TECHNICAL REPORT
No. TR TRHP - 03

Semantic Web Services Composition based on Planning as satisfiability

-

Σύνθεση υπηρεσιών ιστού με τη μέθοδο του «Planning as satisfiability»

Παρασκευή Τσούτσα
tsoutsa@teilar.gr

Πάτρα, 2012

Πανεπιστήμιο Πατρών
Τμήμα Μαθηματικών

ΠΕΡΙΕΧΌΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Στις μέρες μας στο διαδίκτυο υπάρχουν αρκετές εκατοντάδες υπηρεσίες ιστού που η καθεμιά προσφέρει κάποια λειτουργικότητα μέσω των operations που διαθέτει. Οι υπηρεσίες αυτές εκτός του ότι αυξάνονται με γοργούς ρυθμούς ενημερώνονται και επεκτείνονται συνεχώς «on the fly». Δεδομένης αυτής της πληθώρας και της πολυπλοκότητας υπάρχει η ανάγκη αυτόματα να μπορούμε να ψάξουμε για κάποιες υπηρεσίες ιστού που χρειαζόμαστε και στη συνέχεια να τις χρησιμοποιήσουμε κατάλληλα ώστε να επιτύχουμε κάποιο στόχο. Οι συγκεκριμένες λειτουργίες των υπηρεσιών πρέπει δυναμικά να συν λειτουργήσουν ώστε να επιτύχουμε τον επιθυμητό στόχο.

Σκοπός της ευρύτερης έρευνας που κάνουμε είναι να προτείνουμε μια μέθοδο για τη μοντελοποίηση της σύνθεσης υπηρεσιών στο σημασιολογικό ιστό. Δεδομένου ότι έχουν παρουσιασθεί μέθοδοι επίλυσης του προβλήματος της σύνθεσης υπηρεσιών που το αντιμετωπίζουν ως πρόβλημα αυτόματου σχεδιασμού ενεργειών μελετήθηκαν μέθοδοι αυτόματου σχεδιασμού ενεργειών (Automated Planning)[10]. Στη συνέχεια μελετήθηκαν κάποια συστήματα λογικής όπως Τροπική Λογική (Modal Logic) [6], Προτασιακή Δυναμική Λογική (PDL)[8], Περιγραφική Λογική(DL)[11] ώστε να βρεθεί το κατάλληλο που θα εκφράσει το πρόβλημα εν γένει. Στις μεθόδους αυτόματου σχεδιασμού ενεργειών σχεδιάζουμε να ακολουθήσουμε εκείνη του planning as satisfiability και για το λόγο αυτό μελετήθηκαν Συστήματα Αυτόματης Απόδειξης Θεωρημάτων και η ικανοποιησιμότητα σε ένα σύστημα Λογικής [9][11].

1.1. Το πρόβλημα σχεδιασμού ενεργειών (Planning Problem)

Ο αυτόματος σχεδιασμός ενεργειών (Automated Planning) είναι κλάδος της Τεχνητής Νοημοσύνης (Artificial Intelligence) και αφορά στη μελέτη του προβλήματος της επιλογής και οργάνωσης ενεργειών(actions) με βάση την πρόβλεψη των προσδοκώμενων αποτελεσμάτων τους[17Οι ενέργειες εκτελούνται από έξυπνους πράκτορες(agents), robots, κ.λπ. με σκοπό να επιτευχθούν συγκεκριμένοι στόχοι(goals) που έχουν προκαθορισθεί.

Ένα πρόβλημα σχεδιασμού ενεργειών (Planning Problem) είναι ένα πρόβλημα όπου ζητάμε την εύρεση μιας ακολουθίας ενεργειών, οι οποίες εφαρμοζόμενες σε μια αρχική κατάσταση, έχουν

ως αποτέλεσμα την επίτευξη συγκεκριμένων στόχων π.χ. μεταφορά φορτίων, πλοήγηση οχημάτων κ.α.

Το πρόβλημα σχεδιασμού στη γενική του μορφή ορίζεται από τρεις περιγραφές :

Της αρχικής κατάστασης (initial state)

Των στόχων (goals)

Των διαθέσιμων ενεργειών (actions)

Η ακολουθία των ενεργειών που αποτελεί τη λύση ενός προβλήματος σχεδιασμού ονομάζεται σχέδιο (plan) ενώ το πρόγραμμα που την παράγει ονομάζεται σχεδιαστής (planner).

Η αρχική κατάσταση θα πρέπει να ληφθεί υπόψη από ένα σχεδιαστή για την εφαρμογή του πλάνου διότι θα δημιουργήσει ένα πλάνο, που όταν εφαρμοσθεί στην αρχική κατάσταση θα οδηγήσει στον επιθυμητό στόχο. Οι στόχοι συνήθως εκφράζονται ως ένα σύνολο από ιδιότητες που πρέπει να έχουν κάποια τιμή σε μια συγκεκριμένη κατάσταση, την κατάσταση που θα επιτευχθεί ο στόχος.

Τα κύρια θέματα που αντιμετωπίζονται για την αυτόματη επίλυση ενός προβλήματος ενεργειών είναι:

- Ο τρόπος αναπαράστασης των καταστάσεων και ενεργειών που πρόκειται να σχεδιαστούν και εξαρτάται από το πεδίο που θα εφαρμοσθεί το πρόβλημα. Η γλώσσα που θα επιλεχθεί για την αναπαράσταση συνήθως χρειάζεται προσαρμογή για τα διαφορετικά πεδία εφαρμογής του προβλήματος.
- Ο τρόπος που θα γίνει η αναζήτηση για μια αποτεσματική λύση του προβλήματος, ο αλγόριθμος αναζήτησης που θα χρησιμοποιηθεί, η πολυπλοκότητα που θα έχει και οι ευριστικές τεχνικές αναζήτησης που θα χρησιμοποιηθούν για την εύρεση του πλάνου.

Η μοντελοποίηση ενός προβλήματος σχεδιασμού ενεργειών χαρακτηρίζεται από τον τρόπο που προσεγγίζονται τα πιο πάνω θέματα. Εχουν προταθεί και εξελιχθεί διάφοροι αλγόριθμοι και συστήματα που διαφέρουν μεταξύ τους, είτε στον τρόπο που μοντελοποιούν τον κόσμο είτε στον τρόπο που αναζητούν την λύση του εκάστοτε προβλήματος και στις ευριστικές τεχνικές που χρησιμοποιούν για επιτάχυνση της διαδικασίας επίλυσης.

Σε ότι αφορά την πολυπλοκότητα του προβλήματος θα πρέπει να αναφέρουμε πως κατά την περιγραφή του προβλήματος σχεδίασης πρακτικά είναι αδύνατον να απαριθμηθούν όλες οι δυνατές καταστάσεις και οι δυνατές μεταβάσεις του συστήματος. Μια τέτοια περιγραφή θα ήταν τόσο εκτεταμένη που η ίδια η ανάπτυξη της θα απαιτούσε περισσότερο χρόνο από το να λύσει κανείς στην πραγματικότητα το πρόβλημα χειροκίνητα. Για το λόγο αυτό χρειάζεται μια αναπαράσταση του προβλήματος που θα μπορούν αυτόματα να υπολογισθούν οι δυναντές καταστάσεις και οι μεταβάσεις των καταστάσεων.

1.2. Επίλυση του προβλήματος Planning και η προσέγγιση του planning as satisfiability

Η κλασική προσέγγιση του Planning που ακολουθείται από σημαντικό αριθμό μεθόδων που έχουν προταθεί είναι του Planning as Deduction [30,31,32]. Οι λεπτομέρειες υλοποίησης των μεθόδων διαφέρουν αλλά όλες χρησιμοποιούν αξιώματα που καθορίζουν ότι τα αποτελέσματα μιας ενέργειας συνεπάγονται από τον εάν υπάρχει το συμβάν της ενέργειας στη δεδομένη κατάσταση και από το εάν ισχύουν οι προυποθέσεις της(preconditions). Στη συνέχεια αναζητείται μια επαγωγική απόδειξη της πρότασης που ισχυρίζεται πως η αρχική κατάσταση μαζί με μια ακολουθία ενεργειών συνεπάγεται την κατάσταση που υλοποιείται ο στόχος.

Στην επόμενη ενότητα παρουσιάζουμε μια διαφορετική τεχνική που συναντάται στη βιβλιογραφία[16] και παρουσιάζει ιδιαίτερο ενδιαφέρον, εκείνη του planning as satisfiability.

1.2.1. Planning as satisfiability

Η γενική ιδέα του Planning as satisfiability είναι να αντιστοιχισθεί το planning πρόβλημα σε ένα άλλο γνωστό πρόβλημα για το οποίο υπάρχουν αποδοτικοί αλγόριθμοι και διαδικασίες. Έτσι λύνοντας το μετασχηματισμένο πρόβλημα επιλύουμε κατ' επέκταση και το πρόβλημα του planning και βρίσκουμε τα έγκυρα πλάνα.

Planning	(encoded)	πρόβλημα Ικανοποίησης
Problem	---------->	πρότασης της Προτασιακής Λογικής

Η μέθοδος προτάθηκε από τους Kautz και Selman το 1992 [16] και υπάρχουν υλοποιημένοι κάποιοι planners όπως οι blackbox, satplan, lama, mp, κ.λπ. που στηρίζονται στη μέθοδο του planning as propositional satisfiability.

Για την κωδικοποίηση σε πρόβλημα Ικανοποιησημότητας Προτασιακής Λογικής παίζουν ρόλο κλειδί οι καταστάσεις του προβλήματος και οι μεταβάσεις των καταστάσεων. Σταδιακά η Φ χτίζεται με τα ακόλουθα πέντε είδη φόρμουλας.

- κωδικοποίηση αρχικής κατάστασης (Initial state)
- κωδικοποίηση τελικής κατάστασης(Goal state)
- κωδικοποίηση ενεργειών (Operators)
- κωδικοποίηση Frame axiom
- κωδικοποίηση Complete exclusion axiom

Κάθε μια από τις παραπάνω φόρμουλες παράγεται από ένα σετ αξιωμάτων του domain. Για ένα δοσμένο αριθμό n βημάτων που αποτελεί το μήκος του πλάνου παράγεται μια προτασιακή φόρμουλα Φn για ένα bounded Planning problem. Αν αυτή η φόρμουλα είναι ικανοποιήσιμη τότε υπάρχει ένα πλάνο με n βήματα. Για τον έλεγχο ικανοποιησιμότητας της Φn θα χρησιμοποιήσουμε έναν αλγόριθμο αναζήτησης ικανοποιησιμότητας λογικών εκφράσεων(theorem prover) για να βρούμε το μοντέλο που αληθεύει.

Μία πολύ γνωστή προσέγγιση στην επιστημονική κοινότητα για την επίλυση προβλημάτων ικανοποιησιμότητας λογικών εκφράσεων (το μετασχηματισμένο πλέον Planning πρόβλημα) είναι ο αλγόριθμος οπισθοδρόμησης (backtracking) των Davis και Putman[2]. Για να εφαρμοσθεί ο αλγόριθμος πρέπει οι προτάσεις να είναι γραμμένες σε κανονική συζευκτική μορφή και με οπισθοδρόμηση διερευνούνται όλες οι μερικώς αληθείς τιμές στις ατομικές προτάσεις της λογικής έκφρασης προκειμένου να βρεθεί κάποιος συνδυασμός τιμών των μεταβλητών της που την ικανοποιούν. Σε περίπτωση που δεν βρεθεί τέτοιος συνδυασμός, τότε ο αλγόριθμος επιστέφει την τιμή και χαρακτηρίζει τη λογική πρόταση ως μη ικανοποιήσιμη. Ο αλγόριθμος λειτουργεί αρκετά ικανοποιητικά ακόμα και για περιπτώσεις σύνθετων προβλημάτων, κυρίως λόγω του γεγονότος ότι μπορεί να απορίψει υποπροτάσεις που είναι ολόκληρα κλαδιά στο δέντρο αναζήτησης χωρίς να απαιτείται εξερεύνηση των φύλλων τους.

Η τυποποίηση ενός προβλήματος σχεδιασμού ενεργειών ως πρόβλημα ικανοποιησιμότητας έχει κάποια πλεονεκτήματα σε σχέση με την τυποποίηση ως deductive planning [16]. Με τον πρώτο τρόπο είναι εφικτό να ορίσουμε περιορισμούς σε οποιοδήποτε ενδιάμεσο στάδιο και όχι μόνο στο αρχικό ή τελικό στάδιο. Για παράδειγμα εάν θέλω να επιβεβαιώσω πως κάτι υπάρχει πάνω από το φορτίο C ή D τη χρονική στιγμή 5 δηλώνω στον ορισμό του προβλήματος

$$\neg clear(C,5) \ V \ \neg clear(D,5)$$

Τέτοιου είδους υποθέσεις είναι δύσκολο να περιγραφούν στην επαγωγική προσέγγιση. Επιπλέον στο [16] οι συγγραφείς αποδεικνύουν πως όταν το πρόβλημα επεκτείνεται επειδή προσθέτουμε περισσότερα αξιώματα αυτό μπορεί να βελτιώσει δραματικά την αποδοτικότητα των αλγορίθμων ικανοποιησιμότητας που βασίζονται σε greedy local search.

1.2.2.Παράδειγμα κωδικοποίησης ενός planning προβλήματος με την προσέγγιση του Planning as satisfiability

Ένα παράδειγμα που συνήθως χρησιμοποίειται στο σχεδιασμό ενεργειών είναι το παράδειγμα του Gripper [21], η μετακίνηση ενός robot από μια τοποθεσία σε κάποια άλλη και η μεταφορά φορτίων. Σε αυτή την ενότητα θα παρουσιάσουμε τον τρόπο κωδικοποίησης του προβλήματος σε Planning as satisfiability χρησιμοποιώντας ένα αντίστοιχο ιδιαίτερα απλοποιημένο παράδειγμα.

Στο πεδίο που θα κωδικοποιηθεί υπάρχει ένα `robot` το `r1` και δύο διαφορετικές τοποθεσίες οι `l1` και `l2`. Σκοπός είναι να μετακινηθεί το robot από την αρχική του θέση `l1` στον προορισμό του που είναι η τοποθεσία `l2`.

Robot: r1	Places : l1,l2	Operators	: move(r,l,l')
		Prec	: **a**t(r1,l1,0)
		Effects	: **a**t(r1,l2,1)

Σταδιακά θα χτίζουμε τη Φ με τα ακόλουθα πέντε είδη φόρμουλας.[22]

a) κωδικοποίηση αρχικής κατάστασης (Initial state)
b) κωδικοποίηση τελικής κατάστασης(Goal state)
c) κωδικοποίηση ενεργειών (Operators)
d) κωδικοποίηση Frame axiom
e) κωδικοποίηση Complete exclusion axiom

Το βήμα επίλυσης του προβλήματος είναι 1.

Α) Κωδικοποιούμε την αρχική κατάσταση(initial state) του προβλήματος. Η αρχική κατάσταση κωδικοποιείται με μια πρόταση που αποτελεί σύζευξη όσων ισχύουν στην αρχική κατάσταση και η άρνηση όσων δεν ισχύουν. Όλα αυτά είναι που συμβάινουν στο πρώτο βήμα , το βήμα 0.

Initial state **a**t(r1,l1,0) ∧ $\neg at(r1, l2,0)$

Β) Κωδικοποιούμε τις τελικές καταστάσεις(goal states) του προβλήματος. Η τελική κατάσταση κωδικοποιείται με μια πρόταση που αποτελεί σύζευξη των προτάσεων που ισχύουν σε όλες τις goal states.

goal state **a**t(r1,l2,1) ∧ $\neg at(r1, l1,1)$

c) Κωδικοποιούμε τους Operators του domain. Εδώ κωδικοποιούμε το γεγονός ότι μια ενέργεια, όταν εφαρμοσθεί σε μια κατάσταση, ισχύουν κάποια preconditions και όταν τελικά εφαρμοσθεί έχει κάποια effects. Στην κατάσταση που εφαρμόσθηκε(0) λοιπόν ισχύουν τα Precondition και στην επόμενη κατάσταση(1) φαίνονται τα effect αυτής. Για να μετακινηθελι από τη θέση 1 το robot στη θέση 2 σημαίνει πως ήταν στη θέση 1 τη χρονική στιγμή 0 (precondition). Αφού μετακινηθεί το robot στη θέση 2 σημαίνει πως βρίσκεται στη θέση 2 το robot(effect) τη χρονική στιγμή 1.

Operators

$move(r1, l1, l2,0) \Longrightarrow at(r1, l1,0) \wedge at(r1, l2,1)$ move1

$\equiv \quad \neg\, move(r1, l1, l2,0) \vee [at(r1, l1,0) \wedge at(r1, l2,1)]$

$move(r1, l2, l1,0) \Longrightarrow at(r1, l2,0) \wedge at(r1, l1,1$ move2

$\equiv \quad \neg\, move(r1, l2, l1,0) \vee [at(r1, l2,0) \wedge at(r1, l1,1)]$

d) Κωδικοποιούμε το Frame axiom. Πρέπει να δείξουμε ότι μια ενέργεια επηρεάζει μόνο ότι έχει ως effect και τίποτε άλλο. Δείχνουμε λοιπόν τι επηρεάστηκε και τι δεν επηρεάστηκε για να δηλώσουμε τα αποτελέσματα της εκτέλεση της κάθε ενέργειας ξεκάθαρα.

Frame axiom

$$at(r1, l1,0) \wedge \neg at(r1, l1,1) \Longrightarrow move(r1, l1, l2,0)$$
$$\neg at(r1, l2,0) \wedge at(r1, l2,1) \Longrightarrow move(r1, l1, l2,0)$$
$$at(r1, l2,0) \wedge \neg at(r1, l2,1) \Longrightarrow move(r1, l2, l1,0)$$
$$\neg at(r1, l1,0) \wedge at(r1, l1,1) \Longrightarrow move(r1, l2, l1,0)$$

d) Κωδικοποιούμε το complete exclusion axiom. Εδώ δηλώνουμε ότι μόνο ένας Operator εκτελείται σε κάθε στάδιο. Για κάθε στάδιο κ, και κάθε ζευγάρι των operators o_k kai o_k ' η έκφραση είναι

$$\neg \boldsymbol{o_\kappa} \vee \neg \boldsymbol{o'_\kappa}$$

Που είναι λογικά ισοδύναμο με το $\neg(\boldsymbol{o_\kappa} \vee \boldsymbol{o'_\kappa})$ και σημαίνει όχι και τα δύο να ισχύουν στην ίδια κατάσταση.

Complete exclusion axiom

$$\neg move(r1, l1, l2,0) \vee \neg move(r1, l2, l1,0)$$

Η προτασιακή φόρμουλα Φ που εκφράζει το Planning πρόβλημα σε ένα πρόβλημα ικανοποιησιμότητας είναι η λογική σύζευξη όλων των παραπάνω υποπροτάσεων. Η Φ που κατασκευάζεται είναι η ακόλουθη :

$\boldsymbol{\Phi} =$ at(r1,l1,0) $\wedge$ $\neg at(r1, l2,0) \wedge$ at(r1,l2,1) $\wedge$ $\neg at(r1, l1,1)] \wedge$

$[\neg\, move(r1, l1, l2,0) \vee \big(at(r1, l1,0) \wedge at(r1, l2,1)\big)] \wedge$

$[\neg\, move(r1, l2, l1,0) \vee (at(r1, l2,0) \wedge at(r1, l1,1)] \wedge$

$[\neg at(r1, l1,0) \vee at(r1, l1,1) \vee move(r1, l1, l2,0)] \wedge$

$[at(r1, l2,0) \vee \neg at(r1, l2,1) \vee move(r1, l1, l2,0)] \wedge$

$[\neg at(r1, l2,0) \vee at(r1, l2,1) \vee move(r1, l2, l1,0)] \wedge$

$[at(r1, l1,0) \vee \neg at(r1, l1,1) \vee move(r1, l2, l1,0)] \wedge$

$[\neg move(r1, l1, l2,0) \vee \neg move(r1, l2, l1,0)]$

Αν αυτή η φόρμουλα είναι ικανοποιήσιμη τότε υπάρχει κάποιο πλάνο και μπορούμε να το συμπεράνουμε από το μοντέλο που αληθεύει. Για τον έλεγχο ικανοποιησιμότητας της Φ θα τη χρησιμοποιήσουμε έναν αλγόριθμο εύρεσης ικανοποιησιμότητας λογικών εκφράσεων(theorem prover) για να βρούμε το μοντέλο που αληθεύει.

Μετά τη δημιουργία της συνολικής Φ τη μετατρέπουμε σε CNF μορφή και αναζητούμε το μοντέλο που αληθεύει με τον αλγόριθμο των Davis Putman. Το μοντέλο που προκύπτει ότι αληθεύει η Φ είναι :

M = {$\mathbf{at}(\mathbf{r1}, \mathbf{l1}, \mathbf{0})$, at(r1,l2,1) , $\neg\, move(r1, l2, l1,0)$, $move(r1, l1, l2,0)$ }

και μας δείχνει το πλάνο για την μετακίνηση του robot. Η εκτέλεση της ενέργειας $move(r1, l1, l2,0)$ θα μετακινήσει το robot από τη θέση 1 στη θέση 2.

1.3. Η σύνθεση υπηρεσιών ιστού ως ένα πρόβλημα σχεδιασμού ενεργειών

Η σύνθεση των υπηρεσιών ιστού είναι η χρησιμοποίηση κάποιων υπηρεσιών με την κατάλληλη σειρά για να επιτύχουμε ένα συγκεκριμένο στόχο, το αίτημα του χρήστη. Αυτό το πρόβλημα είναι ένα πρόβλημα σχεδιασμού ενεργειών. Θα μπορούσε να εκφρασθεί ως ένα πρόβλημα σχεδιασμού P που περιγράφεται από την τριάδα (s,g,a) όπου s είναι η αρχική κατάσταση πριν τη σύνθεση, g είναι ο στόχος (αίτημα του χρήστη) και a το σύνολο των ενεργειών που πρέπει να γίνουν. Οι ενέργειες στο πρόβλημα μας είναι οι λειτουργίες των υπηρεσιών. Ο planner θα κληθεί να λύσει το πρόβλημα ακολουθώντας κάποιο σχέδιο, την εκτέλεση συγκεκριμένων λειτουργιών υπηρεσιών ιστού με προκαθορισμένη σειρά. Το πρόβλημα θα λυθεί εάν η εκτέλεση των συγκεκριμένων λειτουργιών του σχεδίου μεταφέρουν το πρόβλημα από την αρχική κατάσταση στην κατάσταση που περιγράφεται από τον στόχο και είναι το αίτημα του χρήστη. Η γενική παραδοχή των μεθόδων που αντιμετωπίζουν τη σύνθεση ως πρόβλημα σχεδιασμού ενεργειών είναι πως κάθε υπηρεσία ιστού μπορεί να προσδιοριστεί από τα precondition and effects της. Κάθε υπηρεσία λαμβάνει κάποιο input και παράγει κάποιο output και μετά την εκτέλεση της αλλάζει την κατάσταση του κόσμου στον οποίο εκτελείται.

Στο [1,7] έχουμε αναφέρει κάποιες από τις κυριότερες μεθόδους που συναντήσαμε στην βιβλιογραφία και αντιμετωπίζουν το πρόβλημα της σύνθεσης των υπηρεσιών ιστού ως ένα πρόβλημα σχεδιασμού ενεργειών. Παρόλο που σε όλες τις μεθόδους το πρόβλημα επιλύεται ως ένα πρόβλημα σχεδιασμού ενεργειών η υλοποίηση της κάθε μεθόδου διαφέρει. Διαφέρουν στην γλώσσα που χρησιμοποιούν για αναπαράσταση των καταστάσεων και ενεργειών, στον αλγόριθμο που χρησιμοποιούν για την αναζήτηση του πλάνου, στην πολυπλοκοτητα, στις ευριστικές τεχνικές

αναζήτησης. Στην αναφορά[15] οι McIlraith and Son παρουσιάζουν μια μέθοδο για σύνθεση υπηρεσιών εφαρμόζοντας τεχνικές λογικής παραγωγής και προτείνουν σύνθεση με τη χρήση της situation calculus. Στις αναφορές [20,19] η γενική ιδέα πίσω από την εφαρμογή της μεθόδου είναι η περιγραφή του προβλήματος σε planning domain definition language (PDDL) και σύνθεση μέσω κλασικού σχεδιασμού ενεργειών. Ο Rao στο [13] προτείνει μια μέθοδο για αυτοματοποιημένη σύνθεση υπηρεσιών ιστού και χρησιμοποιεί για την περιγραφή του συστήματος ένα fragment της Γραμμικής Λογικής (Linear Logic) και στη συνέχεια επιλύει το πρόβλημα ως planning as satisfiability πρόβλημα. Στην αναφορά [14] οι συγγραφείς λαμβάνοντας υπόψη τις υπηρεσίες ως καθήκοντα ανάγουν το πρόβλημα της σύνθεσης σε Hierarchical Task Network (HTN) Planning πρόβλημα.

Υπάρχουν πολλές ακόμη προτάσεις για τη σύνθεση σημασιολογικών υπηρεσιών ιστού και αυτό οφείλεται στο γεγονός ότι η ερευνητική κοινότητα ασχολείται όλο και περισσότερο με το θέμα αυτό. Ταυτόχρονα όμως υπάρχουν αρκετά ζητήματα που παραμένουν ανοιχτά όσον αφορά στη σύνθεση υπηρεσιών ιστού. Τα ζητήματα αυτά έχουν να κάνουν με τον βαθμό αυτοματοποίησης της διαδικασίας αυτής, με θέματα ασφάλειας, με θέματα προτυποποίησης της σημασιολογικής περιγραφής, κλίμακα υλοποίησης κ.α.

Από τις μεθόδους που μελετήθηκαν διακρίνουμε εκείνες που προσεγγίζουν το πρόβλημα ως Planning as satisfiability για τα πλεονεκτήματα που παρουσιάζουν έναντι των άλλων[18]. Σε αυτές τις μεθόδους η γλώσσα Περιγραφής του προβλήματος είναι η Προτασιακή Λογική και έμμεσα εκφράζεται η αλλαγή της κατάστασης όταν εκτελείται κάποια ενέργεια. Εναλλακτικά αναζητούμε κάποια γλώσσα που να περιέχει την κατάλληλη εκφραστικότητα για να περιγράψει το πρόβλημα. Μελετήσαμε τη γλώσσα της Propositional Dynamic Logic (PDL)[5] που δημιουργήθηκε για program verification [5], μπορεί να περιγράψει την ακολουθία των καταστάσεων μετά την εκτέλεση των ενεργειών και εν' συντομία την παρουσιάζουμε στην επόμενη ενότητα. Για τη γλώσσα αυτή έχει μελετηθεί το θέμα της ικανοποιησιμότητας και έχει αποδειχθεί[5] ότι είναι εκθετικά πλήρες (EXP-πλήρες).

1.4. Προτασιακή Δυναμική Λογική - Propositional Dynamic Logic

Η Προτασιακή δυναμική λογική (PDL) είναι μια μια επέκταση της τροπικής λογικής που πρωτοπαρουσιάσθηκε από τους Fisher Ladner[3][4] στα τέλη της δεκαετίας του 1970 για τον έλεγχο ορθότητας προγραμμάτων. Έχει χρησιμοποιηθεί όχι μόνο για program verification αλλά και σε άλλους τομείς της πληροφορικής όπως knowledge representation and artificial intelligence.

Στις ενότητες που ακολουθούν θα δούμε μια σύντομη περιγραφή της γλώσσας PDL, την περιγραφή του αλγόριθμου ικανοποιησιμότητας της PDL και την παρουσίαση ενός theorem prover που αναπτύξαμε για την PDL. Η εφαρμογή αυτή υλοποιήθηκε χρησιμοποιώντας τον αλγόριθμο ικανοποιησιμότητας του Pratt έχει ως σκοπό να αποδεικνύει εάν μια πρόταση της PDL γλώσσας είναι ικανοποιήσιμη ή όχι.

1.4.1. ΣΥΝΤΑΞΗ ΤΗΣ PDL

Η σύνταξη προϋποθέτει τον καθορισμό του αλφαβήτου της γλώσσας, δηλαδή του συνόλου των συμβόλων με τα οποία μπορεί να κατασκευαστούν αποδεκτές προτάσεις της PDL. Συντακτικά η PDL αποτελείται από ένα μίγμα τριών κλασικών συστατικών της προτασιακής λογικής, της τροπικής λογικής και της άλγεβρας των κανονικών εκφράσεων. Μια πρόταση της PDL μπορεί να περιλαμβάνει εκφράσεις από προγράμματα και από ατομικές ή σύνθετες προτάσεις.

Χρησιμοποιούμε P_0 για να δηλώσουμε το σύνολο των ατομικών προγραμμάτων, και S_0 για να δηλώσουμε τις ατομικές φόρμουλες. Οι τύποι και τα προγράμματα της PDL ορίζονται, αντίστοιχα, από τους ακόλουθους κανόνες γραμματικής:

$$\varphi := p\ (p \in At\ S_0) \mid \neg\ \varphi \mid \varphi \rightarrow \varphi \mid [\alpha]\varphi$$
$$\alpha:\ = \pi\ (\pi \in At\ P_0)\ \mid \varphi?\ \mid \alpha; \alpha \mid\ \alpha \cup \alpha \mid \alpha^*$$

Με βάση τα παραπάνω μπορεί να οριστεί ο τελεστής <> (diamond) ως εξής:

$$\langle a\rangle\varphi =\ \neg\,[\alpha]\neg\varphi$$

Η διαισθητική ερμηνεία των εκφράσεων που ακολουθούν είναι:

$\langle a\rangle\varphi$: είναι δυνατόν να εκτελεστεί με επιτυχία το πρόγραμμα α και το σύστημα να τερματίσει σε μια κατάσταση που αληθεύει η φ.

$[\alpha]\varphi$: όταν η εκτέλεση του προγράμματος α τερματίσει, θα αληθεύει η φ.

$[\alpha; \beta]\varphi$: το πρόγραμμα $\alpha; \beta$ είναι η ακολουθιακή σύνθεση των προγραμμάτων α και β και εκφράζει την εκτέλεση του προγράμματος β μετά την ολοκλήρωση της εκτέλεσης του προγράμματος α.

$[\alpha \cup \beta]\varphi$: το πρόγραμμα $\alpha \cup \beta$ δηλώνει μη ντετερμινιστική επιλογή εκτέλεσης ενός από τα δύο προγράμματα α, β.

$[\varphi?]\psi$: το πρόγραμμα $\varphi?$ είναι ένας έλεγχος εάν ισχύει η φ , εάν πράγματι αληθεύει η φ τότε σε αυτή την κατάσταση αληθεύει και το ψ.

$[\alpha^*]\varphi$: ο τελεστής * δηλώνει απροσδιόριστη επανάληψη εκτέλεσης του προγράμματος α, πιθανόν και μηδενική. Ξεκινώντας από την τρέχουσα κατάσταση μετά από κάθε εκτέλεση του προγράμματος α οδηγούμαστε σε μια νέα κατάσταση όπου θα ισχύει η φ.

Μια κατάσταση είναι μια στιγμιαία περιγραφή της πραγματικότητας του domain. Ένα πρόγραμμα μπορεί να θεωρηθεί πως είναι μια συνεχόμενη μετάβαση καταστάσεων. Δεδομένης μιας αρχικής κατάστασης, ένα πρόγραμμα θα περάσει από μια σειρά ενδιάμεσων καταστάσεων και πιθανόν τελικά να τερματίσει σε μια τελική κατάσταση η οποία θα μπορεί να περιγραφεί μέσω των ατομικών και σύνθετων προτάσεων.

Οι δομές ερμηνείας στην PDL[6] είναι αυτές των πολυτροπικών συστημάτων, δηλαδή ένα μοντέλο $N = < W, \rho, \twoheadrightarrow >$ αποτελείται από ένα (μη κενό) σύνολο W, μια ερμηνεία ατομικών προτάσεων ως υποσυνόλων του W, ρ(p) $\subseteq W$, και μια ερμηνεία ατομικών προγραμμάτων ως διμελών σχέσεων στο W, $\overset{\pi}{\rightarrow} \subseteq W \times W$.

Η γλώσσα των προγραμμάτων είναι αρκετά επαρκής ώστε να οριστούν τόσο ένα conditional όσο και ένα while loop.

If φ then α else β	=	$\varphi\,?\,\alpha \cup (\neg\varphi)?\,\beta$
While φ do α	=	$(\varphi\,?\,\alpha)^*(\neg\varphi)$

Η προτασιακή δηναμική λογική είναι αποφασίσιμη και έχει δειχθεί ότι το πρόβλημα ικανοποιησιμότητας προτάσεων είναι εκθετικά πλήρες[8]. Στην επόμενη ενότητα θα δούμε τον αλγόριθμο ικανοποιησιμότητας του Pratt με τη βοήθεια του οποίου γίνεται έλεγχος εγκυρότητας κάποιας πρότασης της PDL.

1.4.2. ΑΛΓΌΡΙΘΜΟΣ ΙΚΑΝΟΠΟΙΗΣΙΜΌΤΗΤΑΣ ΤΟΥ PRATT ΓΙΑ ΤΗΝ PDL

Τα βήματα του αλγόριθμου ικανοποιησιμότητας όπως τα περιέγραψε ο Pratt[8] στον αλγόριθμο ικανοποιησιμότητας [7] που έγραψε για την PDL έχουν ως εξής:

Βήμα 1ο – Δημιούργησε πρώτα το Fisher Ladner set

Αυτό αποτελείται από όλες τις καταστάσεις u ώστε $u \subseteq FL(\varphi) \cup \{\neg\psi \,|\psi \in FL\,(\varphi)\}$

και για κάθε ψ που ανήκει στη FL(φ) ένα από τα ψ ή $\neg\psi$ ανήκουν στην κατάσταση u

Βήμα 2ο – Για κάθε κατάσταση $u \in M_0$

έλεγξε εάν στην κατάσταση u ισχύουν τα αξιώματα που περιγράφονται στον Πίνακα Α, εάν όχι απέρριψε αυτή την κατάσταση.

Οι εναπομείναντες καταστάσεις απαρτίζουν το $\mathcal{M}_1$

Το μοντέλο $\mathcal{M}_1$ αποτελείται από τις καταστάσεις που πέρασαν τον παραπάνω έλεγχο και από τις ακμές που δημιουργούνται μεταξύ των καταστάσεων βάσει του ορισμού

$$\mathrm{m}_{m_i}(p) = \{u \in M_i | p \in u\}$$

$$\mathrm{m}_{m_i}(a) = \left\{(u,v) \in M_i^2 \,\middle|\, \gamma\iota\alpha\ \acute{o}\lambda\alpha\ \tau\alpha\ [\alpha]\psi\ \in FL(\varphi), \varepsilon\acute{\alpha}\nu\ [\alpha]\psi \in u, \tau\acute{o}\tau\varepsilon\ \psi \in v\right\}$$

Βήμα 3ο – Για κάθε κατάσταση $u \in \mathcal{M}_1$ κάνε τον επιπλέον έλεγχο

Βρες μια φόρμουλα της μορφής $[\alpha]\psi\ \in FL(\varphi)$ και μια κατάσταση που παραβιάζει τη συνθήκη

$$(\forall v\ \left((u,v) \in m_{m^i}(a) \Rightarrow \psi \in v\right)) \Rightarrow [a]\psi\ \in u$$

σε αυτή την περίπτωση διέγραψε την κατάσταση u_i που παραβιάζει τη συνθήκη και κατασκεύασε το μοντέλο $\mathcal{M}_{i+1}$ χωρίς το u_i

Επανέλαβε το τελευταίο βήμα έως ότου δεν υπάρχουν άλλες καταστάσεις u_i να διαγραφούν.

Ο αλγόριθμος θα τερματίσει δεδομένου ότι υπάρχει πεπερασμένος αριθμός καταστάσεων και για να επαναληφθεί το βήμα τρία θα πρέπει να διαγραφεί κάποια κατάσταση. Επιπλέον, όταν ο αλγόριθμος τερματίσει έχει ως αποτέλεσμα ένα μοντέλο $\mathcal{M}_i$ όπου σε κάθε κατάσταση u που έχει απομείνει στο μοντέλο αυτό η φ είναι ικανοποιήσιμη.

<u>Πίνακας Α</u>

Τα αξιώματα της Κλασικής Προτασιακής Λογικής που θα ελεγχθούν

$$\varphi \rightarrow (\psi \rightarrow \varphi)$$

$$\big(\varphi \rightarrow (\psi \rightarrow \chi)\big) \rightarrow ((\varphi \rightarrow \psi) \rightarrow \chi))$$

$$(\neg\varphi \rightarrow \neg\psi) \rightarrow ((\neg\varphi \rightarrow \psi) \rightarrow \varphi)$$

Τα αξιώματα της PDL που θα ελεγχθούν

$$[\alpha \cup b]\varphi\ \leftrightarrow [\alpha]\varphi \Lambda [\beta]\varphi$$

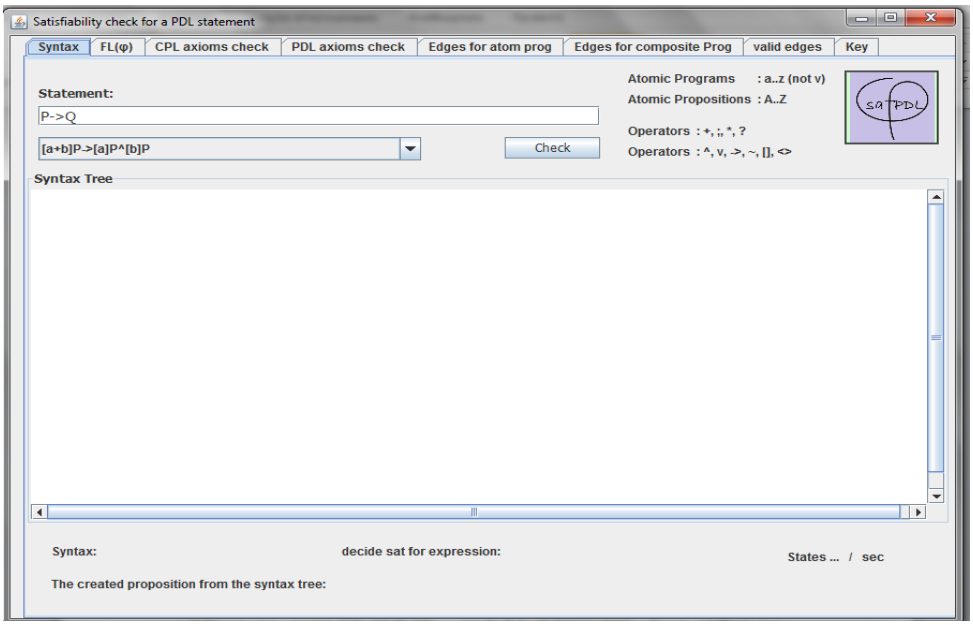
$[\alpha;\beta]\varphi \leftrightarrow [\alpha][\beta]\varphi$

$[\psi?]\varphi \leftrightarrow (\psi \rightarrow \varphi)$

$\varphi \wedge [\alpha][\alpha^*]\varphi \leftrightarrow [\alpha^*]\varphi$

1.4.3. ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ ΜΕ ΒΑΣΗ ΤΟΝ ΑΛΓΟΡΙΘΜΟ ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑΣ ΓΙΑ ΤΗΝ PDL

Στην ενότητα αυτή θα παρουσιάσουμε εν συντομία την εφαρμογή satPDL που αναπτύξαμε και υλοποιεί τον αλγόριθμο ικανοποιησιμότητας για την PDL. Στην εικόνα που ακολουθεί είναι το user interface της εφαρμογής. Ο χρήστης καταχωρεί την πρόταση που θέλει να ελέγξει και στη συνέχεια μπορεί να δει το συντακτικό δέντρο της πρότασης καθώς και να προχωρήσει στον έλεγχο ικανοποιησιμότητας της πρότασης. Υπάρχει η δυνατότητα ο χρήστης «να εκτελέσει» βήμα-βήμα τον αλγόριθμο επιλέγοντας τις αντίστοιχες καρτέλες ώστε να παρακολουθεί την προοδευτική απόρριψη των καταστάσεων μετά από τους ελέγχους που γίνονται σύμφωνα με τον αλγόριθμο ή να ζητήσει απ' ευθείας την εμφάνιση του μοντέλου στο οποίο η πρόταση είναι ικανοποιήσιμη.



Ως παράδειγμα θα δείξουμε τον έλεγχο της πρότασης της PDL $\varphi = [\alpha;b]P \rightarrow [a][b]P$. Στην αρχή σύμφωνα με τον αλγόριθμο γίνεται η δημιουργία του Fisher Ladner set και αυτό φαίνεται στην αντίστοιχη καρτέλα. Μετά τη δημιουργία του σετ δημιουργείται ένας πίνακας με 2^5= 32 γραμμές που είναι οι αρχικές 32 δυνατές καταστάσεις. Μετά τον έλεγχο των αξιωμάτων της CPL

απομένουν 20 από τις 32 καταστάσεις ενώ μετά τον έλεγχο των αξιωμάτων της PDL απομένουν 8 από τις 32 καταστάσεις. Κατά τον έλεγχο του τρίτου βήματος του αλγορίθμου ικανοποιησιμότητας δεν απορρίπτεται κάποια κατάσταση και το τελικό μοντέλο έχει τις καταστάσεις

FL($[\alpha;b]P \rightarrow [a][b]P$) = {

	$[\alpha;b]P \rightarrow [a][b]P$	$[\alpha;b]P$,	$[a][b]P$,	[b]P)	P }
u0.	1	1	1	1	1
u1.	1	0	0	1	1
u2.	1	1	1	0	1
u3.	1	0	0	0	1
u4.	1	1	1	1	0
u5.	1	0	0	1	0
u6.	1	1	1	0	0
u7.	1	0	0	0	0

Οι ακμές του μοντέλου στο οποίο ικανοποιείται η πρόταση φ παρουσιάζονται στον πίνακα που ακολουθεί.

πρόγραμμα	state from /	state to
edge_0.	a, 0,	0,
edge_1.	a, 0,	1,
edge_2.	a, 0,	4,
edge_3.	a, 0,	5,
edge_4.	a, 2,	0,
edge_5.	a, 2,	1,
edge_6.	a, 2,	4,
edge_7.	a, 2,	5,
edge_8.	a, 4,	0,
edge_9.	a, 4,	1,
edge_10.	a, 4,	4,
edge_11.	a, 4,	5,
edge_12.	a, 6,	0,
edge_13.	a, 6,	1,
edge_14.	a, 6,	4,
edge_15.	a, 6,	5,

πρόγραμμα	state from /	state to
edge_0.	b, 0,	0,
edge_1.	b, 0,	1,
edge_2.	b, 0,	2,
edge_3.	b, 0,	3,

edge_4. b, 1, 0,

edge_5. b, 1, 1,

edge_6. b, 1, 2,

edge_7. b, 1, 3,

edge_8. b, 4, 0,

edge_9. b, 4, 1,

edge_10. b, 4, 2,

edge_11. b, 4, 3,

edge_12. b, 5, 0,

edge_13. b, 5, 1,

edge_14. b, 5, 2,

edge_15. b, 5, 3,

1.5. Γενική Περιγραφή της Μεθόδου που θα χρησιμοποιηθεί για τη σύνθεση Υπηρεσιών Ιστού.

Το πρόβλημα της σύνθεσης υπηρεσιών ιστού θα μπορούσε να περιγραφεί σε μια πρόταση ως εξής: δεδομένου ενός σύνθετου στόχου που θα μπορούσε να παρουσιασθεί ως μια υπηρεσία με pre- conditions και effects, σύνθεσε κάποιες από τις κατάλληλες και διαθέσιμες υπηρεσίες του domain ώστε να ικανοποιηθεί ο στόχος. Η πρώτη πρόκληση είναι να βρεθεί η γλώσσα που θα μπορεί να περιγράψει τις υπηρεσίες προς σύνθεση και στη συνέχεια να βρεθεί ο τρόπος για να αυτοματοποιηθεί η διαδικασία σε πλαίσια χρόνου και κόστους που είναι υλοποιήσιμα.

Στη γλώσσα της PDL που μελετήσαμε η έκφραση $\langle\pi\rangle\psi$ εκφράζει πως το πρόγραμμα π τελειώνει σε μια κατάσταση που ισχύει η ψ. Η έκφραση φ->$\langle\pi\rangle\psi$ είναι έγκυρη εάν για κάθε κατάσταση s που ικανοποιείται η precondition φ η εκτέλεση του προγράμματος π αφενός τερματίζει σε μια νέα κατάσταση s' και αφετέρου σε αυτή την κατάσταση που τερματίζει ισχύουν τα effects που περιγράφονται από την πρόταση ψ.

Το παραπάνω πρόβλημα θα μπορούσε να παρουσιασθεί με την ακόλουθη λογική έκφραση στη γλώσσα PDL [12] η οποία περιγράφει το planning πρόβλημα που πρέπει να επιλύσουμε:

$$\phi \text{ -> } < (\alpha_1 \cup \alpha_2 .... \cup \alpha_n) *> \psi$$

Η διαισθητική ερμηνεία της πρόταση της PDL είναι: δοσμένης μια αρχικής κατάστασης που περιγράφεται από την πρόταση φ έλεγξε εάν υπάρχει μια πεπερασμένη ακολουθία ενεργειών του

συνόλου {α1, α2, .. , αν} που εάν εκτελεστεί θα τερματίσει σε μια κατάσταση που θα ισχύει ο επιθυμητός στόχος που περιγράφεται από την πρόταση ψ.

Ο έλεγχος ύπαρξης ενός πλάνου το οποίο όταν εφαρμόζεται σε μια αρχική κατάσταση που ικανοποιεί την ιδιότητα φ επιτυγχάνει έναν επιθυμητό στόχο ψ ανάγεται στον έλεγχο εγκυρότητας της παραπάνω πρότασης. Το φ περιγράφει την αρχική κατάσταση του συστήματος όταν ο χρήστης αιτείται την εκτέλεση κάποιας λειτουργίας. Εάν η λειτουργία που αιτείται ο χρήστης είναι ατομική θα εκτελεστεί κάποια από τις ατομικές ενέργειες, σε διαφορετική περίπτωση θα πρέπει να βρεθεί η κατάλληλη σύνθεση των ενεργειών α1, α2, .. , αn να εκτελεσθεί ώστε το σύστημα να μεταβεί στην κατάσταση στόχο που περιγράφεται από την ψ (τελική κατάσταση) .

Η προσέγγιση που προτείνουμε είναι μια μέθοδος για την αυτόματη και δυναμική σύνθεση υπηρεσιών ιστού που βασίζεται σε περιγραφή του προβλήματος σε ένα σύστημα λογικής που θα βασίζεται στην PDL. Το πρόβλημα περιγράφεται με αξιώματα σε αυτό το σύστημα, εκφράζονται οι παράμετροι που προσδιορίζουν την αρχική κατάσταση του συστήματος, οι τρόποι μετάβασης μεταξύ των καταστάσεων και η τελική κατάσταση που επιθυμούμε να μεταβεί το σύστημα μετά τη σύνθεση. Το πρόβλημα αυτό αποτελεί ένα πρόβλημα σχεδιασμού ενεργειών και η μέθοδος που ακολουθούμε για να το επιλύσουμε είναι του planning as satisfiability.

Αφού συνταχθεί η πρόταση που θα ελεγθεί με τη χρήση ενός theorem prover κάνουμε έλεγχο εγκυρότητας αυτής και βρίσκουμε το μοντέλο στο οποίο ικανοποιείται. Ο theorem prover θα στηριχθεί στην επέκταση του αλγόριθμου ικανοποιησιμότητας του Pratt για την PDL. Το πλάνο των ενεργειών προς εκτέλεση για να εκτελεστεί η σύνθετη υπηρεσία προκύπτει άμεσα από το μοντέλο στο οποίο ικανοποιείται η πρόταση.

1.6 Μελλοντικοί στόχοι

Δόθηκε μια γενική περιγραφή για την οπτική της μεθόδου που θα προταθεί για τη μοντελοποίηση της σύνθεσης των υπηρεσιών ιστού.

Για την υλοποίηση της μεθόδου που αντιμετωπίζουν το πρόβλημα της σύνθεσης οι στόχοι είναι:

- να προσδιορισθεί η περιγραφής της αρχιτεκτονικής που θα λυθεί το πρόβλημα
- να προσδιοριστούν οι απαιτήσεις της γλώσσας για την περιγραφή του προβλήματος της σύνθεσης και εάν η εκφραστικότητα της PDL είναι αρκετή
- να προσδιορισθεί η σύνταξη για τις λειτουργίες των υπηρεσιών, actions, inputs, outputs, preconditions, effects, parameters

- να διερευνηθεί κατά πόσο μπορεί να προσδιοριστεί η προτεραιότητα κάποιων ενεργειών στο στάδιο της περιγραφής του προβλήματος (partial ordering)
- να διερευνηθεί η επικοινωνία των υπηρεσιών και το πέρασμα τιμών
- να περιγραφεί στην παραπάνω γλώσσα το πρόβλημα της σύνθεσης
- να αποδειχθεί ο τρόπος της επίλυσης του προβλήματος
- να επεκταθεί ο αλγόριθμος ικανοποιησιμότητας του Pratt για την PDL.

ΠΑΡΑΡΤΗΜΑ

ΕΛΕΓΧΟΙ ΕΚΦΡΑΣΕΩΝ ΠΟΥ ΠΡΑΓΜΑΤΟΠΟΙΗΘΗΚΑΝ ΣΤΗΝ ΕΦΑΡΜΟΓΗ SATPDL

- [a+b]P->[a]P^[b]P
- [a]P^[b]P->[a+b]P
- [a;b]P->[a][b]P
- [a][b]P->[a;b]P
- [P?]Q->(P->Q)
- (P->Q)->[P?]Q
- [a*]P->P^[a][a*]P
- P^[a][a*]P->[a*]P
- F->(G->F)
- (F->(G->M))->((F->G)->(F->M))
- (~G->~F)->((~G->F)->G)
- [P?]Qv[~P?]R
- [(P^Q)?]Q
- ((P->Q)->P)->P
- P^(P->Q)->Q
- ~(((P->Q)->P)->P)
- P->~(Q->P)
- ((P->[a]P)->P)->P
- [a+b]P^[b]P->[a+b]P
- [(P?;a)*]P->[(P?;a;a)*]P
- [(P?;a)*]P->[(P?;a);a]P
- [(P?;a)*]P->[(P?;a);a][(P?;a)*]P
- [(P?;a)*]P->[(P?;a);a][((P?;a);a)*]P
- (P->[a](Q^X))^[a](X->[b]Y)^[a;b](Y->[c+d]Z)^P->[a;b;(c+d)]Y

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Π.Τσούτσα, " Composition of Semantic Web Services ", 2011.

[2] Davis, M. and H. Putnam (1960). A Computing Procedure for Quantification Theory. Journal of the ACM 7, 201–215.

[3] Michael J. Fischer, Richard E. Ladner: Propositional Dynamic Logic of Regular Programs. J. Comput. Syst. Sci. 18(2): 194-211 (1979)

[4] Michael J. Fischer, Richard E. Ladner: Propositional Modal Logic of Programs (Extended Abstract) STOC 1977: 286-294

[5] D. Kozen and J. Tiuryn, Logics of programs, Handbook of Theoretical Computer Science — Formal Models and Semantics (Jan van Leeuwen, ed.), Elsevier Science Publishers, Amsterdam, 1990

[6] Χ.Χαρτώνας, Λογική και Πληροφορική,2000

[7] Π.Τσούτσα, "SOA Architecture and Web Services", 2010.

[8] Harel, D., Kozen, D., Tiuryn, J., Dynamic Logic, MIT Press, Foundations of Computing Series, 2000.

[9] Handbook of Automated Reasoning, J. Alan Robinson, Andrei Voronkov

[10] Automated Planning: Theory & Practice (The Morgan Kaufmann Series in Artificial Intelligence)

[11] Handbook of Logic in Computer Science, S. Abramsky, Dov M. Gabbay, T. S. E. Maibaum

[12] On the Dynamic Logic of Agency and Action,T.Hartonas, Technical Report, June 2012, TEI Larissa, Greece.

[13] J.Rao, Application of Linear Logic to Web Service Composition, First International Conference on Web Services, Las Vegas, 2003.

[14] Evren Sirin, B.Parsia, Dan Wu, J.Hendler, Dana Nau, HTN Planning for web service composition using shop2, Elsevier, 2004.

[15] S. McIlraith and T. C. Son. Adapting Golog for composition of Semantic Web services. In Proceedings of the 8th International Conference on Knowledge.

[16] Planning as satisfiability, H.Kautz and Bart Selman, Proc. ECAI-92, Vienna, Austria, 1992, 359–363.

[17] Automated Planning: Theory & Practice, Malik Ghallab (Author), Dana Nau (Author), Paolo Traverso (Author), (The Morgan Kaufmann Series in Artificial Intelligence)

[18] Allen, J.F.,H. Kautz, R. Pelavin, and J. Tenenberg Reasoning.About Plans Morgan Kaufmann, 1991.

[19] J. Peer, «A PDDL Based Tool for Automatic Web Service Composition».Principles and Practice of Semantic Web Reasoning, 2004.

[20] McDermott, D.: Estimated-regression planning for interactions with Web services. In: Proceedings of the 6th International Conference on AI Planning and Scheduling, Toulouse, France. AAAI Press, Menlo Park (2002)

[21] AIPS-98 Planning Competition, http://www.cs.cmu.edu/~aips98/

[22] Planning Algorithms, Steven M. LaValle,2006