Ατομική Διπλωματική Εργασία

ΥΛΟΠΟΙΗΣΗ ΔΙΑΣΥΝΔΕΣΗΣ ΑΝΑΜΕΣΑ ΣΕ ΕΡΓΑΛΕΙΟ ΜΕ ΓΡΑΦΙΚΗ ΔΙΕΠΙΦΑΝΕΙΑ ΥΛΟΠΟΙΗΜΕΝΟ ΣΕ JAVA ΚΑΙ ΕΡΓΑΛΕΙΟ ΑΝΑΛΥΣΗΣ ΑΝΑΣΤΡΕΨΙΜΩΝ ΔΙΚΤΥΩΝ ΠΕΤΡΙ ΜΕ ΤΗ ΧΡΗΣΗ ANSWER-SET PROGRAMMING.

Δημήτρης Βερέης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ιούνιος 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τίτλος Ατομικής Διπλωματικής Εργασίας:
Υλοποίηση διασύνδεσης ανάμεσα σε εργαλείο με γραφική διεπιφάνεια υλοποιημένο σε Java και εργαλείο ανάλυσης αναστρέψιμων δικτύων Πέτρι με τη χρήση Answer-Set Programming.

Δημήτρης Βερέης

Επιβλέπων Καθηγητής
Άννα Φιλίππου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Ιούνιος 2023

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια μου Δρ. Άννα Φιλίππου για την καθοδήγηση και υποστήριξη την οποία μου παρείχε κατά την εκπόνηση της Διπλωματικής Εργασίας.

Επιπρόσθετα θα ήθελα να ευχαριστήσω την οικογένεια μου για την στήριξη που μού παρείχε καθ΄ όλη την διάρκεια των σπουδών μου.

Περίληψη

Η ατομική διπλωματική εργασία έχει να κάνει κατά βάση με τα αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ίδιου τύπου (Multi-Reversing Petri Nets - MRPN), που αποτελούν ένα μαθηματικό μοντέλο το οποίο επεκτείνει τα Αναστρέψιμα Δίκτυα Πέτρι (Reversing Petri Nets - RPN), με την προσθήκη πολλαπλών μάρκων (tokens) ίδιου τύπου. Τα μοντέλα αυτά έχουν την δυνατότητα υποστήριξης του Αναστρέψιμου Υπολογισμού.

Στην εργασία, υπήρξε η ανάγκη της δημιουργίας διασύνδεσης μεταξύ δύο εργαλείων, προερχόμενα από δύο υφιστάμενες διπλωματικές εργασίες [15] [16]. Το πρώτο εργαλείο [15] είναι μια desktop εφαρμογή, υλοποιημένη στην Java, η οποία προσφέρει μια γραφική διεπιφάνεια μέσω της οποίας ο χρήστης έχει την δυνατότητα να δημιουργεί, να τροποποιεί και να αποθηκεύει, σε μορφή XML, Αναστρέψιμα Δίκτυα Petri με πολλαπλές μάρκες του ίδιου τύπου. Επίσης προσφέρεται η δυνατότητα προσομοίωσης της εμπρόσθιας και της αντίστροφης εκτέλεσης (forward and backward execution). Ακόμη, ο χρήστης μπορεί να εκτελέσει οποιαδήποτε εφικτή μετάβαση εντός του δικτύου ενώ μπορεί επίσης, να διεξάγει δοκιμές προσβασιμότητας προσδιορίζοντας την δυνατότητα επίτευξης μια συγκεκριμένης μελλοντικής κατάστασης. Η υλοποίηση αυτή βασίστηκε σε πρώτιστο βαθμό σε προηγούμενη διατριβή η οποία επικεντρώθηκε στην ανάπτυξη μιας αντίστοιχης εφαρμογής για Αναστρέψιμα Δίκτυα Petri τα οποία στερούνται εγγενώς την πρόβλεψη για μάρκες πανομοιότυπων τύπων, επεκτείνοντας την, προκειμένου να υπάρχει πρόνοια για επεξεργασία πολλαπλών μαρκών ίδιου τύπου.

Το δεύτερο εργαλείο, επικεντρώνεται στην κωδικοποίηση των αναστρέψιμων δικτύων Petri με πολλαπλά σημεία (MRPNs) στη γλώσσα Answer-Set Programming (ASP). Η γλώσσα αυτή στοχεύει στην επίλυση NP-δύσκολων προβλημάτων και έτσι τα MRPNs είναι η βάση για την περιγραφή προβλημάτων με τέτοια σχετική δυσκολία. Η υλοποίηση αυτής της προσέγγισης συνδυάζεται με την ανάλογη βελτιστοποίηση της απόδοσης και της υπολογιστικής μνήμης. Οι λειτουργίες οι οποίες προσφέρει μεταξύ άλλων είναι η εμπρόσθια εκτέλεση (Forward Execution), η Αντίστροφη Εκτέλεση (Reversible Execution) αλλά και ο έλεγχος προσβασιμότητας (Reachability) του δικτύου σε μια μελλοντική κατάσταση [5].

Περιεχόμενα

Κεφάλαιο 1 - Εισαγωγή

1.1 Εισαγωγή

Το μαθηματικό μοντέλο Αναστρέψιμων Δικτύων Πέτρι (Reversing Petri Net) [1] αποτελεί επέκταση του μαθηματικού μοντέλου Πέτρι (Petri Nets) [5] το οποίο εφευρέθηκε από τον Carl Adam Petri το 1962. Οι κύριες χρήσεις των εν λόγω μοντέλων αφορούν την περιγραφή, τον σχεδιασμό και τη μελέτη διακριτών δυναμικών συστημάτων με βάση τα γεγονότα, τα οποία χαρακτηρίζονται ως ταυτόχρονα, ασύγχρονα, κατανεμημένα, παράλληλα, τυχαία ή/και μη ντετερμινιστικά. Η κύρια διαφορά των Αναστρέψιμων Δικτύων Πέτρι (Reversing Petri Net) [1] με των απλών δικτύων Πέτρι είναι η εισαγωγή της έννοιας της αναστρεψιμότητας η οποία πηγάζει από το χαρακτηριστικό της οπισθοδρόμησης (backtracking) αλλά και την χρήση ή μη της αιτιολογικής σειράς (casual order reversibility), κάτι που μας παρέχει την δυνατότητα επαναφοράς σε μια κατάσταση στην οποία υπήρξε το δίκτυο ή στην μετάβαση σε μια καινούργια κατάσταση. Η έρευνα σε θέματα του αναστρέψιμου υπολογισμού είναι ευρέως διαδεδομένη τα τελευταία χρόνια με εφαρμογές σε ποικίλα συστήματα. Η εμπρόσθια εκτέλεση (forward execution) και η αντίστροφη εκτέλεση (backward execution) αποτελούν τα δύο είδη υπολογισμών. Η έννοια της αντίστροφης εκτέλεσης (backward execution) έχει να κάνει κυρίως με τις εφαρμογές σε βιολογικά συστήματα αλλά και σε αναστρέψιμες κβαντικές λειτουργίες. Τα αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ίδιου τύπου (Multi-Reversing Petri Nets - MRPN) [4] αποτελούν επέκταση των αναστρέψιμων δικτύων Πέτρι με την προσθήκη ισοδύναμων πολλαπλών μάρκων ίδιου τύπου με την προσθήκη της συνθήκης ότι οι μάρκες ίδιου τύπου

αντιμετωπίζονται ως ίδιες και έτσι επιτρέπεται η αντιστροφή μιας μετάβασης με την χρήση οποιασδήποτε μάρκας.

1.2 Συμβολή

Ο απώτερος σκοπός της διπλωματικής εργασίας είναι η δημιουργία διασύνδεσης μεταξύ δύο εργαλείων, προερχόμενα από δύο υφιστάμενες διπλωματικές εργασίες [15] [16] προκειμένου να παρέχεται η είσοδος στο εργαλείο της Clingo [6] μέσω τις γραφικής διεπιφάνειας και να επιστρέφεται το αποτέλεσμα μέσω αυτής. Συγκεκριμένα, το εργαλείο το οποίο αναπτύχθηκε στα πλαίσια της παρούσα διατριβής, στην γλώσσα Java [10], παρέχει στον χρήστη την ευκαιρία να ελέγξει την προσβασιμότητα ενός δικτύου MRPN σε μια μελλοντική κατάσταση με την χρήση της γλώσσας ASP. Η διεπιφάνεια η οποία βοηθά τον χρήστη στις δράσεις αυτές είναι μια επέκταση της διεπιφάνειας της διατριβής [15], η οποία δίνει την δυνατότητα στον χειριστή του εργαλείου να σχεδιάσει, να επεξεργαστεί και να αποθηκεύσει ένα δίκτυο MRPN. Η χρησιμότητα αυτής της επέκτασης πηγάζει από το γεγονός ότι το εργαλείο της διατριβής [16] δεν διαθέτει γραφική διεπιφάνεια κάτι που δυσχεραίνει σε μεγάλο βαθμό τη χρήση του και προϋποθέτει ότι ο χρήστης διαθέτει γνώσεις προγραμματισμού στην γλώσσα ASP. Δημιουργήθηκε λοιπόν η ανάγκη για σύνδεση αυτών των δύο εργαλείων καθώς στη διπλωματική αυτή ο χρήστης είναι σε θέση να σχεδιάσει ένα δίκτυο MRPN και να ελέγξει την προσβασιμότητά του σε μια πιθανή μελλοντική κατάσταση με τη χρήση της γλώσσας ASP, η οποία ειδικεύεται στην επίλυση NP-δύσκολων προβλημάτων. Εφόσον τα δίκτυα MRPN μπορούν να χρησιμοποιηθούν ως βάση για τέτοιου είδους προβλήματα, ο χειριστής της διεπιφάνειας θα μπορεί να τα μοντελοποιήσει γραφικά και να προχωρήσει στην μελέτη τους με πιο εύκολο τρόπο. Για να επιτευχθεί η σύνδεση των δύο εργαλείων έπρεπε να ακολουθηθεί η διαδικασία με τον εξής τρόπο:

α) Σχεδιασμός και μετάφραση του δικτύου MRPN από XML μορφή στην γλώσσα ASP.

β) Μετάφραση επιθυμητής κατάστασης όπως λαμβάνεται από την διεπιφάνεια σε ASP.

γ) Αποθήκευση των δύο αρχείων, σε .lp για να είναι σε θέση να εκτελεστούν σε Clingo [6].

δ) Εκτέλεση σε Clingo των δύο αρχείων μαζί το αρχείο των κωδικοποιημένων κανόνων των MRPNs σε ASP [16].

ε) Επιστροφή αποτελεσμάτων της Clingo στο πρόγραμμα της Java.

στ) Αναγραφή στην διεπιφάνεια αν η επιθυμητή κατάσταση είναι μελλοντικά προσβάσιμη με βάση το αποτέλεσμα της Clingo.

ζ) Εφόσον η επιθυμητή κατάσταση είναι μελλοντικά προσβάσιμη, αναγραφή του μονοπατιού που ακολουθείται για να φτάσει το δίκτυο εκεί στην διεπιφάνεια, με βάση το αποτέλεσμα της Clingo.

η) Εφόσον η επιθυμητή κατάσταση είναι μελλοντικά προσβάσιμη παρέχεται η δυνατότητα στον χρήστη να δει γραφικά πώς τροποποιείται το δίκτυο ούτως ώστε να φτάσει στην επιθυμητή κατάσταση.

1.3 Μεθοδολογία

Σε πρώτο στάδιο χρειάστηκε η μελέτη όλων των εκδοχών των δικτύων Πέτρι. Ήταν σημαντικό, εφόσον το πεδίο ήταν ξένο, να κατανοηθεί πλήρως η σημασιολογία και οι ορισμοί των δικτύων ούτως ώστε να είμαστε σε θέση να αναγνωρίζουμε τα βασικά συστατικά ενός δικτύου. Στη συνέχεια προχωρήσαμε στην μελέτη της υλοποίησης της πρώτης διατριβής [15], καθώς αυτή έμελλε να επεκταθεί, δηλαδή του Desktop Application. Υπήρξε πλήρης κατανόηση στον τρόπο αποθήκευσης των δικτύων MRPN σε μορφή XML. Ακολούθως έλαβε χώρα η μελέτη και κατανόηση του κώδικα της δεύτερης διατριβής [16] και πιο συγκεκριμένα του τρόπου κωδικοποίησης ενός δικτύου MRPN. Το επόμενο βήμα ήταν η προσπάθεια χειροκίνητης μετάφρασης ενός δικτύου από την μορφή XML στην κωδικοποιημένη μορφή στην γλώσσα ASP προκειμένου να τρέξει σε Clingo [5] το οποίο έχει την ικανότητα να διαπιστώσει αν υπάρχει κάποιο συντακτικό λάθος. Μετά από την διακρίβωση για το πώς κωδικοποιείται ένα δίκτυο MRPN από μορφή XML στην ASP, ξεκίνησαν οι υλοποιήσεις των συναρτήσεων οι οποίες θα λάμβαναν ως είσοδο ένα αρχείο XML και με την απαραίτητη διαδικασία θα το αντικατόπτριζαν σε μορφή κώδικα ASP. Έπειτα, έπρεπε να τροποποιηθεί η καρτέλα της διεπιφάνειας η οποία ελέγχει αν ένα δίκτυο είναι μελλοντικά προσβάσιμο σε μια κατάσταση. Έτσι προστέθηκε ακόμη ένα κουμπί με ίδια χαρακτηριστικά του υφιστάμενου, το οποίο θα διεκπεραιώνει την διαδικασία του ελέγχου προσβασιμότητας μέσω κωδικοποίησης σε γλώσσα ASP. Το επόμενο βήμα ήταν η σύνδεση της γραφικής απεικόνισης του κουμπιού με κώδικα για να αναδύεται ένα παράθυρο στο οποίο θα

δηλώνεται η επιθυμητή μελλοντική κατάσταση. Υπήρξε επίσης μέριμνα για σύνδεση του εκάστοτε κουμπιού από το αναδυόμενο παράθυρο το οποίο θα αντιστοιχεί στην λειτουργία που του αναλογεί. Σειρά είχε η υλοποίηση της αποθήκευσης των δύο κωδικοποιημένων αρχείων, σε αρχεία με κατάληξη .lp για να είναι σε θέση να εκτελεστούν σε Clingo [6]. Με την αντίστοιχη επιλογή (RUN), συνδέθηκε η εκτέλεση σε Clingo των δύο αρχείων μαζί το αρχείο των κωδικοποιημένων κανόνων των MRPNs σε ASP [16] και η λήψη των αποτελεσμάτων από την εκτέλεση αυτή. Με την αντίστοιχη επιλογή (APPLY), συνδέθηκε η γραφική απεικόνιση των αποτελεσμάτων από την εκτέλεση με χρονική σειρά.

1.4 Δομή Διατριβής

Το δεύτερο κεφάλαιο εμβαθύνει στις θεμελιώδεις θεωρίες και τεχνικές έννοιες που αποτελούν τη βάση της διατριβής.

Το τρίτο κεφάλαιο διερευνά τα υφιστάμενα συστήματα και μελέτες που σχετίζονται με το θέμα της έρευνας, παρέχοντας πληροφορίες σχετικά με τις υπάρχουσες προσεγγίσεις, ενώ καταγράφονται οι λειτουργικές και μη λειτουργικές απαιτήσεις τις οποίες καλούνται να ικανοποιηθούν μέσω της υλοποίησης.

Το τέταρτο κεφάλαιο, πραγματεύεται την σχεδιαστική εκτέλεση του συστήματος, επικεντρώνεται επίσης στην πρακτική υλοποίηση της διεπιφάνειας, παρουσιάζοντας τους αλγορίθμους μέσω των οποίων το εργαλείο αναμένεται να εκπληρώσει τις διάφορες λειτουργίες.

Το πέμπτο κεφάλαιο, αξιολογεί κριτικά το υλοποιημένο εργαλείο, προσμετρώντας την απόδοση σε σχέση με προκαθορισμένα κριτήρια. Η αξιολόγηση αυτή χρησιμεύει για την επικύρωση της λειτουργικότητας του συστήματος και τον εντοπισμό τομέων για περαιτέρω βελτίωση.

Τέλος, στο έκτο κεφάλαιο, συνοψίζονται τα βασικά ευρήματα και περιγράφονται πιθανές κατευθύνσεις για μελλοντική έρευνα και βελτιώσεις του συστήματος.

Κεφάλαιο 2 - Θεωρητικό και Τεχνικό Υπόβαθρο

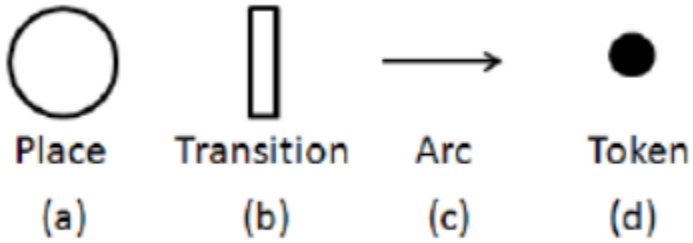
2.1 Θεωρητικό Υπόβαθρο

2.1.1 Δίκτυα Πέτρι

Τα δίκτυα Πέτρι [5] αποτελούν επινόηση του Γερμανού Μαθηματικού Carl Adam Petri και είναι ένα μαθηματικό εργαλείο το οποίο αποσκοπεί στην μοντελοποίηση, την ανάλυση, τη σύνθεση, την αξιολόγηση αποδοτικότητας και τον εποπτικό έλεγχο διακριτών δυναμικών συστημάτων με βάση τα γεγονότα, τα οποία χαρακτηρίζονται ως ταυτόχρονα, ασύγχρονα, κατανεμημένα, παράλληλα, τυχαία ή/και μη ντετερμινιστικά.

2.1.1.1 Βασικά Στοιχεία

Τα βασικά στοιχεία τα οποία απαρτίζουν ένα δίκτυο Πέτρι είναι το στοιχείο της θέσης (place), το στοιχείο της μετάβασης (transition), το στοιχείο του τόξου (arc) και της μάρκας (token).



Εικόνα 2.1: Βασικά στοιχεία δικτύου Πέτρι.[8]

α) Θέση (Place): Το χαρακτηριστικό σύμβολο για περιγραφή μιας τοποθεσίας ή/και μιας κατάστασης. Συμβολίζεται με κύκλο και έχει την δυνατότητα να συνδεθεί με μετάβαση μέσω της χρησιμοποίησης ενός τόξου (arc).

β) Μετάβαση (Transition): Περιγράφει την μετάβαση από μία θέση (place) σε μια άλλη και συμβολίζεται ως ορθογώνιο.

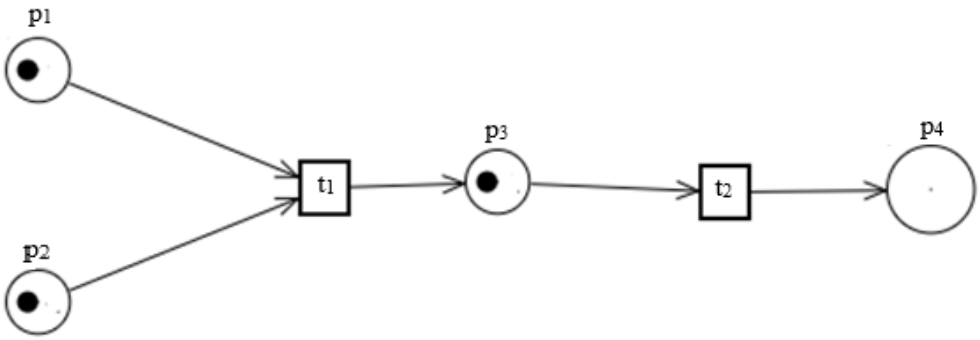
γ) Τόξο (arc): Συνδέει δύο θέσεις μεταξύ τους με φορά καθορισμένη από τον χρήστη. Το εισερχόμενο τόξο (arc) αναπαριστά τις προϋποθέσεις και το εξερχόμενο το αποτέλεσμα της μετάβασης μετά το πέρας της διεργασίας. Συμβολίζεται με ένα κατευθυνόμενο τόξο το οποίο δείχνει την κατεύθυνση.

δ) Μάρκα (Token): Αντιπροσωπεύει τις οντότητες οι οποίες μεταφέρονται και τροποποιούνται στο δίκτυο.

2.1.1.2 Μαθηματικός ορισμός

[5] Ένα δίκτυο Πέτρι ορίζεται ως N = (P, T, F, W) και μια αρχική σήμανση (initial marking) M_0, όπου:

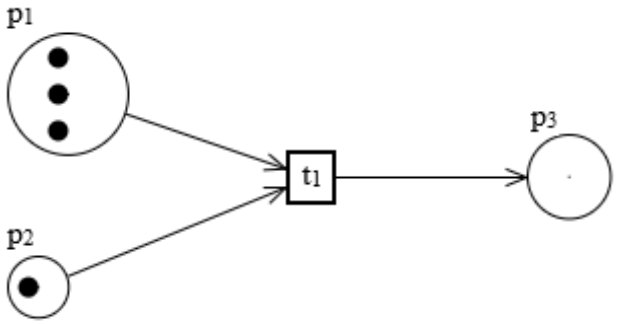
1. P είναι ένα πεπερασμένο σύνολο θέσεων.
2. T είναι ένα πεπερασμένο σύνολο μεταβάσεων, $P \cup T \neq \emptyset$, και $P \cap T = \emptyset$
3. F: είναι το σύνολο των κατευθυνόμενων τόξων από θέσεις σε μεταβάσεις και από μεταβάσεις σε θέσεις, $F \subseteq (P \times T) \cup (T \times P)$.
4. W: $F \rightarrow \mathbb{N}^+$: συνάρτηση που αποδίδει βάρη στα τόξα του δικτύου.
5. M_0: είναι η αρχική ανάθεση μαρκών σε κάθε θέση.



Εικόνα 2.2: Παράδειγμα δικτύου Πέτρι.

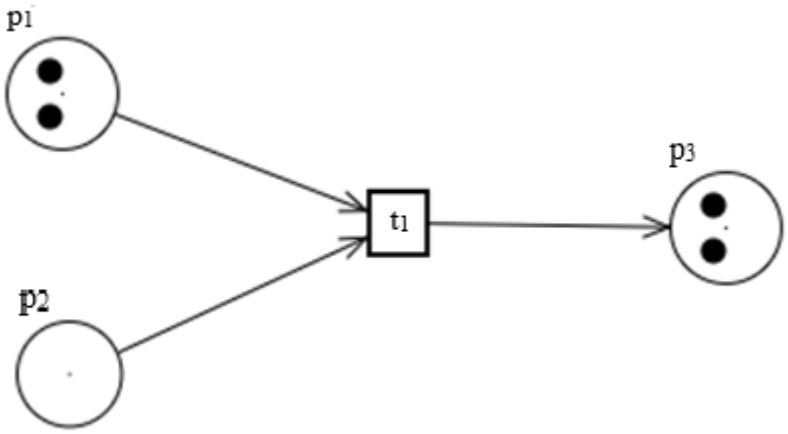
Το πιο πάνω παράδειγμα δικτύου Πέτρι έχει 4 θέσεις (places) και 2 μεταβάσεις (transitions). Στην αρχική σήμανση (initial marking) M0 οι θεσεις p_1, p_2, p_3 έχουν από ένα token. Για την μετάβαση t_1 υπάρχουν δύο εισερχόμενα τόξα, από p_1 και p_2 αντίστοιχα, ενώ υπάρχει ένα εξερχόμενο τόξο το οποίο κατευθύνεται προς το p_3. Για την μετάβαση t_2 υπάρχει εισερχόμενο τόξο από την θέση p_3 και εξερχόμενο τόξο προς την θέση p_4. Το βάρος των μεταβάσεων χωρίς ετικέτες (labels) ορίζεται εκ προεπιλογής με 1. Ως αρχική σήμανση M_0 (initial marking) έχουμε μια μάρκα στο p_1, μια μάρκα στο p_2 και μια μάρκα στο p_3.

Προκειμένου να είναι εφικτή μια πυροδότηση μετάβασης πρέπει όλες οι θέσεις οι οποίες κατευθύνουν τόξα σε μια μετάβαση να έχουν τουλάχιστον πλήθος μαρκών (token), ίσο με το βάρος του εκάστοτε τόξου. Για περαιτέρω κατανόηση παρατηρούμε την Εικόνα 2.3 και βλέπουμε μια αρχική ανάθεση (initial marking). Στο παρακάτω παράδειγμα δικτύου Πέτρι υπάρχουν 3 θέσεις (places) και 1 μετάβαση (transition). Στην αρχική σήμανση (initial marking) M_0 η θέση p_1 έχει 3 tokens και στην θέση p_2 έχει 1 token. Στην θέση p_3 δεν υπάρχει κάποιο token. Για την μετάβαση t_1 υπάρχουν δύο εισερχόμενα τόξα, από p_1 και p_2 αντίστοιχα, ενώ υπάρχει ένα εξερχόμενο τόξο το οποίο κατευθύνεται προς το p_3. Το βάρος των μεταβάσεων χωρίς ετικέτες (labels) ορίζεται εκ προεπιλογής με 1.



Εικόνα 2.3: Παράδειγμα δικτύου Πέτρι.

Βλέπουμε στην Εικόνα 2.3 ότι ικανοποιούνται οι προϋποθέσεις για την πυροδότηση (fire) της μετάβασης t_1. Έτσι μετά από αυτή την εξέλιξη έχουμε το αποτέλεσμα που βλέπουμε παρακάτω στην Εικόνα 2.4.



Εικόνα 2.4: Εικόνα μετά την πυροδότηση της μετάβασης στο δίκτυο Πέτρι της Εικόνας 3.

Μετά από την πυροδότηση της μετάβασης t_1 το δίκτυο έχει την μορφή που περιγράφεται στην Εικόνα 2.4. Δεν υπάρχει άλλη διαθέσιμη πυροδότηση μετάβασης καθώς παρά το γεγονός ότι η θέση p_1 έχει 2 μάρκες, η έτερη θέση η οποία έχει κατευθυνόμενο τόξο στην μετάβαση t_1 δεν περιέχει κανένα token.

2.1.1 Αναστρέψιμος Υπολογισμός

Τα τελευταία χρόνια, ο Αναστρέψιμος Υπολογισμός αποτελεί ένα ενεργό ερευνητικό πεδίο και εφαρμόζεται σε διάφορα συστήματα. Η αντιστρεψιμότητα είναι μια ιδιαίτερα επιθυμητή ιδιότητα στα συστήματα λόγω των πλεονεκτημάτων της όσον αφορά την

υπολογιστική ισχύ και την ευελιξία σε σφάλματα. Ο Αναστρέψιμος Υπολογισμός αναφέρεται σε έναν τύπο υπολογισμού ο οποίος μπορεί να εκτελεστεί τόσο προς τα εμπρός (forward), όσο και προς τα πίσω. Αυτό σημαίνει ότι οποιαδήποτε ακολουθία εκτελέσεων του συστήματος μπορεί να εκτελεστεί με αναστρέψιμο τρόπο, επιτρέποντας στο δίκτυο να επαναφέρει οποιαδήποτε προηγούμενη κατάσταση σε οποιοδήποτε σημείο κατά τη διάρκεια του υπολογισμού. Για να επιτευχθεί ο αναστρέψιμος υπολογισμός, είναι απαραίτητο να δημιουργηθεί μια συσχέτιση 1 - 1 μεταξύ εισόδου και εξόδου, εξασφαλίζοντας ότι μια απλή είσοδος μπορεί να προβλεφθεί με ακρίβεια από την αντίστοιχη έξοδό της.

2.1.1.1 Μορφές Αναστρεψιμότητας

Η οπισθοδρόμηση (backtracking), η αιτιώδης αναστρεψιμότητα (causal reversibility) και η αναστρεψιμότητα μη αιτιολογικής σειράς (out-of-causal-order reversibility) αποτελούν τις βασικές μορφές αναστρεψιμότητας [4] και θα αναλυθούν στα επόμενα υποκεφάλαια.

Οπισθοδρόμηση

Η οπισθοδρόμηση (backtracking) εκτελεί τις ενέργειες αυστηρώς αντίστροφα από την σειρά με την οποία έχουν εκτελεστεί. Έτσι εξασφαλίζεται ότι κάθε κατάσταση έχει μια προηγούμενη κατάσταση δικτύου.

Αιτιώδης Αναστρεψιμότητα

Η Αιτιώδης Αναστρεψιμότητα (Causal Reversibility) επιτρέπει σε μια ενέργεια να επαναφερθεί αν και εφόσον όλες οι εξαρτώμενες ενέργειες έχουν ήδη προβεί σε αναστροφή. Η διαφοροποίηση με την έννοια της οπισθοδρόμησης είναι ότι οι ενέργειες δεν αντιστρέφονται αυστηρώς με την αντίθετη σειρά με την οποία εκτελεστήκαν.

Αναστρεψιμότητα μη αιτιολογικής σειράς

Η αναστρεψιμότητα μη αιτιολογικής σειράς (out-of-causal-order reversibility) είναι ικανή να προβεί στην αντιστροφή μιας κατάστασης ανεξαρτήτως αν τα αποτελέσματα αυτής της κατάστασης έχουν αναιρεθεί. Με αυτή τη διαδικασία δημιουργούνται καταστάσεις οι οποίες δεν επιτυγχάνονται μέσω οποιασδήποτε ακολουθίας εμπρόσθιας εκτέλεσης.

2.1.2 Αναστρέψιμα Δίκτυα Πέτρι

Τα Αναστρέψιμα Δίκτυα Πέτρι [1], αποτελούν επέκταση των Δικτύων Πέτρι. Η διαφοροποίηση οφείλεται στην εισαγωγή της έννοιας της αναστρεψιμότητας, η οποία προσδίδει την δυνατότητα του αναστρέψιμου υπολογισμού (reversible computation) ο οποίος αποσκοπεί στην αναίρεση των συνεπειών μιας προηγούμενης μετάβασης η οποία εκτελέστηκε.

2.1.3 Αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ίδιου τύπου

Τα Αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ίδιου τύπου [4], είναι μια επιπλέον επέκταση του ορισμού Αναστρέψιμων Δικτύων Πέτρι. Στην υφιστάμενη εκδοχή των δικτύων, προστίθεται το χαρακτηριστικό ότι οι μάρκες έχουν ένα τύπο και υπάρχει η δυνατότητα πολλές μάρκες έχουν τον ίδιο τύπο.

2.1.3.1 Έννοια και Βασικά Στοιχεία

Στα Αναστρέψιμα Δίκτυα Πέτρι με πολλαπλές μάρκες ίδιου τύπου, τα βασικά στοιχεία συμβολίζονται με τον ίδιο τρόπο όπως και στα Αναστρέψιμα Δίκτυα Πέτρι. Κάθε μάρκα αντιστοιχεί σε ένα τύπο και ένας τύπος μπορεί να αντιστοιχεί, να κατέχεται δηλαδή, από πολλές μάρκες. Με βάση αυτή την προσέγγιση, η πυροδότηση μιας μετάβασης προσφέρει την δυνατότητα αντιστροφής της μετάβασης από οποιοδήποτε σύνολο μαρκών πληροί τις προϋποθέσεις χωρίς να σημαίνει ότι είναι το ίδιο σύνολο μαρκών ή/και δεσμών το οποίο πυροδότησε την μετάβαση προσθέτοντας ευκολία στην αναστρεψιμότητα μη αιτιολογικής σειράς.

2.1.3.2 Ορισμός

Σύμφωνα με το άρθρο της Άννας Φιλίππου και Κυριακής Ψαρά [4], το Αναστρέψιμο Δίκτυο Πέτρι με πολλαπλές μάρκες ίδιου τύπου ορίζεται ως πλειάδα (P, T, A, A_V, B, F) όπου:

1. P: Πεπερασμένο σύνολο θέσεων (places).
2. T: Πεπερασμένο σύνολο μεταβάσεων (transitions).
3. A: Πεπερασμένο σύνολο τύπων βάσεων ή μάρκων (tokens).
4. A_v: Πεπερασμένο σύνολο μεταβλητών. Γράφουμε type(v) για τον τύπο της μεταβλητής v και υποθέτουμε ότι type(v) ∈ A για κάθε v ∈ A_V.

5. $B \subseteq A \times A$: Πεπερασμένο σύνολο μη κατευθυνόμενων δεσμών (bonds). Αν υφίσταται δεσμός ανάμεσα στις μάρκες a και b, τότε συμβολίζεται ως a-b ή $(a, b) \in B$.
6. $F : (P \times T \cup T \times P) \rightarrow P(A_V \cup (A_V \times A_V))$: Σύνολο με κατευθυνόμενα τόξα με ετικέτα και το κάθε ένα ενώνεται με υποσύνολο από $A_V \cup (A_V \times A_V)$), όπου $(u, v) \in F(x, y)$ υποδηλώνει ότι $u, v \in F(x, y)$ και $\forall t \in T, F(t, x) \cap F(t, y) = \emptyset$.

Οι μεταβάσεις με τις θέσεις και οι θέσεις με τις μεταβάσεις ενώνονται με κατευθυνόμενα τόξα. Δεν είναι δυνατό να ενωθούν δύο θέσεις χωρίς ενδιάμεση μετάβαση, ούτε δύο μεταβάσεις χωρίς ενδιάμεση θέση. Οι προϋποθέσεις – συνθήκες για την πυροδότηση μιας μετάβασης αναγράφονται στην ετικέτα του τόξου το οποίο κατευθύνεται στην μετάβαση και το αποτέλεσμα μετά την πυροδότηση μιας μετάβασης αναγράφεται στο εξερχόμενο τόξο. Όταν ο δεσμός αναγράφεται στο τόξο, οι βάσεις οι οποίες αποτελούν τον δεσμό περιέχονται επίσης στο τόξο. Όσο αφορά τις ετικέτες, οι μεταβλητές των μάρκων, έστω $u \in F(x, t) \cap A_V$ όπου type(u) = a συμβολίζονται με u : a στο τόξο F(x,t). Σε μια μετάβαση t το σύμβολο $\bullet t = \{x \in P \mid F(x, t) \neq \emptyset\}$ αντιπροσωπεύει τις εισερχόμενες θέσεις και το σύμβολο και $t\bullet = \{x \in P \mid F(t, x) \neq \emptyset\}$ τις εξερχόμενες θέσεις.

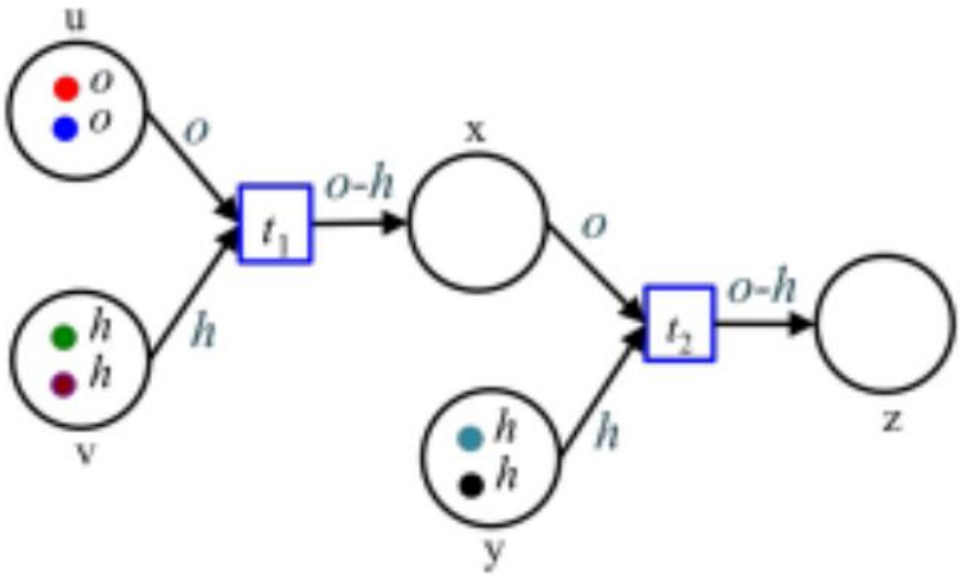
$$pre(t) = \bigcup_{x \in P} F(x,t) \quad και \quad post(t) = \bigcup_{x \in P} F(t,x)$$

Για μια μετάβαση t, οι παραπάνω έννοιες pre και post εκφράζουν την ένωση των ετικετών οι οποίες βρίσκονται στα εισερχόμενα τόξα και εξερχόμενα τόξα αντίστοιχα. Μαρκάρισμα αποτελεί η ανάθεση των δεσμών και των βάσεων στις διάφορες θέσεις.

Για να καταστεί δυνατός ο ορισμός της σήμανσης, απαιτούνται δύο πρόσθετοι μηχανισμοί. Οι εμφανίσεις συμβόλων του ίδιου τύπου διακρίνονται χρησιμοποιώντας το a για να αντιπροσωπεύσει την $i^{οστή}$ εμφάνιση του συμβόλου τύπου a. Κατά συνέπεια, συμβολίζουμε το A_I ως τη συλλογή όλων των εμφανίσεων των συμβόλων $B_I = A_I \times A_I$ για το σύνολο των δεσμών τους. Με αυτόν τον τρόπο, μια μάρκα (token) ορίζεται ως $M: P \rightarrow 2^{A_I \cup B_I}$. Επιπλέον, κάθε μετάβαση διαθέτει το δικό της ιστορικό, το οποίο συμβολίζεται ως $H: T \rightarrow \mathbb{N}$. Όταν το ιστορικό μιας μετάβασης, H(t), είναι ίσο με 0, υποδηλώνει ότι η μετάβαση δεν έχει εκτελεστεί. Από την άλλη πλευρά, εάν H(t) = k, όπου το k είναι μεγαλύτερο από 0, σημαίνει ότι η μετάβαση έχει εκτελεστεί προς τα

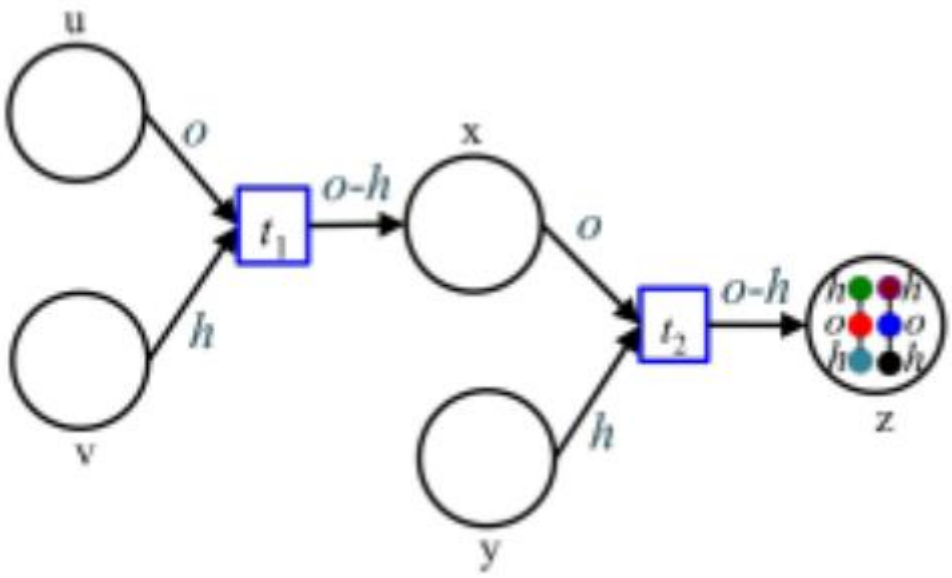
εμπρός k φορές. Η κατάσταση ενός MRPN περιγράφεται με δυαδική αναπαράσταση <M, H>.

2.1.3.3 Παράδειγμα Δικτύου MRPN



Εικόνα 2.5: Αρχικοποίηση δεύτερου παραδείγματος [1]

Παραθέτουμε στην Εικόνα 2.5, το παράδειγμα δικτύου το οποίο αποτελεί ένα Αναστρέψιμο Δίκτυο Πέτρι με πολλαπλές μάρκες ίδιου τύπου, με το παράδειγμα μοντελοποίησης της χημικής ένωσης του νερού. Οι μάρκες τύπου o συμβολίζουν το άτομο του οξυγόνου και οι μάρκες τύπου h αντιστοιχούν στο άτομο του υδρογόνου. Έχουμε δύο μεταβάσεις, την t_1 και την t_2, τις θέσεις u, v, x, y, z, και τις ανάλογες ακμές οι οποίες ενώνουν τις οντότητες των μεταβάσεων με αυτές των θέσεων και αντίστροφα. Η αρχική κατάσταση αντιστοιχεί στην Εικόνα 2.5. Με την πυροδότηση των μεταβάσεων t_1, t_2 (κατά σειρά) η νέα μορφή του δικτύου έχει την παρακάτω μορφή (Εικόνα 2.6). Η διαδικασία με την ακολουθία πυροδοτήσεων θα χρειαστεί να λάβει χώρα δύο φορές.



Εικόνα 2.6: Αποτέλεσμα μετά την πυροδότηση των μεταβάσεων [1]

Το αποτέλεσμα το οποίο έχουμε στην Εικόνα 2.6 είναι μετά την πυροδότηση των μεταβάσεων. Βλέπουμε πως εφόσον πληρούνται οι προυποθέσεις των εισερχόμενων ακμών σε μια μετάβαση, η παράβαση πυροδοτείται συνθέτοντας εν τέλει 2 δεσμούς από δύο άτομα υδρογόνου και ένα άτομο οξυγόνου ο καθένας. Προκειμένου να σπάσουμε τον δεσμό θα αντιστρέψουμε (reverse step) την μετάβαση t_2 και ο δεσμός o-h θα διασπαστεί κάτι που θα φέρει τον δεσμό o-h στην θέση x και την μάρκα h στην θέση y.

2.1.3.4 Προσβασιμότητα (reachability)

Η έννοια της προσβασιμότητας στην ανάλυση του δικτύου περιστρέφεται γύρω από τη διερεύνηση της ικανότητας του δικτύου να φτάσει τελικά σε έναν συγκεκριμένο μαρκάρισμα στο μέλλον, διασχίζοντας επιτυχώς μια ακολουθία μεταβάσεων. Ο συμβολισμός ↦, απεικονίζει το συνδυασμό εμπρόσθιας και αντίστροφης εκτέλεσης (→ ∪ ⇝).

Ας θεωρήσουμε ένα MRPN (A, P, B, T, F) μαζί με την αρχική του κατάσταση που συμβολίζεται ως $<M_0, H_0>$. Σε αυτό το πλαίσιο, ορίζουμε μια κατάσταση <M, H> ως προσβάσιμη, όπου το M αντιπροσωπεύει το μαρκάρισμα και το H το ιστορικό, εάν, ξεκινώντας από την αρχική κατάσταση $<M_0, H_0>$, είναι εφικτό να προχωρήσουμε μέσω μιας σειράς μεταβάσεων και τελικά να φτάσουμε στην κατάσταση πρόσβασης <M, H>

αν υπάρχουν καταστάσεις <M_i, H_i>, i ≤ n για κάποιο n ≥ 0 τέτοιες ώστε <M_0, H_0> $\overset{t1}{\mapsto}$ <M_1, H_1> $\overset{t2}{\mapsto}$ . . . $\overset{tn}{\mapsto}$ <M_n, H_n> = <M, H>.

2.2 Τεχνικό Υπόβαθρο

2.2.1 Java

Η Java αποτελεί μια υψηλού επιπέδου αντικειμενοστρεφή γλώσσα προγραμματισμού, παρουσιάστηκε αρχικά από τη Sun Microsystems το 1995. Με την πάροδο των ετών, διατήρησε και αύξησε τη δημοτικότητά της όντας από τις πλέον χρησιμοποιούμενες γλώσσες προγραμματισμού. Χρησιμεύει ως ένα αξιόπιστο εργαλείο για τη δημιουργία διαφόρων υπηρεσιών και εφαρμογών [9].

Ένα από τα βασικά πλεονεκτήματα της Java είναι η ανεξαρτησία της πλατφόρμας. Με τη χρήση του API της Java και του JVM, οι προγραμματιστές μπορούν να γράψουν κώδικα που μπορεί να εκτελεστεί σε πολλές πλατφόρμες χωρίς την ανάγκη σημαντικών τροποποιήσεων. Αυτή η δυνατότητα "write once, run anywhere" έχει συμβάλει σημαντικά στην ευρεία υιοθέτηση και εξάπλωση της Java [10]. Το Java API λειτουργεί ως μια βιβλιοθήκη εντολών Java, ενώ η JVM διερμηνεύει τον κώδικα Java σε γλώσσα μηχανής. Αυτή η αρχιτεκτονική επιτρέπει τη συμβατότητα μεταξύ διαφορετικών πλατφορμών, επιτρέποντας την εκτέλεση εφαρμογών σε πολλά συστήματα. Το API και το JVM συνεργάζονται για να διασφαλίσουν ότι τα προγράμματα παραμένουν ανεξάρτητα από το υποκείμενο υλικό [10].

2.2.2 Eclipse IDE

Το Eclipse IDE (Integrated Development Environment) είναι μια ευρέως χρησιμοποιούμενη πλατφόρμα ανάπτυξης λογισμικού open-source κώδικα. Προσφέρει μια σειρά εργαλείων και χαρακτηριστικών για τη συγγραφή, τον έλεγχο και την αποσφαλμάτωση κώδικα. Με υποστήριξη για πολλές γλώσσες προγραμματισμού, συμπεριλαμβανομένων των Java, C/C++ και Python, το Eclipse παρέχει ένα φιλικό προς τον χρήστη περιβάλλον εργασίας, ολοκληρωμένα εργαλεία αποσφαλμάτωσης και ενσωμάτωση συστήματος ελέγχου εκδόσεων. Η επεκτασιμότητά του IDE, μέσω plugins,

επιτρέπει στους προγραμματιστές να προσαρμόζουν το περιβάλλον τους ανάλογα με τις προτιμήσεις τους [11].

2.2.3 JavaFX

Το JavaFX είναι μια συλλογή από πακέτα γραφικών και πολυμέσων που δίνει τη δυνατότητα στους προγραμματιστές να σχεδιάζουν, να δημιουργούν, να δοκιμάζουν, να αποσφαλματώνουν και να τροποποιούν το δικό τους λογισμικό. Επιτρέπει τη δημιουργία εφαρμογών που λειτουργούν με συνέπεια σε διάφορες πλατφόρμες, ενώ επιτρέπει επίσης, το διαχωρισμό της λογικής του front-end (UI) και του back-end. Η ανάπτυξη του front-end μπορεί να πραγματοποιηθεί εύκολα με τη χρήση της γλώσσας σεναρίων FXML, ενώ η λογική του back-end υλοποιείται με τη χρήση κώδικα Java. Επιπλέον, υποστηρίζει τη χρήση Cascading Style Sheets (CSS) για την προσαρμογή της εμφάνισης και του στυλ της εφαρμογής [13]. Για να δημιουργηθεί ένα έργο JavaFX στο Eclipse, πρέπει να εγκατασταθεί το πρόσθετο plugin e(fx)clipse.

2.2.4 Answer Set Programming (ASP)

Η γλώσσα Answer-Set Programming αποτελεί μια γλώσσα δηλωτικού προγραμματισμού που αποσκοπεί στην επίλύση δύσκολων και NP-δύσκολων προβλημάτων, δηλαδή προβλημάτων τα οποία αποτελούν τα προβλήματα τα οποία θεωρούνται δύσκολο να λυθούν αποδοτικά. Βασίζεται στο σταθερό μοντέλο [17] σημασιολογίας του λογικού προγραμματισμού το οποίο εφαρμόζει ιδέες από την αυτοεπιστημική λογική [18] και την προκαθορισμένη λογική [19] στη μελέτη της άρνησης ως αποτυχία.

2.2.5 Clingo

Το Clingo είναι ένα robust σύστημα που χρησιμοποιείται για τη θεμελίωση και την επίλυση λογικών προγραμμάτων από την Potsdam Answer Set Solving Collection (Potassco) [8]. Η κρίσιμη διαδικασία θεμελίωσης περιλαμβάνει τον υπολογισμό ενός ισοδύναμου προγράμματος χωρίς μεταβλητές και το Clingo αξιοποιεί το gringo [6] ως υποδειγματικό πρόγραμμα θεμελίωσης. Για την επέκταση των δυνατοτήτων της εξόδου του gringo, μπορεί να γίνει περαιτέρω επεξεργασία με τη χρήση των clasp, claspfolio ή clingcon. Το clasp, ο καθορισμένος solver που χρησιμοποιείται στο Clingo [7], υπερέχει στην επίλυση (εκτεταμένων) κανονικών και διαζευκτικών λογικών προγραμμάτων ενσωματώνοντας απρόσκοπτα τις πρωτοποριακές τεχνικές επίλυσης περιορισμών

Boolean με τις εξαιρετικές δυνατότητες προσομοίωσης του Προγραμματισμού Συνόλων Απαντήσεων (ASP). Η δυναμική του αλγορίθμου clasp έγκειται στη conflict-driven μάθηση nogood, η οποία έχει αποδειχθεί εκτενώς ότι είναι μια εξαιρετικά ισχυρή τεχνική για τον έλεγχο ικανοποιησιμότητας (SAT).

2.2.6 Python

Η Python αποτελεί μια δυναμικά διερμηνευόμενη γλώσσα προγραμματισμού υψηλού επιπέδου που ακολουθεί μια αντικειμενοστραφή προσέγγιση. Η σύνταξη της γλώσσας είναι απλή και εύκολα κατανοητή, προάγοντας την αναγνωσιμότητα του κώδικα και ελαχιστοποιώντας την προσπάθεια που απαιτείται για τη συντήρηση του λογισμικού. Οι ενσωματωμένες δομές δεδομένων της Python, σε συνδυασμό με τις δυνατότητες δυναμικής τυποποίησης και δέσμευσης, την καθιστούν ιδιαίτερα κατάλληλη για ταχεία ανάπτυξη εφαρμογών. Επιπλέον, η Python χρησιμεύει ως αποτελεσματική γλώσσα σεναρίων ή ως “glue” language, διευκολύνοντας την απρόσκοπτη ενσωμάτωση προϋπαρχόντων στοιχείων. Καταλυτικός είναι ο ρόλος της στην σύνδεση πολλαπλών προγραμμάτων τα οποία είναι γραμμένα σε διαφορετικές γλώσσες.

Κεφάλαιο 3 - Προϋπάρχοντα Συστήματα

Η Διπλωματική Εργασία βασίστηκε σε δύο προηγούμενες διατριβές. Η μια αποτελεί την υλοποίηση του γραφικού εργαλείου επεξεργασίας και προσημείωσης δικτύων Πέτρι με πολλαπλές μάρκες ίδιου τύπου [15], και η άλλη την κωδικοποίηση δικτύων Πέτρι με πολλαπλές μάρκες ίδιου τύπου, σε γλώσσα Answer Set Programming [16].

3.1 Γραφικό Εργαλείο επεξεργασίας και προσημείωσης δικτύων Πέτρι με πολλαπλές μάρκες ίδιου τύπου

Η διπλωματική εργασία του Ραφαήλ Ιακώβου [15] είχε ως πρωταρχικό στόχο την ανάπτυξη μιας εφαρμογής με τη χρήση της γλώσσας προγραμματισμού Java, η οποία δίνει τη δυνατότητα στους χρήστες να δημιουργούν, να τροποποιούν και να αποθηκεύουν αναστρέψιμα δίκτυα Petri που περιέχουν πολλαπλές μάρκες του ίδιου τύπου. Επιπλέον, προσφέρει δυνατότητες προσομοίωσης, επιτρέποντας στους χρήστες να εκτελούν αυτά τα δίκτυα τόσο προς τα εμπρός όσο και προς τα πίσω. Οι χρήστες μπορούν να τροχοδρομήσουν οποιαδήποτε εφικτή μετάβαση εντός του δικτύου και να διεξάγουν δοκιμές προσβασιμότητας, προσδιορίζοντας τη δυνατότητα επίτευξης μιας συγκεκριμένης κατάστασης στο μέλλον.

Βασίστηκε σε πρώτιστο βαθμό σε προηγούμενη διατριβή η οποία επικεντρώθηκε στην ανάπτυξη μιας desktop εφαρμογής για αναστρέψιμα δίκτυα Petri την οποία τροποποίησε,

για να ενσωματώσει τις λειτουργικότητες που αναφέρθηκαν παραπάνω, ειδικά για την επεξεργασία πολλαπλών μαρκών ίδιου τύπου.

3.2 Encoding Multi-Token Reversing Petri Nets in Answer Set Programming for Simulation-Based Reasoning

Η διπλωματική εργασία του Μιχαήλ Άγγελου Πιττάκαρα [16] επικεντρώνεται στην κωδικοποίηση των δικτύων Petri με πολλαπλές μάρκες ίδιου τύπου (MRPN) στη γλώσσα προγραμματισμού ASP. Επιπλέον, δημιουργήθηκαν πρόσθετες επεκτάσεις για έναν αλγόριθμο που σχεδιάστηκε για την αντιμετώπιση του προβλήματος βελτιστοποίησης του Antenna Selection, το οποίο εμπίπτει στην κατηγορία των προβλημάτων πολλαπλών εισόδων και πολλαπλών εξόδων (MIMO) και είναι γνωστό ότι αποτελεί πρόβλημα δυσκολίας NP [3]. Για τον υπολογισμό της σύνθετης εξίσωσης που υπολογίζει την αθροιστική χωρητικότητα ενός δικτύου κεραιών, χρησιμοποιήθηκε η Python σε συνδυασμό με το ASP ενώ εφαρμόστηκαν βελτιστοποιήσεις για την ενίσχυση της απόδοσης του προγράμματος. Επιπρόσθετα, το πρόγραμμα εφαρμόστηκε σε ένα case study το οποίο αφορούσε αλγόριθμο Antenna Selection. Οι λειτουργίες οι οποίες προσφέρει μεταξύ άλλων είναι η εμπρόσθια εκτέλεση (Forward Execution), η Αντίστροφη Εκτέλεση (Reversible) αλλά και η προσβασιμότητα (reachability) του δικτύου σε μια μελλοντική κατάσταση. Δεν υπάρχει κάποιο interface το οποίο να απλοποιεί την διαδικασία για τον χρήστη αλλά εντούτοις επεξεργάζεται διάφορες εκδοχές των δικτύων Πέτρι.

3.3 Πλεονεκτήματα και Σύγκριση

Το κύριο πλεονέκτημα της πρώτης διπλωματικής [15] έναντι της δεύτερης [16] είναι η ύπαρξη μιας διεπιφάνειας (interface), φιλικής και κατανοητής προς τον χρήστη η οποία τον βοηθά να μοντελοποιήσει τα MRPNs χωρίς μεγάλη δυσκολία, σε αντίθεση με την δεύτερη διατριβή [16] η οποία δεν έχει κάποια κάποια διεπιφάνεια κάτι που απαιτεί την καλή γνώση της γλώσσας ASP. Από την άλλη, η διατριβή [16], προκειμένου να εξετάσει την προσβασιμότητα του δικτύου σε μια πιθανή μελλοντική κατάσταση επιλύει τις κωδικοποιημένες σε ASP μορφές της κατάστασης του δικτύου και της επιθυμητής

μελλοντικής κατάστασης, με την βοήθεια των επίσης κωδικοποιημένων σε ASP κανόνων των MRPN. Από την άλλη, στην διατριβή [15], επιλέγεται ο αλγόριθμος ανάπτυξης γράφου ο οποίος περιέχει όλες τις πιθανές μελλοντικές καταστάσεις. Έπειτα εφαρμόζεται μια αναζήτηση εις πλάτος (Breath-First-Search), προκειμένου να βρεθεί η επιθυμητή μελλοντική κατάσταση αν και εφόσον υπάρχει.

Γνωρίζοντας ότι η αναζήτηση σε ένα μεγάλο γράφο, δηλαδή σε ένα δίκτυο με πολλές θέσεις, μεταβάσεις και μάρκες, δύναται να αυξήσει σημαντικά τον χρόνο ελέγχου της προσβασιμότητας, θα επιχειρήσουμε να προσθέσουμε την επιλογή εύρεσης της μελλοντικής προσβάσιμης κατάστασης (εφόσον υπάρχει), μέσω της κωδικοποίησης του δικτύου στην γλώσσα ASP, έχοντας εις γνώση ότι η γλώσσα ASP ειδικεύεται στην επίλυση NP-δύσκολων προβλημάτων, δηλαδή προβλημάτων τα οποία αποτελούν τα προβλήματα τα οποία θεωρούνται δύσκολο να λυθούν αποδοτικά.

3.4 Απαιτήσεις Υλοποίησης

3.4.1 Λειτουργικές Απαιτήσεις

Οι λειτουργικές απαιτήσεις διατυπώνουν το ακριβές σύνολο λειτουργιών, δυνατοτήτων και χαρακτηριστικών που πρέπει να διαθέτει το εργαλείο προκειμένου να εκπληρώνει αποτελεσματικά τον επιδιωκόμενο σκοπό του και να ανταποκρίνεται στις ανάγκες των χρηστών του. Οι απαιτήσεις αυτές καθορίζουν τις συγκεκριμένες συμπεριφορές, λειτουργίες που πρέπει να είναι σε θέση να εκτελεί το εργαλείο.

1. Η λήψη των περιγραφών των MRPN με τον τρόπο που γίνεται στο υφιστάμενο εργαλείο [15], και η μετατροπή τους στην γλώσσα λογικού προγραμματισμού ASP.
2. Έλεγχος προσβασιμότητας του δικτύου σε μια μελλοντική κατάσταση μέσω Clingo και επιστροφή αντίστοιχων αποτελεσμάτων.
3. Επεξεργασία αποτελεσμάτων από την εκτέλεση μέσω Clingo και επιστροφή σε αναγνώσιμη μορφή στην αντίστοιχη διεπιφάνεια, με την διευκρίνηση αν υπάρχει ακολουθία από πυροδοτήσεις μεταβάσεων από την οποία μπορεί να περάσει η αρχική μορφή του δικτύου προκειμένου να φτάσει στην τελική επιθυμητή κατάσταση. Αν η επιθυμητή κατάσταση του δικτύου είναι προσβάσιμη να

αναπαρίσταται γραφικά η διαδρομή του δικτύου από την αρχική στην τελική κατάσταση ανά χρονοστιγμή (timestamp).

4. Κατάλληλη τροποποίηση της διεπιφάνειας ούτως ώστε να ικανοποιούνται οι απαιτήσεις οι παραπάνω απαιτήσεις.

3.4.2 Μη Λειτουργικές Απαιτήσεις

Οι μη λειτουργικές απαιτήσεις καθορίζουν ένα σύνολο κριτηρίων που περιγράφουν την αναμενόμενη συμπεριφορά και απόδοση ενός συστήματος, εστιάζοντας σε πτυχές πέραν των ρητών λειτουργιών του.

1. Οι προσθήκες στο εργαλείο πρέπει να χαρακτηρίζονται από ευχρηστία εύκολες στην εκμάθηση, επιτρέποντας στους χρήστες να κατανοήσουν γρήγορα πώς να εκτελούν εργασίες και να περιηγούνται στη διεπαφή.
2. Το προσθήκες στο εργαλείο πρέπει χαρακτηρίζονται από υψηλή απόκριση και ορθότητα.

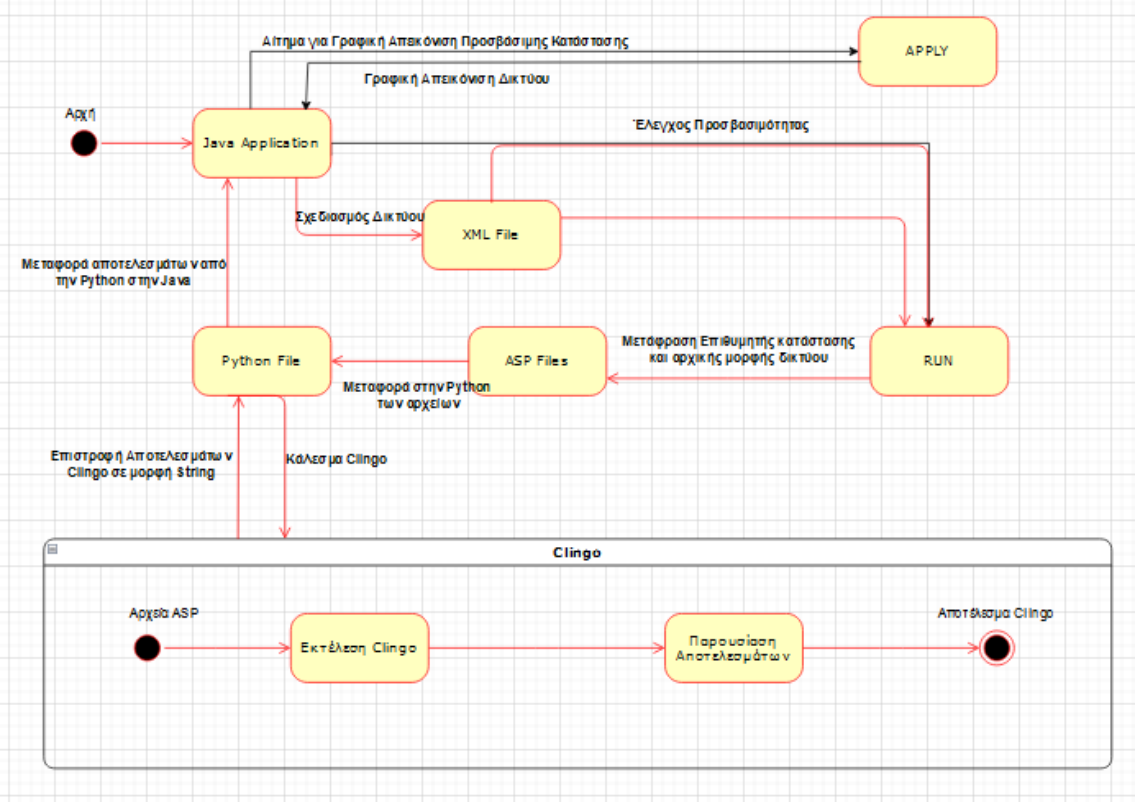
Κεφάλαιο 4 – Σχεδίαση και Υλοποίηση Συστήματος

Στο επικείμενο κεφάλαιο παρατίθεται η διαδικασία και η επιχειρησιακή πορεία η οποία καταγράφηκε προκειμένου να υλοποιηθεί η επέκταση του εργαλείου.

4.1 Αξιοποίηση προϋπαρχόντων συστημάτων

Η υλοποίηση κινήθηκε στους πυλώνες τους οποίους όρισαν οι δύο διατριβές [15] και [16]. Αρχικά αναλύθηκε πλήρως ο τρόπος κωδικοποίησης των δικτύων Πέτρι με πολλαπλές μάρκες ίδιου τύπου. Έτσι, έγινε αντιληπτό ότι θα έπρεπε να παραχθεί κώδικας στην γλώσσα Answer-Set-Programming (ASP), ενώ θα έπρεπε να λαμβάνεται ως είσοδος το δίκτυο Πέτρι το οποίο δημιουργείται και επεξεργάζεται από το γραφικό εργαλείο της διατριβής [15].

Η διαδικασία αυτή περιγράφεται διαγραμματικά στην Εικόνα 4.1.1 και επεξηγείται στα υπόλοιπα υποκεφάλαια:

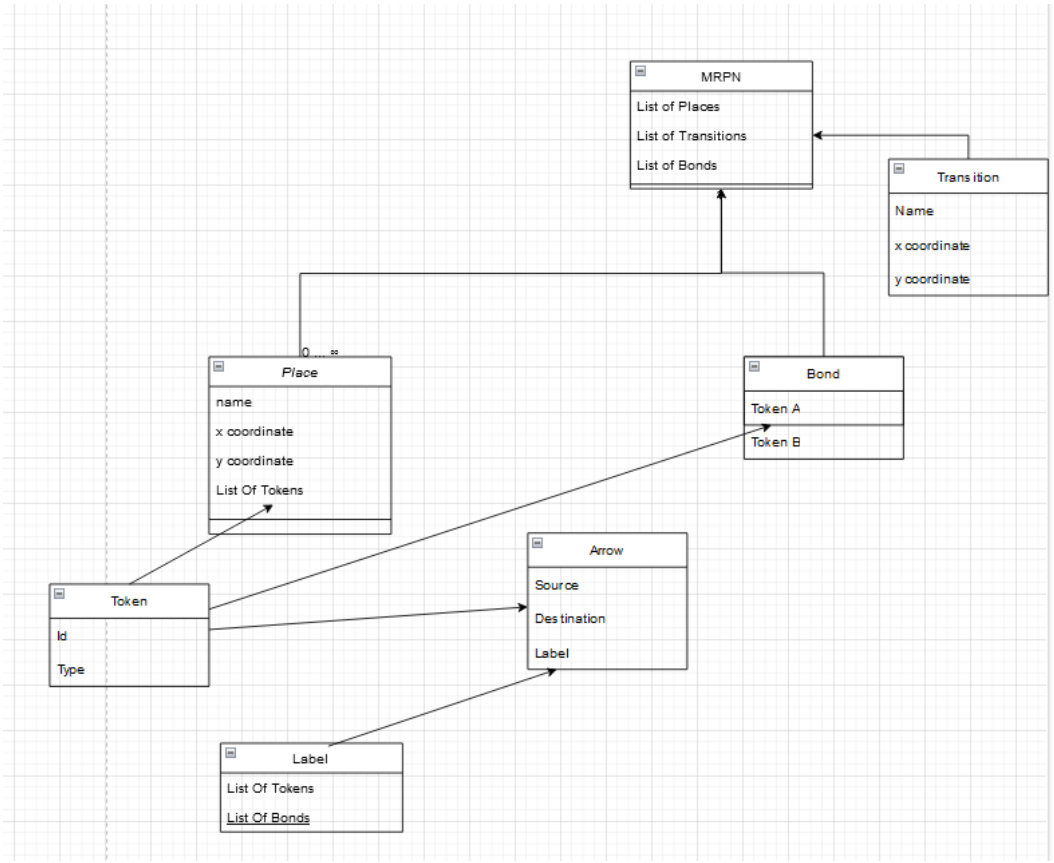


Εικόνα 4.1.1: Διάγραμμα Δράσεων Χρήστη – Εργαλείου - Clingo

4.2 Κωδικοποίηση αρχείου XML σε ASP

Το πρώτο βήμα ήταν η δημιουργία του αλγόριθμου ο οποίος λαμβάνει το αρχείο XML, που παράγεται από το εργαλείο και αντιστοιχεί στο δίκτυο το οποίο σχεδίασε ο χρήστης.

Η λογική που ακολουθήθηκε ήταν αρχικά, η δημιουργία της συνάρτησης η οποία διαβάζει το αρχείο και η αποθήκευση των χαρακτηριστικών του αρχείου στα αντίστοιχα αντικείμενα (objects) τα οποία συνάδουν με τες πληροφορίες σχετικά με τα βασικά συστατικά του δικτύου.



Εικόνα 4.2.1: Το διάγραμμα κλάσεων για το αντικείμενο MRPN.

Τα αντικείμενα Arrow, Label, Place, Token, Transition, Bond κληροδοτούν στο αντικείμενο Mrpn όπως φαίνεται και στην Εικόνα 4.2.1.

Στην συνέχεια έχει δημιουργηθεί η συνάρτηση translateToAsp η οποία όταν καλείται έχει την δυνατότητα να παίρνει το αρχείο XML το οποίο παράγεται από το εργαλείο της Java εφαρμογής, διαβάζει το αρχείο XML ως αρχείο κειμένου με αναδρομικό τρόπο.

Παρατίθεται ο ψευδοκώδικας και στο Παράρτημα Γ αυτούσιος ο κώδικας για την μετάφραση από XML αρχείο σε γλώσσα ASP.

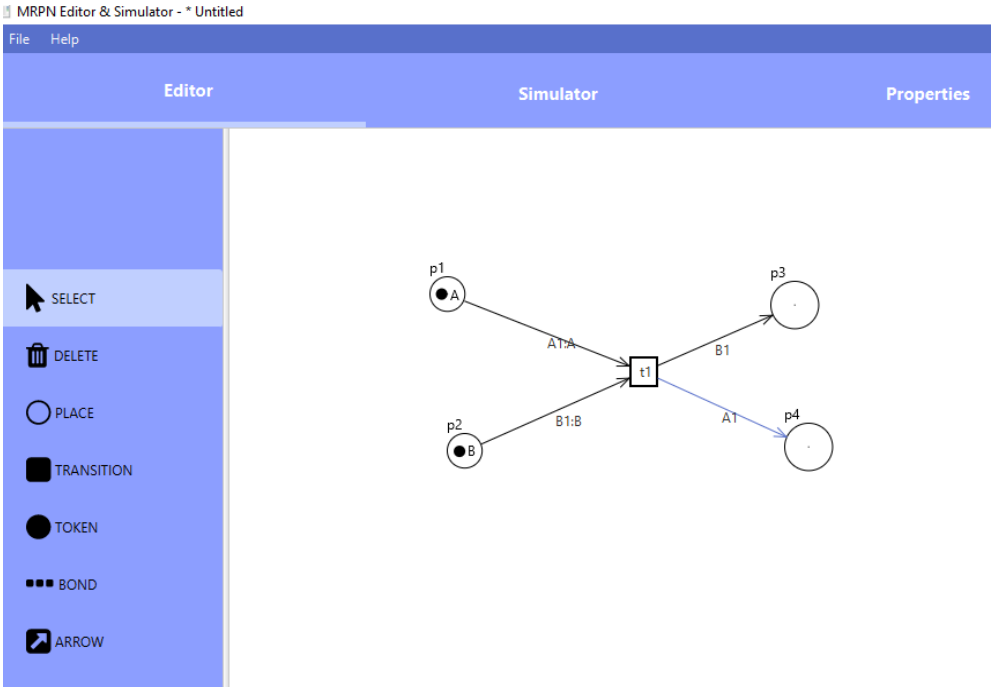
```
function readXML(node, level):
    indent = "  " * level
    s = ""
    s += indent + node.getNodeName() + ": " + node.getTextContent()
    checkNode(node)

    childNodes = node.getChildNodes()
    for i = 0 to childNodes.getLength() - 1:
        readXML(childNodes.item(i), level + 1)

    s1 += s

function checkNode(node):
    if node.getNodeName() equals "places":
        Set appropriate flags
    else if node.getNodeName() equals "name":
        Get name
    else if node.getNodeName() equals "x":
        Get x-coordinate
    else if node.getNodeName() equals "y":
        Get y-coordinate and create Transition object
    else if node.getNodeName() equals "transitions":
        Set appropriate flags
    else if placesFlag and node.getNodeName() equals "token":
        Create Token object and add it to Place object
    else if node.getNodeName() equals "arrows":
        Set appropriate flags
    else if node.getNodeName() equals "arrow" and arrowsFlag:
        Process arrow content
    else if node.getNodeName() equals "source" and arrowsFlag:
        Get source
    else if node.getNodeName() equals "destination" and arrowsFlag:
        Get destination
    else if node.getNodeName() equals "token" and arrowsFlag and not arrowBondsFlag and not totalBondsFlag:
        Process token content
    else if node.getNodeName() equals "bond" and totalBondsFlag:
        Process bond
    else if node.getNodeName() equals "token" and tokenBondFlag:
        Process token for bond
    else if node.getNodeName() equals "bonds":
        Process bonds
```

Εικόνα 4.2.2: Αλγόριθμος για την αποθήκευση των στοιχείων του XML αρχείου στα αντίστοιχα αντικείμενα της Java

```
function produceAsp():
    aspCode <- ""

    // Generate ASP code for places
    for i <- 0 to placeList.size() - 1:
        place <- placeList.get(i)
        aspCode <- aspPlaceRepresentation
        tokens <- place.getTokens()
        for j <- 0 to tokens.size() - 1:
            token <- tokens.get(j)
            aspCode <- aspTokenRepresentation

    // Generate ASP code for transitions
    for i <- 0 to transitionList.size() - 1:
        transition <- transitionList.get(i)
        aspCode <- aspTransitionRepresentation

    // Generate ASP code for arrows
    for i <- 0 to arrowList.size() - 1:
        arrow <- arrowList.get(i)
        aspCode <- aspArrowRepresentation

    // Generate ASP code for arrow bonds
    for i <- 0 to arrowList.size() - 1:
        arrow <- arrowList.get(i)
        bonds <- arrow.getBonds()
        for j <- 0 to bonds.size() - 1:
            bond <- bonds.get(j)
            aspCode <- aspArroBondRepresentation

    // Generate ASP code for total bonds
    for i <- 0 to totalBonds.size() - 1:
        totalBond <- totalBonds.get(i)
        if totalBond != null:
            aspCode <- aspForTotalBondsRepresentation

    // Write ASP code to file
    writeToFile(aspCode, pathForFiles + "aspCode.lp")
```

Εικόνα 4.2.3: Αλγόριθμος για την μετατροπή των αντικειμένων της Java σε κώδικα ASP

Βλέπουμε στην Εικόνα 4.2.4 το αρχικό δίκτυο:



Εικόνα 4.2.4: Η αρχική κατάσταση δικτύου

Το δίκτυο αυτό μεταφράζεται με την επιλογή της καρτέλας Properties στην εξής μορφής XML που βλέπουμε στην Εικόνα 4.2.5.

Εικόνα 4.2.5: Η απεικόνιση σε XML του δικτύου της Εικόνας 4.2.4

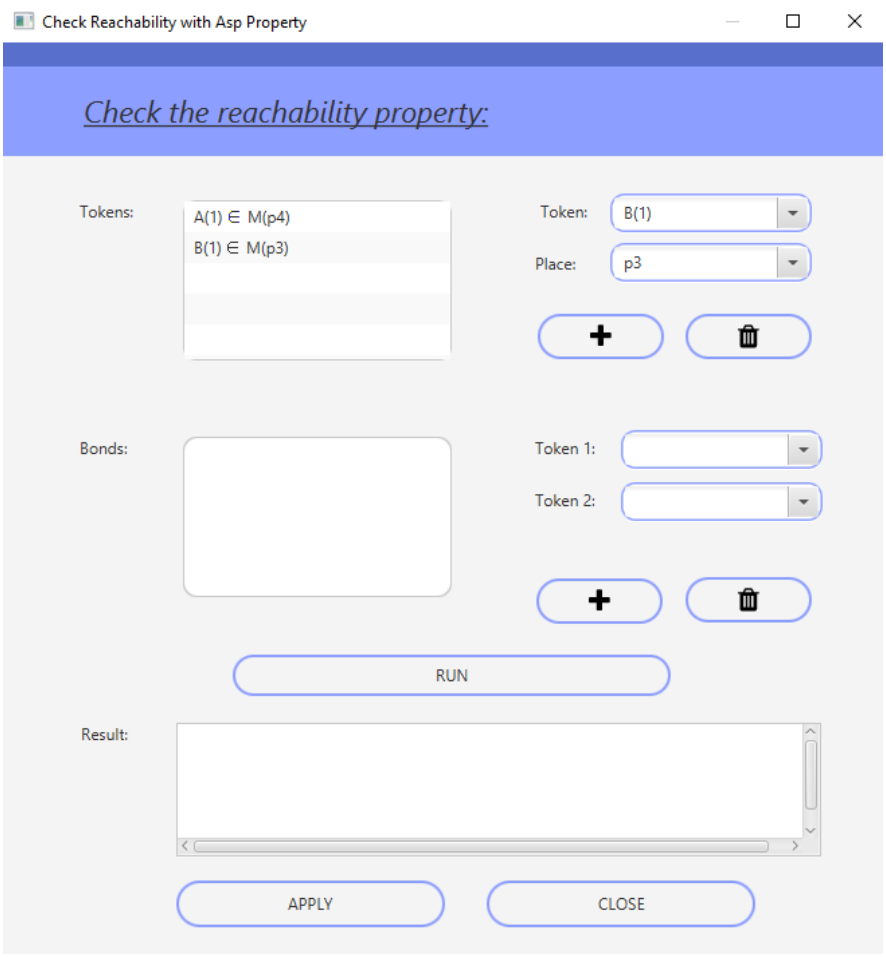
Το αρχείο αυτό αποθηκεύεται στο περιβάλλον του εργαλείου. Στην συνέχεια κάθε στοιχείο του αποθηκεύεται στα αντίστοιχα αντικείμενα. Και έτσι στην Εικόνα 4.2.6 βλέπουμε την μορφή του δικτύου στην γλώσσα ASP. Σημαντική λεπτομέρεια είναι το γεγονός ότι δεν επιτρέπεται από την Clingo η αναπαράσταση με κεφαλαίο γράμμα κάποιας θέσης, μετάβασης, μάρκας και τύπου μάρκας. Έτσι πριν γραφτεί ο κώδικας στο αρχείο προνοείται η μετατροπή του σε μικρά γράμματα όπου χρειαστεί.

```
holds(p1,b1,0).
holds(p2,b2,0).

type(a,b1).
type(b,b2).

ptarc(p1,t1,a1,a).
ptarc(p2,t1,b1,b).

tparc(t1,p3,b1,b).
tparc(t1,p4,a1,a).
```

Εικόνα 4.2.6: Το δίκτυο κωδικοποιημένο σε ASP

4.3 Κωδικοποίηση επιθυμητής κατάστασης σε ASP

Στην συνέχεια το εργαλείο ζητά από τον χρήστη να καταχωρήσει την κατάσταση η οποία θα ελεγχθεί αν είναι μελλοντικά προσβάσιμη με αυτό το δίκτυο.



Εικόνα 4.3.1: Η δήλωση της μελλοντικής επιθυμητής κατάστασης

Στο συγκεκριμένο παράδειγμα στην Εικόνα 4.3.1 δίνονται ως επιθυμητές συνθήκες, να βρίσκεται η μάρκα με τύπο A στην θέση p4 και η μάρκα με τύπο B στην θέση p3. Ακολούθως έχουμε την επιλογή να τρέξουμε το παράδειγμα προκειμένου να δούμε αν βρεθεί ακολουθία πυροδότησης μεταβάσεων μέσω του πατήματος του κουμπιού με την λέξη “RUN”. Για να επιτευχθεί αυτό, χρειάστηκε να κωδικοποιηθούν οι συνθήκες που επέλεξε ο χρήστης. Αυτές οι συνθήκες αποθηκεύονται ως HashMap, τα οποία αντικατοπτρίζουν τις μάρκες (tokens) και την θέση που βρίσκεται και τους δεσμούς (bonds) μαζί με την θέση που βρίσκονται. Αυτά τα Hashmap λαμβάνονται ως Strings και

μετατρέπονται μέσω συναρτήσεων στο String το οποίο θα αντιγραφεί στο αρχείο το οποίο θα τρέχει στο περιβάλλον της Clingo. Το πιο πάνω παράδειγμα έχει την εξής μορφή στην γλώσσα ASP.

```
1 goal:- holds(p3,b2,TS),
2 holds(p4,b1,TS),TS<4.
3
4
5 :- not goal.
6
```

Εικόνα 4.3.2: Η κωδικοποιημένη σε ASP μορφή της επιθυμητής κατάστασης

4.4 Αλγόριθμος Εκτέλεσης σε Clingo

Μετά από την κωδικοποίηση της επιθυμητής κατάστασης σε ASP, έχουμε δύο αρχεία σε γλώσσα ASP, το ένα με την αρχική κατάσταση του δικτύου και το άλλο την επιθυμητή κατάσταση σε ASP. Σε αυτό το σημείο θα χρησιμοποιηθεί η κωδικοποιημένη μορφή των κανόνων των MRPNs, αρχείο παραγμένο στα πλαίσια της διπλωματικής [16]. Αυτά τα τρία αρχεία θα εκτελεστούν στην Clingo, από την οποία αναμένουμε τα αποτελέσματα. Για να επιτευχθεί η επικοινωνία μεταξύ της Java και της Clingo χρειάστηκε η δημιουργία ενός προγράμματος στην γλώσσα Python η οποία ειδικεύεται στην επικοινωνία μεταξύ των περιβαλλόντων. Ο κώδικας της Python επισυνάπτεται στο Παράρτημα Δ. Το πρόγραμμα της Python είναι υπεύθυνο να λαμβάνει τα αρχεία που χρειάζονται, να επικοινωνεί με την Clingo, και να επιστρέφει στην Java τα αποτελέσματα από την εκτέλεση. Στο Παράρτημα Α επισυνάπτεται ένα παράδειγμα από τα αποτελέσματα μιας εκτέλεσης μέσω Clingo.

4.5 Επιστροφή Αποτελεσμάτων

Εφόσον τα αποτελέσματα από την Clingo επιστρέψουν στην Java, υπάρχει η πρόνοια για επεξεργασία των αποτελεσμάτων. Τα αποτελέσματα επιστρέφονται σε μορφή συμβολοσειράς (String). Έτσι ελέγχεται αρχικά αν το αποτέλεσμα περιέχει την λέξη “SATISFIABLE”, που αντιστοιχεί στο ότι η μελλοντική κατάσταση είναι μελλοντικά

προσβάσιμη ή “UNSATISFIABLE” η οποία υποδηλώνει το γεγονός ότι η μελλοντική κατάσταση δεν είναι εφικτή να επιτευχθεί μελλοντικά με το υφιστάμενο δίκτυο. Αν η μελλοντική κατάσταση η οποία καταχωρήθηκε είναι προσβάσιμη, το αποτέλεσμα συμπεριλαμβάνει την διαδικασία μετακίνησης και τροποποίησης των μαρκών στις θέσεις του δικτύου. Για το παράδειγμα το οποίο αναπτύχθηκε στις εικόνες αυτού του υποκεφάλαιου η απάντηση της Clingo, η οποία επιστρέφεται στην Java είναι η εξής:

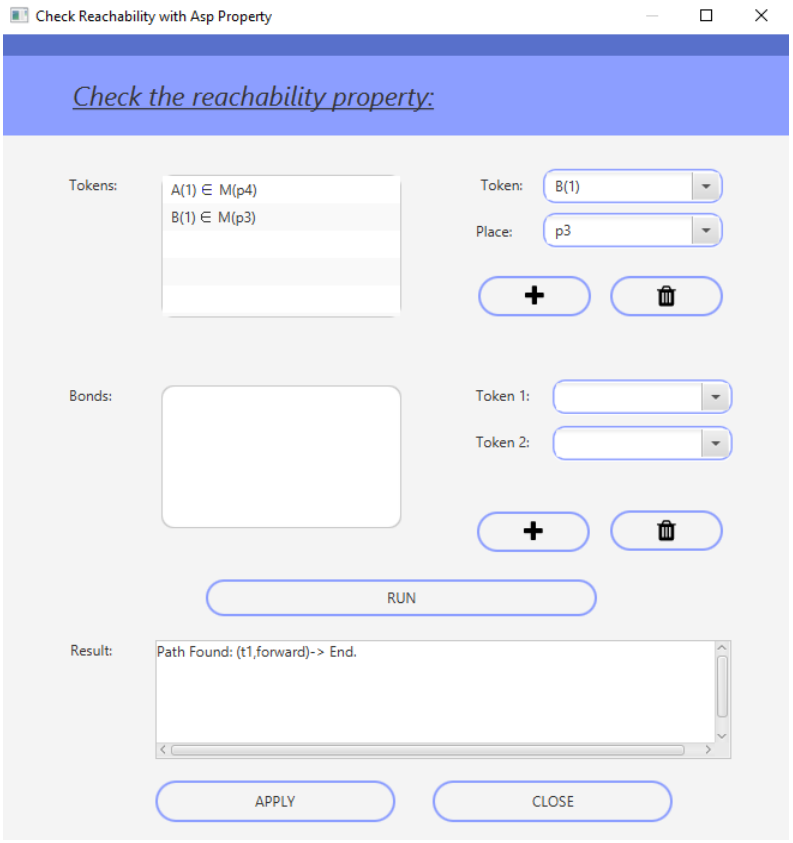
```
Solving...
Answer: 1
holds(p1,b1,0) holds(p2,b2,0) time(0) time(1) time(2)
time(3) time(4) time(5) fires(t1,0) firesb(t1,1) fires
(t1,2) firesb(t1,3) fires(t1,4) assigned(p1,t1,a1,b1,0
) satisfied(p1,t1,a1,b1,0) assigned(p2,t1,b1,b2,0) sat
isfied(p2,t1,b1,b2,0) holds(p4,b1,1) holds(p3,b2,1) co
ll_assigned(p4,t1,a1,b1,1) coll_assigned(p3,t1,b1,b2,1
) holds(p1,b1,2) holds(p2,b2,2) assigned(p1,t1,a1,b1,2
) satisfied(p1,t1,a1,b1,2) assigned(p2,t1,b1,b2,2) sat
isfied(p2,t1,b1,b2,2) holds(p4,b1,3) holds(p3,b2,3) co
ll_assigned(p4,t1,a1,b1,3) coll_assigned(p3,t1,b1,b2,3
) holds(p1,b1,4) holds(p2,b2,4) assigned(p1,t1,a1,b1,4
) satisfied(p1,t1,a1,b1,4) assigned(p2,t1,b1,b2,4) sat
isfied(p2,t1,b1,b2,4) holds(p4,b1,5) holds(p3,b2,5) co
ll_assigned(p4,t1,a1,b1,5) coll_assigned(p3,t1,b1,b2,5
) holds(p1,b1,6) firesb(t1,5) holds(p2,b2,6) enabled(t
1,0) enabled(t1,2) enabled(t1,4) enabled_coll(t1,1) en
abled_coll(t1,3) enabled_coll(t1,5)
SATISFIABLE
```

Εικόνα 4.5.1: Αποτέλεσμα μετά από εκτέλεση των αρχείων στην Clingo

4.6 Αναπαράσταση αποτελεσμάτων

Έπειτα αυτά τα αποτελέσματα αποθηκεύονται σε μια λίστα από συμβολοσειρές (Strings), προκειμένου να είναι πιο εύκολο να επεξεργαστούν. Ο τελευταίος αριθμός της παρένθεσης αντικατοπτρίζει την χρονική στιγμή στην οποία λαμβάνει χώρα το εκάστοτε γεγονός. Το μήνυμα fires συμβολίζει την πυροδότηση της μετάβασης που αναγράφεται στην πρώτη λέξη της παρένθεση την χρονική στιγμή που αναγράφεται στην δεύτερη λέξη. Το μήνυμα fires αντικατοπτρίζει το κείμενο το οποίο θα γραφτεί στην διεπιφάνεια σε περίπτωση που η μελλοντική κατάσταση είναι προσβάσιμη. Με την ίδια λογική και το μήνυμα holds περιέχει αντιστοίχως την θέση, το id της μάρκας και την χρονική στιγμή που βρίσκεται στην συγκεκριμένη θέση. Το συγκεκριμένο μήνυμα μαζί με το μήνυμα holdsbonds το οποίο συμβολίζει που βρίσκεται ένας δεσμός σε μια χρονική στιγμή,

φτιάχνουν την γραφική απεικόνιση του τρόπου με τον οποίο τροποποιείται το δίκτυο ούτως ώστε να φτάσει στην τελική επιθυμητή κατάσταση.



Εικόνα 4.6.1: Αποτέλεσμα και ένδειξη ακολουθίας πυροδοτήσεων

Βλέπουμε ότι αναγράφεται η ένδειξη ότι βρέθηκε μονοπάτι το οποίο οδηγεί το δίκτυο στην μελλοντική επιθυμητή κατάσταση που καταχωρήθηκε από τον χρήστη. Συγκεκριμένα αναγράφεται η πυροδότηση της μετάβασης t1.

Για την γραφική απεικόνιση λαμβάνουμε τα holds και τα holdsbonds τα οποία δεν υπερβαίνουν την χρονική στιγμή (Timestamp) στην οποία το δίκτυο βρέθηκε στην επιθυμητή κατάσταση. Αυτό γίνεται για τον λόγο ότι τα βήματα φαίνονται μέχρι εκεί που ορίζεται από το σενάριο του δικτύου με την εντολή Time(0..x). , με x να είναι οι χρονικές στιγμές για τις οποίες τρέχει το δίκτυο. Ο περιορισμός όμως, για το πότε ο χρήστης επιθυμεί να φτάσει το δίκτυο στην τελική κατάσταση κωδικοποιείται στην εντολή TS <

x όπου x η στιγμή στην οποία ελέγχεται ότι το δίκτυο έχει φτάσει ή περάσει από την επιθυμητή κατάσταση.

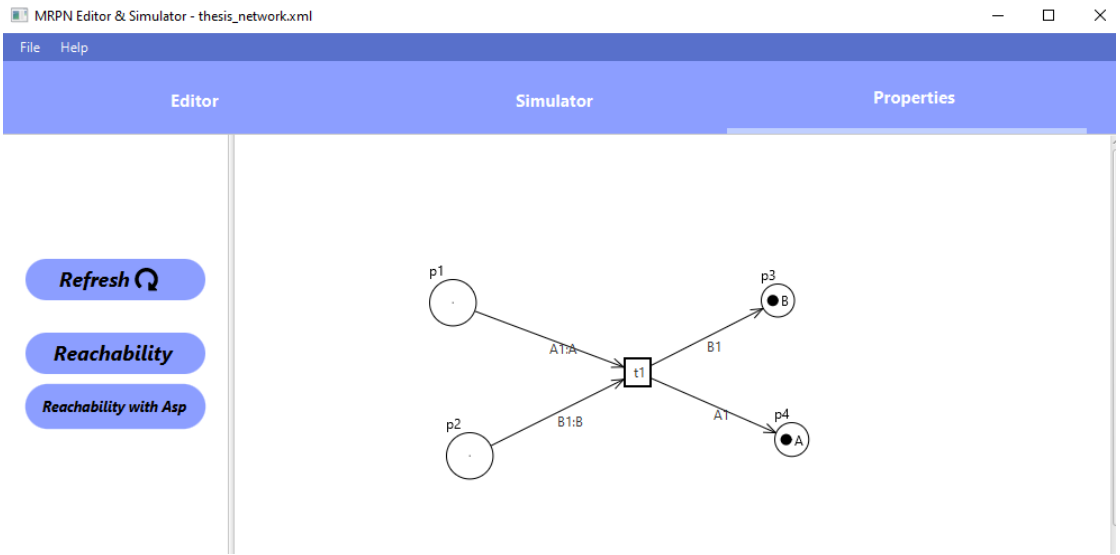
Αν και εφόσον έχει βρεθεί μονοπάτι το οποίο μας οδηγεί στην επιθυμητή κατάσταση η οποία καταχωρήθηκε, μπορούμε να δούμε γραφικά τα βήματα που ακολουθήθηκαν από το δίκτυο προκειμένου να φτάσει σε αυτή την κατάσταση, πατώντας το κουμπί APPLY. Η επιλογή αυτή μας δείχνει βήμα-βήμα την διαδικασία αυτή, δηλαδή για κάθε χρονική στιγμή. Οι πληροφορίες αυτές λαμβάνονται από την λίστα στην οποία αποθηκεύτηκαν οι εντολές που περιέχουν είτε την λέξη holds ή την λέξη holdsbonds. Επιλέχτηκε να αποφευχθεί η ολική δημιουργία ενός XML αρχείου για κάθε κατάσταση και να προτιμηθεί η μετακίνηση των μαρκών στο δίκτυο με βάση την χρονική στιγμή και το που υπάρχουν τα μηνύματα holds και holdsbonds τα οποία στέλνονται από την Clingo. καθώς μεταβάσεις και οι θέσεις σε ένα δίκτυο αφού είναι αμετάβλητες και δεν χρειάζονται να επαναπροσδιορίζονται σε κάθε χρονική στιγμή. Στην προκειμένη περίπτωση από την Εικόνα 4.5.1 βλέπουμε ότι η λίστα περιέχει όλες τις συμβολοσείρες με τις λέξεις holds ή holdsbonds αλλά η γραφική απεικόνιση σταματάει όταν η κατάσταση του δικτύου ικανοποιεί τις συνθήκες που ορίστηκαν, Άρα από την λίστα κρατούνται μόνο οι συμβολοσειρές “holds(p4,b1,1)” και “holds(p3,b2,1)” κάτι που αντικατοπτρίζεται και στα βήματα τα οποία εκτελούνται.

Στην Εικόνα 4.6.2 παρατίθεται ο ψευδοκώδικας για τον αλγόριθμο γραφικής αναπαράστασης και στο Παράρτημα Ε αυτούσιος ο κώδικας.

```
function apply():
    if not UNSATISFIABLE:

        if initial state is not rechable or holdsSteps.size() equal 0:
            // Retrieve necessary components from MRPN
            get scroll area, places, bonds and tokens

            Remove all tokens and bonds from the MRPN

            Refresh Scroll Area

            // Process holdsSteps
            for each step in holdsSteps:
                if step.contains("holdsbonds"):
                    // Extract holdsbonds function parameters
                    params = extractHoldsBondsParams(step)

                    // Add bonds based on holdsbonds function parameters
                    for each token in tokens:
                        if token.name.equals(params[1]):
                            for each otherToken in tokens:
                                if otherToken.name.equals(params[2]):
                                    for each place in places:
                                        if place.name.equals(params[0]):
                                            addBond(token, otherToken, place)

                else if step.contains("holds"):
                    // Extract holds function parameters
                    params = extractHoldsParams(step)

                    // Add tokens based on holds function parameters
                    for each token in tokens:
                        if token.name.equals(params[1]):
                            newToken = createToken(mrpn, params[0], params[1], token.type)
                            mrpn.addToken(newToken)

                refresh scroll area

                if time == holdsStepsCounter - 1:
                    holdsStepsCounter--
                    applyFlag = true

                if mrpn.getTokens().length == 0:
                    mrpn.scrollArea = scr
                    mrpn.scrollArea.refreshAll()

            holdsStepsCounter++
```

Εικόνα 4.6.2: Ψευδοκώδικας για το κουμπι Apply

Η γραφική απεικόνιση με την επιλογή του κουμπιού APPLY έχει ως εξής:



Εικόνα 4.6.3: Επιθυμητή κατάσταση

4.7 Τροποποίηση Διεπιφάνειας

Η υφιστάμενη διεπιφάνεια [15] υπέστη κάποιες αλλαγές και τροποποιήθηκε προκειμένου να υλοποιούνται οι λειτουργίες οι οποίες προστέθηκαν.

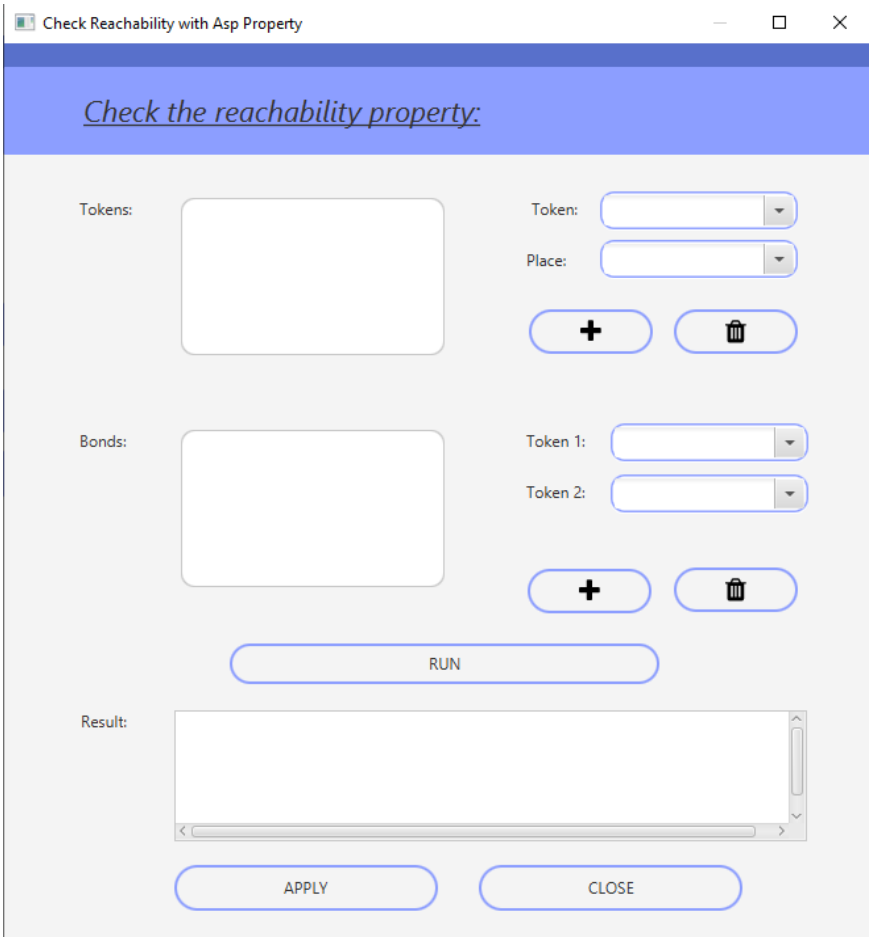
Τροποποίηση καρτέλας Properties

Υπήρξε η ανάγκη για προσθήκη του κουμπιού το οποίο θα ανοίγει το αναδυόμενο παράθυρο προκειμένου να ελεγχθεί η προσβασιμότητα. Στην Εικόνα 4.7.1 βλέπουμε πώς διαφοροποιείται η διεπιφάνεια στην καρτέλα Properties.

Εικόνα 4.7.1: Αριστερά η εικόνα με την προηγούμενη και δεξιά η εικόνα με την νέα έκδοση της καρτέλας Properties

Τροποποίηση Αναδυόμενου Παράθυρου για τον έλεγχο Προσβασιμότητας.

Με την επιλογή του κουμπιού “Reachability with Asp” ανοίγει το εξής παράθυρο:



Εικόνα 4.7.2: Το αναδυόμενο παράθυρο για τον έλεγχο της προσβασιμότητας

Το παράθυρο δεν έχει μεγάλες διαφορές οπτικά, από την προηγούμενη υλοποίηση του εργαλείου. Η μόνη διαφορά είναι η μη συμπερίληψη της επιλογής “Strengthened Bond” για τον κάθε δεσμό καθώς η γλώσσα ASP δεν υποστηρίζει αυτή την ιδιότητα του δεσμού. Η επιλογή strengthened bond υποδηλώνει ότι οι μάρκες του δεσμού αυτό δεν μπορούν να έχουν δεσμό με άλλες μάρκες, οι οποίες δεν είναι δηλωμένες σαν strengthened bond με τις μάρκες αυτού του δεσμού.

Επίσης έχει ανακατευθυνθεί το κουμπί RUN το οποίο μεταφράζει το δίκτυο σε ASP, μεταφράζει την επιθυμητή κατάσταση σε ASP, επικοινωνεί με την Clingo για να ελέγξει την προσβασιμότητα και επιστρέφει τα αποτελέσματα στην Java.

Ακόμη το κουμπί APPLY έχει τροποποιηθεί ούτως ώστε να λαμβάνει ως είσοδο την λύση που προέκυψε από την Clingo.

Κεφάλαιο 5– Αξιολόγηση Συστήματος

Αυτό το υποκεφάλαιο επικεντρώνεται στην αξιολόγηση του συστήματος που αναπτύχθηκε στο πλαίσιο της παρούσας διατριβής. Σκοπός της αξιολόγησης αυτής είναι να εκτιμηθεί η απόδοση του συστήματος στην επίτευξη των επιδιωκόμενων στόχων του.

5.1 Επίδοση

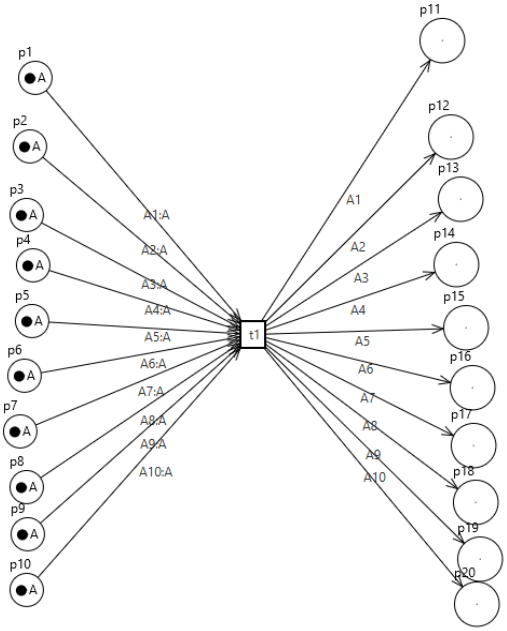
Η απόδοση του συστήματος αναφέρεται στα μετρήσιμα χαρακτηριστικά και τις δυνατότητες ενός συστήματος όσον αφορά την ταχύτητα, την αποδοτικότητα, την απόδοση, την απόκριση και τη χρήση των πόρων. Σχετίζεται με το πόσο αποτελεσματικά χρησιμοποιούνται οι διαθέσιμοι πόροι για την επίτευξη αποτελεσμάτων από το εργαλείο.

Χαρακτηριστικά Υπολογιστή:

Επεξεργαστής	AMD Ryzen 5 3500U with Radeon Graphics 2.10 GHz
RAM	8.00 GB
Λειτουργικό Σύστημα	Windows 10 Home
Τύπος Συστήματος	64-bit operating system, x64-based processor

Πίνακας 5.1.1: Χαρακτηριστικά Υπολογιστή για τις πειραματικές μετρήσεις

Θέτουμε το παράδειγμα δικτύου (Εικόνα 5.1.2) με το οποίο θα ελέγξουμε την επίδοση του αλγόριθμου.



Εικόνα 5.1.2 Παράδειγμα δικτύου 1

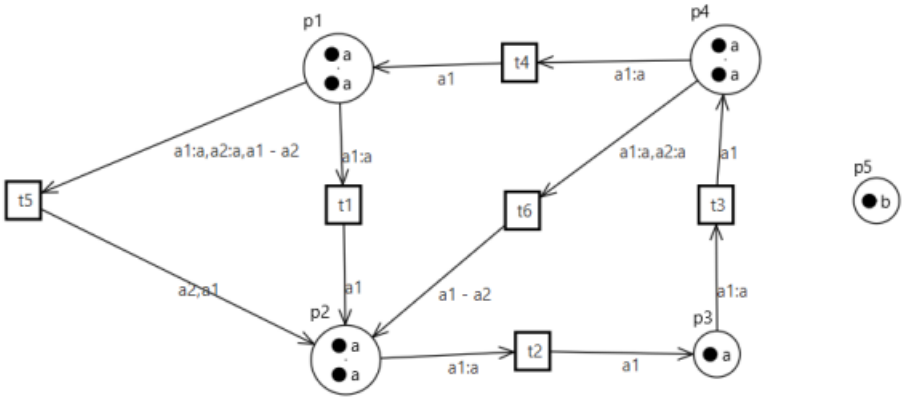
Επιθυμούμε να υπάρχει μια μάρκα σε κάθε θέση που δέχεται τόξο από την μετάβαση t1, με τρόπο ώστε p11, p12, p13… p20 να έχουν από μια μάρκα A. Έτσι τρέχουμε τον αλγόριθμο και έχουμε το εξής αποτέλεσμα:

Χρόνος Επίλυσης (ms)	6 (0.05%)
Διάρκεια Μετάφρασης (ms)	72 (6.4 %)
Διάρκεια επικοινωνίας Java – Clingo (ms)	828 (74.2%)
Λοιπές λειτουργίες	209 (18.7%)
Συνολική Διάρκεια Αλγορίθμου (ms)	1115

Πίνακας 5.1.3: Μετρήσεις επίλυσης με κωδικοποίηση δικτύου σε ASP για το δίκτυο 1

Βλέπουμε ότι ο συνολικός χρόνος για τον έλεγχο της προσβασιμότητας διαχωρίζεται σε επί μέρους χρόνους. Μεγάλο μέρος του συνολικού χρόνου λαμβάνεται από την ανάγκη του εργαλείου για επικοινωνία μέσω Clingo. Σχετικά μικρό επίσης είναι το ποσοστό του χρόνου που χρειάστηκε για να μεταφραστεί το δίκτυο από το αρχείο XML στην γλώσσα ASP.

Προχωρούμε στην δημιουργία ενός νέου δικτύου το οποίο χαρακτηρίζεται από την δημιουργία περισσότερων μελλοντικών καταστάσεων δικτύου προκειμένου να εξακριβωθεί πώς επηρεάζεται χρονικά ο αλγόριθμός.



Εικόνα 5.1.4 Παράδειγμα δικτύου 2

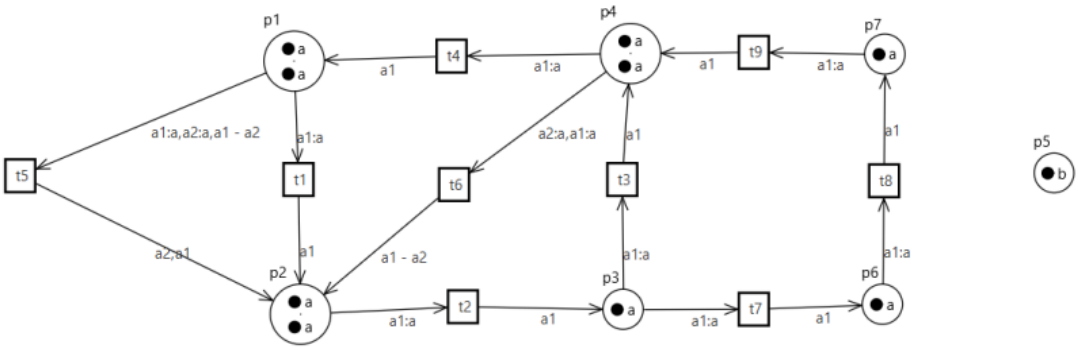
Αριθμός διαφορετικών πιθανών μελλοντικών καταστάσεων: 187

Χρόνος Επίλυσης (ms)	39 (3.4%)
Διάρκεια Μετάφρασης (ms)	64 (5.7%)
Διάρκεια επικοινωνίας Java – Clingo (ms)	840 (74.6%)
Λοιπές Λειτουργίες (ms)	372 (33.0%)
Συνολική Διάρκεια Αλγορίθμου (ms)	1125

Πίνακας 5.1.5: Μετρήσεις επίλυσης με κωδικοποίηση δικτύου σε ASP για το δίκτυο 2

Ο χρόνος επικοινωνίας της Java με την Clingo διατηρείται περίπου ο ίδιος και διαπιστώνεται ότι ο χρόνος για επεξεργασία του επιστρεφόμενου μηνύματος (λοιπές λειτουργίες) αυξάνεται αρκετά κάτι που οφείλεται στην έκταση του μηνύματος το οποίο στέλνεται από την Clingo. Επίσης ο χρόνος μετάφρασης από XML αρχείο στην γλώσσα ASP μειώνεται ελαφρώς και αυτό οφείλεται στο γεγονός ότι ο αριθμός των θέσεων είναι μικρότερος από το αριθμό του δικτύου στην Εικόνα 5.1.2.

Στην συνέχεια εξετάζουμε ένα δίκτυο με περισσότερες πιθανές μελλοντικές καταστάσεις, προκειμένου να δούμε πώς επηρεάζεται το εργαλείο.



Εικόνα 5.2.6 Παράδειγμα δικτύου 3

Το εργαλείο δεν ανταποκρίνεται στο πλήθος των καταστάσεων (13637) τις οποίες πρέπει να δημιουργήσει και έτσι ορίσαμε τον αλγόριθμο να τρέξει για 16 χρονοστιγμές μέσω της πιο κάτω γραμμής. Έπειτα εξετάσαμε επί μέρους τους χρόνους για τις υπόλοιπες λειτουργίες προκειμένου να λάβουμε τις απαραίτητες μετρήσεις.

```
time(0..15).
```

Εικόνα 5.2.7 Ορισμός χρονοστιγμών

Επιχειρήθηκε να τρέξει για περισσότερες χρονοστιγμές αλλά το σύστημα του προσωπικού υπολογιστή αδυνατούσε να ολοκληρώσει την εκτέλεση.
Επίσης λόγω αδυναμίας του αλγοριθμού να τρέξει μέσω eclipse (καθορισμός εκ προεπιλογής χρήσης περιορισμένου χώρου μνήμης RAM), τρέξαμε με clingo μέσω PowerShell και έχουμε το εξής αποτέλεσμα όσο αφορά τους χρόνους.

```
Models       : 1+
Calls        : 1
Time         : 1005.477s (Solving: 985.97s 1st Model: 975.50s Un
CPU Time     : 994.625s
PS C:\Users\dvere\OneDrive\Υπολογιστής\demetris vereis\Demetris
\MRPN-UI-master\MRPN_UI>
```

Εικόνα 5.2.8 Αποτελέσματα Clingo

Αριθμός διαφορετικών πιθανών μελλοντικών καταστάσεων: 13637

Χρόνος Επίλυσης (ms)	1005477 (97.5%)
Διάρκεια Μετάφρασης (ms)	125 (≈ 0%)
Διάρκεια επικοινωνίας Java – Clingo (ms)	920 (0.01%)
Λοιπές λειτουργίες (ms)	24528 (2.4%)
Συνολική Διάρκεια Αλγορίθμου (ms)	1031050

Πίνακας 5.2.9: Μετρήσεις επίλυσης με κωδικοποίηση δικτύου σε ASP για το δίκτυο 3

Από τις μετρήσεις του Πίνακα 5.2.9 διαπιστώνεται ότι η διεπιφάνεια χρειάζεται μεγάλο χρόνο για να δείξει τα αποτελέσματα λόγω του γεγονότος ότι απαιτείται μεγάλος χρόνος για την επίλυση των δικτύων μέσω Clingo γεγονός που αποδεικνύεται και στο ποσοστό σε σχέση με τον συνολικό χρόνο ο οποίος χρειάζεται.

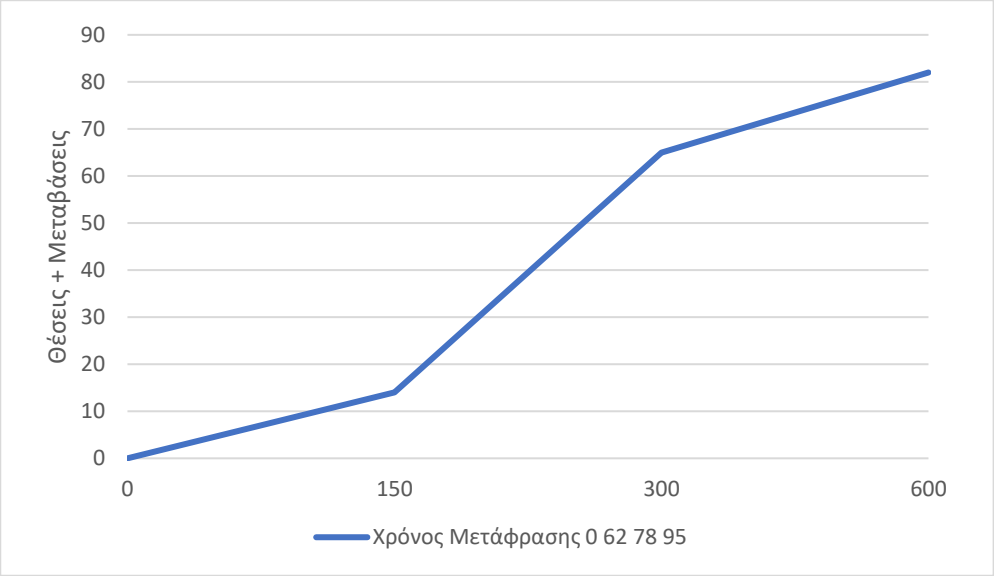
Αξιολόγηση Αλγόριθμου Μετάφρασης

Για τις ανάγκες τις αξιολόγησης της επίδοσης του Αλγόριθμου Μετάφρασης XML αρχείου σε γλώσσα ASP δημιουργήθηκαν δίκτυα με μεγάλο πλήθος θέσεων και μεταβάσεων.

Ο ενδεικτικός πίνακας των αποτελεσμάτων είναι ο εξής:

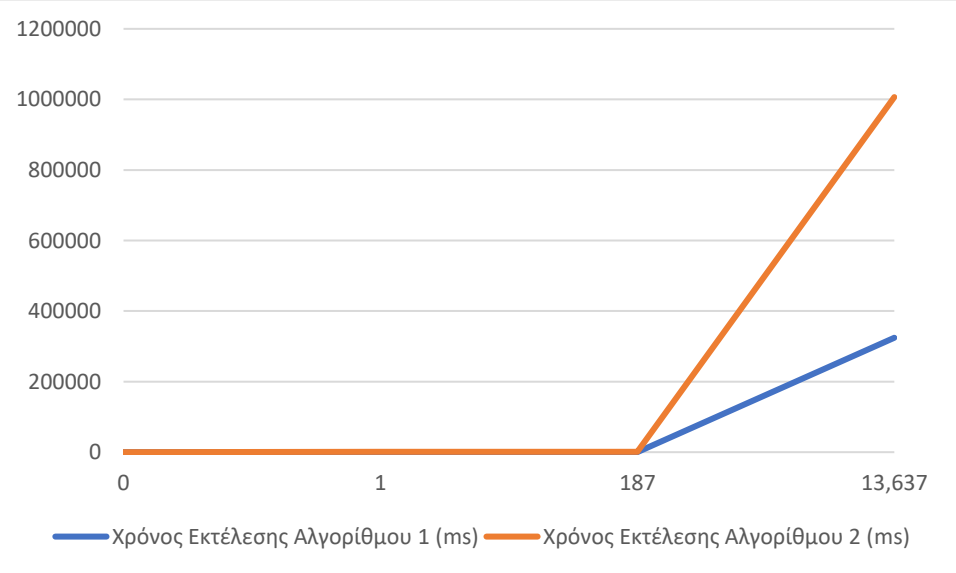
Άθροισμα Πλήθους Θέσεων + Μεταβάσεων	Χρόνος Μετάφρασης
150	62
300	78
600	95

Πίνακας 5.2.10: Χρόνος Μετάφρασης XML αρχείου σε γλώσσα ASP



Πίνακας 5.1.11: Γραφική Παράσταση Αποτελεσμάτων Πίνακα 5.1.10

Μα βάσει τις μετρήσεις του Πίνακα 5.1.10 και την γραφική παράσταση του Πίνακα 5.1.11, διαπιστώνουμε ότι με την αύξηση του αριθμού των θέσεων και των μεταβάσεων δεν επηρεάζεται σημαντικά ο χρόνος μετάφρασης καθώς το αρχείο διαβάζεται με την μορφή αρχείου κειμένου. Επίσης το δέντρο XML με τον τρόπο τον οποίο δημιουργείται, είναι αδύνατο να διαθέτει πολλά επίπεδα κάτι που θα αύξανε την πολυπλοκότητα της αναζήτησης των κόμβων του δέντρου.



Πίνακας 5.1.12 Γραφική Παράσταση Χρόνου – Πλήθους Πιθανών Καταστάσεων

Στην Πίνακα 5.1.12 βλέπουμε τις διαφορές στους συνολικούς χρόνους εκτέλεσης του αλγορίθμου σε σχέση με το πλήθος των πιθανών καταστάσεων. Το συμπέρασμα από αυτή την ακολουθία παρατηρήσεων είναι ότι η γλώσσα λογικού προγραμματισμού ASP απαιτεί αρκετά περισσότερο χρόνο για να αναπτύξει τις μελλοντικές καταστάσεις όσο ο αριθμός αυτών αυξάνεται. Επίσης ο χρόνος ο οποίος απαιτείται για την επικοινωνία ανάμεσα στο πρόγραμμα της Java, με το πρόγραμμα της Python το οποίο τρέχει με την σειρά του τα αρχεία στο περιβάλλον Clingo και η επιστροφή του αποτελέσματος με την αντίστροφη διαδικασία αναδεικνύει τον λόγο για τον οποίο δεν αποτελεί παρόμοια χρονικά λύση σε σχέση με την λύση η οποία υλοποιείται με την ανάπτυξη του γράφου ο οποίος υλοποιείται στην διατριβή [15] (Χρόνος Εκτέλεσης Αλγορίθμου 1).

Στην συνέχεια εξετάζεται ο χρόνος απόκρισης της εφαρμογής όταν επιλέγουμε το κουμπί APPLY. Προκειμένου η σύγκριση μεταξύ των μεγεθών να είναι εφικτή λαμβάνεται υπόψη ο μέσος όρος από την κάθε χρονική στιγμή. Τα αποτελέσματα έχουν όπως βλέπουμε στον Πίνακα 5.1.13.

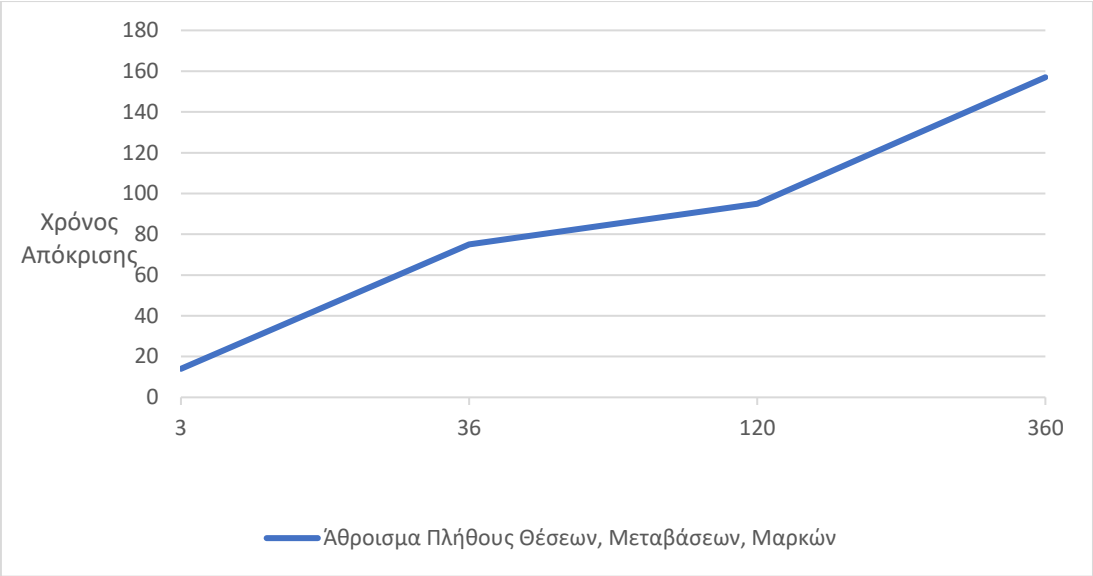
Πλήθος Θέσεων + Μεταβάσεων + Μαρκών	**Μέσος όρος χρόνου (ms)**
3	14
36	75
120	95
360	157

Πίνακας 5.1.13: Πίνακας Μετρήσεων για το κουμπί Apply

Για την γραφική απεικόνιση επιλέχτηκε να αποφευχθεί η ολική δημιουργία ενός XML αρχείου για κάθε κατάσταση και να προτιμηθεί η μετακίνηση των μαρκών στο δίκτυο με βάση την χρονική στιγμή και το που υπάρχουν τα μηνύματα holds και holdsbonds τα οποία στέλνονται από την Clingo. Με την επιλογή αυτή δεν χρειάζεται να επαναπροσδιορίζονται οι μεταβάσεις και οι θέσεις σε ένα δίκτυο αφού είναι αμετάβλητες.

Στον Πίνακα 5.1.14 φαίνεται πώς μεταβάλλεται ο χρόνος απόκρισης του κουμπιού APPLY σε σχέση με το άθροισμα των θέσεων, των μεταβάσεων και των μαρκών του

δικτύου. Στο άθροισμα προστέθηκε και το πλήθος των μαρκών για τον λόγο ότι αυτές μετακινούνται στο δίκτυο και έπειτα από εμπειρική χρήση φάνηκε ότι αποτελούν ένα από τους παράγοντες οι οποίοι επηρεάζουν τον χρόνο απόκρισης της γραφικής αναπαράστασης.



Πίνακας 5.1.14: Γραφική Παράσταση Καταστάσεων – Χρόνου

Κεφάλαιο 6 – Συμπεράσματα - Μελλοντική Εργασία

6.1 Συμπεράσματα

Στόχος της διπλωματικής ήταν η δημιουργία διασύνδεσης μεταξύ δύο εργαλείων, προερχόμενα από δύο υφιστάμενες διπλωματικές εργασίες [15] [16] προκειμένου να ελέγχεται η προσβασιμότητα του μοντέλου σε μια πιθανή μελλοντική κατάσταση. Συγκεκριμένα, υπήρξε η επέκταση του υφιστάμενου εργαλείου [15] το οποίο αναπτύχθηκε στην γλώσσα Java [10] το οποίο παρέχει στον χρήστη την ευκαιρία να σχεδιάσει, επεξεργαστεί, να αποθηκεύσει ένα δικτύου MRPN. Σε αυτές τις λειτουργίες ο χρήστης πλέον έχει την δυνατότητα να ελέγξει την προσβασιμότητα αυτού του δικτύου σε μια μελλοντική κατάσταση με την χρήση του εργαλείου [16]. Η χρησιμότητα αυτής της επέκτασης πηγάζει από το γεγονός ότι το εργαλείο της διατριβής [16] δεν διαθέτει γραφική διεπιφάνεια κάτι που δυσχεραίνει σε μεγάλο βαθμό την χρήση του, και προϋποθέτει ότι ο χρήστης διαθέτει γνώσεις προγραμματισμού στην γλώσσα ASP. Δημιουργήθηκε λοιπόν οι ανάγκη για σύνδεση αυτών των δύο εργαλείων καθώς στην διπλωματική αυτή ο χρήστης είναι σε θέση να σχεδιάσει ένα δίκτυο MRPN και να ελέγξει την προσβασιμότητά του σε μια πιθανή μελλοντική κατάσταση με την χρήση της γλώσσας ASP, η οποία ειδικεύεται στην επίλυση NP-δύσκολων προβλημάτων. Εφόσον τα δίκτυα MRPN μπορούν να χρησιμοποιηθούν ως βάση για τέτοιου είδους προβλήματα, ο χειριστής της διεπιφάνειας θα μπορεί να τα μοντελοποιήσει γραφικά και να προχωρήσει στην μελέτη τους με πιο εύκολο τρόπο.

Η πρόσθετη λειτουργία για την προσβασιμότητα σε μελλοντική κατάσταση μέσω κωδικοποίησης του δικτύου σε ASP εγγυάται την επίλυση NP-δύσκολων προβλημάτων, με την προϋπόθεση της αύξησης της υπολογιστικής ισχύς και της μνήμης RAM του υπολογιστή. Αξιοσημείωτο είναι το γεγονός ότι η νέα μέθοδος μειονεκτεί σε συνολική

ταχύτητα από αυτή που έχει η υφιστάμενη υλοποίηση, γεγονός που πηγάζει από τον τρόπο αναζήτησης της επιθυμητής μελλοντικής κατάστασης του δικτύου. Το μειονέκτημα, το οποίο προβλέφθηκε εκ των προτέρων ήταν ο χρόνος που χρειάζεται για την επικοινωνία ανάμεσα στο πρόγραμμα της Java, με το πρόγραμμα της Python το οποίο τρέχει με την σειρά του τα αρχεία στο περιβάλλον Clingo. Η επικοινωνία της Java με την Python φάνηκε βάσει δοκιμών ότι δεν ήταν χρονοβόρα έτσι προτιμήθηκε η συγκεκριμένη γλώσσα για την πρακτικότητά της. Επίσης η τροποποίηση του κώδικα ASP, κυρίως όταν μιλούμε για μια μεγάλη κατάσταση δικτύου ή/και μια μεγάλη επιθυμητή μελλοντική κατάσταση, σημαίνει και αύξηση του χρόνου εκτέλεσης καθώς έχουμε να κάνουμε με επεξεργασία κειμένου κατά την παραγωγή κώδικα. Έτσι δεν κατέστη δυνατό η διαδικασία να γίνει πιο αποδοτική από την υφιστάμενη καθώς η προηγούμενη τεχνική για έλεγχο της προσβασιμότητας δεν χρειαζόταν να επικοινωνεί με άλλα περιβάλλοντα γεγονός που φαίνεται τελικά ότι κόστισε χρονικά, όπως και η αδυναμία του πειραματικού υπολογιστή να μεγιστοποιήσει την απόδοση της ASP.

6.2 Μελλοντική Εργασία

Υπάρχουν σημαντικές δυνατότητες βελτίωσης του συστήματος. Ένας από τους κύριους τομείς εστίασης αφορά τη βελτίωση του αλγορίθμου προσβασιμότητας για την ελαχιστοποίηση της πολυπλοκότητάς του προκειμένου να καταστεί δυνατός ο εξορθολογισμός της διαδικασίας ανάλυσης προσβασιμότητας, αυξάνοντας έτσι την αποδοτικότητα και την απόδοση του συστήματος.

Ακόμη, εφόσον είναι θεμιτό, οι κωδικοποιημένοι κανόνες σε ASP από την προηγούμενη διπλωματική [16], θα μπορούσαν να τροποποιηθούν με τέτοιο τρόπο ούτως ώστε οι λύσεις να είναι βέλτιστες. Βελτιώνοντας τον αλγόριθμο προσβασιμότητας, μπορούν να επιτευχθούν αξιοσημείωτα κέρδη όσον αφορά την αποδοτικότητα του χρόνου εκτέλεσης, επιτρέποντας την ταχύτερη ανάλυση και προσομοίωση δικτύων. Αυτή η βελτίωση όχι μόνο βελτιώνει τη συνολική εμπειρία του χρήστη, αλλά επιτρέπει επίσης στους ερευνητές και τους επαγγελματίες να χειρίζονται μεγαλύτερα και πιο σύνθετα δίκτυα με μεγαλύτερη ευκολία.

Επιπροσθέτως, το αναδυόμενο παράθυρο για τον έλεγχο της προσβασιμότητας με κωδικοποίηση ASP θα μπορούσε να περιέχει την επιλογή για τρέξιμο του αλγόριθμου για ένα συγκεκριμένο αριθμό χρονικών στιγμών. Αυτό δεν πραγματοποιήθηκε γιατί απαιτούσε εκτεταμένη επεξεργασία των αρχείων JavaFX άρα και περαιτέρω εξοικείωση στην εν λόγω γλώσσα κάτι που δεν κατέστη δυνατό λόγω χρόνου.

Το υφιστάμενο εργαλείο θα μπορούσε επίσης, να επεκταθεί και στην καρτέλα Simulator προκειμένου να υπάρχει εμπρόσθια και αντίστροφη εκτέλεση (forward and backward execution) μέσω της γλώσσας ASP.

Βιβλιογραφία

[1] A. Philippou and K. Psara, "A collective interpretation semantics for reversing Petri nets," Theoretical Computer Science, In Press, 2022. Available: https://dl.acm.org/doi/abs/10.1016/j.tcs.2022.05.016

[2] Y. Dimopoulos, E. Kouppari, A. Philippou, and K. Psara, "Encoding Reversing Petri Nets in Answer Set Programming.", 2020. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7342048/

[3] A. Ozgur, O. Lévêque, and D. Tse, "Spatial degrees of freedom of large distributed MIMO systems and wireless ad hoc networks," IEEE Journal on Selected Areas in Communications, vol. 31, no. 2, pp. 202-214, 2013.

[4] A. Philippou and K. Psara, "Reversible computation in Petri nets," in International Conference on Reversible Computation, pages 136-148, Springer, 2018. Available: https://doi.org/10.48550/arXiv.1804.04607

[5] C. A Petri, "Kommunikation mit Automaten," Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

[6] Clingo and Gringo. Available: https://potassco.org/clingo/

[7] Clingo API documentation. Available: https://potassco.org/clingo/python-api/5.4

[8] Potassco.org. Available: https://potassco.org/

[9] "What is Java technology and why do I need it?" Available: https://www.java.com/en/download/faq/whatis_java.xml.

[10] "The Java Programming Language and the Java Platform." Available: https://www.oracle.com/technetwork/topics/newtojava/downloads/index.html.

[11] Eclipse IDE Home Page. Available: https://www.eclipse.org/ide/.

[12] PowerShell Documentation. Available: https://learn.microsoft.com/en-us/powershell/

[13] "JavaFX: Getting Started with JavaFX." Available: https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.html

[14] "JavaFX Scene Builder - A Visual Layout Tool for JavaFX Applications." Available: https://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html

[15] Ρ. Ιακώβου, "Ανάπτυξη γραφικού εργαλείου για την επεξεργασία και προσομοίωση αναστρέψιμων δικτύων Petri με πολλαπλές μάρκες ιδίου τύπου," Ατομική Διπλωματική Εργασία, Πανεπιστήμιο Κύπρου, 2022.

[16] M. A. Pittakaras, "Encoding multi-token reversing Petri nets in Answer Set Programming for simulation-based reasoning," Ατομική Διπλωματική Εργασία, Πανεπιστήμιο Κύπρου, 2021.

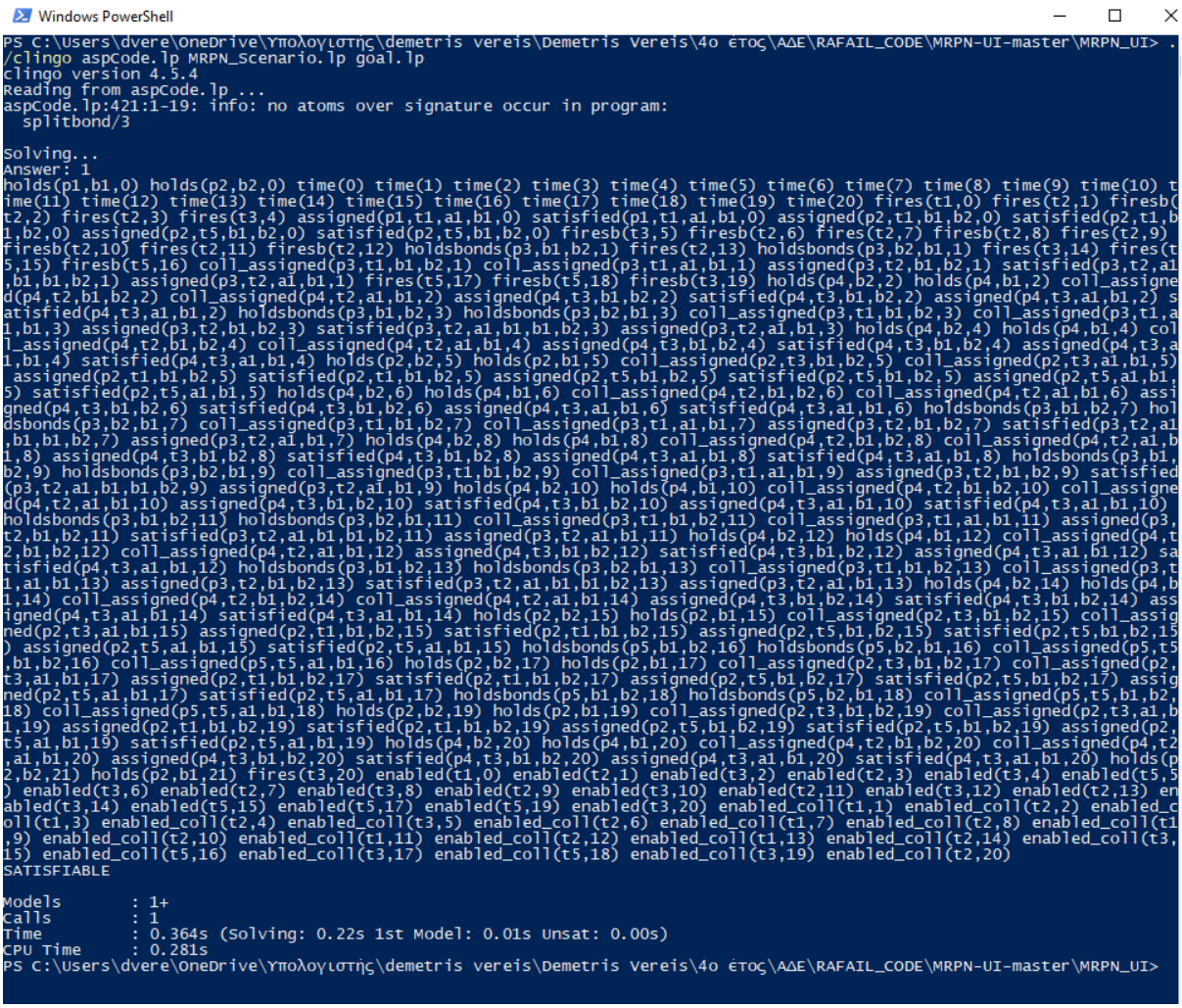
[17] M. Gelfond and V. Lifschitz, "The stable model semantics for logic programming," in Proceedings of International Logic Programming Conference and Symposium, R. Kowalski and K. Bowen (Eds.), MIT Press, pp. 1070-1080, 1988.

[18] R. Moore, "Semantical considerations on non-monotonic logic," Artificial Intelligence, vol. 25, no. 1, pp. 75-94, 1985.

[19] R. R., "A logic for default reasoning," Artificial Intelligence, vol. 13, pp. 81-132, 1980.

Παράρτημα Α

Παράδειγμα εκτέλεσης Clingo σε Powershell.



Παράρτημα Β

Μπορείτε να βρείτε τον κώδικα αναρτημένο στον εξής σύνδεσμο:
https://github.com/dverei01/MRPN_UI

Παράρτημα Γ

Κώδικας για μετάφραση XML αρχείου σε γλώσσα ASP:

```
public void readXML(Node node, int level) {
        StringBuilder indent = new StringBuilder();
        for (int i = 0; i < level; i++) {
                indent.append("  ");
        }
        String s = "";
        s += indent + node.getNodeName() + ": " + node.getTextContent();
        checkNode(node);
        NodeList childNodes = node.getChildNodes();
        for (int i = 0; i < childNodes.getLength(); i++) {
                readXML(childNodes.item(i), level + 1);
        }
        s1 += s;
}

public void checkNode(Node node) {
        if (node.getNodeName().equals("places")) {
                placesFlag = true;
                transitionsFlag = false;
                arrowsFlag = false;
                arrowBondsFlag = false;
                totalBondsFlag = false;
        } else if (node.getNodeName().equals("name")) {
                name = node.getTextContent();
        } else if (node.getNodeName().equals("x")) {
                x = Double.parseDouble(node.getTextContent());
        } else if (node.getNodeName().equals("y")) {
                y = Double.parseDouble(node.getTextContent());
                if (transitionsFlag) {
                        tr = new Transition(name, x, y);
                        transitionList.add(tr);
                        transitionList.add(tr);
                }
        } else if (node.getNodeName().equals("transitions")) {
                transitionsFlag = true;
                placesFlag = false;
                arrowsFlag = false;
                arrowBondsFlag = false;
                totalBondsFlag = false;
        } else if (placesFlag && node.getNodeName().equals("token")) {
                if (node.getTextContent().length() > 1) {
                        t = new Token(node.getTextContent().substring(0, node.getTextContent().length() - 1),
                                        node.getTextContent().charAt(node.getTextContent().length() - 1));
                } else {
                        t = null;
                }
                p = new Place(name, x, y);
                p.addToken(t);
                placeList.add(p);
        } else if (node.getNodeName().equals("arrows")) {
                arrowsFlag = true;
                transitionsFlag = false;
                placesFlag = false;
                arrowBondsFlag = false;
                totalBondsFlag = false;
        } else if (node.getNodeName().equals("arrow") && arrowsFlag) {
                arrowBondsFlag = false;
                String arrowContent = node.getTextContent();
                ArrayList<String> parts = new ArrayList<String>();
                parts = getStringSplited(arrowContent);
        } else if (node.getNodeName().equals("source") && arrowsFlag) {
                source = node.getTextContent();
        } else if (node.getNodeName().equals("destination") && arrowsFlag) {
                destination = node.getTextContent();
```

```
        } else if (node.getNodeName().equals("token") && arrowsFlag && !arrowBondsFlag && !totalBondsFlag) {
                arrowBondsFlag = false;
                String tokenContent = node.getTextContent();
                Token t;
                if (tokenContent.length() % 2 == 1)
                        t = new Token(tokenContent.substring(0, 2));
                else
                        t = new Token(tokenContent.substring(0));
                arrowTokens.add(t);
                if (!Character.isDigit(tokenContent.charAt(tokenContent.length() - 1)))
                        t.setType(tokenContent.charAt(tokenContent.length() - 1));
                else {
                        t.setType(tokenContent.charAt(0));
                }
        } else if (node.getNodeName().equals("bond") && totalBondsFlag) {
                tokenBondFlag = true;
                if (bondTokens.size() != 0) {
                        totalBonds.add(new Bond(bondTokens.get(0), bondTokens.get(1)));
                        totalBonds = new ArrayList<Bond>();
                }
        } else if (node.getNodeName().equals("token") && tokenBondFlag) {
                bondTokens.add(new Token(node.getTextContent()));
                if (bondTokens.size() > 1) {
                        Bond b = null;
                        b = new Bond(bondTokens.get(0), bondTokens.get(1));
                        totalBonds.add(b);
                        bondTokens = new ArrayList<Token>();
                }
        } else if (node.getNodeName().equals("bonds")) {
                if (arrowsFlag) {
                        arrowBondsFlag = true;
                        String bondsContent = node.getTextContent();
                        arrowBondCounter = bondsContent.length() / 4;
                        arrowBonds.clear();
                        for (int i = 0, start = 0; i < arrowBondCounter; i++, start += 4) {
                                arrowBonds.add(new Bond(new Token(bondsContent.substring(start, start + 2)),
                                                new Token(bondsContent.substring(start + 2, start + 4))));
                        }
                        Label label = new Label(arrowTokens, arrowBonds);
                        arrowTokens = new ArrayList<Token>();
                        arrowBonds = new ArrayList<Bond>();
                        Arrow arr = new Arrow(new Token(source), new Token(destination), label);
                        arrowList.add(arr);
                }
        } else if (node.getNodeName().equals("totalBonds")) {
                totalBondsFlag = true;
        } else if (node.getNodeName().equals("token") && totalBondsFlag) {
                // bondTokens.add(new Token(node.getTextContent()));
        }
}
```

Παράρτημα Δ

Κώδικας Python για επικοινωνία με Clingo:

```
import subprocess
import sys
import time

class LineRecord:
    def __init__(self, line, count, timestamp):
        self.line = line
        self.count = count
        self.timestamp = timestamp

def run(clingo_path, MRPN_scenario, file_MRPN, file_goal, timeToRun):
    start_time = time.time()
    run_command_scenario = ['java', '-classpath', 'C:/Users/dvere/eclipse-workspace/test/bin', 'test.translator2']
    input_data = b'This is a test input'
    process = subprocess.Popen(run_command_scenario, stdin=subprocess.PIPE, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    output, error = process.communicate(input=input_data)

    clingo_cmd = [clingo_path, file_MRPN, MRPN_scenario, file_goal, '--quiet=1']
    result = subprocess.run(clingo_cmd, capture_output=True, text=True)
    result = result.stdout.replace(") ", ")\n")
    result = result.replace(") ", ")\n")

    listOfMoves = [[], [], [], [], [], [], [], []]
    labels = ["Coll_assigned", "Assigned", "Holds", "Enabled", "Fires", "Satisfied", "Enabled_coll", "Holds_Bonds"]
    lines = result.split('\n')

    Time = 0
    CPUtime = 0
    Models = 0
    cnt = 0

    for line in lines:
        if "coll_assigned" in line:
            timestamp = line.split("(")[-1].split(")")[0]
            listOfMoves[0].append(LineRecord(line, cnt, timestamp.split(",")[-1]))
        elif "assigned" in line:
            timestamp = line.split("(")[-1].split(")")[0]
            listOfMoves[1].append(LineRecord(line, cnt, timestamp.split(",")[-1]))
        elif "holds" in line:
            timestamp = line.split("(")[-1].split(")")[0]
            if "holdsbonds" in line:
                listOfMoves[7].append(LineRecord(line, cnt, timestamp.split(",")[-1]))
            else:
                listOfMoves[2].append(LineRecord(line, cnt, timestamp.split(",")[-1]))
        elif "enabled" in line:
            timestamp = line.split("(")[-1].split(")")[0]
            if "enabled_coll" in line:
                listOfMoves[6].append(LineRecord(line, cnt, timestamp.split(",")[-1]))
            else:
                listOfMoves[3].append(LineRecord(line,cnt, timestamp.split(",")[-1]))
        elif "fires" in line:
            timestamp = line.split("(")[-1].split(")")[0]
            listOfMoves[4].append(LineRecord(line, cnt, timestamp.split(",")[-1]))
        elif "satisfied" in line:
            timestamp = line.split("(")[-1].split(")")[0]
            listOfMoves[5].append(LineRecord(line, cnt, timestamp.split(",")[-1]))
        elif "Time" in line and "Solving" in line:
            Time = float(line.split(":")[1].split("s")[0].strip())
        elif "CPU" in line:
            CPUtime = float(line.split(":")[1].split("s")[0].strip())
        elif "Models" in line:
            if line.split(":")[1].split("s")[0].strip().isdigit():
                Models = int(line.split(":")[1].split("s")[0].strip())
            else:
                Models = 0
        else:
            cnt = cnt - 1
        cnt = cnt + 1
```

```
    for i, moves in enumerate(listOfMoves):
        print(f"->Moves type {labels[i]}:")
        for move in moves:
            print(move.line, move.count, move.timestamp, ">>")

    cnt = 0

    for i in range(0, timeToRun+1):
        print((f"->TimeStamp {i}:"))
        for inner_list in listOfMoves:
            for item in inner_list:
                if int(item.timestamp) == i:
                    cnt = cnt + 1
                    print(item.line)

    print("Time: ", Time, "s")
    print("CPUtime: ", CPUtime, "s")
    print("Models: ", Models)

    SAT = False
    if "UNSATISFIABLE" not in result:
        SAT = True

    print("SAT:", SAT)
    end_time = time.time()

    print("Execution time: {:.2f} seconds".format(end_time - start_time))

if len(sys.argv) != 6:
    run(clingo_path = r'C:\Users\dvere\OneDrive\Υπολογιστής\demetris vereis\Demetris Vereis\4ο έτος\ΑΔΕ\clingo-4.5.4-win64\clingo.exe', MRPN_scenario = "MRPN_Scenario.lp",
        file_MRPN =  'C:\\Users\\dvere\\PycharmProjects\\pythonProject\\Ptixiaki\\MRPN.lp', file_goal = 'C:\\Users\\dvere\\PycharmProjects\\pythonProject\\Ptixiaki\\test.lp', timeToRun = 5)
    print("Run With User")
else:
    run(clingo_path=sys.argv[1], MRPN_scenario=sys.argv[2], file_MRPN=sys.argv[3], file_goal=sys.argv[4], timeToRun = sys.argv[5])
```

Παράρτημα Ε

```
void apply() {
        if (!UNSAT) {
                if (!initalIsFinalFlag || holdsSteps.size() == 0) {
                        ScrollArea scr = mrpn.getScrollArea();
                        if (applyFlag)
                                holdsStepsCounter = time;
                        Place places[] = mrpn.getPlaces();
                        Bond b[] = mrpn.getBonds();
                        Token t[] = mrpn.getTokens();
                        ArrayList<Token> tokens = new ArrayList<Token>();

                        // Collect all tokens in the MRPN
                        for (int i = 0; i < t.length; i++)
                                tokens.add(t[i]);

                        // Add source and destination tokens of bonds to the token list
                        for (int i = 0; i < b.length; i++) {
                                if (!tokens.contains(b[i].getSource()))
                                        tokens.add(b[i].getSource());
                                else if (!tokens.contains(b[i].getDestination()))
                                        tokens.add(b[i].getDestination());
                        }

                        // Convert the token list back to an array
                        t = new Token[tokens.size()];
                        for (int i = 0; i < t.length; i++)
                                t[i] = tokens.get(i);

                        String tokenTypes[] = new String[t.length];
                        boolean moved[] = new boolean[t.length];
                        Place tokenPlaceBeforeApply[] = new Place[t.length];

                        // Store the original place and type of each token
                        for (int i = 0; i < t.length; i++) {
                                moved[i] = false;
                                tokenPlaceBeforeApply[i] = t[i].getPlace();
                                tokenTypes[i] = t[i].getType();
                        }

                        // Remove all tokens and bonds from the MRPN
                        for (int i = 0; i < t.length; i++) {
                                String p = t[i].getPlace().name;
                                mrpn.removeToken(t[i]);
                        }

                        for (int i = 0; i < b.length; i++)
                                mrpn.removeBondFromMRPN(b[i].getSource(), b[i].getDestination());

                        mrpn.scrollArea.refreshAll();


                for (int i = 0; i < holdsSteps.size(); i++) {
                        if (holdsSteps.get(i).indexOf(',') != -1) {
                                if (holdsSteps.get(i)
                                                .substring(holdsSteps.get(i).lastIndexOf(',') + 1, holdsSteps.get(i).indexOf(')'))
                                                .equals("" + holdsStepsCounter)) {
                                        if (holdsSteps.get(i).contains("holdsbonds")) {
                                                String s[] = parseFunctionHoldsBonds(holdsSteps.get(i), 3);

                                                // Add bonds based on holdsbonds function parameters
                                                for (int j = 0; j < t.length; j++)
                                                        if (t[j].name.equals(s[1])) {
                                                                for (int k = 0; k < t.length; k++) {
                                                                        if (t[k].name.equals(s[2]))
                                                                                for (int l = 0; l < places.length; l++)
                                                                                        if (places[l].name.equals(s[0]))
                                                                                                addBond(t[j], t[k], s[0]);
                                                                }
                                                        }
                                        } else if (holdsSteps.get(i).contains("holds")) {
                                                String s[] = parseFunctionHolds(holdsSteps.get(i), 2);

                                                // Add tokens based on holds function parameters
                                                for (int j = 0; j < t.length; j++) {
                                                        if (t[j].name.equals(s[1])) {
                                                                Token t1 = new Token(mrpn, mrpn.getPlace(s[0]), s[1], t[j].type);
                                                                mrpn.addToken(mrpn.getPlace(s[0]), t1.name, t1.type);
                                                        }
                                                }
                                        }
                                }
                        }
                        mrpn.scrollArea.refreshAll();

                        if (time == holdsStepsCounter - 1) {
                                holdsStepsCounter--;
                                applyFlag = true;
                        }

                        if (mrpn.getTokens().length == 0) {
                                mrpn.scrollArea = scr;
                                mrpn.scrollArea.refreshAll();
                        }
                }

                holdsStepsCounter++;
        }

        System.out.println("apply mrpn!");
} else {
        System.out.println("Unsatisfiable Situation No Apply Effect");
}
}
```