβολές συμβαίνουν στις ιδιότητες ενός χρονικού μέρους διατηρώντας την οντότητα ως σύνολο.

Το σχήμα δύο δείχνει πώς η πληροφορία για τους επενδυτές και τις μετοχές αναπαρίσταται σε μια οντολογία. Υπάρχουν δύο ενέργειες (γεγονότα): InvestorSellShare και InvestorBuyShare. Τα χρονικά στιγμιότυπα της μετοχής

έχουν ως ιδιότητα (σε κάθε χρονική στιγμή) την αξία της μετοχής. Αυτός ο κόσμος μπορεί να περιγραφεί από την οντολογία που έχει μια κλάση Επενδυτή, μια κλάση Μετοχής, μια κλάση Συμβάντος (με τις δύο ενέργειες που περιγράφονται παραπάνω ως υποκλάσεις) και μια κλάση FourDFluents που έχει την Time Slice και την Time Interval ως υποκλάσεις. Η κλάση Time Slice περιέχει όλα τα χρονικά κομμάτια και η κλάση Time Interval περιέχει τα χρονικά διαστήματα.όταν δημιουργείται μια νέα μετοχή στον κόσμο προστίθεται ένα καινούργιο στιγμιότυπο στην κλάση της Μετοχής. Κάθε φορά που αλλάζει η τιμή αυτής της μετοχής δημιουργείται ένα καινούργιο στιγμιότυπο της κλάσης Time Slice και συνδέεται με το στιγμιότυπο της κλάσης Time Slice. Η αξία της ιδιότητας του τύπου δεδομένων κατέχει την καινούργια αξία της μετοχής και συνδέεται με το νέο στιγμιότυπο της κλάσης Time Slice. Τέλος, δημιουργείται ένα στιγμιότυπο της κλάσης Time Interval και συνδέεται με το καινούργιο στιγμιότυπο της κλάσης Time Slice.

Ας υποθέσουμε ότι στο χρόνο Τ ένας επενδυτής αποφασίζει να πουλήσει μια μετοχή. ¨Όταν συμβαίνει ένα τέτοιο γεγονός (InvestorSellShare) δημιουργούνται τρία νέα στιγμιότυπα της κλάσης Time Slice. Το πρώτο συνδέεται με τον επενδυτή, το δεύτερο με τη μετοχή και το τρίτο με το συμβάν (InvestorSellShare). Το Time Slice του InvestorSellShare επίσης συνδέεται με το Time Slice του Επενδυτή μέσω της ιδιότητας του αντικειμένου share Seller και επίσης συνδέεται με το Time Slice της Μετοχής μέσω της ιδιότητας του αντικειμένου share Sell. Τέλος δημιουργείται ένα στιγμιότυπο της κλάσης Time Interval (η οποία αφορά το χρόνο που συνέβη το InvestorSellShare και συνδέεται με όλα τα νέα στιγμιότυπα της Time Slice.

3.2 Τυπολογία

Τα προβλήματα πλαισίου, εξειδίκευσης και παραφυάδων:

Παραδοσιακά υπάρχουν τρία προβλήματα που τα περισσότερα προγράμματα συλλογιστικής προσπαθούν να επιλύσουν. Το *πρόβλημα πλαισίου* το οποίο αναφέρεται στην αναγνώριση των κατηγορημάτων ή των λειτουργιών που παραμένουν αμετάβλητα ως αποτελέσματα ενεργειών. Το *πρόβλημα των παραφυάδων* το οποίο αναφέρεται στον καθορισμό των έμμεσων αποτελεσμάτων των ενεργειών και το *πρόβλημα εξειδίκευσης* το οποίο αναφέρεται στον καθορισμό των προϋποθέσεων που πρέπει να ικανοποιούνται πριν την εκτέλεση μιας ενέργειας.

Τυπολογία για συλλογισμό σχετικά με ενέργεια σε συνάρτηση με το χρόνο: Οι σημαντικότερες τυπολογίες για συλλογισμό σχετικά με ενέργειες και μεταβολές είναι: ο λογισμός των καταστάσεων [16], ο λογισμός ρευστών, ο λογισμός των συμβάντων [14], οι γλώσσες ενεργειών και ο λογισμός ενεργειών και η χρονική λογική ενεργειών (TAL) [15].

Ο λογισμός των καταστάσεων αποτελεί τη δεύτερη γλώσσα εντολών που έχει σχεδιαστεί για την αναπαράσταση των μεταβολών που συμβαίνουν σ' ένα κόσμο αυξομειώσεων. Όλες οι μεταβολές που συμβαίνουν σ' ένα κόσμο είναι αποτέλεσμα της εκτέλεσης κάποιων πράξεων. Ο κόσμος περιγράφεται από ρευστά (κατηγορήματα και συναρτήσεις). Μια πιθανή εξέλιξη του κόσμου είναι μια ακολουθία ενεργειών που αναπαρίσταται από τον πρώτο όρο εντολής που αποκαλείται κατάσταση. Ο λογισμός των μεταβολών και ο λογισμός των ενεργειών αποτελούν εναλλακτικές του λογισμού των καταστάσεων.

Ο λογισμός των συμβάντων περιέχει μια σειρά ρευστών, μια σειρά ενεργειών, μια σειρά χρονικών σχέσεων μεταξύ των ενεργειών και των ρευστών και μια μερικώς διατεταγμένη σειρά χρονικών σημείων.

Η Λογική Χρονικών Ενεργειών (TAL) είναι μια λογική που εμπερικλείει τον χρόνο. Η TAL αποτελείται από μια γλώσσα συμπιεσμένης επιφάνειας L(SD) που μεταφράζεται σε μια γλώσσα πρώτης εντολής L(FL).

Οι γλώσσες Λογικής Περιγραφής [13] θεωρούνται ως ο πυρήνας των συστημάτων αναπαράστασης της γνώσης, συμπεριλαμβάνοντας και την δομή της γνωστικής βάσης DL και τις σχετιζόμενες υπηρεσίες συλλογιστικής. Τα καθορισμένα προβλήματα συμπερασμού που ένα πρόγραμμα συλλογιστικής Λογικής Περιγραφής (DL) [13] μπορεί να απαντήσει είναι: υπαγωγή, ικανοποιητικότατα, συνέπεια και προσδιορισμός. Οι συλλογιστικοί αλγόριθμοι μπορεί να εξαρτώνται από την εκφραστικότητα της λογικής που υιοθετείται.

Σ' αυτό το έργο ένα πρόγραμμα χρονικής συλλογιστικής δομείται πάνω στο λογισμό καταστάσεων. Παρακάτω θα παρουσιάσουμε μια **Προέκταση στο Λογισμό Καταστάσεων** με σκοπό να μπορέσουμε να εμπερικλείσουμε τον χρόνο.

Για κάθε ρευστό *f*, προστίθεται ένα όρισμα *L*, όπου *L* είναι μια λίστα από χρονικά διαστήματα $[a,b]$, $a < b$. Κάθε $[a,b]$ αναπαριστά τα χρονικά στιγμιότυπα

μεταξύ του x: *{x {a < x < b } }*. Το ρευστό *f* επαληθεύεται στα χρονικά διαστήματα που περιέχονται στην λίστα L.

Ορίζουμε τις συναρτήσεις *start (a)* και *end (a)*, όπου *a* είναι μια ενέργεια. Το πρώτο επιστρέφει το χρονικό σημείο εκκίνησης για την ενέργεια α ενώ το τελευταίο επιστρέφει το χρονικό σημείο ολοκλήρωσης.

Οι ενέργειες εκτελούνται με την παρακάτω σειρά: $a_1 < a_2 < \ldots < a_n$, όπου $start(a_1) < start(a_2) < \ldots < start(a_n)$.

Οι (στιγμιαίες) ενέργειες $a_1, a_2, \ldots, a_n$ εκτελούνται ταυτόχρονα αν $start(a_1) = start(a_2) = \ldots = start(a_n)$.

Το κατηγόρημα *actionHappen*(a,t) σημαίνει ότι η ενέργεια *a* εκτελείται τη χρονική στιγμή *t*.

Ορίζουμε τις συναρτήσεις *start (S)* και *end (S)*, όπου *S* είναι μια κατάσταση. Η πρώτη συνάρτηση επιστρέφει το χρονικό σημείο εκκίνησης για το *S* ενώ το τελευταίο επιστρέφει το χρονικό σημείο ολοκλήρωσης.

Ορίζουμε ως χρονική κατάσταση μια κατάσταση που περιέχει όλα τα ρευστά με την λίστα των χρονικών διαστημάτων στα οποία τα ρευστά επαληθεύονται.

Η συνάρτηση *FluentHold(S,t)* επιστρέφει όλα τα ρευστά που επαληθεύονται τη χρονική στιγμή *t*. Για τη συνάρτηση ρευστού το *FluentHold(S,t)* επιστρέφει την τιμή που έχει η συνάρτηση στο χρονικό σημείο *t*. Η κατάσταση *S* αποτελεί χρονική κατάσταση.

3.3 Υπάρχοντα προγράμματα συλλογιστικής

Τα οντολογικά προγράμματα συλλογιστικής μπορούν να κατηγοριοποιηθούν σύμφωνα με το υποσύνολο OWL πάνω στο οποίο εκτελείται ο συλλογισμός. Τα τρέχοντα προγράμματα συλλογιστικής επικεντρώνονται στο υποσύνολο OWL-DL. Αυτό δεν εκπλήσσει από τη στιγμή που το OWL-DL παρέχει τη μέγιστη εκφραστικότητα ενώ παράλληλα διατηρεί της επιθυμητές υπολογιστικές ιδιότητες όπως η αποφασιστικότητα. Παραδείγματα πολύ γνωστών προγραμμάτων συλλογιστικής DL είναι τα FaCT[4] και Racer. Από την άλλη πλευρά δεν υπάρχουν υπολογιστικές εγγυήσεις για την OWL-Full Παρόλα αυτά υπάρχουν προγράμματα συλλογιστικής τα οποία στοχεύουν την OWL-Full (πχ. cwm, surnia) αλλά αυτά τα προγράμματα παρέχουν ατελή συλλογισμό. Είναι απίθανο οποιοδήποτε πρόγραμμα συλλογιστικής να καταφέρει να υποστηρίζει ολοκληρωμένο συλλογισμό για κάθε πλευρά του OWL-Full. Μια ενδιαφέρουσα κατηγορία των προγραμμάτων συλλογιστικής OWL βασίζεται στην Prolog. Με την εφαρμογή του εργαλείου SWI-Prolog [3] η οντολογία μετατρέπεται από OWL σε κατηγορήματα prolog (υποκείμενο, κατηγόρημα, αντικείμενο). Έτσι ένα πρόγραμμα συλλογιστικής μπορεί να εφαρμοστεί σε Prolog. Με αυτόν τον τρόπο, η σημασιολογία ανοικτού κόσμου της OWL γίνεται άμεσα προσβάσιμη εντός του συστήματος Prolog. Στη συνέχεια, οι ερωτήσεις στις οντολογίες μπορούν να βασιστούν σε γλώσσες ερωτημάτων στην Prolog όπως η SeRQL [7]. Άλλες ενδιαφέρουσες κατηγορίες προγραμμάτων συλλογισμού (προγραμματισμένες σε Java) είναι η Bossam και η Palet [6]. Ο συλλογιστικός αλγόριθμος βασίζεται στην περιγραφική λογική. Τα προγράμματα συλλογιστικής αυτής της κατηγορίας (επίσης η παραπάνω κατηγορία) εφαρμόζουν μια διαδικασία αποφάσεων βασισμένη σε πίνακα για τα γενικά TBoxes (υπαγωγή, ικανοποιητικότατα, κατηγοριοποίηση) και για τα ABoxes (ανάκτηση, συζευκτική αναζήτηση απαντήσεων).

Όλα τα παραπάνω προγράμματα συλλογιστικής δεν μπορούν να χειριστούν τις μεταβολές που προκύπτουν με την πάροδο του χρόνου. Μερικά από αυτά υποστηρίζουν τον χειρισμό του χρόνου αλλά χειρίζονται τον χρόνο όπως κάθε άλλη ιδιότητα (πχ. Pellet). Επίσης αυτά τα προγράμματα συλλογιστικής δεν υποστηρίζουν τις σχέσεις σε χρονικά διαστήματα.

Το χρονικό πρόγραμμα συλλογιστικής για KBs όπως το ConGolog είναι ακατάλληλο για OWL-time διότι βασίζονται σε συγκεκριμένη διαμόρφωση των KBs. Προτείνουμε το PROTON ένα ολοκληρωμένο σύστημα που λαμβάνει ως είσοδο τη γνώση σε οντολογία OWL-time και τη μετατρέπει σε προτάσεις Prolog (που είναι τα KBs για τον χρονικό συλλογισμό). Το πρόγραμμα χρονικού συλλογισμού αποτελείται από πέντε συστατικά μέρη:

α) Μετατροπή από οντολογίες OWL-time σε προτάσεις Prolog

β) Μια σειρά λειτουργιών σε χρονικές περιόδους (σχέσεις Allen).

γ) Συναρτήσεις για τον καθορισμό της αξίας κάθε ιδιότητας σε κάθε χρονικό στιγμιότυπο Τα

δ) Μια σειρά κατηγορημάτων που καθορίζουν πότε συμβαίνει ένα γεγονός

ε) Μια σειρά κατηγορημάτων που εκτελούν κανόνες όταν προκύπτει ένα συμβάν ή όταν μεταβάλλεται η αξία μιας ιδιότητας. (Για παράδειγμα όταν ένα γεγονός InvestorBuyShare συμβαίνει τότε πρέπει να προστεθεί η μετοχή στο επενδυτικό χαρτοφυλάκιο ή όταν η τιμή αυξάνει κατά 10% θα πρέπει να πουληθεί).

Στο επόμενο κεφάλαιο θα παρουσιάσουμε αυτά τα πέντε συστατικά στοιχεία του προγράμματος χρονικού συλλογισμού PROTON.

ΚΕΦΑΛΑΙΟ 4

PROTON: ΕΝΑ ΠΡΟΓΡΑΜΜΑ ΣΥΛΛΟΓΙΜΟΥ ΤΗΣ PROLOG ΓΙΑ ΤΙΣ ΧΡΟΝΙΚΕΣ ΟΝΤΟΛΟΓΙΕΣ

Το PROTON βασίζεται στην μετατροπή των δηλώσεων σε OWL σε γεγονότα της μορφής «κατηγόρημα (υποκείμενο, αντικείμενο)». Αρχικά, μετατρέπουμε το αρχείο OWL σε τριπλέτες (υποκείμενο κατηγόρημα αντικείμενο) χρησιμοποιώντας το εργαλείο SWI-Prolog. Έπειτα τις τριπλέτες τις μορφής:

(S,'http://www.w3.org/ 1999/02/22-rdf-syntax-ns#type',O)

Μετατρέπονται σε O(S) (έπειτα μπορούμε να προσθέσουμε στην KBase τα γεγονότα πχ. fact(class('Zolinza')))

Οι τριπλέτες της μορφής (X,Y,literal(Z))[4] μετατρέπονται σε Y(X,Z) τα οποία εισάγονται στην γνωστική βάση.

Για παράδειγμα, μια οντότητα OWL της μορφής

<owl:ObjectPropertyrdf:ID="tsTimeInterval">

<rdfs:domain rdf: resource="#tsTimeSliceof'/>

<rdfs :range rdf:resource="#TimeInterval"/>

</owl:ObjectProperty>

μετατρέπεται αρχικά σε μια τριπλέτα του είδους

(tsTimeSliceof, onproperty.....)

(tsTimeSliceof,disjointwith,tsTimeInterval)

κι έπειτα στα ακόλουθα γεγονότα KB:

fact(disjointwith('tsTimeSliceof`;'tsTimeInterval')).

[4] ΣτΜ: literal σημαίνει λεκτικό

fact(onproperty('__Dcscription2', tsTimeSlice O f)).

Οι τριπλέτες της μορφής (X,Y,Z) μετατρέπονται σε γεγονότα Y(X,Z)

ΟΙ τριπλέτες της μορφής

(S,' http://www.w3. org/2000/0l/rdf-schema#subClassOf',O)

μετατρέπονται σε **O(X): -S(X)**

Οι τριπλέτες της μορφής

(S,'http:/www.w3 .org/2000/ 1 /rdf-schema#subPropertyOf', O)

μετατρέπονται σε **O(X,Y): - S(X,Y)**

Αφού έχουμε μετατρέψει όλες τις οντότητες OWL σε γεγονότα, μπορούμε να εκμεταλλευτούμε τους μηχανισμούς που προσφέρει η Prolog για την εφαρμογή ενός προγράμματος συλλογιστικής. Έτσι, η KB στην Prolog μπορεί να εμπλουτιστεί περαιτέρω με τους αναγκαίους συμπερασματικούς κανόνες για τον συλλογισμό. Η παραπάνω διαδικασία μπορεί να μεταβληθεί για να παράγει μια βάση γνώσεων για κάθε μοντέλο (λογισμός καταστάσεων, λογισμός συμβάντων, λογισμός ρευστών, κτλ.).

Η Βάση γνώσεων αποτελείται από κατηγορήματα που παράγονται αυτόματα μέσω της προαναφερθείσας διαδικασίας. Μερικά από αυτά τα κατηγορήματα είναι κοινά για όλες τις οντολογίες ενώ άλλα διαφέρουν (εξαρτώνται από την εφαρμογή).

Για να τα παρουσιάσουμε χρησιμοποιούμε το παράδειγμα του συστήματος διαχείρισης μετοχών στο οποίο κάναμε μια εισαγωγή στο Κεφάλαιο 1 (Σχήμα 2). Θα δείξουμε τα πέντε μέρη των προγραμμάτων συλλογιστικής, στα οποία αναφερθήκαμε στο τέλος του προηγούμενου κεφαλαίου. Πρώτα θα δώσουμε ένα παράδειγμα πώς τα αρχεία σε OWL μετατρέπουν τα κατηγορήματα της Prolog. Ας υποθέσουμε την παρακάτω ιδιότητα δεδομένων σε OWL

<ow1:DatatypeProperty rdf:ID="caption">

<rdfs:domain rdf:resource="#share"/>

<rdfs:range rdf:resource="...#string"/>

Στην δική μας ΚΒ αυτή η ιδιότητα μετατρέπεται σε δύο αντίστοιχα κατηγορήματα (α) **caption/2** το οποίο έχει ένα μοναδικό αναγνωριστικό όνομα (πχ. Share_1) ως πρώτο όρισμα και το όνομα της μετοχής (π.χ. metoxh_1) ως δεύτερο, (β) **share/1** το οποίο περιέχει το αναγνωριστικό όνομα της μετοχής.

Το κατηγόρημα **event/1** καθορίζει ένα καινούργιο γεγονός. Στο παράδειγμα διαχείρισης των μετοχών ένα συμβάν λαμβάνει χώρα όταν ένας επενδυτής αγοράζει ή πουλά μια μετοχή/caption. Το κατηγόρημα event/1 καθορίζει ένα καινούργιο συμβάν το οποίο αφορά ένα επενδυτή Χ και μια μετοχή/caption Υ. Αυτά τα δύο συνδέονται χρησιμοποιώντας τα time Slices:

Πρώτα το συμβάν συνδέεται σε ένα timeSlice **K** χρησιμοποιώντας το κατηγόρημα **IsTimeSlice/2** και μετά το timeSlice **K** συνδέεται με το **shareSlice** (που συνδέεται με την μετοχή) και σ' ένα **investorSlice** (που συνδέεται με ένα επενδυτή) με τα κατηγορήματα **shareBought/2** και **share Buyer/2**. Το shareSlice καθώς και το investorSlice είναι timeSlices. Έπειτα, και τα δύο συνδέονται με την ίδια χρονική περίοδο με το κατηγόρημα **IsTimeInterval/2** και τελικά με το χρονικό αποτύπωμα (όταν γίνεται το συμβάν) με το κατηγόρημα **date/2**. Τέλος το κατηγόρημα eventHappen/3 δημιουργεί μια λίστα δύο στοιχείων περιλαμβάνοντας τον Επενδυτή Χ και τη μετοχή/caption Υ που εμπλέκονται στο συμβάν Ε στο χρονικό αποτύπωμα D (βλ. Σχήμα 6).

Παρακάτω παρουσιάζουμε το μέρος (β) του προγράμματος χρονικής συλλογιστικής. Θα πρέπει να ορίσουμε νέα κατηγορήματα για τον χειρισμό του χρόνου. Έχουμε εφαρμόσει, στα κατηγορήματα Prolog όλες τις χρονικές σχέσεις του Allen (πριν, ισούται, συναντά, επικαλύπτει, κατά τη διάρκεια, αρχίζει, τελειώνει).

Το κατηγόρημα **date/2**

Το πρώτο όρισμα είναι το **timeInterval** και το δεύτερο ένα χρονικό αποτύπωμα σε μια διάταξη συμβολοσειράς της μορφής: "**yyy-mm-ddThh-mm-ss**". Το κατηγόρημα επαληθεύεται αν το δεύτερο όρισμα ανήκει στο χρονικό διάστημα.

Ας πάρουμε για παράδειγμα το κατηγόρημα **before/2**. Αυτό το κατηγόρημα χρησιμοποιείται για τον καθορισμό του προτιμότερου από τα δύο χρονικά αποτυπώματα που δίνονται ως ορίσματα και επιτυγχάνει μόνο αν το πρώτο όρισμα αναπαριστά ένα νωρίτερο χρονικό σημείο απ' ότι το δεύτερο.

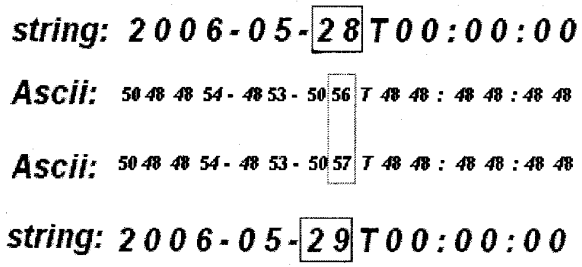
Before(X,Y):-name(X,L),name(Y,G),check(L,G).

Κάθε ένα από τα δύο ορίσματα αποτελεί ένα χρονικό αποτύπωμα της μορφής "yyy-mm-ddThh-mm-ss" . Χρησιμοποιούμε το κατηγόρημα name/2 για να μετατρέψουμε κάθε χρονικό αποτύπωμα (το οποίο είναι αλφαριθμητικό) σε μια λίστα χαρακτήρων ASCII που προσμετρά την τιμή τους. Έπειτα συνεχίζουμε με το κατηγόρημα check/2 που παίρνει τις δύο λίστες ASCII και συγκρίνει κάθε ένα από τα στοιχεία τους ένα προς ένα. Παρακάτω βρίσκεται το κατηγόρημα **check/2**:

check(]AtL],[B G]):-A<B.

check([A|L],[A|G]):- check(L,G).

Το κατηγόρημα επιτυγχάνει μόνο όταν κάθε στοιχείο στην πρώτη λίστα έχει μικρότερη αξία στον πίνακα ASCII απ' ότι στην δεύτερη λίστα. Αυτό συμβαίνει εξετάζοντας αναδρομικά ένα προς ένα όλα τα στοιχεία και από τις δύο λίστες. Οι τιμές ASCII για τους αριθμητικούς χαρακτήρες βρίσκονται σε ανιούσα σειρά όπως ακριβώς οι αριθμοί. Έτσι αν ο αριθμός Α έχει μικρότερη αριθμητική αξία απ' ότι ο αριθμός Β έπειτα η τιμή ASCII του αριθμού Α είναι επίσης μικρότερη. Δίνουμε ένα παράδειγμα στο σχήμα 3.



Σχήμα 3. Το κατηγόρημα Before/2

Όπως βλέπουμε στο σχήμα 3, τα δύο χρονικά αποτυπώματα είναι σχεδόν τα ίδια και όλες οι τιμές του πίνακα ASCII ταιριάζουν, εκτός από μία. έτσι, για κάθε αριθμό ο οποίος είναι ο ίδιος και στις δύο λίστες, το κατηγόρημα check/2 θα επαναληφθεί και όταν θα φτάσει στο σημείο να συγκρίνει τα ψηφία "8" και "9" (οι τιμές ASCII "56" και "57") θα επιτύχουν διότι η πρώτη ημερομηνία βρίσκεται νωρίτερα απ' ότι η δεύτερη. Οι χαρακτήρες που χωρίζουν τα ψηφία (" -" ," T" and " : ") έχουν τις ίδιες τιμές στον πίνακα ASCII και όλες ταιριάζουν μεταξύ τους.

Τώρα θα παρουσιάσουμε το τρίτο μέρος (γ) του προγράμματος χρονικής συλλογιστικής (πώς μπορούμε να καθορίσουμε την αξία κάθε ιδιότητας σε κάθε χρονικό σημείο). Θα πρέπει να ορίσουμε τα κατηγορήματα **value/3** και **hold/3.**

Το κατηγόρημα **value/3**

Αυτό το κατηγόρημα έχει τρία ορίσματα και επιτυγχάνει αν η λεζάντα έχει μια ειδική τιμή σε συγκεκριμένη ημερομηνία (χρονικό αποτύπωμα)[5].

value(X,Y,Z):- caption(SHARE,X), share(SHARE), timeSliceOf(SLICE,SHARE),isTimeInter-

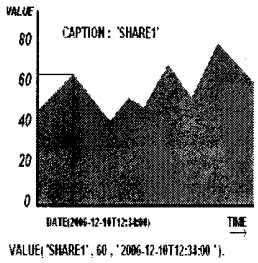
val(SLICE, INTERVAL),value(INTERVAL, Y), date(INTERVAL,Z), timeInterval(INTERVAL).

Το όρισμα **X** είναι η λεζάντα του ορίσματος που είναι αλφαριθμητική, το όρισμα **Y** είναι η τιμή του ορίσματος (αριθμητική τιμή) και τέλος το όρισμα **Z** είναι ένα χρονικό αποτύπωμα που αναπαριστά την συγκεκριμένη ημερομηνία.

Έτσι αν η λεζάντα X υπάρχει στην λίστα μετοχών έπειτα το πρώτο που εξετάζουμε είναι σε ποια time Slice ανήκει αυτή η μετοχή και έπειτα

[5] Το κατηγόρημα value/3 ορίζεται για όλα τα κατηγορήματα που παράγονται από όλες τις ιδιότητες του τύπου δεδομένων (π.χ. μετοχή).

εξετάζουμε αν υπάρχει χρονικό διάστημα που συνδέεται με αυτή τη φέτα. Έπειτα πρέπει να υπάρχει μια τιμή **Y** που να συνδέεται με αυτό το διάστημα και τέλος να συνδέεται με αυτό μια ημερομηνία **Z**.



Σχήμα 4. Το κατηγόρημα value/3

Όπως βλέπουμε στο παράδειγμα του σχήματος 4 θα επαληθευτεί η τιμή (sharel',60,'2006-12-lOT:12:34:00').

Το κατηγόρημα **hold/3**

Όμοια με το κατηγόρημα της τιμής αυτό το κατηγόρημα επιτυγχάνει αν μια λεζάντα διατηρήσει την τιμή της κατά τη διάρκεια ενός συγκεκριμένου χρονικού διαστήματος. Η τιμή της λεζάντας πρέπει να διατηρεί την αρχική τιμή της κατά τη διάρκεια του χρονικού διαστήματος και να μεταβάλει τη τιμή του στο τέλος του χρονικού διαστήματος. Διαφορετικά το κατηγόρημα αποτυγχάνει. Εφαρμογή:

hold(X,A,[T,EI) :- value(X,A,T), iiot(value(X,A,E)),
not((value(X,B,F),B\=A,before(F,E),before(T,F))).

Για να εφαρμόσουμε αυτό το κατηγόρημα χρησιμοποιούμε το κατηγόρημα value/3 διότι πρέπει να γνωρίζουμε εάν η λεζάντα έχει ειδική τιμή σε κάποιες

χρονικές αποτυπώσεις και επίσης να εξασφαλίσουμε ότι αυτή η τιμή δεν μεταβάλλεται κατά τη διάρκεια του χρονικού διαστήματος.

Το όρισμα **X** είναι η λεζάντα ορίσματος και το ***A*** είναι η τιμή του ορίσματος όπως είδαμε και πριν.

Το τρίτο όρισμα αυτού του κατηγορήματος είναι μια λίστα με δύο στοιχεία. Αυτή η λίστα αναπαριστά το χρονικό διάστημα όπου η λεζάντα θα πρέπει να διατηρήσει την αρχική τιμή της. Το πρώτο στοιχείο της λίστας είναι το αρχικό χρονικό αποτύπωμα του χρονικού διαστήματος και το δεύτερο είναι το τελικό χρονικό αποτύπωμα.

Θα πρέπει να εξασφαλίσουμε τα παρακάτω:

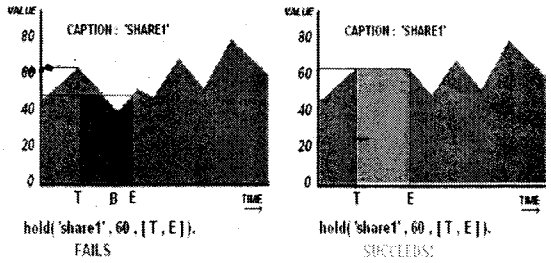
1) Η τιμή της λεζάντας στο χρονικό αποτύπωμα **T** θα πρέπει να είναι ***A*→ τιμή (X,*A*,T)**

2) Η τιμή της λεζάντας στο χρονικό αποτύπωμα **E** θα πρέπει να είναι **≠ *A* →¬ (value(X,*A*,E))**

3) Το χρονικό αποτύπωμα **F, F ∈ [T , E]** (όπου η τιμή της λεζάντας στο χρονικό αποτύπωμα **B ≠ *A***) δεν θα έπρεπε να υπάρχει.

Για να ικανοποιήσουμε αυτούς τους κανόνες που αναφέρονται στο (3) θα πρέπει να διασφαλίσουμε ότι αν υπάρχει μια τιμή **B** και το χρονικό αποτύπωμα **F** που κάνει τη πρόταση **value(X,B,F)** να επιτυγχάνει όταν το χρονικό αποτύπωμα **F** πρέπει να μην ανήκει στο χρονικό διάστημα **[T, E]**.

Ειδικά → **¬((value(X,B,F), BSA , before(F,E) ,before(T,F)))**.



Σχήμα 5. Το κατηγόρημα hold/3

Μπορούμε να δούμε ένα παράδειγμα στο σχήμα 5. Ο λόγος που αποτυγχάνει ο κανόνας στο πρώτο διάγραμμα είναι επειδή υπάρχει τουλάχιστον μια τιμή **B ≠ 60** (**60** είναι όπως βλέπουμε η αρχική τιμή στην αρχή του χρονικού διαστήματος) για ένα χρονικό αποτύπωμα το οποίο βρίσκεται στο χρονικό διάστημα **[T, E]**.

Αντίθετα το δεύτερο διάγραμμα παρουσιάζει σαφώς ότι η λεζάντα διατηρεί την τιμή της κατά τη διάρκεια του χρονικού διαστήματος **[T, E]** και για κάθε τιμή **B** κατά τη διάρκεια αυτού του διαστήματος έχουμε **B=60**. Επίσης βλέπουμε ότι η λεζάντα αλλάζει τιμή μόνο στο τέλος του διαστήματος στο χρονικό αποτύπωμα **E**.

Το κατηγορούμενο **holding/3**

Το κατηγορούμενο **holding/3** καθορίζει τη τιμή για όλες τις λεζάντες σε ένα συγκεκριμένο χρονικό σημείο. Αυτό είναι πολύ σημαντικό για να αντιμετωπίσουμε το πρόβλημα πλαισίου. Το κατηγορούμενο παίρνει το χρονικό αποτύπωμα ως τρίτο όρισμα και επιστρέφει μια λίστα με ένα ζευγάρι λεζάντας και τιμής (caption_i, value_i) ως δεύτερο όρισμα. Κάθε ζευγάρι (caption_i, value_i) είναι μοναδικό και σημαίνει ότι η λεζάντα caption_i έχει τη τιμή της value_i σε συγκεκριμένο χρονικό σημείο.

Αν το κατηγόρημα **value/3** δεν υπάρχει για μια λεζάντα σε συγκεκριμένο χρονικό αποτύπωμα τότε το κατηγόρημα **holding/3** χρησιμοποιεί το κατηγορούμενο **find_last_value/3** για να βρει το τελευταίο κατηγορούμενο **value/3** που υπάρχει στη

βάση δεδομένων για αυτή τη λεζάντα (τη τελευταία φορά που μεταβλήθηκε η τιμή της λεζάντας).

Εφαρμογή:

holding(L,L2,Date):-caption(Share,Name), not(member([Name,Date],L2)), share(Share) (date(Interval,Date),ísTimelnterval(Slice,lnterval); ísTimelnterval(Slice,Ipterval)), timeSliceOf(Slice,Share), newpr(Intervai, Date, Share, V), timelntervai(Interval), holding(L, I [Name,Date] L2], Date),!.

holdiiig(L,L,Date):-**newpr(Interval,Date,Share,V):- date(Interval,Date) , value(Interval,V).newpr(i,D,S,V):- fiadlast_value(S,V, D).**

Το κατηγόρημα σαρώνει τη βάση δεδομένων για λεζάντες και για κάθε λεζάντα ακολουθεί τον ίδιο αλγόριθμο:

Αν η λεζάντα **C** υπάρχει τότε:

1) Εξετάζουμε αν η λεζάντα **C** υπάρχει στη λίστα **L**

2) Αν ναι τότε σταματάμε και πηγαίνουμε στο **6** διαφορετικά πηγαίνουμε στο **3**

3) Προσθέτουμε τη λεζάντα **C** στην λίστα **L**

4) Εξετάζουμε αν η λεζάντα **C** έχει καινούργια τιμή (ορισμένη από το κατηγόρημα **value/3**) σε τρέχουσα ημερομηνία **Date** η οποία χρησιμοποιείται ως όρισμα στο κατηγόρημα **holding/3**

5) Αν ναι τότε η τιμή **V** είναι η τιμή που ορίζεται από το κατηγόρημα **value/3**, αν όχι τότε τρέξτε το κατηγόρημα **findlast value/3** για να βρούμε τη τελευταία φορά που μεταβλήθηκε η τιμή αυτής της λεζάντας. Η τιμή **V** ορίζεται τότε από το κατηγόρημα **findlast value/3**.

6)Βρείτε μια καινούργια λεζάντα στη βάση δεδομένων.

Το κατηγόρημα εφαρμόζεται ως εξής: Κάθε λεζάντα που σαρώνεται από τη βάση δεδομένων επισυνάπτεται σε μια λίστα μαζί με την τρέχουσα ημερομηνία. Με αυτό τον τρόπο έχουμε ζευγάρια ([λεζάντα, ημερομηνία]) στη λίστα μας έτσι που να μην χρειάζεται να σαρώνουμε ξανά ότι ήδη έχουμε δώσει κάποια τιμή. Αυτή η επαλήθευση γίνεται με το κατηγόρημα μέλος. Εμείς σαρώνουμε μόνο λεζάντες που δεν είναι μέλη (μαζί με την τρέχουσα ημερομηνία) της λίστας μας.

Το επόμενο που κάνει το κατηγόρημα είναι να εξετάσει αν η τρέχουσα λεζάντα έχει τιμή στο τρέχων χρονικό αποτύπωμα ή αν ακόμα διατηρεί τη προηγούμενη τιμή της.

Αυτό γίνεται από το κατηγόρημα **newpr/4**. Αν υπάρχει μια πρόταση ημερομηνίας με ένα χρονικό διάστημα που να αντιστοιχεί σε μια τιμή στην τρέχουσα ημερομηνία τότε αυτό σημαίνει ότι οι λεζάντα λαμβάνει μια καινούργια τιμή στην τρέχουσα ημερομηνία. Αν αυτό δεν ισχύει, τότε χρησιμοποιούμε το κατηγόρημα **find_last_value/3** για να βρούμε τη τελευταία τιμή της λεζάντας.

find_last_value(Share,Value,Date):-
share(Share),timeSliceOf(Slice,Share), isTimeInterval(Shce, Interval),value(Interval,Value), date(Interval,Date 1),before(Datel,Date), not((share(Share), timeSliceOf(Slice1i, Share), isTimeInterval(Slicesı, IntervaLn), value(IatervaLn, Valuem), date(Interval ii, Date n), before(Date1, Dater), before(Daten, Date))).

Το κατηγόρημα **find_last_value/3** σαρώνει τη βάση δεδομένων για να βρει μια προηγούμενη τιμή **V** για την συγκεκριμένη μετοχή (που χρησιμοποιείται σαν όρισμα) με την προϋπόθεση ότι η ημερομηνία **Date 2** κατά την οποία η μετοχή μεταβλήθηκε έχει τιμή **πριν** την ημερομηνία **Date** και δεν υπάρχει καμιά άλλη ημερομηνία **Date_n** έτσι ώστε **Date .2 <Date ıı < Date.**

Τώρα θα παρουσιάσουμε το τέταρτο μέρος του προγράμματος χρονικής

συλλογιστικής, δηλαδή πώς μπορούμε να αναγνωρίσουμε ότι προκύπτει ένα συμβάν.

Το κατηγόρημα **eventHappen/3**

Αυτό το κατηγόρημα επιτυγχάνει αν υπάρχει ένα συμβάν (αγορά ή πώληση μιας μετοχής)[6] που συμβαίνει σε συγκεκριμένο χρονικό αποτύπωμα κι αν υπάρχει ένα συγκεκριμένο ζευγάρι επενδυτή και μετοχής που συμβαίνουν στην συναλλαγή αυτού του συμβάντος. Το κατηγόρημα επιστρέφει μια λίστα (τρίτο όρισμα) το οποίο περιλαμβάνει τον επενδυτή και την λεζάντα. Εφαρμογή:

eventHappen(X,Time,List):-event(X),
isTimeS-IiceOf(GETSLICE,X),
date(INTERVAL,Time),timeInterval(INTERVAL),timeSlice(GETSLICE),tímeSlice(SHARESLICE) , isTimeSliceOf(SHARESLICE,SHARE), isTimeInterval(SHARESLICE, INTERVAL), share(SHARE) ,caption(ShIARE, NAME), shareBuyed(GETSLICE, SHARESLICE), shareBuyer(GETSLICE, INVESTORSLICE), isTimeSliceOf(GET- SLICE, SIIARESLICE), isTimeInterval(GETSLICE, INTERVAL), isTimeInterval(INVESTORSLICE, INTERVAL), isTimeSliceOf(INVESTORSLICE, INVESTOR), timeSlice(INVESTORSLICE), investor(INVESTOR), List = [INVESTOR,NAME].

Αυτό το κατηγόρημα έχει τρία ορίσματα. Το Συμβάν **X** που προκύπτει, το χρονικό αποτύπωμα **Time** που αποτελεί τον ακριβή χρόνο όταν το Συμβάν έγινε και τελικά μια λίστα με δύο στοιχεία που περιέχουν τον επενδυτή και την μετοχή σε αυτή τη συναλλαγή Συμβάν.

Ένα Συμβάν αποτελεί μια συναλλαγή που προκύπτει σε συγκεκριμένο χρονικό αποτύπωμα και συνδέεται πάντοτε με δύο ρευστά: Τον επενδυτή (τον

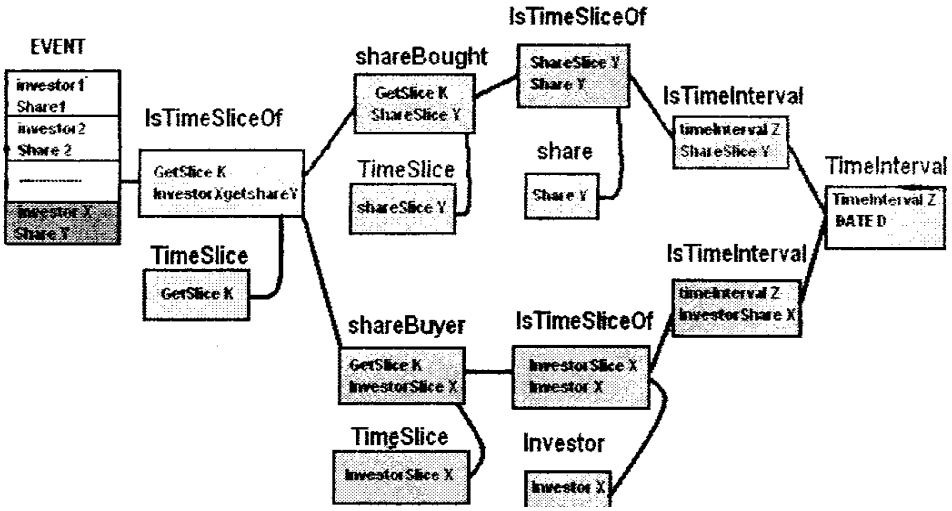
[6] Όμοια με όλα τα συμβάντα.

αγοραστή ή τον πωλητή της μετοχής) και τη λεζάντα (τη μετοχή που αγοράζεται ή πωλείται). Αυτή η συναλλαγή είναι πάντοτε μοναδική από τη στιγμή που δεν υπάρχει δυνατότητα ο ίδιος επενδυτής να μπορεί να αγοράσει/πουλήσει την ίδια μετοχή ακριβώς στον ίδιο χρόνο.

Το Συμβάν τότε συνδέεται μ'ένα TimeSlice και έπειτα αυτή η φέτα (GetSlice K στο σχήμα 6) συνδέεται μ' ένα μοναδικό ρευστό (χρησιμοποιώντας τη πρόταση IsTimeSliceOf) που αναπαριστά την συναλλαγή. Ας ονομάσουμε αυτή τη φέτα InvestorXshareY ώστε να ξέρουμε ότι ο επενδυτής X αγοράζει και πωλεί μια μετοχή Y. Στη συνέχεια η χρονική φέτα (GetSlice K) συνδέεται σε μια shareSlice και επίσης σε μια μετοχή και το investorSlice μ' έναν επενδυτή. Με αυτόν τον τρόπο γνωρίζουμε τη μετοχή και τον επενδυτή που παίρνουν μέρος στη συναλλαγή.

Τελικά το shareSlice και η φέτα του επενδυτή συνδέονται με το ίδιο TimeInterval που αναπαριστά ένα συγκεκριμένο χρονικό αποτύπωμα (Ημερομηνία).

Τώρα παρουσιάζουμε το πέμπτο μέρος του προγράμματος χρονικής συλλογιστικής, πώς μπορούμε να εκτελέσουμε μια σειρά κανόνων για να «αιχμαλωτίσουμε» τα έμμεσα αποτελέσματα μερικών πράξεων (συμβάντων)[7]



[7] Αν αυτοί οι κανόνες παράγονται από περιορισμούς της ακεραιότητας [9] τότε θα πρέπει να εξετάσουμε το πρόβλημα των παραφυάδων

Σχήμα 6. Η αντιστοιχία μεταξύ Χρόνου-Συμβάντων-Καταστάσεων

Το κατηγόρημα **sales/2**

Σε αυτό το σημείο θα δώσουμε ένα παράδειγμα πώς εφαρμόζουμε κανόνες για να ικανοποιήσουμε κάποιους περιορισμούς. Ας υποθέσουμε ότι θέλουμε να δημιουργήσουμε ένα κανόνα που θα πρέπει όταν εκτελείται να ικανοποιεί τον ακόλουθο περιορισμό: αν η τιμή μιας μετοχής αυξηθεί κατά 10% τότε θα πρέπει να πωληθεί.

Το κατηγόρημα **sales/2** χρησιμοποιεί το κατηγόρημα **mysales/3** για να καθορίσει αυτές τις λεζάντες που η αξία τους έχει αυξηθεί περισσότερο από 10% (σε σύγκριση με την προηγούμενη αξία της μετοχής) μέχρι και μια συγκεκριμένη ημερομηνία. Η λίστα των μετοχών σαρώνεται και οι λεζάντες αυτών των μετοχών που ικανοποιούν το κατηγόρημα προστίθενται στην λίστα. Εφαρμογή: **sales(L, DATE_1):-mysales(L,[]DATE_1),!.**

Το κατηγόρημα των πωλήσεων παράγεται από δύο ορίσματα:

1) Η λίστα **L** η οποία πρόκειται να περιλαμβάνει τις λεζάντες που ικανοποιούν αυτόν τον κανόνα και

2) Η ημερομηνία **DATE.1**. Για να ικανοποιηθεί αυτός ο κανόνας η μετοχή θα πρέπει να αυξήσει την αξία της (σε σχέση με την τελευταία φορά που ανανεώθηκε η συγκεκριμένη αξία) περισσότερο από 10%.

Έπειτα το κατηγόρημα **mysales/3** χρησιμοποιείται αφού έχει ακόμα ένα όρισμα, μια κενή- αρχικά- λίστα (λίστα L2), που χρησιμοποιείται για να αποθηκεύσει

τις λεζάντες και έπειτα να τις αντιγράφει στην αρχική λίστα ορισμάτων L. Εφαρμογή:

mysales(L,L2,DATE I):- caption(SHARE,N ME), share(SHARE), timeSliceOf(SLICE_1,SIIARE), isTimeInterval(SLICE_l, INTERVAL_!), value(INTERVAL_I,V1), date(INTERVAL_1, DATE-1), timeInterval(INTERVAL_1) , timeSliceOf(SLICE_2,SHARE), isTimeInterval(SLICE_2, INTERVAL_2), value(INTERVAL_2, V2), date(INPERVAL 2,DATE1),timeInterval(INPERVAL_2), Vl>V2+0.1*V2,before(DATE2,DATE_1), not((timeSliceOf(SLICE_3,SHARE), isTimehterval(SLICE3, IN-TERVAL_3), value(INTERVAL_ 3,V3), date(INTERVAL3,DATE3), timeInterval(INPERVAL3), V3\= V1, before(DATE2,DATE3),before(DATE3, DATE-!))), not(member(NAME,L2)),mysales(L,[NAME L2],DATE_1).mysales(L,L,DATE_1):-!.

Αυτό το κατηγόρημα σαρώνει τη λίστα με τις λεζάντες και επιτυγχάνει μόνο όταν ικανοποιούνται κάποιοι κανόνες. Πρώτα θα πρέπει να υπάρχει μια μετοχή που να συνδέεται με δύο διαφορετικά χρονικά διαστήματα έχοντας μια διαφορετική αξία σε κάθε διάστημα. Αυτό σημαίνει ότι αν έχουμε μια μετοχή **SHARE** , που συνδέεται με τη χρονική φέτα **SLICE_1**, τότε αυτή η μετοχή θα πρέπει επίσης να συνδέεται με μια άλλη χρονική φέτα **SLICE_2**. Η **SLICE_1** συνδέεται με ένα χρονικό διάστημα που αντιστοιχεί σε μια τιμή **(V1)** και σ' ένα χρονικό αποτύπωμα **(DATE_1)**. Η **SLICE_2** συνδέεται με ένα διαφορετικό χρονικό διάστημα που αντιστοιχεί σε μια διαφορετική τιμή (V2) κι ένα διαφορετικό χρονικό αποτύπωμα **(DATE_2)**. Οι τιμές **V1** και **V2** αναφέρονται στην τιμή της μετοχής **SHARE** σε δύο

διαφορετικά χρονικά αποτυπώματα **DATE_1** και **DATE_2**. Υπάρχουν ακόμα 3 περιορισμοί:

1) **V1>V2+0.1*V2** που σημαίνει ότι η τιμή της μετοχής στη **DATE_1** θα πρέπει να αυξηθεί για τουλάχιστον 10% επί της αξίας της μετοχής κατά την **DATE_2**

2) Ως συνέπεια αυτού, η **DATE_2** πρέπει να είναι η πιο πρόσφατη από τις δύο ημερομηνίες για να ικανοποιείται το κατηγόρημα **before/2**: **before (DATE_2, DATE_1)**

3) Υπάρχει περίπτωση οι δύο περιορισμοί που αναφέρονται παραπάνω να ικανοποιούνται αλλά η μετοχή να αλλάζει τιμή κατά το χρονικό διάστημα **[DATE_2, DATE_1]**. Για μειώσουμε αυτήν την πιθανότητα χρειαζόμαστε ένα τρίτο περιορισμό. Αν υπάρχει η χρονική φέτα **SLICE_3**, που να αναφέρεται σε ένα άλλο χρονικό διάστημα INTERVAL_3 το οποίο να αντιστοιχεί δε άλλη τιμή **V3** και μια άλλη **DATE_3** όπου **V3≠V1** και η **DATE_3** ανήκει στο χρονικό διάστημα **[DATE_2, DATE_1]**, οπότε το κατηγόρημα αποτυγχάνει.

Το κατηγόρημα **run**

run(DateList,L1,L2,L3):- date(Date) , not(member(Date,D'eList), append([Date],DateList,L) , holding(L1,[],Date) , nl, (eventHappen(X,Date,L2);write('No Events!')), sales(L3,Date),write("Raised Value By 10% :"), write(L3),nl, run(L,L4,L5,L6).

Το κατηγόρημα run/4 είναι ο προσομοιωτής για το σύστημα διαχείρισης μετοχών. Το κατηγόρημα λειτουργεί βασιζόμενο στον ακόλουθο αλγόριθμο (σχήμα 7):

1) Αναζήτηση στην βάση δεδομένων για μια νέα ημερομηνία εγγραφής.

2) Εξέταση αν αυτή η ημερομηνία μπορεί να βρεθεί στην DateList. Αν ναι τότε πηγαίνετε στο 5 αλλιώς πηγαίνετε στο 3.

3) Τρέξτε διαδοχικά τα κατηγορήματα holding/3, eventHappen/3 και sales/2 με νέα ημερομηνία που θα αποτελεί το όρισμα, και γράψτε τα αποτελέσματα στην έξοδο.

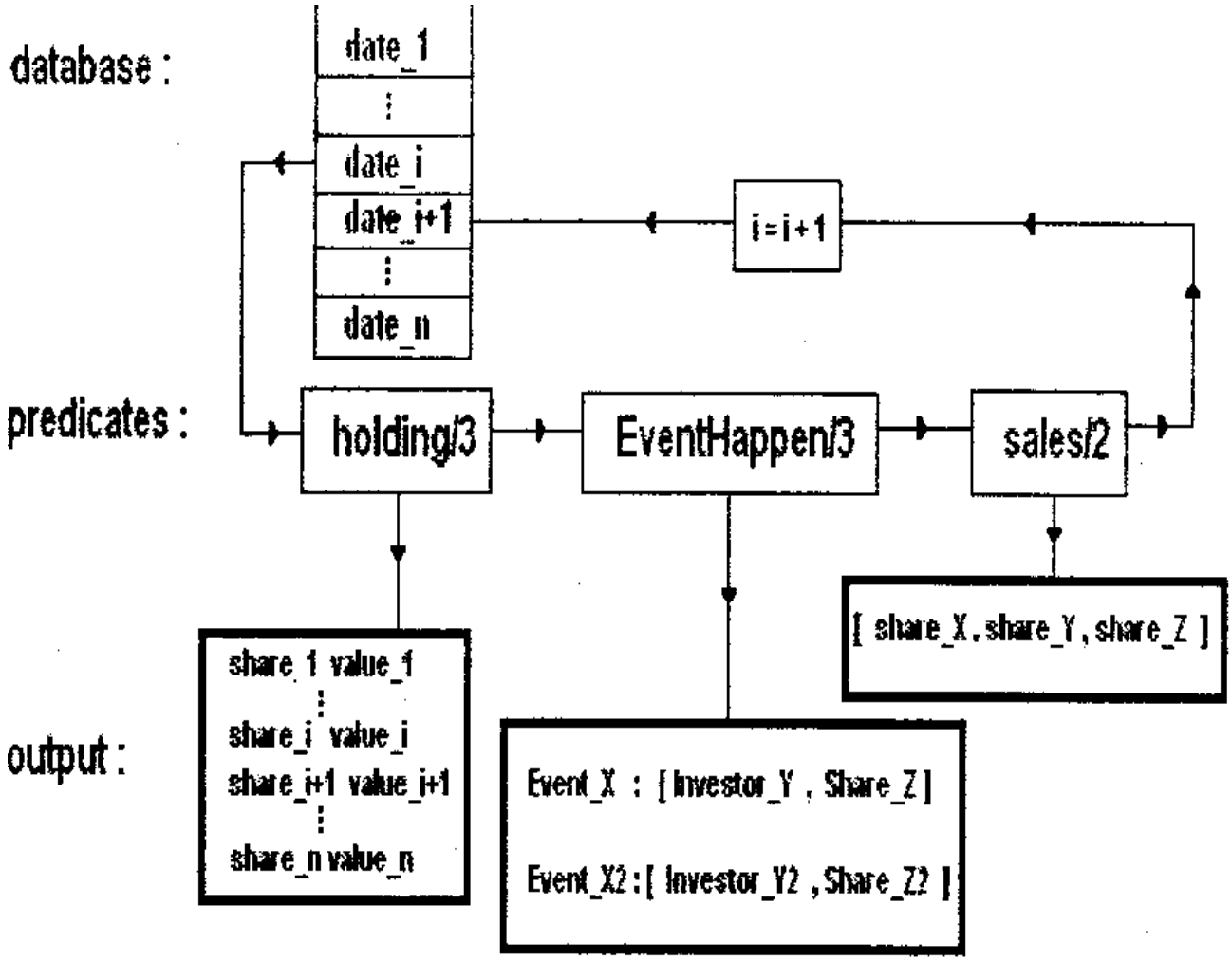
4) Τρέξτε το κατηγόρημα run/4 με την ενημερωμένη DateList.

5) Βρείτε την date2 ≠ date στην βάση δεδομένων. Αν όχι τότε τελειώνετε.

Συμπέρασμα

Σε αυτή τη δημοσίευση παρουσιάσαμε το PROTON, ένα πρόγραμμα συλλογιστικής που εφαρμόζεται στην Prolog για τη διαχείριση της χρονικής πληροφορίας αναφορικά με τις οντολογίες OWL. Περιγράψαμε (α) πώς μετατρέψαμε τις οντολογίες σε ισότιμες τριπλέτες με το εργαλείο SWI-Prolog, (β) πώς τις μετατρέψαμε σε κατηγορήματα Prolog έτσι ώστε να υπάρχει αντιστοιχία με το λογισμό της χρονικής κατάστασης (γ) το εφαρμόσαμε στην Prolog.

Ως μελλοντική εργασία σκοπεύουμε να διευρύνουμε το πρόγραμμα συλλογιστικής ώστε να μπορούμε να χρησιμοποιήσουμε μια σειρά περιορισμών ως είσοδο και να παράγουμε αυτόματα μια σειρά κανόνων που να εφαρμόζονται στην Prolog τα οποία να αναφέρονται στο πρόβλημα των παραφυάδων.



Σχήμα 7. *Ο προσομοιωτής*

ΑΝΑΦΟΡΕΣ

[1] Tobias Matzner, Pascal Hitzler "Any-World Access to OWL from Prolog" KI 2007,LNCS τομ. 4667,σελ. 84-98.

[2] Allen, J.F.: Maintaining Knowledge about Temporal Intervals.Cornmunications of the ACM 26), 832843,1983

[3] OWL-Thea http://www.semanticweb.gr/TheaOWLLib/

[4] Dimitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In IJCAR 2006, τομ. 4130 LNAζP σελ. 292-297.

[5] Bossam OWL reasoner http://bossam.wordpress_com/

[6] Pellet an owl-reasoner, http://pellet.owldl.com/

[7] SeRQL, http://www.openrdf.org/doc/sesame/users/.

[8] Amineh Fadhil, Volker Haarslev, "A Visual Query Language for OWL Ontologies", 2007

[9] Nikos Papadakis,Dimitris Plexousakis.Action with Duration and Constraints: The Ramification problem in Temporal Databases. IJTAI σελ.315-353, τομ. 12,2003.

[10] J. Avery, J. Vearwood dOWL: A DYNAMIC ONTOLOGY LANGUAGE School of Information Technology and Mathematical Sciences Univ. Ballarat,2003

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

[12] Christopher Welty , Richard Fikes and Selene Makarios A Reusable Ontology for Fluents in OWL", IBM Research paper, 2004.

[13] F. Bander, D. Calvanese, D. Mc Guinness, D. Nardi, and P. Patel-Schneider, editors. The Description Logic Handbook. Cambridge Press, 2002.

[14] R.A. Kowalski, F. Sadri, Reconciling the event calculus with the situation calculus, Journal of Logic Programming 31 (1-3) (1997) 39-58.

[15] E. Giunchiglia, V. Lifschitz, Action languages, temporal action logics and the situation calculus, Link. Elec. Art. in CS 4 (040) (1999).

[16] R. Reiter, Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems, MIT Press, Cambridge, MA, 2001.

[17] C. Elkan. Reasoning about action in first order logic. Proceedings of the Conference of the Canadian Society for Comptutational Studies in Intelligence (CSCSI), pp 221-227, Vancouver, May 1992.

[18]Edmund Clarke, Jeannette Wing, Formal Methods: State of the Art and Future Directions, ACM Computing Surveys, Vol. 28, No. 4, December 1996, pp. 626-643.

[19] Marc Denecker and Eugenia Ternovska. Inductive situation calculus,Artificial Intelligence archive Volume 171 , Issue 5-6, Pages: 332-360 April 2007

[20]M. Ginsberg and D. Smith. Reasoning about action I: A possible worlds approach. Artificial Intelligence, 35:165-195, 1988.

[21] A. Fusaoka. Situation Calculus on a Dense Flow of Time, Proceedings of AAAI-96, pp.
633-638, 1996

[22] Peter Lindsay, A Survey of Mechanical Support for Formal Reasoning, Software Engineering Journal, vol. 3, no. 1, January 1988, pp.3-27.

[23]A.C. Kakas, R.S. Miller and F. Toni, E-RES: Reasoning about Actions, Events and Observations, in Proceedings of LPNMR2001, pp. 254-266, Springer Verlag, 2001.

[24] Antonis Kakas and Rob Miller, A Simple Declarative Language for Describing Narratives with Actions, The Journal of Logic Programming, Vol 31(1-3) (Special Issue on Reasoning about Action and Change), pages 157-200, Elsevier, 1997.

[25]R.A. Kowalski. Database updates in the event calculus. Journal of Logic Programming, 1992.

[26]V. Lifshitz. Towards a metatheory of action. In J.F. Allen, R. Fikes, and E. Sandewall, editors, Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, pages 376-386, Cambridge, MA, 1991.

[27] V. Lifshitz. Frames in the space of situations, Artificial Intelligence, 46:365-376, 1990.

[28] V. Lifschitz. Towards a metatheory of action. Proceedings of KR'91, pp. 376-386, Cambridge, MA, 1991.

[29] N. McCain and H. Turner. A causal theory of ramifications and qualifications. Proceedings of IJCAI-95, pp. 1978-1984, Montreal, Canada, August 1995.

[30] J. McCarthy and P.J. Hayes. Some philophical problem from the standpoint of artificial intelligence. In B. Meltzer and D. Mitchie, editors, Machine Intelligence 4, pp. 463-502. American Elsevier, 1969.

[31] Rob Miller and Murray Shanahan. The Event Calculus in Classical Logic - Alternative Axiomatisations. Linkping Electronic Articles in Computer and Information Science, 4(16), 1999.

[32] J. Pinto. Temporal Reasoning in the Situation Calculus. Ph.D. Thesis, Dept. of Computer Science, Univ. of Toronto, Jan. 1994.

[33] J. Pinto and R. Reiter. Temporal Reasoning in Logic Programming: A Case for the Situation Calculus, Proceedings of 10th Int. Conf. on Logic Programming, Budapest, Hungary, June 21-24, 1993.

[34] M. Thielscher. Ramification and causality. Artifical Intelligence, 89(1-2):317-364, 1997.

[35] M. Thielscher. Reasoning about actions: Steady versus stabilizing state constraints. Artifical Intelligence, 104:339-355, 1988.

[36] M. Winslett. Reasoning about action using a possible models approach. Proceedings of AAAI-88, pp. 89-93, Saint Paul, MN, August 1988.

ΠΑΡΑΡΤΗΜΑ

```
%   Technological Education  Institute Of Crete –Applied information and multimedia department
%
%           KEMESIDIS THODORIS A.M:955-PAIDARAKIS EYTIXIS A.M:1116
%
%     ***********************************************
%    *          SHARE MANAGMENT SYSTEM          *
%     ***********************************************
%
%
% Tested and runned at SWI-Prolog.
%


timeInterval('timeInterval_1').
timeInterval('timeInterval_2').
timeInterval('timeInterval_3').
timeInterval('timeInterval_4').
timeInterval('timeInterval_5').
timeInterval('timeInterval_6').
timeInterval('timeInterval_7').
```

```
timeInterval('timeInterval_8').
timeInterval('timeInterval_9').
timeInterval('timeInterval_10').

timeInterval('timeInterval_11').
timeInterval('timeInterval_12').
timeInterval('timeInterval_13').
timeInterval('timeInterval_14').
timeInterval('timeInterval_15').
timeInterval('timeInterval_16').
timeInterval('timeInterval_17').
timeInterval('timeInterval_18').
timeInterval('timeInterval_19').
timeInterval('timeInterval_20').

timeInterval('timeInterval_21').
timeInterval('timeInterval_22').
timeInterval('timeInterval_23').
timeInterval('timeInterval_24').

/* Value */

/* share 1 */
value('timeInterval_1',10).
date('timeInterval_1','2008-06-30T07:00:00').
isTimeInterval('timeSlice_1','timeInterval_1').
timeSliceOf('timeSlice_1','share_1').
```

```
value('timeInterval_2',60).
date('timeInterval_2','2008-06-30T09:00:00').
isTimeInterval('timeSlice_2','timeInterval_2').
timeSliceOf('timeSlice_2','share_1').

value('timeInterval_3',30).
date('timeInterval_3','2008-06-30T10:00:00').
isTimeInterval('timeSlice_3','timeInterval_3').
timeSliceOf('timeSlice_3','share_1').

value('timeInterval_4',10).
date('timeInterval_4','2008-06-30T13:00:00').
isTimeInterval('timeSlice_4','timeInterval_4').
timeSliceOf('timeSlice_4','share_1').

value('timeInterval_5',40).
date('timeInterval_5','2008-06-30T14:00:00').
isTimeInterval('timeSlice_5','timeInterval_5').
timeSliceOf('timeSlice_1','share_1').

/* share 2 */


value('timeInterval_6',20).
date('timeInterval_6','2008-06-30T07:00:00').
isTimeInterval('timeSlice_6','timeInterval_6').
```

timeSliceOf('timeSlice_6','share_2').

value('timeInterval_7',30).
date('timeInterval_7','2008-06-30T08:00:00').
isTimeInterval('timeSlice_7','timeInterval_7').
timeSliceOf('timeSlice_7','share_2').

value('timeInterval_8',10).
date('timeInterval_8','2008-06-30T09:00:00').
isTimeInterval('timeSlice_8','timeInterval_8').
timeSliceOf('timeSlice_8','share_2').

value('timeInterval_9',40).
date('timeInterval_9','2008-06-30T11:00:00').
isTimeInterval('timeSlice_9','timeInterval_9').
timeSliceOf('timeSlice_9','share_2').

value('timeInterval_10',10).
date('timeInterval_10','2008-06-30T12:00:00').
isTimeInterval('timeSlice_10','timeInterval_10').
timeSliceOf('timeSlice_10','share_2').

value('timeInterval_11',40).
date('timeInterval_11','2008-06-30T13:00:00').
isTimeInterval('timeSlice_11','timeInterval_11').
timeSliceOf('timeSlice_11','share_2').

```
value('timeInterval_12',60).
date('timeInterval_12','2008-06-30T14:00:00').
isTimeInterval('timeSlice_12','timeInterval_12').
timeSliceOf('timeSlice_12','share_2').

/* share 3 */

value('timeInterval_13',0).
date('timeInterval_13','2008-06-30T07:00:00').
isTimeInterval('timeSlice_13','timeInterval_13').
timeSliceOf('timeSlice_13','share_3').

value('timeInterval_14',20).
date('timeInterval_14','2008-06-30T08:00:00').
isTimeInterval('timeSlice_14','timeInterval_14').
timeSliceOf('timeSlice_14','share_3').

value('timeInterval_15',40).
date('timeInterval_15','2008-06-30T09:00:00').
isTimeInterval('timeSlice_15','timeInterval_15').
timeSliceOf('timeSlice_15','share_3').

value('timeInterval_16',70).
date('timeInterval_16','2008-06-30T10:00:00').
isTimeInterval('timeSlice_16','timeInterval_16').
timeSliceOf('timeSlice_16','share_3').
```

```
value('timeInterval_17',90).
date('timeInterval_17','2008-06-30T11:00:00').
isTimeInterval('timeSlice_17','timeInterval_17').
timeSliceOf('timeSlice_17','share_3').

value('timeInterval_18',70).
date('timeInterval_18','2008-06-30T12:00:00').
isTimeInterval('timeSlice_18','timeInterval_18').
timeSliceOf('timeSlice_18','share_3').

/* share 4 */

value('timeInterval_19',60).
date('timeInterval_19','2008-06-30T07:00:00').
isTimeInterval('timeSlice_19','timeInterval_19').
timeSliceOf('timeSlice_19','share_4').

value('timeInterval_20',90).
date('timeInterval_20','2008-06-30T08:00:00').
isTimeInterval('timeSlice_20','timeInterval_20').
timeSliceOf('timeSlice_20','share_4').

value('timeInterval_21',70).
date('timeInterval_21','2008-06-30T09:00:00').
isTimeInterval('timeSlice_21','timeInterval_21').
timeSliceOf('timeSlice_21','share_4').
```

value('timeInterval_22',40).

date('timeInterval_22','2008-06-30T10:00:00').

isTimeInterval('timeSlice_22','timeInterval_22').

timeSliceOf('timeSlice_22','share_4').

value('timeInterval_23',50).

date('timeInterval_23','2008-06-30T11:00:00').

isTimeInterval('timeSlice_23','timeInterval_23').

timeSliceOf('timeSlice_23','share_4').

value('timeInterval_24',80).

date('timeInterval_24','2008-06-30T14:00:00').

isTimeInterval('timeSlice_24','timeInterval_24').

timeSliceOf('timeSlice_24','share_4').

timeSlice('timeSlice_1').

timeSlice('timeSlice_2').

timeSlice('timeSlice_3').

timeSlice('timeSlice_4').

timeSlice('timeSlice_5').

timeSlice('timeSlice_6').

timeSlice('timeSlice_7').

timeSlice('timeSlice_8').

timeSlice('timeSlice_9').

timeSlice('timeSlice_10').

timeSlice('timeSlice_11').

```
timeSlice('timeSlice_12').
timeSlice('timeSlice_13').
timeSlice('timeSlice_14').
timeSlice('timeSlice_15').
timeSlice('timeSlice_16').
timeSlice('timeSlice_17').
timeSlice('timeSlice_18').
timeSlice('timeSlice_19').
timeSlice('timeSlice_20').
timeSlice('timeSlice_21').
timeSlice('timeSlice_22').
timeSlice('timeSlice_23').
timeSlice('timeSlice_24').

share('share_1').
share('share_2').
share('share_3').
share('share_4').


caption('share_1','metoxh_1').
caption('share_2','metoxh_2').
caption('share_3','metoxh_3').
caption('share_4','metoxh_4').


investor('investor_1').
```

```
investor('investor_2').
investor('investor_3').
investor('investor_4').

/* EVENT FACTS */


event('investor1getShare1').
event('investor2getShare2').
event('investor3getShare3').
event('investor3getShare4').
event('investor4getShare1').

isTimeSliceOf('getSlice_1','investor1getShare1').
isTimeSliceOf('getSlice_2','investor2getShare2').
isTimeSliceOf('getSlice_3','investor3getShare3').
isTimeSliceOf('getSlice_4','investor3getShare4').
isTimeSliceOf('getSlice_5','investor4getShare1').


timeSlice('investorSlice_1').
timeSlice('investorSlice_2').
timeSlice('investorSlice_3').
timeSlice('investorSlice_4').
timeSlice('investorSlice_5').
```

timeSlice('getSlice_1').

timeSlice('getSlice_2').

timeSlice('getSlice_3').

timeSlice('getSlice_4').

timeSlice('getSlice_5').

timeSlice('shareSlice_1').

timeSlice('shareSlice_2').

timeSlice('shareSlice_3').

timeSlice('shareSlice_4').

timeSlice('shareSlice_5').

shareBuyer('getSlice_1','investorSlice_1').

shareBuyer('getSlice_2','investorSlice_2').

shareBuyer('getSlice_3','investorSlice_3').

shareBuyer('getSlice_4','investorSlice_4').

shareBuyer('getSlice_5','investorSlice_5').

shareBuyed('getSlice_1','shareSlice_1').

shareBuyed('getSlice_2','shareSlice_2').

shareBuyed('getSlice_3','shareSlice_3').

shareBuyed('getSlice_4','shareSlice_4').

shareBuyed('getSlice_5','shareSlice_5').

isTimeSliceOf('getSlice_1','shareSlice_1').

isTimeSliceOf('getSlice_2','shareSlice_2').

isTimeSliceOf('getSlice_3','shareSlice_3').

isTimeSliceOf('getSlice_4','shareSlice_4').

isTimeSliceOf('getSlice_5','shareSlice_5').

isTimeSliceOf('investorSlice_1','investor_1').

isTimeSliceOf('investorSlice_2','investor_2').

isTimeSliceOf('investorSlice_3','investor_3').

isTimeSliceOf('investorSlice_4','investor_3').

isTimeSliceOf('investorSlice_5','investor_4').

isTimeSliceOf('shareSlice_1','share_1').

isTimeSliceOf('shareSlice_2','share_2').

isTimeSliceOf('shareSlice_3','share_3').

isTimeSliceOf('shareSlice_4','share_4').

isTimeSliceOf('shareSlice_5','share_1').

isTimeInterval('getSlice_1','timeInterval_1').

isTimeInterval('getSlice_2','timeInterval_8').

isTimeInterval('getSlice_3','timeInterval_17').

isTimeInterval('getSlice_4','timeInterval_23').

isTimeInterval('getSlice_5','timeInterval_5').

isTimeInterval('investorSlice_1','timeInterval_1').

isTimeInterval('investorSlice_2','timeInterval_8').

isTimeInterval('investorSlice_3','timeInterval_17').

isTimeInterval('investorSlice_4','timeInterval_23').

isTimeInterval('investorSlice_5','timeInterval_5').

```
isTimeInterval('shareSlice_1','timeInterval_1').
isTimeInterval('shareSlice_2','timeInterval_8').
isTimeInterval('shareSlice_3','timeInterval_17').
isTimeInterval('shareSlice_4','timeInterval_23').
isTimeInterval('shareSlice_5','timeInterval_5').



%            Predicate :  Check/2
%
```

```
% This predicate checks one by one the characters in the
% argument Lists and compares their ASCII values
%

check([A|L],[B|G]):- A<B.
check([A|L],[A|G]):- check(L,G).



%            Predicate :  Before/2
```

```
%
```

```
% This predicate is using the predicate Check/2 to define
% the if Date X is before Date Y.
%
%


before(X,Y):-name(X,L),name(Y,G),check(L,G).




%              Predicate :  Value/3
%
```

```
%
% Checks if Caption Name has a value of V at the date Date.
%
%


value(Name,V,Date):-caption(Share,Name) ,share(Share),
(date(Interval,Date),isTimeInterval(Slice,Interval);isTimeInterval(Sli
ce,Interval)),timeSliceOf(Slice,Share),
newpr(Interval,Date,Share,V),
timeInterval(Interval).
```

```
%               Predicate :  Hold/3
%
```

```
%
% Simmilar to the Value/3 predicate, this predicate checks
% if the caption X maintains value A at the Time Interval [T,E]
%


hold(X,A,[T,E]):-value(X,A,T),not(value(X,A,E)),
not( (value(X,B,F),B\=A,before(F,E),before(T,F) )).



%               Predicate :  Event_Happen/3
%
```

```
%
%  This predicate defines the share and investor who are
%  involved in the transaction that is described by the Event X.
%  The names of the Caption and the Investor are apended into
%  the List.
```

```
        %
        %


        eventHappen(X,Time,List):-
event(X),isTimeSliceOf(GETSLICE,X),date(INTERVAL,Time),timeInterval(I
NTERVAL),
        timeSlice(GETSLICE)
,timeSlice(SHARESLICE),isTimeSliceOf(SHARESLICE,SHARE),
        isTimeInterval(SHARESLICE,INTERVAL), share(SHARE)
,caption(SHARE,NAME),
        shareBuyed(GETSLICE,SHARESLICE),
shareBuyer(GETSLICE,INVESTORSLICE),isTimeSliceOf(GETSLICE,SHARE
SLICE) ,
        isTimeInterval(GETSLICE,INTERVAL),isTimeInterval(INVESTORSLI
CE,INTERVAL),isTimeSliceOf(INVESTORSLICE,INVESTOR),
        timeSlice(INVESTORSLICE),investor(INVESTOR),write('Event :
'),write(X),write(' Involving : '),
        List = [INVESTOR,NAME],write(List),nl.


        %               Predicate :  Sales/3
        %
_____________________________________________________________________
_____
        %
        %  The sales predicate scans the caption list for shares that
```

```
        %   have raised their value 10% until date DATE_1. Every share
        %   that satisfies the predicate is added into the 'L' list.
        %


        sales(L,DATE_1):-mysales(L,[],DATE_1),!.

        mysales(L,L2,DATE_1):-
caption(SHARE,NAME),share(SHARE),timeSliceOf(SLICE_1,SHARE),
        isTimeInterval(SLICE_1,INTERVAL_1),value(INTERVAL_1,V1),date(I
NTERVAL_1,DATE_1),
        timeInterval(INTERVAL_1) ,
        timeSliceOf(SLICE_2,SHARE),
isTimeInterval(SLICE_2,INTERVAL_2),value(INTERVAL_2,V2),date(INTER
VAL_2,DATE_2),
        timeInterval(INTERVAL_2),
        V1 > V2 + 0.1 * V2 , before(DATE_2,DATE_1),
        not(
(timeSliceOf(SLICE_3,SHARE),isTimeInterval(SLICE_3,INTERVAL_3),valu
e(INTERVAL_3,V3),date(INTERVAL_3,DATE_3),
        timeInterval(INTERVAL_3), V3\= V1,
before(DATE_2,DATE_3),before(DATE_3,DATE_1))),
        not(member(NAME,L2)),mysales(L,[NAME|L2],DATE_1) .

        mysales(L,L,DATE_1):-!.
```

```
        find_last_value(Share,Value,Date):-
share(Share),timeSliceOf(Slice,Share),
        isTimeInterval(Slice,Interval),value(Interval,Value),date(Interval,Dat
e_1),before(Date_1,Date),
        not( (share(Share),timeSliceOf(Slice_n,Share),
        isTimeInterval(Slice_n,Interval_n),value(Interval_n,Value_n),date(Int
erval_n,Date_n),before(Date_1,Date_n),
        before(Date_n,Date))).


        %               Predicate :  holding/3
        %
_____________________________________________________________________
_____
        %
        %   Prints the value of all the captions at the current time.
        %
        %



        holding(L,L2,Date):-
caption(Share,Name),not(member([Name,Date],L2)),share(Share),
```

```
        (date(Interval,Date),isTimeInterval(Slice,Interval);isTimeInterval(Sli
ce,Interval)),timeSliceOf(Slice,Share),
        newpr(Interval,Date,Share,V),
        timeInterval(Interval),write('caption : '),write(Name),write('  value :
'),write(V),nl,
        holding(L,[[Name,Date]|L2],Date),!.
        holding(L,L,Date):-!.

        newpr(Interval,Date,Share,V):-  date(Interval,Date) ,
value(Interval,V).
        newpr(Interval,Date,Share,V):-find_last_value(Share,V,Date).


        %               Predicate :  run_test/1
        %
_____________________________________________________________________
_____
        %
        %    The simulation program.
        %
        %

        run:-     holding(L,[],'2008-06-30T07:00:00'),nl,
        (eventHappen(X1,'2008-06-30T07:00:00',L2);write('No Events!')),
        sales(L3,'2008-06-30T07:00:00'),write('Raised Value By 10%
:'),write(L3),nl,
```

```
        holding(L4,[],'2008-06-30T08:00:00'),nl,
        (eventHappen(X2,'2008-06-30T08:00:00',L5);write('No Events!')),
        sales(L6,'2008-06-30T08:00:00'),write('Raised Value By 10%
:'),write(L6),nl,

        holding(L7,[],'2008-06-30T09:00:00'),nl,
        (eventHappen(X3,'2008-06-30T09:00:00',L8);write('No Events!')),
        sales(L9,'2008-06-30T09:00:00'),write('Raised Value By 10%
:'),write(L9),nl,

        holding(L10,[],'2008-06-30T10:00:00'),nl,
        (eventHappen(X4,'2008-06-30T10:00:00',L11);write('No Events!')),
        sales(L12,'2008-06-30T10:00:00'),write('Raised Value By 10%
:'),write(L12),nl,

        holding(L13,[],'2008-06-30T11:00:00'),nl,
        (eventHappen(X5,'2008-06-30T11:00:00',L14);write('No Events!')),
        sales(L15,'2008-06-30T11:00:00'),write('Raised Value By 10%
:'),write(L15),nl,

        holding(L16,[],'2008-06-30T12:00:00'),nl,
        ( eventHappen(X,'2008-06-30T12:00:00',L17);write('No Events!')),
        sales(L18,'2008-06-30T12:00:00'),write('Raised Value By 10%
:'),write(L18),nl,

        holding(L19,[],'2008-06-30T13:00:00'),nl,
        ( eventHappen(X,'2008-06-30T13:00:00',L20);write('No Events!')),
```

```
        sales(L21,'2008-06-30T13:00:00'),write('Raised Value By 10%
:'),write(L21),nl,

        holding(L22,[],'2008-06-30T14:00:00'),nl,
        (eventHappen(X,'2008-06-30T14:00:00',L23);write('No Events!')),
        sales(L24,'2008-06-30T14:00:00'),write('Raised Value By 10%
:'),write(L24),nl.



        /*
        metoxh1(L):-met(L,[]),!.

        met(L,L2):-
caption(Share,Name),not(member(Name,L2)),met(L,[Name|L2]).
        met(L,L):-!.

        */
```