ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αυτοματοποιημένος σχεδιασμός ενεργειών για πολλούς πράκτορες, βασισμένος σε προτιμήσεις γραμμικής χρονικής λογικής (LTL)

Σταύρος Ι. Πετσαλάκης

Επιβλέποντες: **Ιωάννης Ιωαννίδης,** Καθηγητής Τμήματος Πληροφορικής & Τηλεπικοινωνιών του ΕΚΠΑ
Σωτήρης Λιάσκος, Καθηγητής Τμήματος Πληροφορικής του York University, Τορόντο, Καναδάς

ΑΘΗΝΑ

Ιούλιος 2015

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αυτοματοποιημένος σχεδιασμός ενεργειών για πολλούς πράκτορες, βασισμένος σε προτιμήσεις γραμμικής χρονικής λογικής (LTL)

ΣΤΑΥΡΟΣ Ι. ΠΕΤΣΑΛΑΚΗΣ

Α.Μ.: 1115200900236

Επιβλέποντες: **Ιωάννης Ιωαννίδης,** Καθηγητής Τμήματος Πληροφορικής & Τηλεπικοινωνιών του ΕΚΠΑ
Σωτήρης Λιάσκος, Καθηγητής Τμήματος Πληροφορικής του York University, Τορόντο, Καναδάς

ΠΕΡΙΛΗΨΗ

Ο Αυτοματοποιημένος Σχεδιασμός, είναι ο τομέας της Τεχνητής Νοημοσύνης που ασχολείται με την επιλογή των κατάλληλων ενεργειών τις οποίες πρέπει να κάνει ένας πράκτορας ή αλλιώς παράγοντας (actor), ώστε να βρεθεί σε μια επιθυμητή κατάσταση. Ειδικότερα, στη σχετική βιβλιογραφία, υπάρχουν πολλά εργαλεία τα οποία ασχολούνται με τον αυτοματοποιημένο σχεδιασμό, κυρίως σε περιπτώσεις σχεδιασμού βασισμένου σε προτιμήσεις, αλλά και σε περιπτώσεις σχεδιασμού για πολλούς πράκτορες. Παρ' όλα αυτά, υπάρχει έλλειψη στον τομέα που συνδυάζει τους παραπάνω, δηλαδή τον σχεδιασμό με προτιμήσεις για πολλούς πράκτορες. Αυτός ο τομέας παρουσιάζει ιδιαίτερο ενδιαφέρον, γιατί εφαρμογές πάνω σε αυτόν είναι άμεσα χρησιμοποιήσιμες σε καθημερινές εφαρμογές, όπως την συλλογική λήψη απόφασης για κάποιο σχέδιο δράσης. Η συνεισφορά της παρούσας εργασίας είναι ένας τρόπος φορμαλισμού τέτοιων προβλημάτων, ώστε να δοθεί η δυνατότητα επίλυσής τους από ήδη υπάρχοντα εργαλεία, δημιουργώντας σχέδια με ένα κεντρικό τρόπο, λαμβάνοντας υπ' όψιν τις προτιμήσεις του κάθε παράγοντα που συμμετέχει στο σχέδιο. Αυτός ο τρόπος φορμαλισμού εφαρμόστηκε με επιτυχία σε διαφορετικές περιπτώσεις ενός παραδείγματος λήψης αποφάσεων: την διαδικασία που θα ακολουθήσει μια επιτροπή όταν καλούνται όλα τα μέλη της να κρίνουν μία επιστολή που τους παραθέτει ο πρόεδρος της επιτροπής.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνητή Νοημοσύνη, Αυτοματοποιημένος Σχεδιασμός Ενεργειών

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Σχεδιασμός ενεργειών, σχεδιασμός πολλών πρακτόρων, προτιμήσεις, στόχοι, προγραμματισμός με περιορισμούς

ABSTRACT

Automated Planning is the area of Artificial Intelligence that deals with choosing the required sequence of actions of an agent or actor to achieve a desired state. Specifically, in the relative bibliography, there are many tools that deal with planning, mostly in cases of preference-based planning or in cases of multi-agent planning. However, there seems to be a lack of research in the area combining the above, that is, preference-based planning for many actors. This area presents particular interest, because such applications are directly applicable to everyday situations, such as collective decision making about an action plan. The contribution of the current thesis is a way of formalizing such problems, so as to allow their solution using existing tools, creating plans in a centralized way, taking into account the preferences of each actor participating in the plan. As a proof of principle this approach was applied successfully in different instances of an example of decision making: specifically the process to be taken by a committee when all members are called to comment officially on a letter given by the chairman of the committee.

SUBJECT AREA: Artificial Intelligence, Automated Planning and Scheduling

KEYWORDS: automated planning , multi-agent planning, preferences, goals, constraint programming

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Αυτή η εργασία εκπονήθηκε κατα την διάρκεια του ακαδημαικού έτους 2014-2015, στο πλαίσιο της ολοκλήρωσης των σπουδών μου στο Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών στο τμήμα Πληροφορικής και Τηλεπικοινωνιών. Η εργασία αυτή έγινε σε συνεργασία με τον κ. Σωτήρη Λιάσκο, καθηγητή του York University στο Τορόντο, Καναδά, υπό την επίβλεψη του κ. Ιωάννη Ιωαννίδη, καθηγητή μου στο τμήμα Πληροφορικής του ΕΚΠΑ.

Ήταν μια πολύ καλή ευκαιρία για εμένα γιατί συνεργαζόμουν με άτομα πλήρως καταρτισμένα στον χώρο, και έμαθα πολλά, τόσο για το θέμα, όσο και για την ερευνητική διαδικασία καθ' αυτή.

Εκφράζω τις ευχαριστίες μου στούς κυρίους καθηγητές, κ. Σωτήρη Λιάσκο και κ. Ιωάννη Ιωαννίδη για την επίβλεψη της εργασίας αυτής.

1. ΕΙΣΑΓΩΓΗ

Ο Αυτοματοποιημένος Σχεδιασμός [1,2,3] είναι ένας κλάδος της Τεχνητής Νοημοσύνης ο οποίος ασχολείται με το πρόβλημα της παραγωγής ενός συνόλου από ενέργειες προς την επίτευξη κάποιας καθορισμένης κατάστασης στόχου, δεδομένης μιας αρχικής κατάστασης του κόσμου. Είναι μία ενεργή περιοχή έρευνας, κεντρική για την ανάπτυξη ευφυειών πρακτόρων και αυτόνομων ρομπότ.

Ο όρος **τεχνητή νοημοσύν**η αναφέρεται στον κλάδο της πληροφορικής ο οποίος ασχολείται με τη σχεδίαση και την υλοποίηση υπολογιστικών συστημάτων που μιμούνται στοιχεία της ανθρώπινης συμπεριφοράς τα οποία υπονοούν έστω και στοιχειώδη ευφυΐα: μάθηση, προσαρμοστικότητα, εξαγωγή συμπερασμάτων, κατανόηση από συμφραζόμενα, επίλυση προβλημάτων κλπ. οι **ευφυείς πράκτορε**ς, είναι αυτόνομο λογισμικό ΤΝ τοποθετημένο σε κάποιο περιβάλλον με το οποίο αλληλεπιδρά. Οι ευφυείς πράκτορες βρήκαν μεγάλο πεδίο εφαρμογών λόγω της εξάπλωσης του Διαδικτύου. Οι πράκτορες στοχεύουν συνήθως στην παροχή βοήθειας στους χρήστες τους, στη συλλογή ή ανάλυση γιγάντιων συνόλων δεδομένων ή στην αυτοματοποίηση επαναλαμβανόμενων εργασιών ενώ στους τρόπους κατασκευής και λειτουργίας τους συνοψίζουν όλες τις γνωστές μεθοδολογίες ΤΝ που αναπτύχθηκαν με το πέρασμα του χρόνου. Τα αυτόνομα ρομπότ εκτελούν έργα με αυτόνομο τρόπο, κάτι που είναι ιδιαίτερου ενδιαφέροντος σε διάφορες εφαρμογές όπως π.χ. εξερεύνησης του διαστήματος, επεξεργασίας των λυμάτων κ.α.

Στην διαδικασία λήψης αποφάσεων από ανθρώπους, οι προτιμήσεις εχουν κεντρικό ρόλο και επίσης καθοδηγούν και τις επακόλουθες πράξεις τους. Επομένως και στις μεθοδολογίες αυτοματοποιημένου σχεδιασμού ΤΝ οι προτιμήσεις είναι πολύ σημαντικές. Δεδομένου ενός έργου, κάποιος χρήστης μπορεί να έχει προτιμήσεις όσον αφορά τους στόχους αλλά και τις περιστάσεις, τις διαδικασίες και τς συνέπειες. [4]

Στο συμβατικό ή κλασσικό σχεδιασμό, τα δεδομένα είναι μια δυναμική περιοχή, μια αρχική κατάσταση και η περιγραφή των στόχων. Το ζητούμενο είναι ο προσδιορισμός μιας πορείας δράσης (ενός σχεδίου) η οποία οδηγεί από την αρχική κατάσταση σε αυτή όπου οι στόχοι έχουν επιτευχθεί. Δηλαδή, ενώ μπορεί να υπάρχουν διάφοροι τρόποι επίτευξης των στόχων, δεν υπάρχουν κριτήρια στον ορισμό του προβλήματος (στα αρχικά δεδομένα) που να διαφοροποιούν τα προτιμητέα σχέδια από άλλα. Παραδείγματος χάριν, το πρόβλημα της επιστροφής στους χώρους ενοικιάσεως ενός συνόλου ενοικιασμένων αυτοκινήτων μπορεί να επιλυθεί μέσω αυτοματοποιημένου σχεδιασμού, όπου επί πλέον μία βελτιστοποίηση του κόστους επιστροφής εφαρμόζεται.

Ο Σχεδιασμός βασισμένος σε προτιμήσεις [5] είναι μία επέκταση του κλασσικού σχεδιασμού και περιλαμβάνει κριτήρια για τη διαφοροποίηση μεταξύ των διαφορετικών σχεδιασμών και τον προσδιορισμό αυτού που καλύτερα εκπληρώνει τις προτιμήσεις του χρήστη. Στο παραπάνω παράδειγμα, οι προτιμήσεις μπορεί να αφορούν συγκεκριμένου τύπου αυτοκίνητα, παλαιότητα, μεγέθους, να έχουν προτεραιότητα για ορισμένους χώρους στάθμευσης, ή χρονική προτεραιότητα. Ο προσδιορισμός των διαφορετικών προτιμήσεων αποτελεί πεδίο έρευνας στον τομέα του σχεδιασμού βασισμένου σε προτιμήσεις. Απαιτείται να εκφραστεί φορμαλιστικά το κριτήριο του προτιμητέου σχεδιασμού σε σχέση με κάποιο άλλο, καθώς και η αποδοτική παραγωγή προτιμητέων σχεδίων κατά το δυνατόν βέλτιστα σε σχέση με το δοθέν κριτήριο. Δηλαδή, ο ορισμός των προτιμήσεων στον αυτόματο σχεδιασμό απαιτεί τη χρήση κάποιας γλώσσας που επιτρέπει επί πλέον και την συγκέντρωση όλων των προτιμήσεων. Αυτό επιτυγχάνεται μέσω της γραμμικής χρονικής λογικής [6,7,8](Linear Temporal Logic (LTL)), όπως θα παρουσιαστεί παρακάτω.

Σε πολλές εφαρμογές πραγματικού κόσμου, υπάρχει πληθώρα έγκυρων σχεδίων, και ο κάθε χρήστης τα κρίνει ανάλογα με το πόσο καλά υπακούν τις προτιμήσεις του. Όπως αναφέρθηκε πιο πάνω, για την παραγωγή έγκυρων σχεδίων υψηλής ποιότητας αυτόματα, ένα σύστημα αυτοματοποιημένου σχεδιασμού πρέπει να παρέχει έναν τρόπο να καθορίζονται οι προτιμήσεις του χρήστη, σε σχέση με το σχεδιαστικό στόχο, καθώς και έναν τρόπο παραγωγής σχεδίων τα οποία (ιδανικά) βελτιστοποιούν αυτές τις προτιμήσεις. Επιπλέον, πολλές φορές σε προβλήματα πραγματικού κόσμου, χρειάζεται να συνεργαστούν διαφορετικοί παράγοντες για την επίτευξη ενός στόχου, και η ποιότητα ενός σχεδίου προς αυτό, πολλές φορές καθορίζεται από προτιμήσεις διαφορετικών παραγόντων, οι οποίες πιθανώς να είναι και αντικρουόμενες μεταξύ τους.

Τα τελευταία χρόνια, έχει υπάρξει σημαντική έρευνα στην περιοχή του σχεδιασμού με προτιμήσεις. Ωστόσο, δεν υπάρχει ανάλογη δραστηριότητα στο σχεδιασμό πολλών πρακτόρων με προτιμήσεις και σε αυτό τον τομέα ευρίσκεται η συνεισφορά της παρούσας εργασίας. Κατ' αρχήν θα παρουσιάσουμε τις τρέχουσες προσεγγίσεις στην αναπαράσταση των προτιμήσεων για σχεδιασμό, καθώς και μερικές τεχνικές παραγωγής προτιμώμενων σχεδίων που έχουν αναπτυχθεί. Επεκτείνοντας αυτές τις τεχνικές, θα εισάγουμε έναν τρόπο φορμαλισμού προβλημάτων αυτοματοποιημένου σχεδιασμού βασιζόμενου σε προτιμήσεις πολλών πρακτόρων. Αυτή η μέθοδος οδηγεί σε λύση τέτοιων προβλημάτων, χρησιμοποιώντας ήδη υπάρχοντα εργαλεία όπως θα παρουσιαστεί με την εφαρμογή της σε παράδειγμα.

1.1 **ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΟΣ ΣΧΕΔΙΑΣΜΟΣ** [1,2,3,9]

Ο αυτοματοποιημένος σχεδιασμός, στην σχετική βιβλιογραφία συχνά αναφέρεται απλά ως σχεδιασμός (planning) είναι ένας κλάδος της Τεχνητής Νοημοσύνης ο οποίος αφορά την πραγματοποίηση στρατηγικών, ή αλλιώς ακολουθιών από ενέργειες, τυπικά για εκτέλεση από ευφυείς πράκτορες, αυτόνομα ρομπότ, και μη επανδρωμένα οχήματα.

Ο αυτοματοποιημένος σχεδιασμός είναι μια υπολογιστική διαδικασία επιλογής και οργάνωσης δράσεων, προβλέποντας τα αποτελέσματα των δράσεων, που αποσκοπεί στην επίτευξη ορισμένων προ-δηλωμένων στόχων. Ο σχεδιασμός βρίσκεται στο κέντρο του ενδιαφέροντος στην Τεχνητή Νοημοσύνη αφού αφορά και την κατανόηση της νοημοσύνης και τούς στόχους της κατασκευής ευφυών οντοτήτων.

Σε γνωστά περιβάλλοντα με διαθέσιμα μοντέλα, ο σχεδιασμός μπορεί να γίνει έτσι ώστε να βρεθούν και να αξιολογηθούν λύσεις πριν την εκτέλεσή τους. Σε δυναμικά άγνωστα περιβάλλοντα, η στρατηγική συχνά χρειάζεται να αναθεωρηθεί κατά την διάρκεια της εκτέλεσης, καθώς έρχονται νέα δεδομένα. Τα μοντέλα και οι πολιτικές πρέπει να προσαρμοστούν. Λύσεις σ' αυτό συνήθως οδηγούν σε επαναληπτικές διαδικασίες δοκιμής και λάθους, οι οποίες παρατηρούνται συχνά στην τεχνητή νοημοσύνη. Αυτές περιλαμβάνουν μεταξύ άλλων δυναμικό προγραμματισμό, ενισχυτική μάθηση, και συνδυαστική βελτιστοποίηση. Οι γλώσσες οι οποίες χρησιμοποιούνται για να περιγραφεί ο σχεδιασμός και η χρονοδρομολόγηση συχνά λέγονται γλώσσες ενεργειών.

Ο σχεδιασμός είναι επίσης συγγενής με την θεωρία αποφάσεων. Η θεωρία αποφάσεων ασχολείται με την αναγνώριση των αξιών, αβεβαιοτήτων, και άλλων θεμάτων σχετικών σε μία δεδομένη απόφαση, την λογικότητα, και την προκύπτουσα ιδανική απόφαση. Η θεωρία αποφάσεων ασχολείται με τις επιλογές επιμέρους πρακτόρων σε μία αλληλεπίδραση.

Επισκόπηση

Δεδομένης μιας περιγραφής των πιθανών αρχικών καταστάσεων του κόσμου, μιας περιγραφής των δεδομένων στόχων, και μιας περιγραφής ενός συνόλου πιθανών ενεργειών, το πρόβλημα σχεδιασμού είναι να βρεθεί ένα σχέδιο το οποίο είναι εγγυημένο (από κάθε μία από τις αρχικές καταστάσεις) να παράγει μία ακολουθία απο ενέργειες που οδηγούν σε μία από τις καταστάσεις στόχου. Σε αντίθεση με τα κλασσικά προβλήματα ελέγχου ή κατηγοριοποίησης, οι λύσεις είναι περίπλοκες και πρέπει να ανακαλυφθούν και να βελτιστοποιηθούν σε πολυδιάστατο χώρο.

Η τυπική περιγραφή ενός προβλήματος σχεδιασμού περιλαμβάνει ένα σύνολο ενεργειών Α, ένα σύνολο καταστάσεων Σ, το οποίο εκφράζεται μέσω ενός συνόλου μεταβλητών Χ, και δίνεται μια αρχική κατάσταση Σ0, καθώς και καταστάσεις στόχου.

Η δυσκολία του σχεδιασμού εξαρτάται από τις υποθέσεις απλοποίησης που λαμβάνονται. Πολλές κλάσεις προβλημάτων σχεδιασμού μπορούν να προσδιοριστούν ανάλογα με τις ιδιότητες που έχουν τα προβλήματα σε διάφορες διαστάσεις.

- Είναι οι ενέργειες ντετερμινιστικές ή μη-ντετερμινιστικές; για μη ντετερμινιστικές ενέργειες, είναι οι συσχετιζόμενες πιθανότητες διαθέσιμες;
- Είναι οι μεταβλητές των καταστάσεων διακριτές ή συνεχείς; Εάν είναι διακριτές, έχουν πεπερασμένο αριθμό πιθανών τιμών;

- Μπορεί η τρέχουσα κατάσταση να παρατηρηθεί απερίφραστα; Πιθανώς να υπάρχει πλήρης δυνατότητα παρατήρησης ή μερική δυνατότητα παρατήρησης.
- Πόσες αρχικές καταστάσεις υπάρχουν; Είναι πεπερασμένος ο αριθμός τους ή αυθαίρετα μεγάλος;
- Οι ενέργειες έχουν διάρκεια ή θεωρούνται στιγμιαίες;
- Μπορούν διάφορες ενέργειες να γίνουν ταυτόχρονα ή είναι μόνο μία ενέργεια δυνατή σε κάθε χρονική στιγμή;
- Είναι ο στόχος ενός σχεδίου να φτάσει σε μία προκαθορισμένη κατάσταση στόχο ή να μεγιστοποιήσει κάποια συνάρτηση απόδοσης;
- Υπάρχει μόνο ένας πράκτορας ή υπάρχουν πολλοί; Οι πράκτορες συνεργάζονται ή είναι εγωιστικοί; Οι πράκτορες δημιουργούν τα σχέδιά τους ξεχωριστά, ή σχεδιάζονται κεντρικά όλα τα σχέδια;

Το πιο απλό πρόβλημα σχεδιασμού, γνωστό ως το Κλασσικό Πρόβλημα Σχεδιασμού, ορίζεται από:

- μια μοναδική αρχική κατάσταση,
- στιγμιαίες ενέργειες,
- ντετερμινιστικές ενέργειες,
- οι οποίες μπορούν να ληφθούν μόνο μία σε κάθε χρονική στιγμή
- και έναν μοναδικό πράκτορα.

Εφόσον η αρχική κατάσταση γνωρίζεται απερίφραστα, και όλες οι ενέργειες είναι ντετερμινιστικές, η κατάσταση του κόσμου μετά από οποιαδήποτε ακολουθία ενεργειών μπορεί να προβλεφθεί με ακρίβεια, και το ερώτημα της δυνατότητας παρατήρησης είναι ανούσιο στον κλασσικό σχεδιασμό.

Επιπλέον, τα σχέδια μπορούν να οριστούν ως ακολουθίες ενεργειών, επειδή είναι πάντα γνωστό εκ των προτέρων ποιες ενέργειες θα χρειαστούν.

Με μη-ντετερμινιστικές ενέργειες ή άλλα γεγονότα εκτός του ελέγχου του πράκτορα, οι πιθανές εκτελέσεις σχηματίζουν ένα δέντρο, και τα σχέδια πρέπει να ορίσουν τις κατάλληλες ενέργειες για κάθε κόμβο του δέντρου.

Ιδιαίτερη σημασία έχουν και οι Μαρκοβιανές διαδικασίες απόφασης διακριτού χρόνου (Discrete-time Markov decision processes) οι οποίες είναι προβλήματα σχεδιασμού με:

- στιγμιαίες ενέργειες
- μη-ντετερμινιστικές ενέργειες με πιθανότητες,
- πλήρη δυνατότητα παρατήρησης,
- μεγιστοποίηση μιας συνάρτησης απόδοσης
- ένα μοναδικό πράκτορα

Όταν η πλήρης δυνατότητα παρατήρησης αντικατασταθεί από μερική δυνατότητα παρατήρησης, το πρόβλημα σχεδιασμού αντιστοιχεί σε μερικώς παρατηρήσιμες Μαρκοβιανές διαδικασίες απόφασης (partially observable Markov decision process).

Ένας σημαντικός κλάδος του αυτοματοποιημένου σχεδιασμού είναι αυτός που ασχολείται με την περίπτωση που υπάρχουν παραπάνω από ένας πράκτορες, που είναι

γνωστή ως σχεδιασμός πολλών πρακτόρων (multi-agent planning), ο οποίος και έχει στενή συγγένεια με την Θεωρία Παιγνίων.

Σχεδιασμός πολλών πρακτόρων [10]

Ως πράκτορας στην Τεχνητή Νοημοσύνη ορίζεται ένα πρόγραμμα υπολογιστή με ικανότητα για ανεξάρτητες ενέργειες, που υπολογίζει τις ενέργειες που πρέπει να γίνουν χωρίς άμεση καθοδήγηση. Επί πλέον, πράκτορες μπορεί να είναι άλλες αυτόνομες οντότητες, όπως άνθρωποι, αντιπρόσωποι εταιρειών κ.α. Συνοπτικά, οι ιδιότητες ενός πράκτορα περιλαμβάνουν:

- Αυτόνομη εκτέλεση προσανατολισμένη προς τους στόχους
- προσανατολισμένη συλλογιστική στο πρόβλημα
- Αντιλαμβάνεται το περιβάλλον του και να ενεργεί επ ' αυτού
- Βρίσκεται σε ένα δυναμικό, πολύπλοκο περιβάλλον / πραγματικό κόσμο
- Επιμονή στούς στόχους
- Ενεργεί για λογαριασμό ενός χρήστη
- Δυνατότητα διαπραγμάτευσης και συντονισμού

Ένα σύστημα πολλών πρακτόρων αποτελείται από ένα αριθμό αλληλεπιδρώντων πρακτόρων οι οποίοι συνεργάζονται, συντονίζονται και διαπραγματεύονται.

Οι μελέτες αυτοματοποιημένου σχεδιασμού ασχολούνται κυρίως με συστήματα ενός πράκτορα. Όμως πολλές εφαρμογές χαρακτηρίζονται από περιβάλλοντα πολλών πρακτόρων και σε αυτές τις περιπτώσεις οι ανάλογες μέθοδοι σχεδιασμού πολλών πρακτόρων απαιτούνται. Ο σχεδιασμός πολλών πρακτόρων ασχολείται με τον συγχρονισμό των πόρων και ενεργειών διαφόρων πρακτόρων. Αποτελείται από πράκτορες που σχεδιάζουν για κάποιο κοινό στόχο, μέχρι κάποιον πράκτορα που συντονίζει σχέδια άλλων, ή ακόμα και πράκτορες που τελειοποιούν τα δικά τους σχέδια ενώ διαπραγματεύονται τους στόχους και τους πόρους με άλλους πράκτορες: Συχνά στά υπάρχοντα συστήματα οι πράκτορες είναι συνεργαζόμενοι και σε τέτοιες περιπτώσεις οι πράκτορες συντονίζονται είτε από κάποιο κεντρικό σύστημα ή από ένα από τους πράκτορες εφ’ όσον έχουν κοινούς στόχους. Ωστόσο, συχνά μπορεί οι πράκτορες να έχουν συγκρουόμενες προτιμήσεις σε κάποιο βαθμό και σε τέτοιες περιπτώσεις απαιτείται διαπραγμάτευση ώστε οι διαφορετικές προτιμήσεις να συνδυαστούν. Το θέμα του σχεδιασμού πολλών πρακτόρων περιλαμβάνει επίσης τους τρόπους με τους οποίους οι πράκτορες μπορούν να πραγματοποιήσουν τα παραπάνω, δηλαδή τον συντονισμό και τη διαπραγμάτευση, σε πραγματικό χρόνο, ενώ εκτελούν σχέδια.

1.2 **Γραμμική χρονική λογική (Linear Temporal Logic, LTL)** [6,7,8]

Όπως αναφέρθηκε πιο πάνω στην Εισαγωγή, για την παραγωγή έγκυρων σχεδίων υψηλής ποιότητας αυτόματα, ένα σύστημα αυτοματοποιημένου σχεδιασμού πρέπει να παρέχει έναν τρόπο για να καθορίζονται οι προτιμήσεις του χρήστη, σε σχέση με το σχεδιαστικό στόχο, καθώς και έναν τρόπο παραγωγής σχεδίων τα οποία (ιδανικά) βελτιστοποιούν αυτές τις προτιμήσεις. Η γραμμική χρονική λογική (LTL) μπορεί να χρησιμοποιηθεί για να κωδικοποιηθούν προτάσεις για το μέλλον κάποιου μονοπατιού, το οποίο παρέχει πολλές δυνατότητες στην έκφραση περιορισμών η/και προτιμήσεων.

Η γραμμική χρονική λογική είναι μια τροπική χρονική λογική. Τροπικές λογικές είναι λογικές στις οποίες μπορούμε να μελετήσουμε προτάσεις, όχι μόνο ως *αληθείς* ή *ψευδείς*, αλλά και περαιτέρω τρόπους που μπορεί να είναι αληθής μία πρόταση (π.χ. “είναι απαραίτητα αληθής”, “πιστεύεται ως αληθής”). Χρονικές λογικές λέγονται οι τροπικές λογικές όπως η LTL των οποίων οι τροπικότητες αναφέρονται στον χρόνο. Ως εκ τούτου, στην LTL, μπορούν να κωδικοποιηθούν προτάσεις για το μέλλον κάποιου μονοπατιού, π.χ. μια συνθήκη να είναι τελικά αληθής, ή να είναι αληθής μέχρι ένα άλλο γεγονός να είναι αληθές. Είναι γνωστή και ως προτασιακή χρονική λογική (propositional temporal logic, PTL). Η γραμμική χρονική λογική προτάθηκε αρχικά για την επίσημη επαλήθευση προγραμμάτων υπολογιστών, απο τον Amir Pnueli το 1977 [11].

Συντακτικό

Αποτελείται απο ένα πεπερασμένο σύνολο προτασιακών μεταβλητών *AP*, τους λογικούς τελεστές ¬ και ∨, και τους χρονικά τροπικούς τελεστές **X** και **U.** Επίσημα, το σύνολο των LTL προτάσεων πάνω στο σύνολο *AP* είναι επαγωγικά ορισμένο όπως ακολουθεί:

- Εάν p ∈ *AP* τότε p είναι LTL πρόταση
- αν ψ και φ είναι LTL προτάσεις τότε ¬ψ, φ ∨ ψ, **X** ψ, και φ **U** ψ είναι επίσης LTL προτάσεις.

Το **X** διαβάζεται ως επόμενο (ne**x**t) και το **U** ώς μέχρι (**u**ntil). Πέρα από αυτούς τους βασικούς τελεστές, υπάρχουν επιπλέον λογικοί και χρονικοί τελεστές, ορισμένοι σε σχέση με τους παραπάνω βασικούς τελεστές, ώστε να γράφονται LTL προτάσεις πιο ευκρινώς. Οι επιπλέον λογικοί τελεστές είναι οι ∧, →, ↔, **true**, και **false**. Παρακάτω οι επιπλέον χρονικοί τελεστές:

- **G** που σημαίνει πάντα (**g**lobaly)
- **F** που σημαίνει κάποτε στο μέλλον (**f**uture)
- **R** που σημαίνει απελευθέρωσε (**r**elease)
- **W** που σημαίνει ασθενώς μέχρι (**w**eakly until)

Σημασιολογία

Μια πρόταση LTL μπορεί να αποτιμηθεί πάνω σε μια άπειρη ακολουθία από αποτιμήσεις αληθείας, με μια θέση πάνω σε αυτό το μονοπάτι. Μια πρόταση LTL ικανοποιείται από ένα μονοπάτι αν και μόνο αν ικανοποιείται στη θέση 0 αυτού του μονοπατιού. Η σημασιολογία των τροπικών τελεστών δίνεται ως εξής:

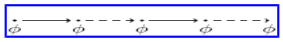
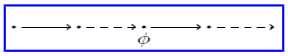
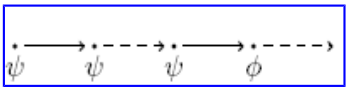
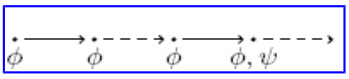
Κείμενο	Επεξήγηση	Διάγραμμα
Μοναδιαίοι Τελεστές:		
X *φ*	ne**X**t: η *φ* πρέπει να ισχύει στην επόμενη κατάσταση.	
G *φ*	**G**lobally: η *φ* πρέπει να ισχύει σε όλο το υπόλοιπο μονοπάτι.	
F *φ*	**F**inally: η *φ* πρέπει τελικά να ισχύει (οπουδήποτε στο υπόλοιπο μονοπάτι).	
Δυαδικοί Τελεστές		
ψ **U** *φ*	**U**ntil: η *ψ* πρέπει να ισχύει *τουλάχιστον* μέχρι τη *φ*, η οποία ισχύει στην τρέχουσα θέση ή σε κάποια θέση στο μέλλον.	
φ **R** *ψ*	**R**elease: η *φ* πρέπει να είναι αληθής μέχρι και τη στιγμή που η *ψ* γίνεται αληθής για πρώτη φορά - αν η *ψ* δε γίνει ποτέ αληθής, η *φ* πρέπει να μείνει αληθής για πάντα.	

Table 1: Πίνακας επεξήγησης Σημασιολογίας Γραμμικής Λογικής

1.3 **Σχεδιασμός Βασισμένος σε Προτιμήσεις** [12,13,14,15]

Οι προτιμήσεις παίζουν σημαντικό ρόλο στην ανθρώπινη λήψη αποφάσεων, ιδιαίτερα επειδή εν μέρει καθορίζουν την συμπεριφορά του ανθρώπου. Ως εκ τούτου, δεν προκαλεί έκπληξη το γεγονός ότι οι προτιμήσεις αποτελούν εξίσου σημαντικό ρόλο στην Τεχνητή Νοημοσύνη, και ιδιαίτερα στον αυτοματοποιημένο σχεδιασμό, όπου χρησιμοποιούνται για να παρέχουν έναν τρόπο να προσδιοριστούν αυτές οι ιδιότητες ενός σχεδίου που το καθιστούν υψηλής ποιότητας.

Δεδομένης κάποιας εργασίας προς επίτευξη, οι χρήστες μπορεί να έχουν προτιμήσεις όσον αφορά τους στόχους που θέλουν επιτύχουν και τις συνθήκες υπό τις οποίες θα γίνει αυτό. Μπορεί επίσης να έχουν προτιμήσεις ως προς τον τρόπο με τον οποίο θα επιτευχθεί ο στόχος-- ιδιότητες του κόσμου που πρέπει να διατηρηθούν ή αποφευχθούν κατά την διάρκεια της εκτέλεσης του σχεδίου. Με εξαίρεση τις Μαρκοβιανές διαδικασίες αποφάσεων (MDPs), οι μη τετριμμένες προτιμήσεις χρηστών έχουν μόλις πρόσφατα ενσωματωθεί στον αυτοματοποιημένο σχεδιασμό.

Ο σχεδιασμός βασισμένος στις προτιμήσεις, είναι μία επέκταση του κλασσικού προβλήματος σχεδιασμού στον οποίο, εκτός από τα δεδομένα που δίνονται στον κλασσικό σχεδιασμό, δίνονται και κριτήρια για να καθορίσουμε πότε ένα σχέδιο είναι προτιμότερο από ένα άλλο. Για αυτό τον σκοπό, συνήθως λαμβάνουμε υπ' όψιν τις σχετικές αξίες των ιδιοτήτων που ικανοποιούν τα διάφορα επιθυμητά σχέδια. Για παράδειγμα, εάν ο κόσμος μας αποτελείται από ένα δίκτυο πόλεων και πακέτα τα οποία μεταφέρονται μεταξύ πόλεων με την χρήση φορτηγών, στο κλασσικό σχεδιαστικό πρόβλημα θα μας ενδιέφερε να βρούμε οποιαδήποτε ακολουθία ενεργειών καταλήγει σε κατάσταση στην οποία όλα τα πακέτα βρίσκονται στους προορισμούς τους. Οι προτιμήσεις προκύπτουν άμεσα σε αυτόν τον κόσμο. Για παράδειγμα, ένας χρήστης μπορεί να θέσει μεγαλύτερη προτεραιότητα σε κάποια πακέτα, ώστε αυτά να παραδοθούν πρώτα. Επιπλέον, μπορεί να έχει κάποια προτίμηση ως προς τα φορτηγά τα οποία θα χρησιμοποιηθούν στο σχέδιο, πιθανώς λόγω κατανάλωσης, ή ασφάλειας των πακέτων. Ο τρόπος με τον οποίο θα προσδιοριστούν οι προτιμήσεις των χρηστών είναι μία από τις σημαντικές κατευθύνσεις έρευνας τον τελευταίο καιρό.

Εφόσον γνωρίζουμε τις προτιμήσεις του χρήστη, πρέπει να τις συσχετίσουμε με προτιμήσεις πάνω στο σχέδιο. Συγκεκριμένα, πρέπει να ορίσουμε επίσημα πότε ένα σχέδιο προτιμάται έναντι ενός άλλου. Στο παράδειγμά μας, κάποιος μπορεί να θεωρεί πως η ικανοποίηση της έγκυρης παράδοσης πακέτων υψηλής προτεραιότητας είναι πιο σημαντική από τον τύπο φορτηγών που θα χρησιμοποιηθούν. Επιπλέον, μπορεί να θεωρούμε την ποσότητα βενζίνης η οποία καταναλώθηκε από τα φορτηγά ως παράγοντα σύγκρισης σχεδίων. Για να ορίσουμε την σχέση των προτιμήσεων ανάμεσα στα σχέδια, χρειαζόμαστε μια γλώσσα η οποία α) επιτρέπει να ορίσουμε προτιμήσεις, και β) επιτρέπει να συνδυάζουμε διαφορετικές προτιμήσεις.

1.4 **Ο Σχεδιαστής PPLAN** [12,13,14,15,16]

Ο σχεδιαστής PPLAN αναπτύχθηκε στο Πανεπιστήμιο του Τορόντο από τους Meghyn Bienvenu, Sheila McIlrath, Shirin Sohrabi Araghi, και Christian Fritz.

Σ' αυτό το τμήμα θα παρουσιάσουμε τον σχεδιαστή PPLAN, και την θεωρία στην οποία είναι βασισμένος, ώστε να αποσαφηνιστεί ο τρόπος με τον οποίο αντιμετωπίζονται τα σχεδιαστικά προβλήματα τέτοιου τύπου.

Επισκόπηση

Ο PPLAN είναι ένας βέλτιστος (όσον αφορά την ποιότητα του σχεδίου) σχεδιαστής για σχεδιασμό με μη-Μαρκοβιανές ποιοτικές προτιμήσεις. Περιλαμβάνει:

- Μία γλώσσα πρώτης-τάξης η οποία χρησιμοποιείται για τον προσδιορισμό μη-Μαρκοβιανών ποιοτικών προτιμήσεων (του χρήστη) πάνω σε πιθανά σχέδια, και
- Έναν σχεδιαστή με αλγόριθμο αναζήτησης πρώτα κατά το καλύτερο (best-first search)

Η γλώσσα είναι πλούσια, και είναι επιδεκτική στην ενσωμάτωση με πολλούς από τους υπάρχοντες σχεδιαστές, και πέρα από τον προγραμματισμό, μπορεί να χρησιμοποιηθεί για την υποστήριξη αυθαίρετων δυναμικών συλλογιστικών στόχων. Η σημασιολογία της πρώτης-τάξης γλώσσας προτιμήσεων ορίζεται σύμφωνα με τον **λογισμό καταστάσεων**. Η αλήθεια ή το ψεύδος μιας πρότασης προτίμησης αξιολογείται ως μια πρόταση λογισμού καταστάσεων. Η σχετική προτίμηση των εναλλακτικών σχεδίων αντιπροσωπεύεται ως ένα βάρος, το οποίο είναι μια συνάρτηση των βαρών των ιδιοτήτων του κάθε σχεδίου. Έτσι, κάθε πρόταση προτίμησης παρέχει μια μερική διάταξη των σχεδίων που οδηγούν στον στόχο, ώστε να προτιμηθεί η καλύτερη.

Ο σχεδιαστής PPLAN παίρνει ως είσοδο, μία θεωρία ενεργειών, ένα καθορισμό για την αρχική κατάσταση του συστήματος, μία πρόταση προτιμήσεων, και ένα στόχο.

Λογισμός Καταστάσεων

Ο λογισμός καταστάσεων είναι μια λογική γλώσσα που χρησιμοποιείται για τον προσδιορισμό και την συλλογιστική σχετικά με δυναμικά συστήματα (Reiter 2001). Στο λογισμό καταστάσεων, η κατάσταση του κόσμου εκφράζεται όσον αφορά τις λειτουργίες και τις σχέσεις (fluents) σε σχέση με μια συγκεκριμένη κατάσταση s, π.χ., F (x, s). Διακρίνουμε μεταξύ του συνόλου των ρευστών κατηγορημάτων F και το σύνολο των μη-ρευστών κατηγορημάτων R τα οποία αντιπροσωπεύουν τις ιδιότητες που δεν αλλάζουν με την πάροδο του χρόνου. Μια κατάσταση s είναι μια *ιστορία* των πρωτόγονων δράσεων α ∈ A οι οποίες εκτελέστηκαν από μια αρχική κατάσταση S0. Η συνάρτηση *do*(A, S) αντιστοιχίζει μια κατάσταση και μια ενέργεια σε μια νέα κατάσταση. Η θεωρία γεννά ένα δέντρο καταστάσεων ριζωμένη στο S0.

Μια βασική θεωρία ενεργειών του λογισμού καταστάσεων D περιλαμβάνει τέσσερα ανεξάρτητα από το πεδίο θεμελιακά αξιώματα και ένα σύνολο από αξιώματα τα οποία εξαρτώνται από το πεδίο. Τα θεμελιώδη αξιώματα Σ καθορίζουν τις καταστάσεις, τον τρόπο με τον οποίο διακλαδώνονται, και την σχέση διάταξης μεταξύ καταστάσεων. Η κατάσταση s προηγείται της s' όταν η κατάσταση s προηγείται της κατάστασης s' στο δέντρο καταστάσεων. Η Σ περιλαμβάνει ένα δεύτερης τάξης αξίωμα επαγωγής. Τα αξιώματα τα οποία εξαρτώνται από το πεδίο είναι αυστηρά πρώτης τάξης και έχουν την εξής μορφή:

- αξιώματα διαδοχής καταστάσεων, ένα για κάθε ρευστό κατηγόρημα, τα οποία συγκεντρώνουν τα αποτελέσματα των ενεργειών, δεδομένου ότι ισχύει το κατηγόρημα
- αξιώματα προαπαιτούμενων ενεργειών, ένα για κάθε ενέργεια α στο πεδίο, τα οποία καθορίζουν εάν είναι δυνατό να εκτελεστεί η ενέργεια α
- αξιώματα που καθορίζουν την αρχική κατάσταση
- μοναδικά ονόματα αξιωμάτων για τις ενέργειες.

Μία πρώτης-τάξης γλώσσα προτιμήσεων

Σ' αυτό το τμήμα θα παρουσιάσουμε την γλώσσα που χρησιμοποιείται από τον σχεδιαστή PPLAN για να εκφραστούν προτιμήσεις για δυναμικά συστήματα. Αποτελεί μία τροποποίηση και επέκταση της γλώσσας προτίμησης *PP* η οποία προτάθηκε από τους Son και Pontelli [17]. Διατηρεί την ιεραρχία των βασικών τύπων επιθυμιών, τους ατομικών τύπων προτίμησης, και γενικών τύπων προτιμήσεων, όπως αυτοί ορίζονται στην *PP* και επιπλέον περιλαμβάνει μια νέα κατηγορία, αυτή των συγκεντρωτικών τύπων προτίμησης. Μεταγενέστερες αναφορές σε έναν τύπο προτίμησης αναφέρονται σε ένα συγκεντρωτική τύπο προτίμησης, η οποία περιλαμβάνει τύπους βασικής επιθυμίας, τύπους ατομικής προτίμησης, και τύπους γενικής προτίμησης.

Για να περιγράψουμε το συντακτικό και την σημασιολογία της γλώσσας αυτής θα χρησιμοποιήσουμε το παράδειγμα που χρησιμοποιούν οι ίδιοι στην σχετική βιβλιογραφία [12]:

Το Παράδειγμα Δείπνου: *Βράδιασε και η Claire είναι κουρασμένη και πεινασμένη. Στόχος της είναι να βρίσκεται στο σπίτι της χορτασμένη. Υπάρχουν τρεις δυνατοί τρόποι να φάει κάτι η Claire: Μπορεί να μαγειρέψει κάτι στο σπίτι, να παραγγείλει από κάποιο εστιατόριο που κάνει διανομή, ή να πάει σε κάποιο εστιατόριο. Για να μαγειρέψει ένα γεύμα, η Claire πρέπει να ξέρει την συνταγή αυτού, και πρέπει να έχει τα απαραίτητα συστατικά, τα οποία ενδέχεται να προμηθευτεί από κάποιο μανάβικο. Επίσης, για να μαγειρέψει η Claire, χρειάζεται να είναι καθαρή η κουζίνα της. Στην περίπτωση της παραγγελείας η διαδικασία είναι πολύ πιο απλή: πρέπει μόνο να παραγγείλει και να φάει το γεύμα. Η τρίτη εναλλακτική απαιτεί να φτάσει στο εστιατόριο, να παραγγείλει το φαγητό, και στη συνέχεια να επιστρέψει στο σπίτι.*

Ορισμός 1 (Τύποι Βασικών Επιθυμιών, Basic Desire Formula (BDF)).

Ένας βασικός τύπος επιθυμίας είναι μια πρόταση αντλημένη από το μικρότερο σύνολο B για το οποίο ισχύει:

1. F ⊂ B όπου F είναι ρευστά κατηγορήματα

2. R⊂ B όπου R είναι μη-ρευστά κατηγορήματα

3. Εάν f ∈ F, τότε final(f) ∈ B

4. Εάν a ∈ A, τότε occ(a) ∈ B

5. Εάν ϕ1 και ϕ2 ανήκουν στο B, τότε ανήκουν και τα ¬ϕ1 , ϕ1 ∧ ϕ2 , ϕ1 ∨ ϕ2 , (∃x)ϕ1 , (∀x)ϕ1 , next(ϕ1), always(ϕ1), eventually(ϕ1), and until(ϕ1,ϕ2).

Οι τύποι βασικών επιθυμιών καθιερώνουν προτιμώμενες περιπτώσεις. Συνδυάζοντας BDFs με δυαδικούς και χρονικούς τελεστές σύνδεσης, μπορούμε να εκφράσουμε μεγάλη γκάμα ιδιοτήτων καταστάσεων.

Παράδειγματα BDF πάνω στο παράδειγμα δείπνου που περιγράψαμε παραπάνω:

hasIngrnts(spag) ∧ knowsHowToMake(spag)	(P1)
(∃x).hasIngrnts(x) ∧ knowsHowToMake(x)	(P2)
final(kitchenClean)	(P3)
(∃x).eventually(occ(cook(x)))	(P4)
(∃x).(∃y).eventually(occ(orderTakeout(x,y)))	(P5)
(∃x).(∃y).eventually(occ(orderRestaurant(x, y)))	(P6)
always(¬((∃x).occ(eat(x)) ∧ chinese(x)))	(P7)

Η πρόταση P1 αναφέρει ότι στην αρχική κατάσταση η Claire έχει τα συστατικά και την τεχνογνωσία για να μαγειρέψει σπαγγέτι. Η P2 είναι πιο γενική, εκφράζοντας ότι στην αρχική κατάσταση η Claire έχει τα συστατικά τρόφιμα για να μαγειρέψει κάτι που ξέρει πώς να κάνει. Παρατηρήστε ότι οι ρευστοί τύποι που δεν είναι μέσα σε χρονικούς τελεστές αναφέρονται μόνο στην αρχική κατάσταση. Η P3 αναφέρει ότι στην τελική κατάσταση η κουζίνα είναι καθαρή. Οι P4 - P6 μας λένε ότι αντίστοιχα σε κάποια στιγμή η Claire μαγείρεψε κάτι, παρήγγειλε κάτι από διανομή, ή παρήγγειλε κάτι σε ένα εστιατόριο. Τέλος, η P7 μας λέει ότι η Claire δεν τρώει ποτέ κανένα κινέζικο φαγητό.

Ενώ οι BDF απο μόνοι τους μας επιτρέπουν να εκφράσουμε κάποιες απλές προτιμήσεις του χρήστη, δεν μπορούμε να εκφράσουμε τις προτιμήσεις μεταξύ των εναλλακτικών λύσεων. Για παράδειγμα, δεν μπορούμε να πούμε ότι η Claire προτιμά το μαγείρεμα από την παραγγελία. Για να γίνει αυτό, ορίζουμε τους Τύπους Ατομικής Προτίμησης, οι οποίες αντιπροσωπεύουν προτιμήσεις πάνω σε εναλλακτικές. Ο ορισμός της γλώσσας αυτής διαφέρει από εκείνο των (Son & Pontelli 2004) στο γεγονός ότι ζητείται από το χρήστη να σχολιάσει τις προτιμήσεις με μια τιμή που δείχνει το επίπεδο της προτίμησης σε σχέση με άλλες. Αυτό που κερδίζεται με αυτόν τον τρόπο είναι η σημαντική μείωση ασύγκριτων καταστάσεων, ένα ευρύτερο φάσμα των μεθόδων συνδυασμού προτίμησης, και μια πιο λεπτομερή απεικόνιση των επιθυμιών του χρήστη.

Ορισμός 2 (Ατομικοί Τύποι Προτιμήσεων, Atomic Preference Formula (APF)).

Έστω V, ένα εντελώς διατεταγμένο σύνολο με ελάχιστο στοιχείο Vmin και μέγιστο στοιχείο Vmax. Ένας τύπος ατομικής προτίμησης είναι μία πρόταση φ0 [v0] >> φ1 [v1] ... >> >> φn [VN], όπου κάθε φi είναι BDF, κάθε vi ∈ V, vi <vj για i <j και v0 = Vmin. Όταν το n = 0, οι ατομικοί τύποι προτίμησης αντιστοιχούν σε BDFs.

Ένας τύπος ατομικής προτίμησης εκφράζει προτίμηση πάνω από εναλλακτικές λύσεις. Σε ό,τι ακολουθεί, θεωρούμε V = [0, 1] για οικονομία, αλλά θα μπορούσαμε κάλλιστα να επιλέξουμε ένα αυστηρά ποιοτικό σύνολο όπως {βέλτιστο < καλό < αδιάφορο < κακό < χείριστο}. Επιστρέφοντας στο παράδειγμά μας, το ακόλουθο APF εκφράζει την προτίμησή της Claire για το τι θα φάει (πίτσα, που ακολουθείται από τα μακαρόνια, που ακολουθείται από κρέπες):

eventually(occ(eat(pizza)))[0]>>

eventually(occ(eat(spag)))[0.4]>>

eventually(occ(eat(crêpes)))[0.5] (P8)

Μπορούμε να δούμε ότι η Claire προτιμά σαφώς την πίτσα, αλλά θεωρεί μακαρονάδες και κρέπες περίπου εξίσου καλές. Αν, αντίθετα, η Claire ενδιαφέρεται περισσότερο για το πόση ώρα θα πρέπει να περιμένει για το γεύμα της, θα μπορούσε να διευκρινίσει τον εξής τύπο APF:

P5[0] >> (P2 ∧ P4) [0.2] >> P6[0,7] >> (¬P2 ∧ P4) [0,9] (P9)

Αυτό λέει ότι η πρώτη επιλογή της Claire είναι να παραγγείλει στο σπίτι, που ακολουθείται από το μαγείρεμα, αν έχει τα συστατικά για κάτι που ξέρει πώς να μαγειρέψει, ακολουθούμενη από τη μετάβαση σε ένα εστιατόριο για να φάει κάτι, και, τέλος, μαγείρεμα όταν απαιτεί ένα ταξίδι στο μανάβικο. Από τις τιμές που δώθηκαν για τις προτιμήσεις της Claire, μπορούμε να δούμε ότι προτιμά περισσότερο επιλογές που δεν προϋποθέτουν την έξοδο.

Για να καθοριστούν πιο περίπλοκες προτάσεις προτιμήσεων, υπάρχει μια τρίτη κλάση προτιμήσεων η οποία δίνει δυνατότητες για υπό συνθήκη προτάσεις, συνδυασμούς αλλά και διαχωρισμούς προτάσεων

Ορισμός 3 (Γενικός Τύπος Προτίμησης, General Preference Formula(GPF)).

Μια πρόταση Φ είναι γενικού τύπου προτίμησης, εάν ισχύει ένα από τα παρακάτω:

• Φ είναι APF

• Φ is γ : Ψ, όπου γ είναι BDF και Ψ είναι GPF [Conditional]

• Φ είναι ένα από

- Ψ0 & Ψ1 & ... & Ψn [General Conjunction]

- Ψ0 | Ψ1 | ... | Ψn [General Disjunction]

όπου n ≥ 1 και κάθε Ψi είναι μια GPF.

Π.Χ.

P2 : P4 (P10)

P8 | P9 (P11)

P8 & P9 (P12)

Η P10 αναφέρει ότι, αν η Claire έχει αρχικά τα συστατικά για κάτι που μπορεί να κάνει, τότε προτιμά να μαγειρέψει. Οι προτιμήσεις P12 και P11 δείχνουν δύο τρόπους με τους οποίους μπορούμε να συνδυάσουμε τα τρόφιμα και τις προτιμήσεις του χρόνου της Claire σε ένα ενιαίο συγκρότημα προτίμησης. Η P12 προσπαθεί να μεγιστοποιήσει την ικανοποίηση και των δύο προτιμήσεων της, ενώ η P11 είναι κατάλληλη εάν η Claire είναι ευχαριστημένη με το ενδεχόμενο μία από τις δύο προτιμήσεις να ικανοποιούνται.

Με τις παραπάνω κλάσεις βλέπουμε πως ο χρήστης μπορεί να περιγράψει πολλές από τις προτιμήσεις του καθώς και να συνδυάσει προτιμήσεις για να περιγράψει οσοδήποτε πολύπλοκες προτιμήσεις. Παρ' όλα αυτά, επειδή μερικές φορές οι προτιμήσεις του χρήστη δεν είναι δυνατό να ικανοποιηθούν, η γλώσσα παρέχει με την τέταρτη κλάση τύπων προτιμήσεων, ένα τρόπο να υποδεικνύει ο χρήστης κάποια χαλάρωση στις προτιμήσεις ώστε να συνεχίσει ο σχεδιαστής σε μία λύση η οποία ικανοποιεί κάποιες εναλλακτικές προτιμήσεις.

Ορισμός 4 (Συγκεντρωτικοί Τύποι Προτιμήσεων, Aggregated Preference Formula (AgPF)).

Μία πρόταση Φ αποτελεί Συγκεντρωτικό Τύπο Προτίμησης εάν:

1) η Φ είναι GPF

2) για GPF Ψi, i απο 1 εως n, η Φ είναι μία από τις παρακάτω

lex(Ψ1,...,Ψn),

lexand(Ψ1,...,Ψn),

lexor(Ψ1,...,Ψn),

leximin(Ψ1,...,Ψn),

sum(Ψ1,...,Ψn).

Οι τύποι αυτοί χρησιμοποιούνται σε περίπτωση που υπάρχουν προτιμήσεις οι οποίες είναι διαφορετικής σημαντικότητας, ώστε να δοθεί μεγαλύτερο βάρος στην πιο σημαντική. Ο τύπος lex(Ψ1,...,Ψn) χρησιμοποιεί την κλασσική λεξικογραφική ταξινόμηση, δηλαδή προσπαθεί να ικανοποιήσει πρώτα την Ψ1, έπειτα την Ψ2, και ούτω καθεξής. Οι lexand και lexor είναι αντίστοιχα με τα λογικά and και or, όμως με την διαφορά ότι για παράδειγμα στην lexand(Ψ1,Ψ2), εάν δεν είναι δυνατό να ικανοποιηθούν και οι δύο, προτιμάται η ικανοποίηση της Ψ1 έναντι της Ψ2. Η leximin πρώτα ταξινομεί το επίπεδο ικανοποίησης των Ψ1,...,Ψn, και μετά χρησιμοποιεί την λεξικογραφική ταξινόμηση, κάτι το οποίο ταιριάζει απόλυτα σε περιπτώσεις που οι Ψ1,...,Ψn είναι περίπου ίδιας σημασίας.

Χρησιμοποιώντας αυτές τις τέσσερεις κλάσεις, ο χρήστης μπορεί να εκφράσει προτιμήσεις πάνω σε οποιοδήποτε σχεδιαστικό πρόβλημα, με πολύ εύχρηστο και ολοκληρωμένο τρόπο. Λεπτομέρειες για την τεχνική ερμηνεία του πότε ικανοποιούνται οι προτάσεις αυτές υπάρχουν στην σχετική βιβλιογραφία[2,3,4].

Ο αλγόριθμος PPLAN

Ο αλγόριθμος του PPLAN χρησιμοποιεί έναν οριοθετημένο πρώτα κατά το καλύτερο αλγόριθμο αναζήτησης για την εύρεση προτιμώμενων σχεδίων. Παίρνει ως είσοδο την αρχική κατάσταση *init*, μία κατάσταση στόχου *goal,* μία πρόταση προτιμήσεων *pref*, και ένα όριο ως προς το μέγιστο μήκος σχεδίου το οποίο είναι αποδεκτό. Επιστρέφει το σχέδιο το οποίο βρίσκει, καθώς και την βαθμολόγησή του σε σχέση με την πρόταση προτιμήσεων *pref*.

Συγκεκριμένα, ξεκινώντας από την αρχική κατάσταση, αναλύει ποιες ενέργειες μπορούν να πραγματοποιηθούν στην αρχική κατάσταση. Έπειτα, στις καταστάσεις στις οποίες φτάνει πραγματοποιώντας τις ενέργειες από την αρχική κατάσταση, βλέπει εάν κάποια από αυτές αποτελεί κατάσταση στόχου, και έπειτα συνεχίζει αναλύοντας εκ νέου τις καταστάσεις στις οποίες μπορεί να φτάσει το σύστημα. Ταξινομεί αυτές τις καταστάσεις μαζί με τις προηγούμενες ανάλογα με το βάρος τους σε σχέση με την πρόταση προτιμήσεων *pref*, και επαναλαμβάνει την διαδικασία έως ότου βρεθεί σε κάποια κατάσταση στόχου.

1.5 Σχόλια και κίνητρο

Παρ' ότι η περιοχή του σχεδιασμού με προτιμήσεις έχει πολύ ενδιαφέρον σε εφαρμογές πραγματικού κόσμου, δεν έχει γίνει πολλή έρευνα σχετικά με εφαρμογές που συμπεριλαμβάνουν ενέργειες που πραγματοποιούνται από πολλαπλούς παράγοντες, με διαφορετικές προτιμήσεις όσον αφορά το σχέδιο που θα πραγματοποιηθεί. Αυτό είναι ιδιαίτερης σημασίας, καθώς ένα τέτοιο εργαλείο θα μπορούσε να χρησιμοποιηθεί για να βελτιστοποιηθούν οι διαδικασίες οι οποίες ακολουθούνται στον πραγματικό κόσμο, έτσι ώστε όλοι οι εμπλεκόμενοι να είναι όσο το δυνατόν πιο ευχαριστημένοι από αυτές. Από πλευράς θεωρίας παιγνίων, κάτι τέτοιο θα αποτελούσε ουσιαστικά μία σύνδεση των στρατηγικών του κάθε παίκτη/πράκτορα σε μία ενιαία στρατηγική, συνδέοντας έτσι την “ευχαρίστηση” ή αλλιώς συμφέρον του καθενός, με την κοινωνική ευημερία, ή αλλιώς το συμφέρον της κοινωνίας ως σύνολο. Η παρούσα συνεισφορά δεν περιλαμβάνει ανάλυση από πλευράς θεωρίας παιγνίων, όμως βλέπουμε πως υπάρχουν αρκετά ενδιαφέρουσες προοπτικές σ' αυτήν την κατεύθυνση. Θα μπορούσε να χρησιμοποιηθεί για να καθοδηγήσει τους εμπλεκόμενους να επιλέξουν στρατηγικές ενεργειών οι οποίες ευνοούν το σύνολο, ή να ξεχωρίσει αυτούς που επιλέγουν στρατηγικές εγωιστικά και να τους απομονώσει από το σύνολο.

Στο κεφάλαιο που ακολουθεί περιγράφεται η συνεισφορά της παρούσας εργασίας στον τομέα αυτό.

2. Μεθοδολογία

2.1 Φορμαλισμός προβλημάτων πολλών πρακτόρων για επίλυση από τον PPLAN

Πολλές φορές σε προβλήματα σχεδιασμού πραγματικού κόσμου, οι παράγοντες (actors) οι οποίοι συνεργάζονται για την επίτευξη ενός στόχου δεν έχουν κοινές προτιμήσεις, και πιθανώς να έχουν διαφορετικούς τρόπους να κρίνουν την ποιότητα του κάθε σχεδίου.

Για να υπάρξει σχεδιασμός τέτοιων προβλημάτων, πρέπει να υπάρχει ένας τρόπος να περιγράψουμε τις πιθανές ενέργειες του κάθε παράγοντα που ενεργεί στο πιθανό σχέδιο, καθώς και τις προτιμήσεις του, με τρόπο που αντιπροσωπεύει την σημαντικότητά του στο σχέδιο, ενώ ταυτόχρονα επιτρέπει να βελτιστοποιηθεί το σχέδιο, λαμβάνοντας υπ' όψιν τις προτιμήσεις όλων των παραγόντων. Δεδομένων των δυνατοτήτων που παρέχονται από τον σχεδιαστή PPLAN για συνδυασμό προσωπικών προτιμήσεων και γενικευμένων τύπων προτιμήσεων, κάτι τέτοιο μπορεί να πραγματοποιηθεί φορμαλιστικά. Ως ενέργειες δηλώνουμε την ένωση των ενεργειών του κάθε παράγοντα, και ως προτιμήσεις, συνδυάζουμε τις προτιμήσεις του κάθε παράγοντα με τρόπο που αναλύουμε παρακάτω.

Μελέτη Περίπτωσης

Παρουσιάζω παρακάτω ένα παράδειγμα περιβάλλοντος για να μπορούμε να εξηγήσουμε περαιτέρω:

Θεωρούμε μία επιτροπή η οποία καλείται να αξιολογήσει ένα έγγραφο με τρόπο τέτοιο ώστε να συμφωνούν όλα τα μέλη της επιτροπής. Θεωρούμε πως η επιτροπή αποτελείται από τον πρόεδρό της (chairman), και τα μέλη. Θεωρούμε επίσης πως υπάρχει στην διάθεση αυτών μία γραμματέας και ένας τεχνικός υπολογιστών, για να βοηθήσει στην διαδικασία αυτή. Σκοπός είναι να διανεμηθεί το έγγραφο από τον πρόεδρο στα υπόλοιπα μέλη της επιτροπής, και αυτά με την σειρά τους να απαντήσουν με σχόλια.

Θεωρούμε στο συμβολισμό μας οτιδήποτε ξεκινά με κεφαλαίο ως μεταβλητή, ενώ αυτά που ξεκινούν με μικρό ως άτομα συγκεκριμένα στον κόσμο μας. π.χ. letter θεωρούμε την επιστολή που παρουσιάζεται προς αξιολόγηση, ενώ Letter είναι μια μεταβλητή η οποία μπορεί να είναι οτιδήποτε. Γενικότερα, χρησιμοποιούμε τον συμβολισμό λογικής που χρησιμοποιείται κατά κόρων στην περιοχή, καθώς και στην γλώσσα προγραμματισμού Prolog στην οποία είναι υλοποιημένος ο σχεδιαστής PPLAN.

Πιθανές ενέργειες οι οποίες μπορούν να πραγματοποιηθούν είναι οι εξής:

- Αποστολή e-mail
- Κρυπτογράφηση του εγγράφου
- Δημιουργία ασφαλούς εξυπηρετητή (secure server) από τον οποίο θα μπορούν να μεταφέρουν δεδομένα χρήστες με ασφαλή και εμπιστευτικό τρόπο.
- Αποστολή FAX.
- Αποστολή ταχυδρομικού γράμματος.

Ανάλογα με την περίσταση, οι γενικές προτιμήσεις που μπορεί να έχει ο κάθε εμπλεκόμενος κυμαίνονται σε ευρύ φάσμα. Για παράδειγμα, μπορεί να είναι επιθυμητό το έγγραφο να παραμείνει κρυφό σε όσους δεν ανήκουν στην επιτροπή, οπότε θα είναι επιθυμητό:

- Το γράμμα να μην ταξιδεύει ποτέ ως έχει, παρά μόνο κρυπτογραφημένο.

always(not(sendEmail(X,Y,letter))).

- Το γράμμα να μην βρεθεί στα εισερχόμενα κάποιου χωρίς να είναι κρυπτογραφημένο.

always(not(inInbox(X,letter))).

- Η γραμματέας και ο τεχνικός υπολογιστών να μην γνωρίζουν τα περιεχόμενα του εγγράφου, εκτός εάν υπογράψουν κάποια συμφωνία εμπιστευτικότητας.

always(not(and(hasaccessto(letter,it) , not(hassignedconfidentiality(it))))).

always(not(and(hasaccessto(letter,secr) , not(hassignedconfidentiality(secr))))).

Επίσης, ανάλογα με το επίπεδο εμπιστευτικότητας πιθανώς να είναι επιθυμητό:

- Το κάθε μέλος της επιτροπής να μην μπορεί να δει τα σχόλια των υπολοίπων.

always(not(hasaccessto(response(CommiteeMember),X))).

- Ο πρόεδρος να μην γνωρίζει ποιος έκανε το κάθε σχόλιο.

always(not(hasaccessto(author(response(X)),chair))).

Ενδιαφέρον παρουσιάζεται όταν προσθέσουμε ειδικότερες προτιμήσεις του κάθε εμπλεκόμενου, όπως για παράδειγμα:

- Η γραμματέας προτιμά να μην αναλάβει ευθύνη για τίποτα.

always(not(hassignedconfidentiality(secr))).

- Ο τεχνικός υπολογιστών προτιμά να μην αναλάβει ευθύνη για τίποτα.
  always(not(hassignedconfidentiality(it))).
- Ένα από τα μέλη της επιτροπής προτιμά να μην χρησιμοποιεί υπολογιστές λόγω έλλειψης τεχνογνωσίας.

always(not(usecomputer(X))).

- Ένα ή περισσότερα από τα μέλη της επιτροπής επιθυμεί να γνωρίζει τα σχόλια κάποιου άλλου.
- Ένα ή περισσότερα από τα μέλη επιθυμεί να αργοπορήσει η διαδικασία είτε λόγω προσωπικής ανάγκης, είτε ακόμα και λόγω κωλυσιεργίας.

Το σχέδιο δράσης μπορεί να απαιτεί την παραβίαση κάποιων εκ των προτιμήσεων ενός η περισσοτέρων εμπλεκομένων (π.χ. στην περίπτωση που κάποιος δεν μπορεί να χρησιμοποιήσει υπολογιστή, οπότε αναζητά βοήθεια είτε από τον τεχνικό υπολογιστών, είτε από την γραμματέα, αναγκάζοντάς τους να υπογράψουν κάποια συμφωνία εμπιστευτικότητας).

Βλέπουμε πως η χρησιμότητα ενός εργαλείου που μπορεί να συνδυάζει προτιμήσεις διαφορετικών παραγόντων περιορίζεται μόνο από την λεπτομέρεια η οποία ενδιαφέρει τον χρήστη. Χρησιμοποιώντας κάτι τέτοιο, θα μπορεί ένας υπεύθυνος να παράγει σχέδια δράσης για προβλήματα, οι οποίες θα αποτελούν την χρυσή τομή ανάμεσα στις προτιμήσεις όλων των εμπλεκόμενων σε αυτή.

Κάτι τέτοιο μπορεί να επιτευχθεί χρησιμοποιώντας τα εργαλεία τα οποία μας δίνονται από τον σχεδιαστή PPLAN. Χρησιμοποιώντας τους Συγκεντρωτικούς Τύπους Προτιμήσεων (Aggregated Preference Formula) οι οποίοι παρέχονται από την γλώσσα προτιμήσεων του PPLAN, μπορούμε να συνδυάσουμε προτιμήσεις διαφορετικών παραγόντων ως εξής:

Έστω ψ1 μια πρόταση προτιμήσεων του παράγοντα A, και ψ2 μια πρόταση προτιμήσεων του παράγοντα B. Μπορούμε να συνδυάσουμε τις προτάσεις αυτές, έτσι ώστε να εκπροσωπεί την σχέση μεταξύ των παραγόντων A και B. Έτσι, εάν οι παράγοντες είναι περίπου ίσης σημαντικότητας στο περιβάλλον μας, θα χρησιμοποιήσουμε την leximin(ψ1,ψ2). Αντίστοιχα, εάν ο παράγοντας A είναι υψηλότερης σημαντικότητας από τον παράγοντα B, θα χρησιμοποιήσουμε την lexand(ψ1,ψ2) και ούτω καθεξής.

Στον σχεδιαστή PPLAN, οι ενέργειες που μπορεί να κάνει ο χρήστης προσδιορίζονται με το κατηγόρημα action και την ενέργεια σε εισαγωγικά. π.χ. action(sendEmailto(Receiver,Mail)). Αυτό μπορεί να μοντελοποιηθεί για να περιλαμβάνει και τον δράστη της ενέργειας (σ' αυτή την περίπτωση αποστολέα) ώς εξής: action(sendEmail(Sender,Receiver,Mail)). Έτσι, δεν χρειάζεται να θεωρούμε πως ο χρήστης του προγράμματος είναι ο δράστης των ενεργειών, και μπορούμε να χρησιμοποιήσουμε τον σχεδιαστή για κεντρικό σχεδιασμό ενεργειών που θα πραγματοποιηθούν από πολλούς πράκτορες

Παραδειγματική μοντελοποίηση περιβάλλοντος

Σ' αυτό το τμήμα θα δείξουμε πώς μοντελοποιείται το παραπάνω πρόβλημα ώστε να αναλυθεί με τον σχεδιαστή PPLAN.

Αρχικά πρέπει να δηλώσουμε με κάποιον τρόπο τους παράγοντες του κόσμου μας

Ορισμός των κατηγορημάτων

–

indPred(isActor(_)).

indPred(isCommitteeMember(_)).

–

committeeMember(john).

committeeMember(maria).

committeeMember(michael).

–

actor(chairman).

actor(secretary).

actor(it).

actor(L):-committeeMember(L).

–

Πρέπει επίσης να ομαδοποιήσουμε όλες τις οντότητες κειμένου, από την επιστολή που πρόκειται να αξιολογηθεί, μέχρι τις διάφορες κρυπτογραφήσεις και κωδικούς.

indPred(isText(_)).

Η επιστολή,

text(letter).

Οι απαντήσεις με τα σχόλια του κάθε μέλους της επιτροπής,

text(response(Actor)):-committeeMember(Actor).

Οι διάφοροι κωδικοί του κάθε μέλους της επιτροπής σε περίπτωση που δημιουργηθεί εξυπηρετητής από τον τεχνικό υπολογιστών.

text(serverPass(Actor)):-committeeMember(Actor).

text(serverPass(chairman)).

Κρυπτογραφημένο κείμενο οποιουδήποτε άλλου κειμένου T

text(ciphertext(T)):-text(T),!.

Κωδικός αναγκαίος για την αποκρυπτογράφηση του κάθε κρυπτογραφημένου κειμένου.

text(pass(ciphertext(T))):-text(T).

Έπειτα, θα ορίσουμε τα ρευστά κατηγορήματα, δηλαδή αυτά τα οποία αλλάζουν ανάλογα με τις ενέργειες οι οποίες πραγματοποιούνται. Κάθε κατάσταση του κόσμου μας περιγράφεται από μία λίστα με ρευστά κατηγορήματα.

fluent(hasaccessto(T,Actor)).

Αυτό το κατηγόρημα θα το χρησιμοποιήσουμε για να δηλώσουμε εάν ο παράγοντας Actor έχει πρόσβαση στο T. Χρησιμοποιώντας αυτό, μπορούμε να πούμε πως ο βασικός στόχος κάποιου σχεδίου είναι να έχει ο πρόεδρος (chairman) πρόσβαση στις απαντήσεις του κάθε μέλους της επιτροπής, δηλαδή μία κατάσταση η οποία περιλαμβάνει την λίστα:

[hasaccessto(response(john),chairman),

hasaccessto(response(maria),chairman),

hasaccessto(response(michael),chairman)]

Ή αλλιως να ισχύει για την κατάστασή μας:

and(hasaccessto(response(john),chairman),

hasaccessto(response(maria),chairman),

hasaccessto(response(michael),chairman))

Αντίστοιχα, η αρχική κατάσταση του κόσμου μας είναι μία κατάσταση στην οποία το μόνο που ισχύει είναι πως ο πρόεδρος της επιτροπής έχει πρόσβαση στην επιστολή:

hasaccessto(letter,chairman).

Θα μπορούσαμε επίσης να δηλώσουμε με το ίδιο κατηγόρημα εάν κάποιος έχει πρόσβαση σε υπολογιστή hasaccessto(computer,Actor), ή ακόμα και σε ταχυδρομείο hasaccessto(postOffice,Actor) εάν επιθυμούσαμε τόση λεπτομέρεια.

Επίσης, για να δίνεται η δυνατότητα να επικοινωνήσουν μέσω ασφαλούς εξυπηρετητή όπως περιγράψαμε παραπάνω, θα πρέπει να γίνουν οι κατάλληλες ενέργειες ώστε να ενεργοποιηθεί αυτός, οπότε και θα χρησιμοποιηθεί το αντίστοιχο κατηγόρημα:

fluent(serverWorking).

Και αντίστοιχα, όταν κάτι έχει μεταφορτωθεί στον εξυπηρετητή, θα χρησιμοποιείται το

fluent(onServer(T)).

Τέλος, πρέπει να ορίσουμε τις ενέργειες του κάθε παράγοντα, καθώς και πότε είναι δυνατό να πραγματοποιηθεί η καθεμία από αυτές, και τι αντίκτυπο έχουν στην κατάσταση του κόσμου.

action(sendEmail(Sender,Receiver,Email)).

Για να μπορεί να στείλει το μύνημα Email ο παράγοντας Sender στον παράγοντα Receiver, πρέπει αρχικά να είναι και οι δύο παράγοντες, αφετέρου το Email να είναι κείμενο, και τέλος, πρέπει ο Sender να έχει πρόσβαση στο κείμενο ώστε να το στείλει.

poss(sendEmail(Sender,Receiver,Email),S):-
isActor(Sender),isActor(Receiver),isText(Email),holdsL([hasaccessto(Email,Sender)],S).

Κάθε ενέργεια έχει ένα αντίκτυπο στον κόσμο μας. Αυτό κωδικοποιείται ως δύο λίστες από ρευστά κατηγορήματα για κάθε ενέργεια, από τις οποίες η μία είναι η λίστα των κατηγορημάτων η οποία προστίθεται στην τρέχουσα κατάσταση, και η άλλη είναι η λίστα αυτών που αφαιρείται. Τυχαίνει στον κόσμο μας οι ενέργειες να μην έχουν αφαιρετικές ιδιότητες, ένα τέτοιο παράδειγμα σε ένα άλλο κόσμο στον οποίο τρώνε deleteList(eat(rice),[has(rice)]) το οποίο θα σήμαινε πως εάν κάποιος φάει ρύζι, στην επόμενη κατάσταση δεν θα έχει ρύζι. Στην περίπτωση της ενέργειας sendEmail, αυτά είναι τα εξής:

addList(sendEmail(_,Receiver,Email),[hasaccessto(Email,Receiver)]).

deleteList(sendEmail(_,_,_),[]).

Αντίστοιχα για την αποστολή γράμματος:

poss(sendLetter(Sender,Receiver,Letter),S):-
isActor(Sender),isActor(Receiver),isText(Letter),holdsL([hasaccessto(Letter,Sender)],S).

addList(sendLetter(_,Receiver,Letter),[hasaccessto(Letter,Receiver)]).

deleteList(sendLetter(_,_,_),[]).

Μπορεί για λόγους εμπιστευτικότητας να χρειάζεται κρυπτογράφηση κάποιου κειμένου.

action(encrypt(Actor,Message)).

Για να μπορέσει να κρυπτογραφήσει ο Actor το κείμενο Message, χρειάζεται αφενός να είναι παράγοντας, αφετέρου το Message να είναι κείμενο, και τέλος, χρειάζεται να έχει πρόσβαση στο κείμενο το οποίο επιθυμεί να κρυπτογραφήσει

poss(encrypt(Actor,Text),S):-
isActor(Actor),isText(Text),holdsL([hasaccessto(Text,Actor)],S).

Η κρυπτογράφηση δίνει στον παράγοντα Actor πρόσβαση στην κρυπτογραφημένη εκδοχή του κειμένου Text.

addList(encrypt(Actor,Text),
[hasaccessto(ciphertext(Text),Actor),hasaccessto(pass(ciphertext(Text)))]).

deleteList(encrypt(_,_),[]).

Αντίστοιχα για την ανάκτηση του αρχικού κειμένου

action(decrypt(Actor,ciphertext(T))).

Για να αποκρυπτογραφήσει το ciphertext(T) πρέπει να έχει αρχικά πρόσβαση σε αυτό, και αφετέρου να έχει πρόσβαση στον κωδικό ο οποίος απαιτείται για την αποκρυπτογράφησή του (pass(ciphertext(T))).

poss(decrypt(Actor,ciphertext(T)),S):-
isActor(Actor),holdsL([hasaccessto(ciphertext(T),Actor),hasaccessto(pass(ciphertext(T))
,Actor)],S).

addList(decrypt(Actor,ciphertext(T)),[hasaccessto(T,Actor)]).

deleteList(decrypt(_,ciphertext(_)),[]).

Για την χρήση ασφαλούς εξυπηρετητή χρειάζεται να δημιουργηθεί αυτός από τον τεχνικό υπολογιστών (it).

action(createServer(it)).

Μόνη προϋπόθεση είναι να μην λειτουργεί ήδη κάποιος εξυπηρετητής (το σύμβολο \+ σημαίνει την άρνηση του επομένου κατηγορήματος).

poss(createServer(it),S):- \+ holdsL([serverWorking],S).

addList(createServer(it),[serverWorking]).

deleteList(createServer(it),[]).

Στον ασφαλή εξυπηρετητή μπορούν να φτιάξουν λογαριασμό οι χρήστες υπογράφοντας συμφωνητικό εμπιστευτικότητας. Έτσι τους δίνεται ένας κωδικός μοναδικός για κάθε χρήστη (securePass(Actor)).

action(createSecureUser(Actor)).

poss(createSecureUser(Actor),S):-isActor(Actor),holdsL([serverWorking],S).

addList(createSecureUser(Actor),[hasaccessto(securePass(Actor),Actor)]).

deleteList(createSecureUser(_),[]).

Μετά από αυτήν την διαδικασία, όσοι έχουν securePass μπορούν να μεταφορτώνουν δεδομένα από και στον εξυπηρετητή

action(postOnSecureServer(Actor,Text)).

action(readFromSecureServer(Actor,Text)).

Για να ανεβάσει κάποιος κάτι στον εξυπηρετητή χρειάζεται να έχει κωδικό securePass, και να έχει πρόσβαση στο κείμενο που επιθυμεί να ανεβάσει.

poss(postOnSecureServer(Actor,Text),S):-
isActor(Actor),isText(Text),holdsL([hasaccessto(securePass(Actor),Actor),hasaccessto(Text,Actor)],S).

Αντίστοιχα για να κατεβάσει, πρέπει να έχει κωδικό securePass, και το κείμενο που θέλει πρέπει να βρίσκεται στον εξυπηρετητή

poss(readFromSecureServer(Actor,Text),S):-
isActor(Actor),isText(Text),holdsL([hasaccessto(securePass(Actor),Actor),onServer(Text)],S).

addList(postOnSecureServer(_,Text),[onServer(Text)]).

addList(readFromSecureServer(Actor,Text),[hasaccessto(Text,Actor)]).

deleteList(postOnSecureServer(_,_),[]).

deleteList(readFromSecureServer(_,_),[]).

Τέλος, η ενέργεια η οποία μένει είναι να απαντήσει ο καθένας στο κείμενο, δημιουργώντας έτσι το κείμενο απάντησης το οποίο είναι μοναδικό για κάθε παράγοντα που απαντάει.

action(respond(Actor)).

Για να απαντήσει πρέπει ο Actor να έχει πρόσβαση στο κείμενο letter το οποίο και σχολιάζει.

poss(respond(Actor),S):-
isCommitteeMember(Actor),holdsL([hasaccessto(letter,Actor)],S).

Πλέον έχει την απάντηση στην διάθεσή του, και μπορεί να την στείλει με όποιον τρόπο επιθυμεί.

addList(respond(Actor),[hasaccessto(response(Actor),Actor)]).

deleteList(respond(_),[]).

2.2 Ενδεικτικές εκτελέσεις

Ο σχεδιαστής καλείται ως pplan(InitState,Goal,Pref, Bound) όπου InitState είναι μια λίστα με ρευστά κατηγορήματα τα οποία ισχύουν στην αρχική κατάσταση, Goal είναι μια λίστα απο ρευστά κατηγορήματα που θέλουμε να ισχύουν στην κατάσταση στόχου, Pref είναι η συνδιαστική πρόταση προτιμήσεων για το σχέδιο δράσης, και Bound είναι ένα μέγιστο όριο ενεργειών για το παραγόμενο σχέδιο:

Περίπτωση 1: Θέλουμε ο πρόεδρος (chairman) να λάβει την απάντηση του μέλους maria.

Ο σχεδιαστής μας απαντά πως ο πρόεδρος πρέπει να στείλει email με το γράμμα στο μέλος maria, στην συνέχεια η maria να παράξει την απάντησή της, και να την στείλει με email ξανά πίσω.

```
?- pplan([hasaccessto(letter,chairman)],[hasaccessto(response(maria),chairman)],final(hasaccessto(let
ter,chairman)),10).
Plan: [sendEmail(chairman,maria,letter),respond(maria),sendEmail(maria,chairman,response(maria))]
Value: 0
Nodes Expanded: 79
Nodes Considered: 1766
true.

?-
```

Περίπτωση 2: Θέλουμε ο πρόεδρος (chairman) να λάβει την απάντηση του μέλους maria, με προτίμηση να μήν αποσταλεί e-mail από τον πρόεδρο σε αυτήν.

Ο σχεδιαστής μας απαντά την αντίστοιχη διαδικασία αλλά μέσω ταχυδρομείου

```
?- pplan([hasaccessto(letter,chairman)],[hasaccessto(response(maria),chairman)],always(notB(occ(sendE
mail(chairman,maria,letter)))),10).
Plan: [sendLetter(chairman,maria,letter),respond(maria),sendEmail(maria,chairman,response(maria))]
Value: 0
Nodes Expanded: 119
Nodes Considered: 2654
true.

?-
```

Περίπτωση 3: Θέλουμε ο πρόεδρος (chairman) να λάβει την απάντηση του μέλους maria, με προτίμηση να μήν αποσταλεί e-mail ούτε γράμμα από τον πρόεδρο σε αυτήν.

Αφού έχουμε απορρίψει τα ενδεχόμενα email και γράμματος από τον πρόεδρο στο μέλος maria, ο πρόεδρος στέλνει μέσω email το γράμμα στην γραμματεία, η οποία στη συνέχεια το προωθεί στην maria.

```
?- pplan([hasaccessto(letter,chairman)],[hasaccessto(response(maria),chairman)],always(andB(notB(occ(
sendEmail(chairman,maria,letter))),notB(occ(sendLetter(chairman,maria,letter))))),10).
Plan: [sendEmail(chairman,secretary,letter),sendEmail(secretary,maria,letter),respond(maria),sendEmai
l(maria,chairman,response(maria))]
Value: 0
Nodes Expanded: 380
Nodes Considered: 9134
true
```

Θα μας ήταν χρήσιμο επίσης να μπορούμε να θέσουμε βάρη στην κάθε προτίμηση, έτσι ώστε να μπορούμε να διαφοροποιήσουμε τις προτιμήσεις μεταξύ τους, ανάλογα με την σημαντικότητά τους. Γι' αυτό προτείνουμε το κατηγόρημα **comG**, το οποίο λαμβάνει μια λίστα απο προτιμήσεις, και έπειτα μία λίστα από αριθμούς που αντιστοιχούν στα βάρη των προηγουμένων : comG([P1,P2,P3],[W1,W2,W3]).

Στο παραπάνω παράδειγμα, προσθέτουμε την περίπρωση η γραμματεία να μην επιθυμεί να εμπλακεί με την επιστολή:

```
?- pplan([hasaccessto(letter,chairman)],[hasaccessto(response(maria),chairman)],
always(comG([notB(occ(sendEmail(chairman,maria,letter))),notB(occ(sendLetter(chairman,maria,letter))),notB(hasaccessto(letter,s
ecretary))],[1,1,0.5])),10).
Plan: [sendEmail(chairman,it,letter),sendEmail(it,maria,letter),respond(maria),sendEmail(maria,chairman,response(maria))]
State: [hasaccessto(response(maria),chairman),hasaccessto(response(maria),maria),hasaccessto(letter,maria),hasaccessto(letter,i
t),hasaccessto(letter,chairman)]
Value: 0.0
Nodes Expanded: 44
Nodes Considered: 526
true
```

3. Σύνοψη

Στα προβλήματα σχεδιασμού, η ποιότητα του αποτελέσματος διαφέρει ανάλογα με τις προτιμήσεις του χρήστη. Όταν στο πρόβλημα συμπεριλαμβάνει πολλούς χρήστες οι οποίοι χρειάζεται να συνεργαστούν για ένα κοινό σκοπό, οι προτιμήσεις του κάθε χρήστη διαφέρουν πολύ.

Πολλές φορές σε προβλήματα σχεδιασμού, οι παράγοντες (actors) οι οποίοι συνεργάζονται για την επίτευξη ενός στόχου δεν έχουν κοινές προτιμήσεις, και πιθανώς να έχουν διαφορετικούς τρόπους να κρίνουν την ποιότητα του κάθε σχεδίου. Είναι λοιπόν χρήσιμο να υπάρχει ένα εργαλείο το οποίο συνδυάζει προτιμήσεις και παράγει σχέδια τα οποία ικανοποιούν στο μέγιστο των δυνατοτήτων τις προτιμήσεις όλων των παραγόντων. Σ' αυτή την εργασία παρουσιάζεται ένας τρόπος να γίνει αυτό κάνοντας χρήση των ήδη υπαρχόντων εργαλείων, σε συνδυασμό με ένα φορμαλισμό προβλημάτων πολλων πρακτόρων. Η χρήση ενός τέτοιου εργαλείου για σχεδιασμό ενεργειών έχει αρκετό ενδιαφέρον, καθώς προσδίδει την έννοια της αλληλεπίδρασης και της συλλογικότητας στο σχέδιο, κάτι το οποίο παρουσιάζει ιδιαίτερο ενδιαφέρον από πλευράς θεωρίας παιγνίων.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Intelligent agent	Έξυπνος Πράκτορας
Artificial Intelligence	Τεχνητή Νοημοσύνη
Linear Temporal Logic	Γραμμική Χρονική Λογική
State	Κατάσταση
Cipher-text	Κωδικοποιημένο κείμενο

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

AgPF	Aggregated Preference Formula
GPF	General Preference Formula
APF	Atomic Preference Formula
BDF	Basic Desire Formula
LTL	Linear Temporal Logic
ΕΚΠΑ	Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΑΝΑΦΟΡΕΣ

[1] Malik Ghallab, Dana Nau, and Paolo Traverso.Automated Planning–Theory and Practice, chapter 1. Elsevier/Morgan Kaufmann, 2004.

[2] Qiang Yang. Intelligent Planning–A Decomposition and Abstraction Based Approach. Springer, 1997.

[3] Dana S. Nau, Automated Planning, CMSC 421, Spring 2010

[4] Jon Doyle, Richmond H. Thomason, Background to Qualitative Decision Theory, Copyright © 2015, Association for the Advancement of Artificial Intelligence (www.aaai.org).

[5] John E. Hopcroft and Jeffrey D. Ullman. Introduction to Automata Theory, Languages, and Computation, chapter 2. Addison Wesley, 1979.

[6] Linear-time Temporal Logic: http://www-step.stanford.edu/tutorial/temporal-logic/temporal-logic.html

[7] *Linear Temporal Logic (LTL): Richard M. Murray Nok Wongpiromsarn Ufuk Topcu California Institute of Technology AFRL, 24 April 2012*

[8] Dov M. Gabbay, A. Kurucz, F. Wolter, M. Zakharyaschev (2003). *Many-dimensional modal logics: theory and applications*. Elsevier. p. 46. ISBN 978-0-444-50826-3.

[9] James Allen, James Hendler, Austin Tate (eds). Readings in Planning. Morgan Kaufmann, 1990.

[10] Roman van der Krogt and Mathijs de Weerdt and Yingqian Zhang: *Of Mechanism Design and Multiagent Planning*

[11] Amir Pnueli: The Temporal Logic of Programs. FOCS 1977: 46-57

[12] Meghyn Bienvenu, Christian Fritz, Sheila A. McIlraith: Planning with Qualitative Temporal Preferences. KR 2006: 134-144

[13] Meghyn Bienvenu, Christian Fritz, Sheila A. McIlraith: Specifying and computing preferred plans. Artif. Intell. 175(7-8): 1308-1345 (2011)

[14] Jorge A. Baier, Sheila A. McIlraith: Planning with Preferences. AI Magazine 29(4): 25-36 (2008)

[15] Judy Goldsmith and Ulrich Junker: Preference Handling for Artificial Intelligence, AI Magazine, Winter, 2008.

[16] PPLAN: http://www.cs.toronto.edu/~sheila/PPLAN/

[17] Tran Cao Son,Enrico Pontelli: Reasoning about Actions and Planning with Preferences Using Prioritized Default Theory. Computational Intelligence 20(2): 358-404 (2004)