Ατομική Διπλωματική Εργασία

ΑΝΑΠΤΥΞΗ ΓΡΑΦΙΚΟΥ ΕΡΓΑΛΕΙΟΥ ΓΙΑ ΤΗΝ ΕΠΕΞΕΡΓΑΣΙΑ ΚΑΙ ΠΡΟΣΟΜΟΙΩΣΗ ΑΝΑΣΤΡΕΨΙΜΩΝ ΔΙΚΤΥΩΝ ΠΕΤΡΙ ΜΕ ΠΟΛΛΑΠΛΕΣ ΜΑΡΚΕΣ ΙΔΙΟΥ ΤΥΠΟΥ

Ραφαήλ Ιακώβου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ιούνιος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη γραφικού εργαλείου για την επεξεργασία

και προσομοίωση Αναστρέψιμων δικτύων Πέτρι με πολλαπλές μάρκες ιδίου τύπου

Ραφαήλ Ιακώβου

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 2022

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά την επιβλέπουσα καθηγήτρια μου Δρ. Άννα Φιλίππου που μου έδωσε την ευκαιρία να εκπονήσω το συγκεκριμένο θέμα διπλωματικής εργασίας καθώς και για τη συνεχή υποστήριξη που μου παρείχε.

Ακόμη δεν μπορώ να παραλείψω την πολύ καλή συνεργασία που είχα με συμφοιτητές μου, καθ' όλη την διάρκεια των σπουδών μου.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για τη στήριξη που μου παρείχε.

Περίληψη

Η διπλωματική μου εργασία ασχολείται με τα αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου (Multi Reversing Petri Nets - MRPN), τα οποία είναι ένα μαθηματικό μοντέλο που επεκτείνει τα αναστρέψιμα δίκτυα Πέτρι (Reversing Petri Nets - RPN) με την προσθήκη πολλαπλών μαρκών (tokens) ιδίου τύπου. Τα μοντέλα αυτά υποστηρίζουν Αναστρέψιμο Υπολογισμό, ο οποίος παρουσιάζεται σε πληθώρα συστημάτων όπως τα βιοχημικά συστήματα.

Η διπλωματική αυτή έχει σκοπό την υλοποίηση μιας desktop εφαρμογής, με την προγραμματιστική γλώσσα Java. Αυτή η εφαρμογή, επιτρέπει στους χρήστες να δημιουργούν, να τροποποιούν και να αποθηκεύουν αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου. Ακόμα, θα υπάρχει η δυνατότητα προσομοίωσης των δικτύων αυτών με εμπρόσθια (forward) ή αντίστροφη (backward) εκτέλεση οποιασδήποτε μετάβασης, καθώς και έλεγχος προσβασιμότητας (reachability) κατάστασης, δηλαδή, κατά πόσο το δίκτυο μπορεί να βρεθεί σε μια συγκεκριμένη κατάσταση στο μέλλον.

Η εφαρμογή που υλοποιήθηκε είναι βασισμένη και επεκτείνει προηγούμενη διπλωματική εργασία, η οποία υλοποίησε μια desktop εφαρμογή με τις παραπάνω λειτουργίες για αναστρέψιμα δίκτυα Πέτρι. Στα αναστρέψιμα δίκτυα Πέτρι, σε αντίθεση με τα αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου, δεν μπορούν να υπάρξουν μάρκες ίδιου τύπου. Έτσι, τροποποιήθηκε αυτή η εφαρμογή με σκοπό να υποστηρίζει τις λειτουργίες που προαναφέρθηκαν για πολλαπλές μάρκες ιδίου τύπου.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Τα Αναστρέψιμα Δίκτυα Πέτρι (Reversing Petri Nets), είναι ένα μαθηματικό μοντέλο που επεκτείνει το μαθηματικό μοντέλο Δίκτυα Πέτρι (Petri Nets) του Carl Adam Petri, με το να ενσωματώνει Αναστρέψιμο Υπολογισμό. Συγκεκριμένα, τα Αναστρέψιμα Δίκτυα Πέτρι υποστηρίζουν οπισθοδρόμηση (backtracking), αναστρεψιμότητα αιτιολογικής σειράς (causal order reversibility) και αναστρεψιμότητα μη αιτιολογικής σειράς (out-of-causal order reversibility).

Ο Αναστρέψιμος Υπολογισμός, τα τελευταία χρόνια είναι ένα ενεργό ερευνητικό πεδίο όπου χρησιμοποιείται σε πληθώρα συστημάτων. Ο υπολογισμός μπορεί να εκτελεστεί προς τα εμπρός (forward) αλλά και προς τα πίσω (backward), μια ιδιότητα η οποία χρησιμοποιείται τόσο σε βιολογικά συστήματα, όσο και σε αντιστρέψιμες κβαντικές λειτουργίες. Παράλληλα, η αντιστρεψιμότητα είναι επιθυμητή ιδιότητα σε συστήματα λόγο της χαμηλής υπολογιστικής ισχύς και την ανοχή σε σφάλματα.

Τα Δίκτυα Πέτρι είναι ένα μαθηματικό εργαλείο που χρησιμοποιείται κυρίως για τη μοντελοποίησης και την ανάλυση της συμπεριφοράς παράλληλων και κατανεμημένων συστημάτων.

Τα Αναστρέψιμα Δίκτυα Πέτρι επεκτείνουν τα Δίκτυα Πέτρι με τη δυνατότητα αναστρεψιμότητας, κάνοντας έτσι εφικτή την επαναφορά σε μια κατάσταση που είχε προηγουμένως το σύστημα. Ένα από τα κύρια συστατικά των δικτιών αυτών είναι οι μοναδικές μάρκες (tokens) που μπορούν να συνδέονται μεταξύ τους για να σχηματίσουν δεσμούς (bonds) και διατηρούνται κατά την εκτέλεση.

Τα Αναστρέψιμα Δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου, επεκτείνουν περεταίρω τα Αναστρέψιμα Δίκτυα Πέτρι με την προσθήκη πολλαπλών μαρκών ιδίου τύπου οι οποίες είναι ισοδύναμες μεταξύ τους. Αυτό επιτρέπει την αντιστροφή μιας μετάβασης με τη χρήση οποιονδήποτε μαρκών ιδίου τύπου με αυτούς που απαιτεί η

μετάβαση, με αποτέλεσμα μεγαλύτερη ευελιξία στην αναστρεψιμότητα μη αιτιολογικής σειράς.

1.1 Συμβολή

Σκοπός αυτής της διπλωματικής είναι η υλοποίηση ενός γραφικού εργαλείου, το οποίο επιτρέπει στους χρήστες να δημιουργούν, να τροποποιούν και να αποθηκεύουν Αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου. Επιπλέων, το εργαλείο θα ελέγχει αν το δίκτυο είναι καλά ορισμένο και σε αντίθετη περίπτωση θα παρέχει ενδεικτικές πληροφορίες για το λόγο που το δίκτυο δεν ακολουθεί τον ορισμό του καλώς ορισμένου δικτύου. Η αποθήκευση και άνοιγμα των αναστρέψιμων δικτύων Πέτρι με πολλαπλές μάρκες ιδίου τύπου γίνετε σε μορφή XML. Ακολούθως, μετά τη δημιουργία του δικτύου θα υπάρχει η δυνατότητα προσομοίωσης, όπου ο χρήστης θα μπορεί να επιλέξει την εκτέλεση οποιασδήποτε έγκυρης μετάβασης, σύμφωνα είτε με τον εμπρόσθιο είτε με τον αναστρέψιμο υπολογισμό, της συγκεκριμένης κατάστασης του δικτύου. Μετά την επιλογή της μετάβασης από τον χρήστη το εργαλείο την εκτελεί και με τον ίδιο τρόπο θα μπορεί να επιλέξει νέα μετάβαση. Τέλος, ο χρήστης μπορεί να ελέγξει την προσβασιμότητα μιας κατάστασης αφού το εργαλείο του επιτρέπει να δηλώσει μια κατάσταση και ελέγχει αν υπάρχει περίπτωση το συγκεκριμένο δίκτυο να οδηγηθεί στη συγκεκριμένη κατάσταση στο μέλλον. Αν όντος υπάρχει, το εργαλείο εμφανίζει την ακολουθία μεταβάσεων μέχρι την συγκεκριμένη κατάσταση και δίνει τη δυνατότητα μετάβασης σε αυτή.

1.2 Δομή Διατριβής

Στο κεφάλαιο 2, αναλύεται το θεωρητικό υπόβαθρο που χρειάζεται για την υλοποίηση του συστήματος καθώς και οι τεχνολογίες που χρησιμοποιήθηκαν. Συγκεκριμένα, αναλύονται οι έννοιες του αναστρέψιμου υπολογισμού, των μοντέλων των Δικτύων Πέτρι, των Αναστρέψιμων δικτύων Πέτρι, των Αναστρέψιμων δικτύων Πέτρι με πολλαπλές μάρκες ιδίου τύπου και της ιδιότητας της προσβασιμότητας.

Στο κεφάλαιο 3, καθορίζονται οι λειτουργικές και μη λειτουργικές απαιτήσεις, οι προδιαγραφές της εφαρμογής και σε ποιους απευθύνεται.

Στο κεφάλαιο 4, παρουσιάζονται οι γραφικές διεπαφές και οι λειτουργίες της εφαρμογής.

Στο κεφάλαιο 5, παρουσιάζεται η αξιολόγηση της ορθότητας και επίδοσης της εφαρμογής.

Τέλος, στο κεφάλαιο 7 παρουσιάζονται τα συμπεράσματα που προέκυψαν καθώς και προτάσεις για μελλοντικές βελτιστοποιήσεις και επεκτάσεις της εφαρμογής.

Κεφάλαιο 2

Θεωρητικό και Τεχνικό Υπόβαθρο

2.1 Θεωρητικό Υπόβαθρο

Το θεωρητικό υπόβαθρο που θα αναλυθεί σε αυτή την υποενότητα είναι οι έννοιες του αναστρέψιμου υπολογισμού, των κλασσικών δικτύων Πέτρι και των αναστρέψιμων δικτύων Πέτρι. Επιπρόσθετα, εξεταστεί το μαθηματικό μοντέλο των αναστρέψιμων δικτύων Πέτρι με πολλαπλές μάρκες ιδίου τύπου, όπου θα παρουσιαστούν τα διάφορα δομικά συστατικά του και ο εμπρόσθιος και αναστρέψιμος υπολογισμός που υποστηρίζει.

2.1.1 Αναστρέψιμος Υπολογισμός

Ο Αναστρέψιμος Υπολογισμός, τα τελευταία χρόνια είναι ένα ενεργό ερευνητικό πεδίο όπου χρησιμοποιείται σε πληθώρα συστημάτων. Η αντιστρεψιμότητα είναι επιθυμητή ιδιότητα σε συστήματα λόγο της χαμηλής υπολογιστικής ισχύς και την ανοχή σε σφάλματα. Ακόμα, είναι καίριας σημασίας τόσο σε βιοχημικά συστήματα [4, 5], όσο και στον κβαντικό υπολογισμό [6].

2.1.1.1 Επεξήγηση έννοιας

Ο αναστρέψιμος υπολογισμός είναι μια μορφή υπολογισμού, όπου μπορεί να εκτελεστεί προς τα εμπρός (forward) αλλά και προς τα πίσω (backward). Κάθε ακολουθία εκτελέσεων που πραγματοποιείται από ένα σύστημα μπορεί στη συνέχεια να εκτελεστεί αναστρέψιμα επιτρέποντας στο σύστημα να ανακτήσει οποιαδήποτε προηγούμενη κατάσταση σε οποιοδήποτε σημείο κατά τη διάρκεια του υπολογισμού [3]. Προϋπόθεση για τον αναστρέψιμο υπολογισμό είναι η 1-1 συσχέτιση ανάμεσα σε είσοδο και έξοδο, δηλαδή από μια έξοδο να μπορεί να προβλεφθεί η μια μοναδική είσοδος.

2.1.1.2 Μορφές αναστρεψιμότητας

Οι επικρατέστερες προσεγγίσεις αναστρεψιμότητας είναι η οπισθοδρόμηση, αιτιώδης αναστρεψιμότητα και αναστρεψιμότητα μη αιτιολογικής σειράς.

2.1.1.2.1 Οπισθοδρόμηση

Η οπισθοδρόμηση αντιστρέφει τις ενέργειες που εκτελέστηκαν ντετερμινιστικά, δηλαδή με ακριβός την αντίθετη σειρά με την οποία εκτελέστηκαν. Αυτός ο τρόπος αναστρεψιμότητας εξασφαλίζει ότι σε κάθε κατάσταση θα υπάρχει το πολύ μια προηγηθείσα κατάσταση.

2.1.1.2.2 Αιτιώδης αναστρεψιμότητα

Η αιτιώδης αναστρεψιμότητα παρέχει μια πιο ευέλικτη μορφή αναστρεψιμότητας, αφού επιτρέπει επαναφορά μιας ενέργειας, εφόσον όλες οι ενέργειες που εξαρτιούνταν από αυτή έχουν ήδη αντιστραφεί. Σε αντίθεση με την οπισθοδρόμηση, δεν είναι απαραίτητο οι ενέργειες να αντιστραφούν με ακριβός την αντίθετη σειρά με την οποία εκτελεστήκαν.

2.1.1.2.3 Αναστρεψιμότητα μη αιτιολογικής σειράς

Η αναστρεψιμότητα μη αιτιολογικής σειράς έχει τη δυνατότητα να αντιστρέψει μια κατάσταση ακόμα και αν οι επιδράσεις της δεν έχουν αναιρεθεί. Επόμενος, με την αναστρεψιμότητα μη αιτιολογικής σειράς υπάρχει η δυνατότητα δημιουργίας καταστάσεων που είναι αδύνατο να επιτευχθούν από οποιαδήποτε ακολουθία εμπρόσθιας εκτέλεσης. Αυτός ο τρόπος αναστρεψιμότητας παρουσιάζεται κυρίως σε βιοχημικά συστήματα.

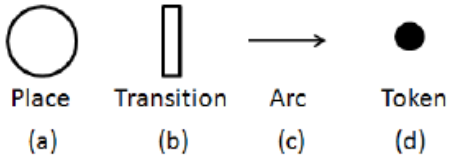
2.1.2 Δίκτυα Πέτρι (Petri Nets)

2.1.2.1 Επεξήγηση έννοιας

Τα δίκτυα Πέτρι [7] επινοήθηκαν από τον Carl Adam Petri και είναι ένα μαθηματικό εργαλείο που χρησιμοποιείται κυρίως για τη μοντελοποίησης και την ανάλυση της συμπεριφοράς παράλληλων και κατανεμημένων συστημάτων. Μπορούν να χρησιμοποιηθούν και ως γραφικά εργαλεία για οπτική απεικόνιση συστημάτων, παρομοίως με ‘flow charts’, ‘block diagrams’, και δίκτυα.

2.1.2.2 Δομικά συστατικά

Τα δομικά συστατικά που αποτελούν ένα δίκτυο Πέτρι είναι: οι θέσεις (places), οι μεταβάσεις (transitions), τα τόξα (arcs) και οι μάρκες (tokens).



Εικόνα 1: Δομικά συστατικά δικτύου Πέτρι [8]

Οι θέσεις, απεικονίζονται γραφικά ως κύκλοι και μπορούν να περιέχουν μία ή περισσότερες μάρκες, και να συνδέεται με μεταβάσεις μέσω ενός τόξου.
Οι μεταβάσεις απεικονίζονται γραφικά ως τετράγωνα ή ορθογώνια και εκφράζουν τις ενέργειες ή διεργασίες που έχει ένα σύστημα. Τα εισερχόμενα τόξα αναπαριστούν τις απαιτήσεις που έχει το σύστημα προκείμενου να εκτελεστεί η συγκεκριμένη διεργασία, ενώ τα εξερχόμενα τόξα αναπαριστούν τα αποτελέσματα που θα προκύψουν με την εκτέλεση της διεργασίας. Ένα τόξο δεν μπορεί να ενώνει δύο ίδιου τύπου κόμβους.
Μια μετάβαση μπορεί να εκτελεστεί μόνο αν πληρούνται η προϋποθέσεις όλων των εισερχόμενων τόξων προς αυτή. Η εκτέλεση της μετάβασης φέρνει ως αποτέλεσμα τη μετακίνηση των μαρκών που απαιτούνται στα εισερχόμενα τόξα, στις θέσεις που υποδεικνύουν τα εξερχόμενα τόξα.

2.1.3 Αναστρέψιμα δίκτυα Πέτρι

2.1.3.1 Επεξήγηση έννοιας

Τα Αναστρέψιμα δίκτυα Πέτρι επεκτείνουν τα Δίκτυα Πέτρι με τη δυνατότητα αναστρεψιμότητας, κάνοντας έτσι εφικτή την επαναφορά σε μια κατάσταση που είχε προηγουμένως το σύστημα. Συγκεκριμένα, υποστηρίζουν τρεις μεθόδους αναστρεψιμότητας: οπισθοδρόμηση, αιτιώδης αναστρεψιμότητα και αναστρεψιμότητα μη αιτιολογικής σειράς.

2.1.3.2 Δομικά συστατικά

Οι θέσεις και οι μεταβάσεις απεικονίζονται και ερμηνεύονται με τον ίδιο τρόπο όπως και στα δίκτυα Πέτρι. Όσο αφορά τις μάρκες, στα αναστρέψιμα δίκτυα Πέτρι η κάθε μάρκα έχει μοναδική ονομασία διαφέροντας από τις υπόλοιπες και δεν μπορεί να καταναλωθεί κατά την εκτέλεση μιας μετάβασης. Μια τέτοια μάρκα ονομάζεται Βάση (base) και απεικονίζεται με ●.

Επιπλέων, επισημαίνεται μία νέα έννοια που σχετίζονται με τις μάρκες. Ονομάζεται Δεσμός (bond), και συνδέει δύο μάρκες μεταξύ τους. Οι δεσμοί μπορούν να δημιουργηθούν ή να καταστραφούν με την πυροδότηση μιας μετάβασης. Απεικονίζονται με μία γραμμή που συνδέει τις δύο μάρκες.

2.1.4 Αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου

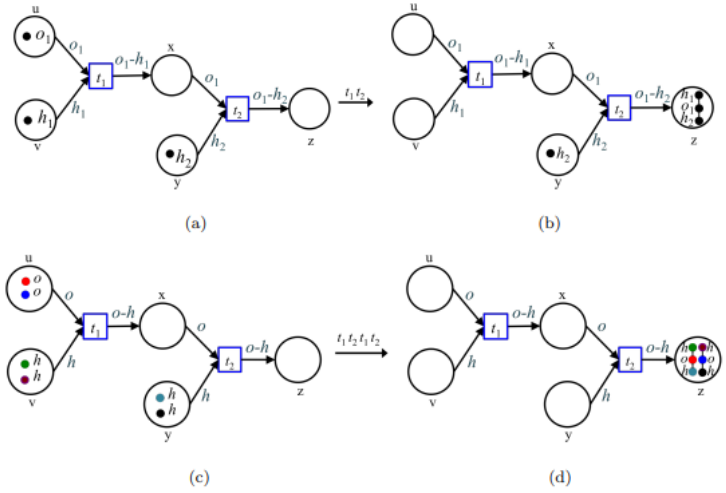
Τα Αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου [1], επεκτείνουν περεταίρω τα Αναστρέψιμα δίκτυα Πέτρι με την προσθήκη πολλαπλών μαρκών ιδίου τύπου τα οποία είναι ισοδύναμα μεταξύ τους. Αυτό επιτρέπει την αντιστροφή μιας μετάβασης με τη χρήση οποιονδήποτε μαρκών ιδίου τύπου με αυτά που απαιτεί η μετάβαση, με αποτέλεσμα μεγαλύτερη ευελιξία στην αναστρεψιμότητα μη αιτιολογικής σειράς.

2.1.4.1 Ορισμός και δομικά συστατικά

Όλα τα δομικά συστατικά απεικονίζονται με τον ίδιο τρόπο με τα Αναστρέψιμα δίκτυα Πέτρι. Η διαφορά όπως προαναφέρθηκε είναι στο ότι οι μάρκες έχουν ένα τύπο και πολλές μάρκες μπορούν να έχουν τον ίδιο τύπο.

Σύμφωνα με αυτή την προσέγγιση, αν μια μετάβαση πυροδοτήσει, είναι δυνατό να αντιστραφεί με οποιοδήποτε σύνολο μαρκών και δεσμών που ικανοποιούν τις απαιτήσεις της μετάβασης. Ακόμα και αν το σύνολο αυτό δεν είναι το ίδιο με αυτό που χρησιμοποιήθηκε κατά εμπρόσθια εκτέλεση.

Με το πιο κάτω σχήμα, παρουσιάζεται γραφικά ένα παράδειγμα για την κατανόηση των διαφορών.



Εικόνα 2: Παράδειγμα κατανόησης μαρκών ιδίου τύπου [1]

Στο πιο πάνω σχήμα απεικονίζεται η γνωστή εξίσωση του νερού. Συγκεκριμένα το (a) και (b) είναι Αναστρέψιμο δίκτυο Πέτρι. Η κάθε μάρκα έχει τη δική της μοναδική ονομασία και σε κάθε μετάβαση απαιτούνται συγκεκριμένες μάρκες, που μόνο αυτές

μπορούν να ικανοποιήσουν τις προϋποθέσεις. Το (b) σχήμα, δείχνει το δίκτυο με την εκτέλεση των μεταβάσεων t1 και t2, όπου δημιουργούνται οι δεσμοί o1-h1 και o1-h2. Για τη δημιουργία ακόμα ένα μόριο νερού, απαιτείται η προσθήκη τριών μαρκών (h3, h4, o2) και η κλωνοποίηση των μεταβάσεων με τα ονόματα των νέων μαρκών στα τόξα τους. Με την αντιστροφή της μετάβασης t2, θα σπάσει ο δεσμός o1-h2, το h2 θε επιστρέψει στη θέση y και ο δεσμός o1-h1 στη θέση x.

Αντίθετα, το (c) και (d) είναι Αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου. Υπάρχουν 2 μάρκες οξυγόνου και 4 μάρκες υδρογόνου, επιτρέποντας με την εκτέλεση των μεταβάσεων t1 και t2, από δύο φορές, τη δημιουργία δύο μορίων νερού, όπως φαίνεται στο σχήμα (d). Με την αντιστροφή της μετάβασης t2, παρατηρείται ότι μπορεί να σπάσει οποιοσδήποτε δεσμός στο δίκτυο, ακόμα και δεσμός που δημιουργήθηκε κατά τη μετάβαση t1.

Στο σημείο αυτό, θα δοθεί ο μαθηματικός ορισμός του αναστρέψιμου δικτύου Πέτρι όπως είναι καθορισμένος στο άρθρο των Φιλίππου και Ψαρά [1]:
Ένα αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου (MRPN) ορίζεται ως μια πλειάδα (P, T, A, A_V, B, F) όπου:

1. P είναι ένα πεπερασμένο σύνολο θέσεων και T είναι ένα πεπερασμένο σύνολο μεταβάσεων.
2. A είναι ένα πεπερασμένο σύνολο τύπων βάσεων ή μαρκών που ορίζονται με a, b, . . .
3. A_V είναι ένα πεπερασμένο σύνολο μεταβλητών. Γράφουμε type(v) για τον τύπο της μεταβλητής v και υποθέτουμε ότι $\text{type}(v) \in A$ για κάθε $v \in A_V$.
4. $B \subseteq A \times A$ είναι ένα πεπερασμένο σύνολο μη κατευθυνόμενων δεσμών που ορίζονται με β, γ, . . . Χρησιμοποιούμε το συμβολισμό $a{-}b$ για ένα δεσμό $(a, b) \in B$.
5. $F : (P \times T \cup T \times P) \rightarrow P(A_V \cup (A_V \times A_V))$ ορίζει ένα σύνολο από κατευθυνόμενα τόξα με ετικέτα, όπου το κάθε ένα συνδέεται με ένα υποσύνολο από $A_V \cup (A_V \times A_V)$), όπου $(u, v) \in F(x, y)$ υποδηλώνει ότι $u, v \in F(x, y)$. Επιπλέον, για όλα τα $t \in T$, $F(t, x) \cap F(t, y) = \emptyset$.

Τα κατευθυνόμενα τόξα με ετικέτα ενώνουν τις θέσεις με τις μεταβάσεις και το αντίστροφο. Η ετικέτα των τόξων εκφράζει τις απαιτήσεις και τα αποτελέσματα των μεταβάσεων με βάση τους τύπους των μαρκών. Κάθε σύνολο μαρκών με ίδιο τύπο και

τυχών δεσμούς με τις μεταβλητές των ετικετών μπορούν να συμμετέχουν στη μετάβαση. Όταν ένας δεσμός αναγράφεται στο τόξο εννοείται ότι και οι βάσεις που αποτελούν το δεσμό περιέχονται στο τόξο. Να σημειωθεί ότι οι μεταβλητές των μαρκών στις ετικέτες $u \in F(x, t) \cap A_V$ του τύπου type(u) = a συμβολίζονται με u : a στο αντίστοιχο τόξο F(x, t).

Ακόμα, ορίζοντα τα σύμβολα $\bullet t = \{x \in P \mid F(x, t) \neq \emptyset\}$ και $t\bullet = \{x \in P \mid F(t, x) \neq \emptyset\}$ για τις εισερχόμενες και εξερχόμενες θέσεις της μετάβασης t αντίστοιχα. Επιπλέων, ορίζονται τις έννοιες pre και post οι οποίες εκφράζουν την ένωση των ετικετών που βρίσκονται στα εισερχόμενα τόξα και στα εξερχόμενα τόξα της μετάβασης t αντίστοιχα.

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

Η κατανομή των δεσμών και των βάσεων στις διάφορες θέσεις ορίζεται ως μαρκάρισμα (marking). Για να είναι δυνατός ο ορισμός του μαρκαρίσματος χρειάζονται ακόμα δύο μηχανισμοί. Οι εμφανίσεις μαρκών ιδίου τύπου ξεχωρίζονται σαν a_i για την $i^{οστή}$ εμφάνιση μάρκας τύπου a. Έτσι, γράφεται A_I για το σύνολο όλων των εμφανίσεων των μαρκών και $B_I = A_I \times A_I$ για το σύνολο των δεσμών τους.

Έτσι, το μαρκάρισμα ορίζεται ως $M : P \rightarrow 2^{A_I \cup B_I}$. Επιπλέον, κάθε μετάβαση, έχει την δική της ιστορία (history), $H : T \rightarrow \mathbb{N}$. Όταν η μετάβαση έχει ως ιστορία H(t) = 0, σημαίνει ότι η συγκεκριμένη μετάβαση δεν έχει εκτελεστεί, ενώ όταν η ιστορία της είναι H(t) = k, k > 0, σημαίνει ότι η συγκεκριμένη μετάβαση έχει εκτελεστεί εμπρόσθια k φορές. Η κατάσταση (state) ενός MRPN, περιγράφεται από τι δυάδα <M, H>.

2.1.4.2 Καλά ορισμένο (Well-formed)

Ένα Multi Reversing Petri Net θεωρείται καλά ορισμένο αν για όλες τις μεταβάσεις $t \in T$ ισχύει ότι:

1. $A_V \cap pre(t) = A_V \cap post(t)$, και
2. $F(t, x) \cap F(t, y) \cap A_V = \emptyset$ για κάθε $x, y \in P, x \neq y$.

Η πρόταση (1) εκφράζει ότι οι μεταβλητές που αναγράφονται στα εισερχόμενα τόξα της μετάβασης πρέπει να αναγράφονται και στα εξερχόμενα τόξα της μετάβασης, που υπονοεί ότι δεν μπορούν να χάνονται μάρκες κατά τη μετάβαση. Η πρόταση (2) εκφράζει

ότι οι μάρκες/δεσμοί δεν μπορούν να κλωνοποιηθούν σε περισσότερες από μία εξερχόμενες θέσεις.

2.1.4.3 Μέθοδοι εκτέλεσης

Αρχικά, ορίζεται ο όρος con(α_i, C), όπου $\alpha_i \in A_I$ και $A_I \cup B_I$, ο οποίος είναι οι συνδεδεμένες μάρκες στο α_i καθώς και οι δεσμοί που δημιουργούν αυτές τις συνδέσεις, συμφώνα με το σύνολο C, όπως και στα RPNs [3]:

$con(a, C) = (\{a\} \cap C) \cup \{x \mid \exists w \text{ s.t. } path(a, w, C), (b, c) \in w, x \in \{(b, c), b, c\}\}$

Όπου $path(a, w, C)$ *αν* $w = <\beta_1, \ldots, \beta_n>$, *και για κάθε* $1 \leq i \leq n$, $\beta_i = (a_i - 1, a_i) \in C \cap B_I$, $a_i \in C \cap A_I$, *και* $a_0 = a$.

Ακόμα, ορίζονται οι δεσμοί οι οποίοι δημιουργούνται ως αυτοί που βρίσκονται στα εξερχόμενα τόξα και όχι στα εισερχόμενα, και οι δεσμοί οι οποίοι καταστρέφονται ως αυτοί που βρίσκονται στα εισερχόμενα τόξα και όχι στα εξερχόμενα. Ορίζονται αντίστοιχα:

$$\text{eff}^{+}(t) = \text{post}(t) - \text{pre}(t)$$

$$\text{eff}^{-}(t) = \text{pre}(t) - \text{post}(t)$$

Τέλος, ορίζεται η συνάρτηση $S : V \rightarrow A_I$, όπου $V \subseteq A_V$, και λέγεται type-respecting assignment αν για κάθε $v \in V$, αν $S(v) = a_i$ τότε $type(v) = a$.

Επεκτείνοντας το πιο πάνω, γράφεται S(a, b) για (S(a), S(b)) και, δεδομένου ενός συνόλου $L \subseteq A_V \cup (A_V \times A_V)$, γράφεται $S(L) = \{S(x) \mid x \in L\}$.

2.1.4.3.1 Εμπρόσθια Εκτέλεση

Μια μετάβαση $t \in T$ μπορεί να εκτελεστεί μπρος τα εμπρός μόνο αν υπάρχει ένα type respecting assignment $S : pre(t) \cap A_V \rightarrow A_I$, τέτοιο ώστε:

1. $S(F(x, t)) \subseteq M(x)$ για κάθε $x \in \bullet t$
2. Αν $u, v \in F(x, t)$ και $S(u, v) \in M(x)$, για κάποια $x \in \bullet t$, τότε $(u, v) \in F(x, t)$
3. Αν $u \in F(t, y1)$ και $v \in F(t, y2)$, $y1 \neq y2$, τότε $con(S(u), comp_f(t, S, M)) \neq con(S(v), comp_f(t, S, M))$.

Όπου $comp_f$ (t, S, M) = ($\bigcup_{x \in \bullet t}$ M(x) ∪ S(eff$^+$(t))) − S(eff$^-$(t)).

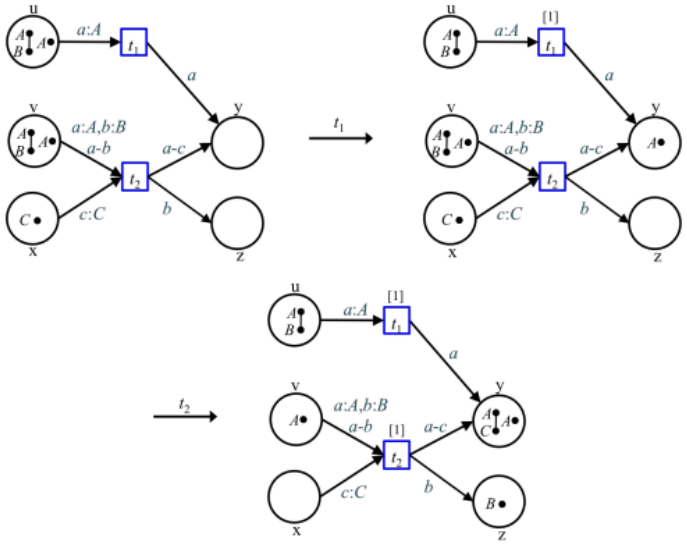
Μια μετάβαση t μπορεί να εκτελεστεί μπρος τα εμπρός όταν (1) οι μάρκες και οι δεσμοί που απαιτούνται από τα εισερχόμενα τόξα της μετάβασης, συμφώνα με το S, είναι διαθέσιμα στις κατάλληλες θέσεις. (2) Αν οι μάρκες που επιλέχθηκαν να μεταφερθούν από τη μετάβαση έχουν δεσμό μετάξι τους, τότε ο δεσμός αυτός πρέπει να υπάρχει και στην ετικέτα του συγκεκριμένου εισερχόμενου τόξου. (3) Αν δύο μάρκες μεταφέρθηκαν από μια μετάβαση σε διαφορετικές εξερχόμενες θέσεις, τότε αυτές οι μάρκες, σε περίπτωση που είχαν δεσμό, δεν πρέπει να συνεχίσουν να είναι συνδεδεμένες (αυτό είναι απαραίτητο για να μην κλωνοποιούνται οι μάρκες).

Όταν εκτελείται μια μετάβαση t με εμπρόσθια εκτέλεση τότε $\langle M, H \rangle \xrightarrow{t} \langle M', H' \rangle$ όπου:

$$M'(x) = \begin{cases} M(x) - \bigcup_{u \in F(x,t)} con(S(u), M(x)) & Αν\ x \in \bullet t \\ M(x) \cup \bigcup_{u \in F(t,x)} con(S(u), compf(t,S,M)) & Αν\ x \in t \bullet \\ M(x), & διαφορετικα \end{cases}$$

και

$$H'(t') = \begin{cases} H(t') + 1, & αν\ t' = t \\ H(t'), & διαφορετικά \end{cases}$$



Εικόνα 3: Παράδειγμα εμπρόσθιας εκτέλεσης [1]

Μετά την εκτέλεση της μετάβασης t, οι επιλεγμένες μάρκες μαζί με τους δεσμούς τους μεταφέρονται στις εξερχόμενες θέσεις της μετάβασης, όπως αυτές ορίζονται από τις

ετικέτες, με τους κατάλληλους δεσμούς να δημιουργούνται και να καταστρέφονται. Ακόμα, η ιστορία της μετάβασης αυξάνεται κατά ένα.
Με το πιο πάνω σχήμα, παρουσιάζεται γραφικά ένα παράδειγμα για την κατανόηση της εμπρόσθιας εκτέλεσης:

Το παράδειγμα παρουσιάζει την εμπρόσθια εκτέλεση των μεταβάσεων t1 και t2. Κατά τη διάρκεια της εκτέλεσης της μετάβασης t1, και υποθέτοντας ότι M(u) = {A1, A2, B1,(A1, B1)}, χρησιμοποιείται το forward-enabling assignment S(a) = A2. Έτσι, η συγκεκριμένη μάρκα μεταφέρεται στη θέση y. Κατά τη διάρκεια της εκτέλεσης της μετάβασης t2, και υποθέτοντας ότι M΄(v) = {A3, A4, B2,(A3, B2)}, M΄(x) = {C1}, χρησιμοποιείται το forward-enabling assignment S΄(a) = A3, S΄(b) = B2 και S΄(c) = C3. Ως αποτέλεσμα, δημιουργείται δεσμός ανάμεσα στο A3 και C1 στη θέση y, ο δεσμός ανάμεσα στο A3 και B2 σπάει και το B2 μεταφέρεται στη θέση z.

2.1.4.3.2 Αντίστροφη Εκτέλεση

Μια μετάβαση μπορεί να αντιστραφεί σε μια συγκεκριμένη κατάσταση, αν έχει ιδεί εκτελεστεί προηγούμενος και υπάρχουν οι κατάλληλες μάρκες και δεσμοί στις εξερχόμενες θέσεις της.

Μια μετάβαση $t \in T$ μπορεί να εκτελεστεί μπρος τα πίσω μόνο αν υπάρχει ένα type respecting assignment R : post(t) ∩ AV → AI, τέτοιο ώστε:

1. R(F(t, x)) ⊆ M(x) για κάθε x ∈ t•.
2. Αν u, v ∈ F(t, x) και R(u, v) ∈ M(x) τότε (u, v) ∈ F(t, x), και
3. Αν u ∈ F(y1, t) και v ∈ F(y2, t), y1 ≠ y2 τότε con(R(u), $comp_r$ (t, R, M)) ≠ con(R(v), $comp_r$ (t, R, M)).
   Όπου $comp_r$ (t, R, M) = ($\bigcup_{x \in t\bullet}$ M(x) ∪ R(eff^-(t))) − R(eff^+(t)).

Μια μετάβαση t μπορεί να εκτελεστεί μπρος τα πίσω όταν (1) οι μάρκες και οι δεσμοί που απαιτούνται από τα εξερχόμενα τόξα της μετάβασης, συμφώνα με το R, είναι διαθέσιμα στις κατάλληλες θέσεις. (2) Αν οι μάρκες που επιλέχθηκαν να μεταφερθούν από τη μετάβαση έχουν δεσμό μετάξι τους, τότε ο δεσμός αυτός πρέπει να υπάρχει και στην ετικέτα του συγκεκριμένου εξερχόμενου τόξου. (3) Αν δύο μάρκες μεταφέρθηκαν από μια μετάβαση σε διαφορετικές εισερχόμενες θέσεις, τότε αυτές οι μάρκες, σε

περίπτωση που είχαν δεσμό, δεν πρέπει να συνεχίσουν να είναι συνδεδεμένες (αυτό είναι απαραίτητο για να μην κλωνοποιούνται οι μάρκες).

Όταν εκτελείται μια μετάβαση t με αντίστροφη εκτέλεση τότε $\langle M, H\rangle \overset{t}{\rightsquigarrow} \langle M', H'\rangle$ όπου:

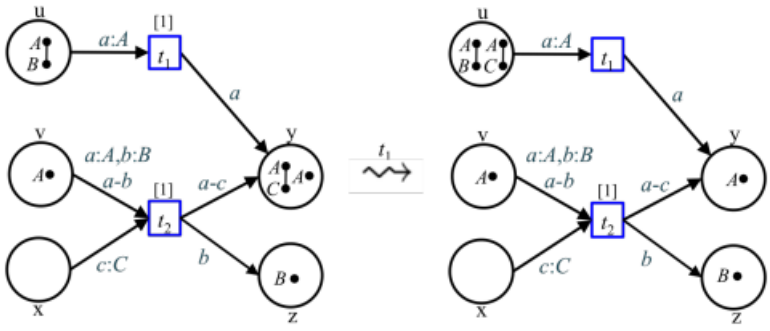
$$M'(x) = \begin{cases} M(x) - \bigcup\limits_{u \in F(t,x)} con\big(R(u), M(x)\big) & Αν\ x \in t\bullet \\ M(x) \cup \bigcup\limits_{u \in F(x,t)} con(R(u), compr(t, R, M)) & Αν\ x \in \bullet t \\ M(x), & διαφορετικα \end{cases}$$

και

$$H'(t') = \begin{cases} H(t') - 1, & αν\ t' = t \\ H(t'), & διαφορετικά \end{cases}$$

Μετά την εκτέλεση της μετάβασης t, οι επιλεγμένες μάρκες μαζί με τους δεσμούς τους μεταφέρονται στις εισερχόμενες θέσεις της μετάβασης, όπως αυτές ορίζονται από τις ετικέτες, με τους κατάλληλους δεσμούς να δημιουργούνται και να καταστρέφονται. Ακόμα, η ιστορία της μετάβασης μειώνεται κατά ένα.

Με το πιο κάτω σχήμα, παρουσιάζεται γραφικά ένα παράδειγμα για την κατανόηση της εμπρόσθιας εκτέλεσης:



Εικόνα 4: Παράδειγμα αντίστροφης εκτέλεσης [1]

Το παράδειγμα παρουσιάζει την αντίστροφη εκτέλεση της μετάβασης t1. Κατά τη διάρκεια της εκτέλεσης της μετάβασης t1, και υποθέτοντας ότι M(y) = {A3, A4, C1,(A3, C1)}. Σε αυτό το παράδειγμα και το A3 και το A4 ικανοποιούν τις προϋποθέσεις, έτσι μπορούν να επιλεχθούν και τα δύο. Χρησιμοποιείται το reverse-enabling assignment R(a) = A3. Έτσι, η συγκεκριμένη μάρκα, μαζί με τη συνδεδεμένη μάρκα C1, μεταφέρονται στη θέση u.

2.1.4.4 Προσβασιμότητα (reachability)

Ο συμβολισμός ↦, απεικονίζει το συνδυασμό εμπρόσθιας και αντίστροφης εκτέλεσης (→ ∪ ⇝). Η προσβασιμότητα μελετά το αν το δίκτυο μπορεί με μια ακολουθία εκτελέσεων μεταβάσεων να φτάσει σε ένα συγκεκριμένο μαρκάρισμα στο μέλλον.

Ορισμός:

Δεδομένου ενός MRPN (A, P, B, T, F) και μια αρχική κατάσταση <M0, H0>, λέμε ότι μια κατάσταση <M, H> είναι πρόσβαση, αν υπάρχουν καταστάσεις <Mi , Hi>, i ≤ n για κάποιο n ≥ 0 τέτοιες ώστε <M0, H0> $\overset{t1}{\mapsto}$ <M1, H1> $\overset{t2}{\mapsto}$ . . . $\overset{tn}{\mapsto}$ <Mn, Hn> = <M, H>.

2.2 Τεχνικό Υπόβαθρο

Το τεχνικό υπόβαθρο που θα αναλυθεί σε αυτή την υποενότητα παρουσιάζονται όλες οι τεχνολογίες που χρησιμοποιηθήκαν.

2.2.1 Java

Η προγραμματιστική γλώσσα Java, είναι μια αντικειμενοστραφής γλώσσα υψηλού επιπέδου, η οποία κυκλοφόρησε για πρώτη φορά από την Sun Microsystems το 1995. Μέχρι και σήμερα, είναι μια από τις δημοφιλέστερες γλώσσες προγραμματισμού, όπου παραχωρεί μια αξιόπιστη πλατφόρμα πάνω στην οποία χτίζονται πολλές υπηρεσίες και εφαρμογές [9].

Τα δύο κύρια στοιχεία της πλατφόρμας Java είναι το Java API, το οποίο είναι μια βιβλιοθήκη εντολών Java και η εικονική μηχανή Java (JVM) που ερμηνεύει τον κώδικα Java σε γλώσσα μηχανής. Η ίδια εφαρμογή μπορεί να εκτελεστεί σε πολλές πλατφόρμες, αφού το API και το JVM καθιστούν το πρόγραμμα ανεξάρτητο από το υποκείμενο υλικό [10].

2.2.2 Eclipse IDE

Το Eclipse IDE είναι διάσημο για το Java Integrated Development Environment (IDE) αλλά υπάρχει και δυνατότητα συνδυασμού πολλών γλωσσών, καθώς και επέκταση με οποιοδήποτε από τα προεπιλεγμένα πακέτα του Eclipse Marketplace ή ακόμα και custom πακέτα [11].

2.2.3 Maven

Το Maven είναι μέρος του Apache Software Foundation. Το Apache Maven είναι ένα εργαλείο διαχείρισης και κατανόησης μεγάλων software projects. Βασισμένο στην έννοια ενός project object model (POM), το Maven μπορεί να διαχειριστεί την κατασκευή, την αναφορά και την τεκμηρίωση ενός project. Με λίγα λόγια, προσφέρει έναν τυπικό τρόπο κατασκευής των projects, έναν σαφή ορισμό του τι αποτελεί αυτά τα projects, έναν εύκολο τρόπο δημοσίευσης πληροφοριών τους και έναν τρόπο κοινής χρήσης JAR σε πολλά projects [12].

2.2.4 JavaFX

Το JavaFX είναι ένα σύνολο πακέτων γραφικών και πολυμέσων που επιτρέπει στους προγραμματιστές να σχεδιάζουν, να δημιουργούν, να ελέγχουν, να διορθώνουν και να αναπτύσσουν εφαρμογές που λειτουργούν με συνέπεια σε διάφορες πλατφόρμες. Δίνει τη δυνατότητα του διαχωρισμού του front-end (UI) και της λογικής back-end. Η ανάπτυξη του front-end μπορεί να γίνει εύκολα στη FXML scripting language και η λογική back-end με χρήση κώδικα Java. Ακόμα, μπορεί να χρησιμοποιηθεί Cascading Style Sheets (CSS) για την προσαρμογή της εμφάνισης και του στυλ της εφαρμογής [13]. Για τη δημιουργία ενός JavaFX project στην Eclipse απαιτείται η εγκατάσταση του e(fx)clipse plugin.

2.2.5 Scene Builder

Το JavaFX Scene Builder είναι ένα εργαλείο οπτικής διάταξης που επιτρέπει στους χρήστες να σχεδιάζουν γρήγορα JavaFX User Interfaces, χωρίς να χρειάζονται να γράψουν κώδικα. Οι χρήστες έχουν τη δυνατότητα να κάνουν drag and drop τα διάφορα συστατικά, να τροποποιήσουν τις ιδιότητές τους και να εφαρμόσουν style sheets. Το αποτέλεσμα είναι ένα αρχείο FXML, το οποίο παράγεται αυτόματα από το Scene Build, που μπορεί στη συνέχεια να συνδυαστεί με ένα πρόγραμμα Java συνδέοντας το front-end και το back-end [14].

Κεφάλαιο 3

Ανάλυση Απαιτήσεων – Προδιαγραφών

Στο κεφάλαιο αυτό, θα παρουσιαστούν τα απαραίτητα δεδομένα με σκοπό τον καθορισμό των αιτήσεων των χρηστών.

3.1 Σκοπός του συστήματος

Αυτό το σύστημα, επιτρέπει στους χρήστες να δημιουργούν, να τροποποιούν και να αποθηκεύουν Αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου. Ακόμα, θα υπάρχει η δυνατότητα προσομοίωσης των δικτύων αυτών με εμπρόσθια ή αντίστροφη εκτέλεση οποιασδήποτε μετάβασης, καθώς και έλεγχος προσβασιμότητας κατάστασης, δηλαδή, κατά πόσο το δίκτυο μπορεί να βρεθεί σε μια συγκεκριμένη κατάσταση στο μέλλον.

3.2 Κατηγορίες Χρηστών

Η κατηγορία χρηστών του συγκεκριμένου συστήματος είναι κυρίως ερευνητές οι οποίοι θα το χρησιμοποιήσουν προκειμένου να μελετήσουν τη συμπεριφορά των Αναστρέψιμων δικτύων Πέτρι με πολλαπλές μάρκες ιδίου τύπου και να μοντελοποιήσουν συστήματα που μπορούν να αναπαρασταθούν με το μοντέλο αυτό.

3.3 Λειτουργικές Απαιτήσεις

Οι λειτουργικές απαιτήσεις είναι οι λειτουργίες που θα πρέπει να υποστηρίζει το σύστημα, δηλαδή τις λειτουργίες που θα μπορεί να διεκπεραιώσει ο χρήστης. Θα χωριστούν και θα αναλυθούν στις δύο μεγάλες υποκατηγορίες του συστήματος, στη δημιουργία και στη προσομοίωση των Αναστρέψιμων δικτύων Πέτρι με πολλαπλές μάρκες ιδίου τύπου.

Λειτουργικές Απαιτήσεις για τη δημιουργία Αναστρέψιμων δικτύων Πέτρι με πολλαπλές μάρκες ιδίου τύπου

1. Ο χρήστης θα έχει τη δυνατότητα να εισάγει θέσεις, μάρκες, δεσμούς ανάμεσα σε δύο μάρκες που βρίσκομαι στην ίδια θέση, μεταβάσεις, και τόξα, με τα οποία θα μπορεί να ενώνει μια θέση με μια μετάβαση και αντίστροφα.

2. Ο χρήστης θα μπορεί να αφαιρέσει οποιοδήποτε από τα πιο πάνω δομικά στοιχεία.
3. Τα δομικά στοιχεία που θα προθέτονται από τον χρήστη, θα ονομάζονται αυτόματα από το σύστημα και στις μάρκες θα δίνεται αυτόματα ένας προκαθορισμένος τύπος.
4. Ο χρήστης θα έχει τη δυνατότητα να μετονομάσει τις θέσεις και τις μεταβάσεις.
5. Ο χρήστης θα μπορεί να αλλάξει τύπο στις μάρκες.
6. Ο χρήστης θα έχει τη δυνατότητα να μετακινήσει τις θέσεις και τις μεταβάσεις στο χώρο.
7. Ο χρήστης θα μπορεί να προσθέσει ετικέτες στα τόξα, οι οποίες θα περιλαμβάνουν όσες μεταβλητές επιθυμεί, με τύπους οι οποίοι υπάρχουν ήδη στο δίκτυο, και δεσμούς ανάμεσα σε αυτές τις μεταβλητές.
8. Αντίστοιχα ο χρήστης θα μπορεί να αφαιρέσει τις πιο πάνω ετικέτες.
9. Ο χρήστης θα μπορεί να αποθηκεύει στη συσκευή στην οποία τρέχει το σύστημα, τα δίκτυα που δημιουργεί.
10. Ο χρήστης θα μπορεί να ανοίγει XML αρχεία που αναπαριστούν Αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου και έχουν δημιουργηθεί με το συγκεκριμένο λογισμικό.
11. Ο χρήστης θα μπορεί να ελέγχει την εγκυρότητα των δικτύων.

Λειτουργικές Απαιτήσεις για την προσομοίωση Αναστρέψιμων δικτύων Πέτρι με πολλαπλές μάρκες ιδίου τύπου

1. Το σύστημα θα παρουσιάζει όλες τις διαθέσιμες μεταβάσεις για εμπρόσθια και αντίστροφη εκτέλεση.
2. Ο χρήστης θα μπορεί να επιλέξει πιο από αυτές τις μεταβάσεις θέλει να εκτελέσει.
3. Ο χρήστης θα μπορεί να δηλώσει ένα μαρκάρισμα του δικτύου για να γίνει ο έλεγχος αν αυτό το μαρκάρισμα είναι προσβάσιμο.
4. Μετά τον έλεγχο προσβασιμότητας, το σύστημα θα παρουσιάζει στο χρήστη την ακολουθία μεταβάσεων για να φτάσει στο επιθυμητό μαρκάρισμα, αν υπάρχει, και θα δίνεται η δυνατότητα εκτελέσεις των συγκεκριμένων μεταβάσεων.
5. Ο χρήστης θα μπορεί να επαναφέρει το δίκτυο στην αρχική του κατάσταση.

3.4 Μη Λειτουργικές Απαιτήσεις

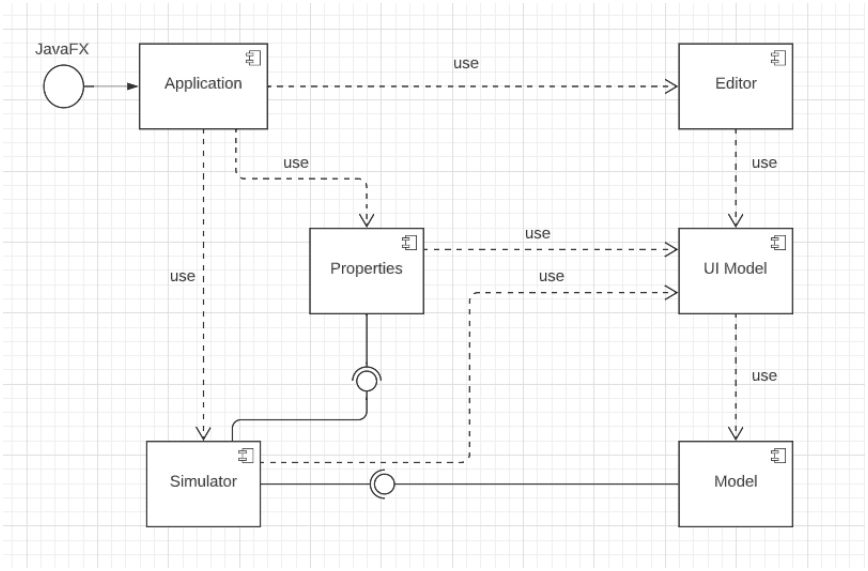
Οι μη λειτουργικές απαιτήσεις παρουσιάζουν τα ποιοτικά χαρακτηριστικά που πρέπει να έχει το σύστημα, δηλαδή το πόσο αποδοτικά το σύστημα υποστηρίζει τις λειτουργικές αποτίσεις.

1. Το σύστημα θα πρέπει να είναι εύχρηστο. Θα πρέπει ο χρήστης να μπορεί να το χρησιμοποιήσει χωρίς να χρειάζεται οδηγίες και να μπορεί να κατανοήσει αμέσως τις λειτουργίες του κάθε κουμπιού ή παραθύρου.
2. Το σύστημα θα πρέπει να ανταποκρίνεται σε πραγματικό χρόνο στις διάφορες λειτουργίες.
3. Το σύστημα θα πρέπει να είναι αξιόπιστο και συνεπές, διεκπεραιώνοντας τις λειτουργίες ορθά.

3.5 Περιορισμοί

1. Μπορούν να ανοιχτούν αρχεία μόνο από αυτό το εργαλείο, λόγο των συντεταγμένων που απαιτούνται για τα αντικείμενα του.
2. Το εργαλείο προσμοιάζει μόνο Αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου.
3. Το σύστημα είναι μόνο desktop εφαρμογή και απαιτείται εγκατάσταση του κώδικα πριν τη χρίση του.

3.6 Αρχιτεκτονική συστήματος



Εικόνα 5: Αρχιτεκτονική συστήματος

Στην πιο πάνω εικόνα φαίνονται τα συστατικά που αποτελούν το σύστημα και τις σχέσεις μεταξύ τους. Τα συστατικά αυτά είναι τα Application, Editor, Simulator, Properties, Model και UI Model. Το Application είναι η γραφική διεπαφή του συστήματος, μέσο της οποίας μπορούν να πραγματοποιηθούν οι τρείς βασικές λειτουργίες.

Η πρώτη είναι η δημιουργία και τροποποίηση των δικτύων που εκφράζεται από το συστατικό Editor.

Η δεύτερη είναι η προσομοίωση και εκτέλεση των επιτρεπόμενων μεταβάσεων του δικτύου που εκφράζεται από το συστατικό Simulator.

Η τρίτη είναι η ανάλυση των διάφορων ιδιοτήτων, στην παρούσα φάση η μόνη υλοποιημένη ιδιότητα είναι αυτή της προσβασιμότητας, που εκφράζεται από το συστατικό Properties.

Και οι τρείς βασικές λειτουργίες επικοινωνούν με το συστατικό UI Model, το οποίο είναι υπεύθυνο για τη γραφική αναπαράσταση των διαφόρων κόμβων του δικτύου.

Για να αναπαραστήσει γραφικά τους κόμβους αυτούς επικοινωνεί με το συστατικό Model, το οποίο εκφράζει τις πληροφορίες που χρειάζονται για τους κόμβους.

Ακόμα, το συστατικό Simulator για να υλοποιήσει τη λειτουργία της προσομοίωσης χρησιμοποιεί το συστατικό Model για να μπορεί να καθορίσει τις επιτρεπόμενες μεταβάσεις προς εκτέλεση.

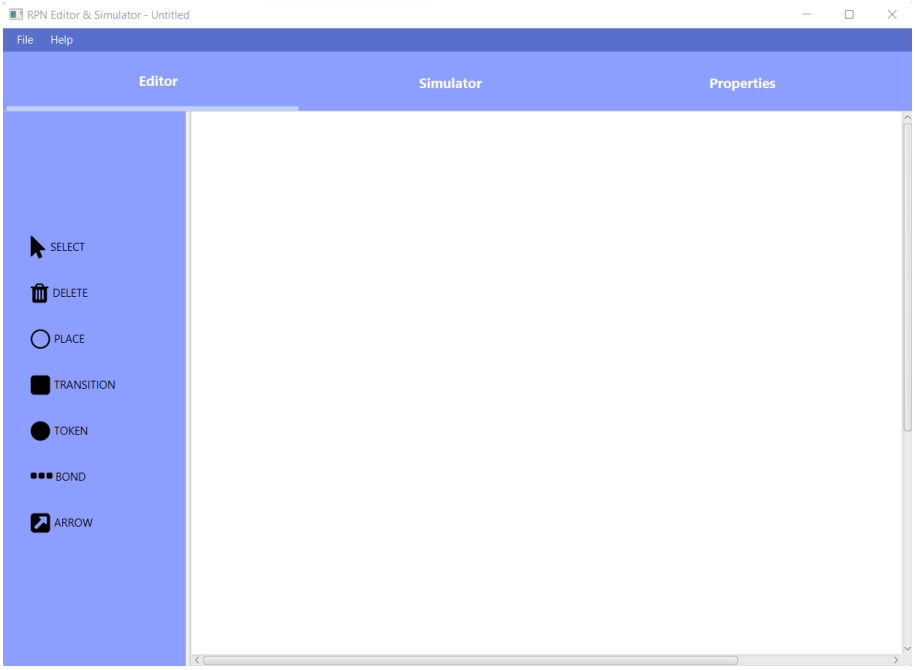
Τέλος, το συστατικό Properties για να υλοποιήσει τη λειτουργία της προσβασιμότητας, χρησιμοποιεί το συστατικό Simulator για την εκτέλεση των μεταβάσεων.

Κεφάλαιο 4

Σχεδίαση και Υλοποίηση Εφαρμογής

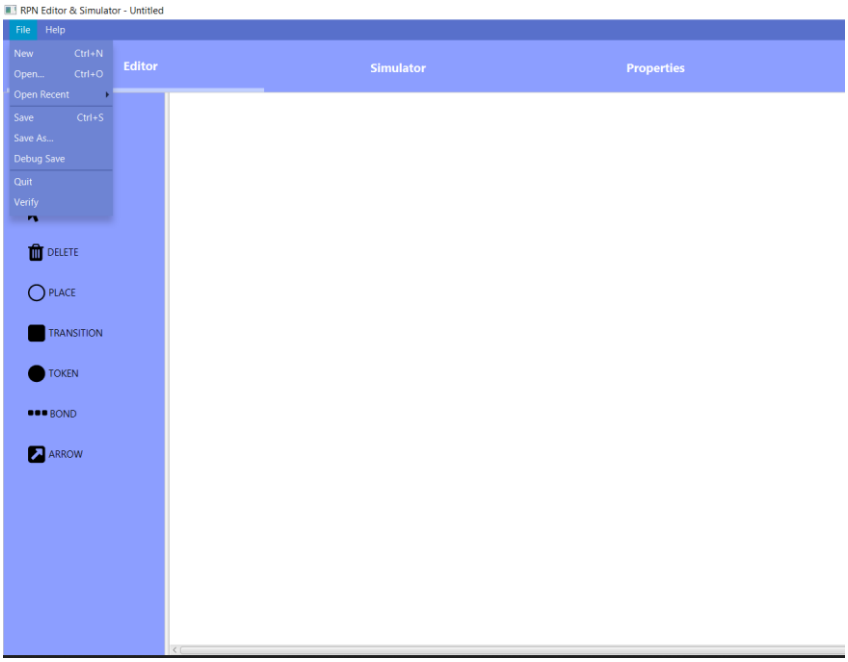
4.1 Αρχική Οθόνη Εφαρμογής

Με το που ξεκινά η εφαρμογή, εμφανίζεται η πιο κάτω οθόνη. Στο πάνω μέρος, υπάρχουν τρία tabs: το Editor tab, το Simulator tab και το Properties tab. Φαίνεται ότι Editor tab είναι προεπιλεγμένο.



Εικόνα 6: Αρχική οθόνη

Στο πάνω αριστερά μέρος της οθόνης υπάρχει το μενού, όπου πατώντας το κουμπί File εμφανίζει επιλογές για άνοιγμα νέου δικτύου, άνοιγμα αποθηκευμένου αρχείου, αποθήκευσης υπάρχοντος δικτύου και έξοδο από την εφαρμογή.

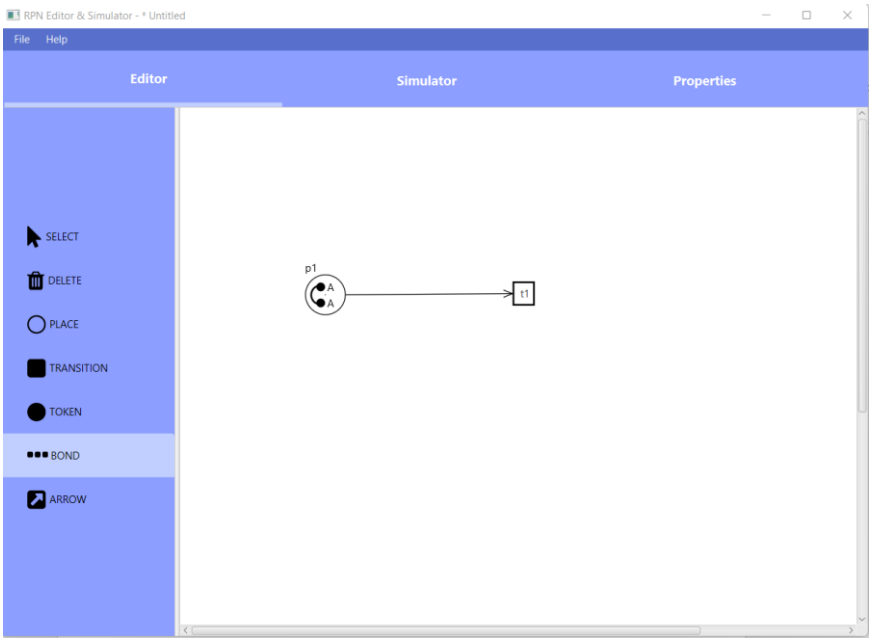


Εικόνα 7: File μενού

4.2 Οθόνη Editor Tab

Στο Editor tab, ο χρήστης μπορεί να κατασκευάσει το δικό του Αναστρέψιμο δίκτυο Πέτρι με πολλαπλές μάρκες ιδίου τύπου. Στο αριστερό μέρος της πιο κάτω οθόνης φαίνονται οι επιλογές που έχει ο χρήστης. Για να προσθέσει ένα αντικείμενο, αρχικά πρέπει να κάνει click σε μία από αυτές τις επιλογές και έπειτα να κάνει click στο σημείο που θέλει να τοποθετήσει αυτό το αντικείμενο, στον άσπρο καμβά. Εκτός από τα αντικείμενα που μπορεί να προσθέσει, υπάρχει η επιλογή Select, όπου ο χρήστης μπορεί να επιλέγει υπάρχοντα αντικείμενα στον καμβά για να τα μετακινεί, και η επιλογή Delete, όπου ο χρήστης μπορεί να διαγράφει υπάρχοντα αντικείμενα στον καμβά κάνοντας click πάνω τους. Στην περίπτωση επιλογής Place (θέση) και Transition (μετάβαση), ο χρήστης με ένα click σε κάποιο κενό σημείο στον καμβά θα προσθέτει το αντίστοιχο στοιχείο. Στην περίπτωση επιλογής Token (μάρκα), για να προστεθεί μια μάρκα στο δίκτυο, ο χρήστης πρέπει να κάνει click πάνω σε κάποια μετάβαση. Στην περίπτωση επιλογής Bond (δεσμός), για να προστεθεί ένας δεσμός στο δίκτυο, ο χρήστης πρέπει να κάνει το πρώτο click πάνω σε κάποια μάρκα και δεύτερο click πάνω σε διαφορετική μάρκα στην ίδια θέση. Στην περίπτωση επιλογής Arrow (τόξο), για να προστεθεί ένα τόξο στο δίκτυο, ο χρήστης πρέπει να κάνει το πρώτο click πάνω σε κάποια θέση ή μετάβαση και δεύτερο click πάνω σε μια μετάβαση ή θέση (επιτρέπεται να προστεθεί τόξο μόνο ανάμεσα σε

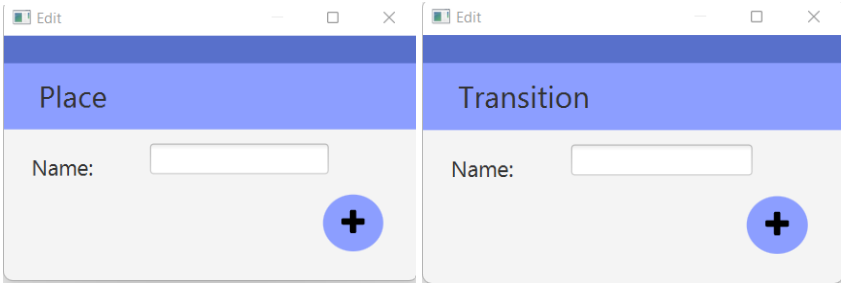
θέση και μετάβαση). Η εικόνα που ακολουθεί δείχνει ένα δίκτυο όπου προστέθηκε κάθε ένας κόμβος από τα παραπάνω.



Εικόνα 8: Προσθήκη κόμβων στην εφαρμογή

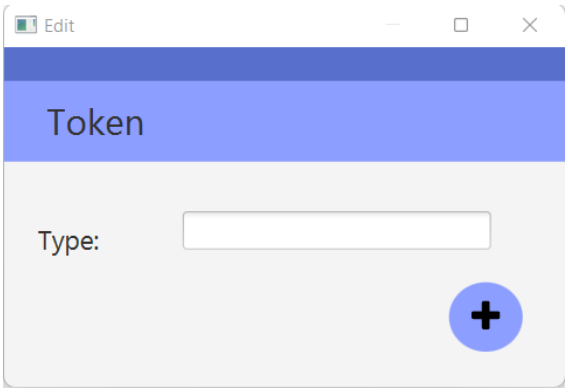
Επιπρόσθετα, ο χρήστης μπορεί να κάνει double click ένα αντικείμενο, και σε κάθε περίπτωση γίνεται:

- Στις θέσεις και μεταβάσεις εμφανίζεται ένα popup window για μετονομασία.



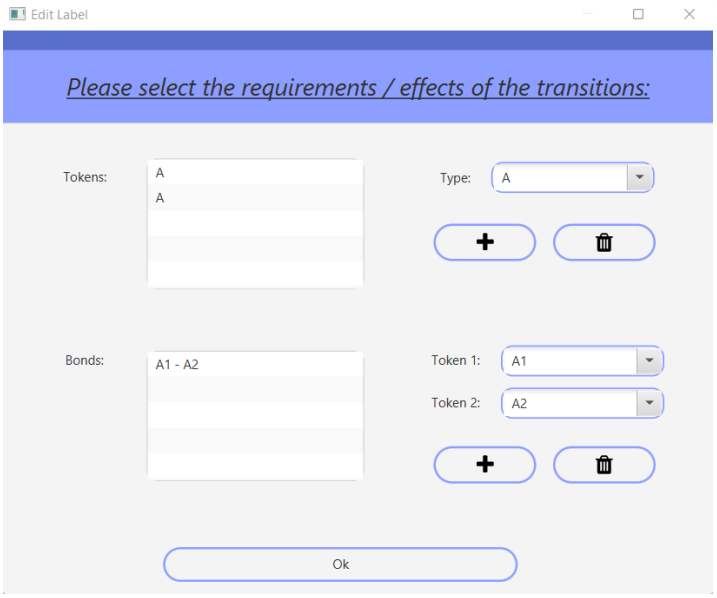
Εικόνα 9: Μετονομασίας θέσης και μετάβασης

- Στις μάρκες εμφανίζεται ένα popup window για αλλαγή τύπου.



Εικόνα 10: Αλλαγή τύπου μάρκας

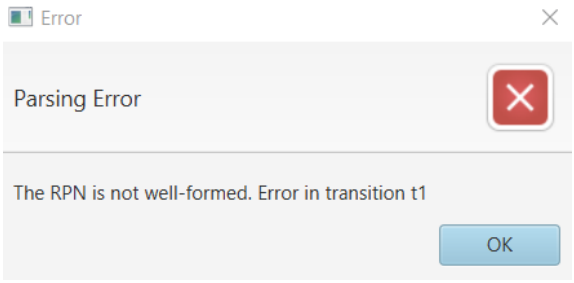
- Στα τόξα εμφανίζεται ένα popup window για προσθήκη ετικέτας. Όπως φαίνεται στην εικόνα πιο κάτω, ο χρήστης μπορεί να προσθέσει μεταβλητές μαρκών, καθορίζοντας τον τύπο της μεταβλητής στο Type και κάνοντας click στο κουμπί '+'. Τότε η μεταβλητή εμφανίζεται στο Tokens listview. Ακόμα, επιλέγοντας μια μεταβλητή από το listview μπορεί να τι διαγράψει με το ανάλογο κουμπί. Αντίστοιχα, μπορεί να προσθέσει και να διαγράψει δεσμό, διαλέγοντας δύο μεταβλητές, που προστέθηκαν πιο πάνω, στα Token 1 και Token 2 (οι μεταβλητές ιδίου τύπου που προστέθηκαν στα Tokens, ονοματίζονται αυτόματα με ένα ακέραιο μετά τον τύπο τους στα Bonds). Ο δεσμός στις ετικέτες απεικονίζετε με το σύμβολο '-' ανάμεσα στις δύο μεταβλητές.



Εικόνα 11: Ενημέρωση ετικέτας

Τέλος, όταν ο χρήστης επιλέξει να μεταβεί σε ένα από τα άλλα δύο tabs, το σύστημα κάνει έλεγχο για το αν το δίκτυο είναι καλά ορισμένο (well-formed). Αν είναι καλά

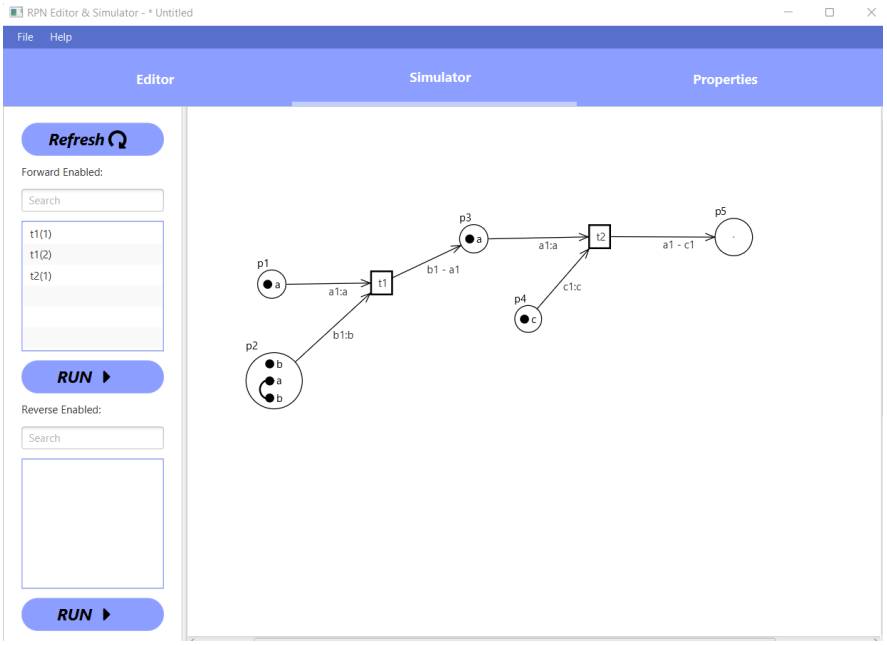
ορισμένο, μεταβαίνει στο tab που επέλεξε. Αν δεν είναι, τότε δεν μεταβαίνει και εμφανίζεται το πιο κάτω error popup, με ένδειξη για το που βρίσκεται το λάθος.



Εικόνα 12: Μήνυμα λάθους

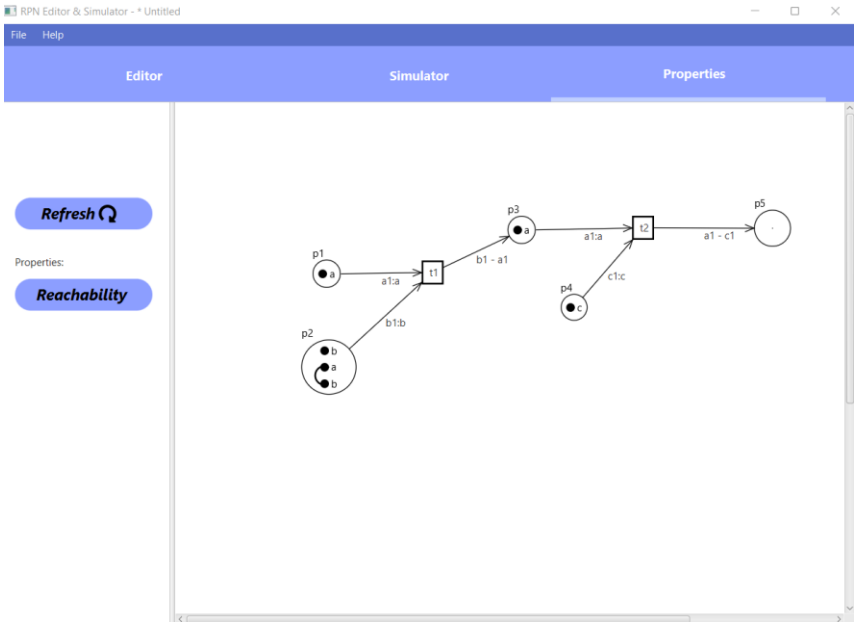
4.3 Οθόνη Simulator Tab

Στο Simulator tab, ο χρήστης μπορεί να εκτελέσει της μεταβάσεις οι οποίες επιτρέπονται να εκτελεστούν, είτε εμπρόσθια είτε αντίστροφα, σύμφωνα με τους κανόνες εκτέλεσης μεταβάσεων. Στην πιο κάτω εικόνα, φαίνεται το Simulator tab για το δίκτυο που βρίσκεται στον καμβά. Στα αριστερά της οθόνης, το σύστημα εμφανίζει αυτόματα όλες τις μεταβάσεις που μπορούν να εκτελεστούν. Στο Forward Enabled, φαίνονται οι εμπρόσθιες εκτελέσεις και στο Reverse Enabled, οι αντίστροφες εκτελέσεις. Στο Forward Enabled φαίνονται τρείς μεταβάσεις, οι t1(1), t1(2) και t2(1). Οι δύο μεταβάσεις t1, είναι λόγο του ότι η μετάβαση αυτή στη θέση p2 μπορεί να επιλεχθεί τόσο η μάρκα b, όσο ο δεσμός a-b. Αυτές οι επιλογές καθιστά τις δύο εκτελέσεις διαφορετικές. Το ίδιο θα ισχύει και στο Reverse Enabled, σε περίπτωση που υπάρχουν μεταβάσεις που μπορούν να εκτελεστούν αντίστροφα. Ο χρήστης μπορεί να επιλέξει μια από αυτές τις μεταβάσεις που εμφανίζονται και πατώντας το κουμπί Run, εκτελείται η μετάβαση που επιλέχθηκε. Τέλος, μπορεί να πατήσει το κουμπί Refresh για να επανέλθει το δίκτυο στην αρχική του κατάσταση.



Εικόνα 13: Οθόνη προσομοιωτή

4.4 Οθόνη Properties Tab



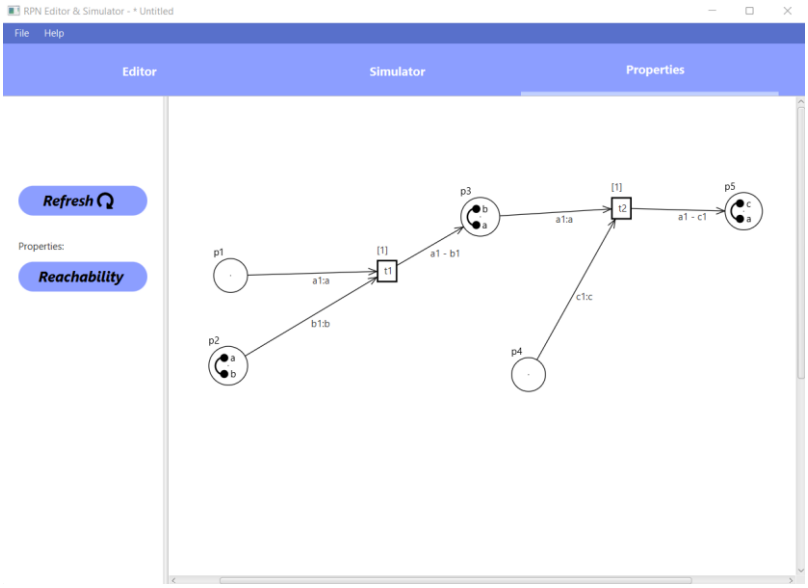
Εικόνα 14: Οθόνη Ιδιοτήτων

Στο Properties tab, ο χρήστης μπορεί να εκτελέσει την ιδιότητα της προσβασιμότητας για ένα μαρκάρισμα το οποίο καθορίζει ο ίδιος. Στην πιο πάνω εικόνα, φαίνεται το Properties tab για το δίκτυο που βρίσκεται στον καμβά.

Εικόνα 15: Αναδυόμενο παράθυρου ελέγχου προσβασιμότητας

Όταν ο χρήστης πατήσει το κουμπί Reachability, εμφανίζεται ένα popup window για να καθορίσει το μαρκάρισμα που θέλει να ελέγξει αν είναι προσβάσιμο. Η εικόνα πιο πάνω δείχνει το πως ορίζεται αυτό το μαρκάρισμα. Στο Token και στο Place, καθορίζει τον τύπο της μάρκας και τη θέση της και πατώντας το κουμπί '+', εμφανίζεται στο listview Tokens. Ακόμα, επιλέγοντας μια μεταβλητή από το listview μπορεί να τι διαγράψει με το ανάλογο κουμπί. Η παρένθεση με το νούμερο δίπλα στον τύπο αναθέτετε αυτόματα από το σύστημα, ανάλογα με τον αριθμό τον μαρκών ιδίου τύπου που υπάρχουν στο δίκτυο. Ο λόγος είναι για να μπορεί ο χρήστης να ξεχωρίζει τις μάρκες που θέλει να τοποθετήσει σε συγκεκριμένες θέσεις όταν μετά θέλει να ορίσει δεσμούς σε αυτές τις μάρκες. Αντίστοιχα, μπορεί να προσθέσει και να διαγράψει δεσμό, διαλέγοντας δύο μάρκες, που προστέθηκαν πιο πάνω, στα Token 1 και Token 2. Η επιλογή strengthened bond υπονοεί ότι οι μάρκες του δεσμού αυτό δεν μπορούν να έχουν δεσμό με άλλες μάρκες, οι οποίες δεν είναι δηλωμένες επίσης σαν strengthened bond με τις μάρκες αυτού του δεσμού. Με το κουμπί RUN, το σύστημα ελέγχει το μαρκάρισμα που οριστικέ, αν είναι προσβάσιμο. Αν δεν είναι, τότε εμφανίζει 'No Path Found'. Αν είναι τότε εμφανίζει την ακολουθία μεταβάσεών που πρέπει να εκτελεστούν για να φτάσει σε αυτό το μαρκάρισμα. Με το κουμπί APPLY, το σύστημα εκτελεί αυτές τις μεταβάσεις και το δίκτυο στον καμβά

μεταβαίνει την κατάσταση μετά από αυτές τις εκτελέσεις. Τέλος, μπορεί να πατήσει το κουμπί Refresh για να επανέλθει το δίκτυο στην αρχική του κατάσταση. Πιο κάτω φαίνεται το δίκτυο μετά την εκτέλεση της ακολουθίας μεταβάσεων που φαίνεται στην Εικόνα 15 (πατώντας το κουμπί APPLY).



Εικόνα 16: Οθόνη Ιδιοτήτων μετά την εκτέλεση της ιδιότητας προσβασιμότητας

4.5 Δομές Υλοποίησης

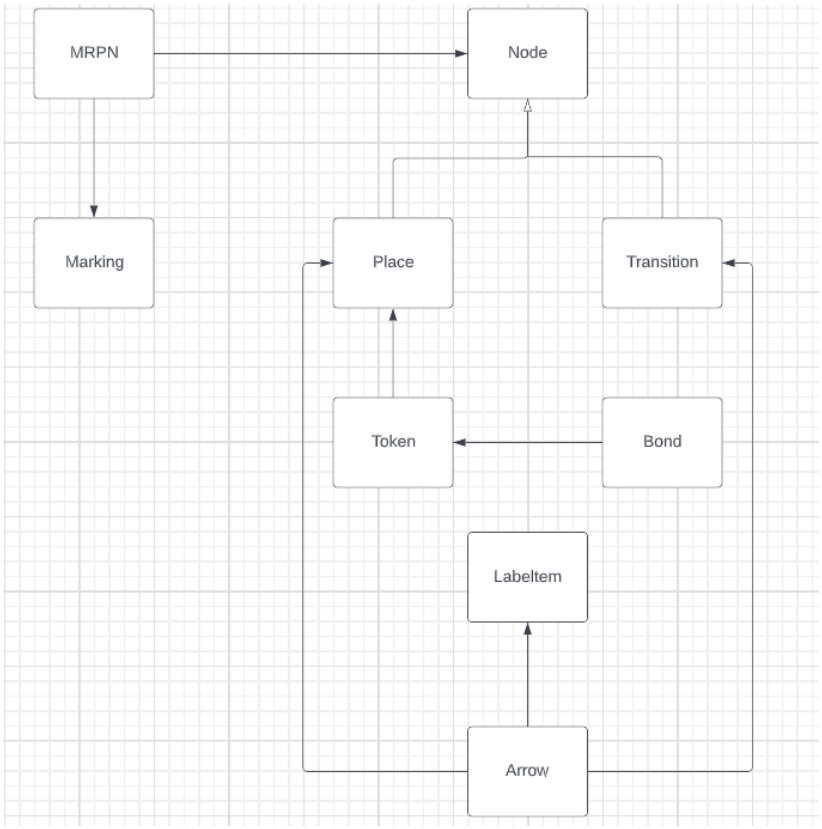
Λόγο του μεγέθους του συστήματος, έχει διαπεραστεί σε πέντε πακέτα με ονόματα, “rpnsim.application”, “rpnsim.application.editor”, “rpnsim.application.model”, “rpnsim.application.ui”, “rpnsim.application.simulator” και “rpnsim.application.properties”. Πιο κάτω θα αναλυθεί το περιεχόμενο του κάθε πακέτου ξεχωριστά.

Το πακέτο με όνομα “rpnsim.application”, είναι υπεύθυνο για την αρχικοποίηση και τη λειτουργία της γραφικής διεπαφής καθώς και για της γενικές λειτουργίες του συστήματος. Σε αυτό το πακέτο γίνεται το διάβασμα και η αποθήκευση του Αναστρέψιμο δίκτυο Πέτρι με πολλαπλές μάρκες ιδίου τύπου σε xml αρχεία, λειτουργίες οι οποίες χρησιμοποιούνται και στο Simulator Tab και στο Properties Tab. Ακόμα, υλοποιούνται οι λειτουργείες του File menu, δηλαδή η αποθήκευση και το άνοιγμα των xml αρχείων από και προς τη συσκευή στην οποία τρέχει το σύστημα.

Το πακέτο με όνομα “rpnsim.application.editor”, είναι υπεύθυνο για τις λειτουργίες σχετικά με την δημιουργία και τροποποίηση ενός δικτύου στο καμβά του Editor Tap. Συγκεκριμένα, η κλάση “EditorArea” καθορίζει κάθε λειτουργία που θα εκτελεστεί

ανάλογα με την επιλογή που έχει επιλεγμένη ο χρήστης. Η κλάση ‘‘EditorToolbox’’ δημιουργεί τα κουμπιά αυτών τον επιλογών και η κλάση ‘‘ToolboxButton’’ παρέχει ανατροφοδότηση για το πιο κουμπί είναι τη δεδομένη στιγμή ενεργοποιημένο. Τέλος, η κλάση ‘‘NameEditPopup’’ μετονομάζει τα αντικείμενα που μπορούν να μετονομαστούν και η κλάση ‘‘TokenEditPopup’’ αλλάζει τύπο στις μάρκες.

Το πακέτο με όνομα “rpnsim.application.model”, είναι υπεύθυνο για την μοντελοποίησης όλων των στοιχείων ενός Multi Reversing Petri Net σε αντικείμενα. Συγκεκριμένα, κάθε κλάση αντιπροσωπεύει ένα στοιχείο του δικτύου και αποθηκεύοντας όλες τις χρήσιμες πληροφορίες του, τις λειτουργίες που του αντιστοιχούν, καθώς και τις λειτουργίες για την αλληλεπίδραση του με τα άλλα στοιχεία. Στην πιο κάτω εικόνα φαίνεται πως συσχετίζονται οι κλάσεις αυτές μετάξι τους.

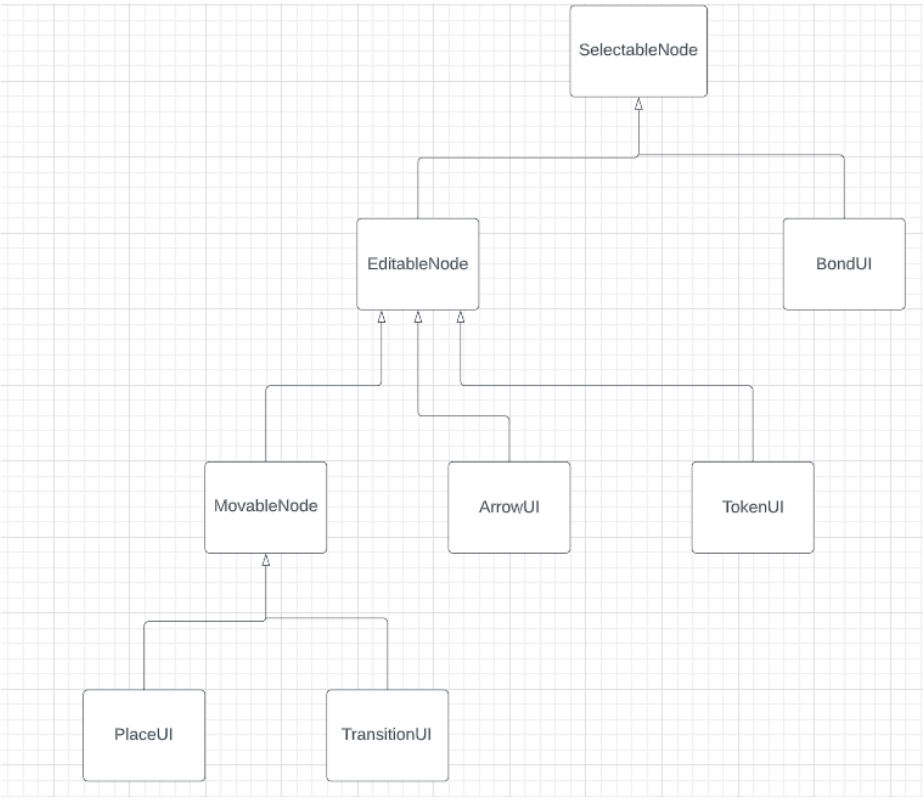


Εικόνα 17: Συσχετίσεις κλάσεων πακέτου model

Ένα Αναστρέψιμο δίκτυο Πέτρι με πολλαπλές μάρκες ιδίου τύπου (MRPN) αποτελείται από τους κόμβους (Node), δηλαδή τις θέσεις (Place) και τις μεταβάσεις (Transition). Οι μάρκες (Token) συμπεριλαμβάνονται μέσα στις θέσει και οι δεσμοί (Bond) δημιουργούνται ανάμεσα σε δύο μάρκες. Τα τόξα (Arrow) ενώνουν μια θέση με μια μετάβαση και το αντίστροφο και σε κάθε τόξο υπάρχουν μια ετικέτα η οποία

συμπεριλαμβάνει πολλές μεταβλητές (LabelItem). Τέλος, υπάρχει το μαρκάρισμα (Marking) μιας χρονικής στιγμής ενός δικτύου.

Το πακέτο με όνομα “rpnsim.application.ui”, είναι υπεύθυνο για την γραφική απεικόνιση και την ενημέρωση των αντικειμένων του δικτύου. Ο χρήστης μπορεί να αλληλοεπιδρά με διαφορετικό τρόπο με κάθε γραφικό αντικείμενο στον καμβά. Έτσι, έχει οριστεί μια ιεραρχία στις κλάσεις, που φαίνεται πιο κάτω, που καθορίζει τις ιδιότητες του κάθε αντικειμένου.



Εικόνα 18: Ιεραρχία UI Αντικειμένων

Τα αντικείμενα που κληρονομούν από την κλάση SelectableNode, μπορούν να επιλεχθούν από τον χρήστη. Αυτά που κληρονομούν από την κλάση EditableNode, μπορούν να τροποποιηθούν από την χρήστη και τέλος, αυτά που κληρονομούν από την κλάση MovableNode μπορούν να μετακινηθούν από τον χρήστη.

Το πακέτο με όνομα “rpnsim.application.simulator”, είναι υπεύθυνο για τον υπολογισμό και την προσομοίωση της συμπεριφοράς του δικτύου με βάση τις μεθόδους εκτέλεσης μεταβάσεων. Συγκεκριμένα, έχουν οριστεί οι κλάσεις ‘‘ForwardExecution’’ και ‘‘ReverseExecution’’ που είναι υπεύθυνες για την εμπρόσθια και την αντίστροφη

εκτέλεση αντίστοιχα. Ακόμα, οι δύο αυτές κλάσεις υπολογίζουν τις μεταβάσεις οι οποίες επιτρέπεται, με βάση τον ορισμό της κάθε εκτέλεσης, να εκτελεστούν.

Το πακέτο με όνομα “rpnsim.application.properties”, είναι υπεύθυνο για την εκτέλεση των ιδιοτήτων των Αναστρέψιμων δικτύων Πέτρι με πολλαπλές μάρκες ιδίου τύπου. Στη συγκεκριμένη διπλωματική, υπάρχει υλοποιημένη μόνο μια ιδιότητα, αυτή της προσβασιμότητας. Η κλάση ‘‘Reachability’’ διαχειρίζεται τις λειτουργίες για δημιουργία του επιθυμητού μαρκαρίσματος από τον χρήστη και με τη χρήση της κλάσεις ‘‘Search’’, εκτελείται ο έλεγχος προσβασιμότητας.

4.6 Δομές Δεδομένων

Οι κύριες δομές δεδομένων που χρησιμοποιήθηκαν είναι οι εξής:

Λίστες (ArrayList)

Οι λίστες χρησιμοποιήθηκαν για την αποθήκευση των αντικειμένων UI στο ScrollArea, γιατί προσφέρουν εύκολη προσθήκη αντικειμένων και κρατούν τη σειρά προσθήκης. Η πολυπλοκότητα αναζήτησης τους είναι O(n).

Σύνολα (Set)

Τα σύνολα χρησιμοποιήθηκαν για την αποθήκευση των αντικειμένων LabelItem στα Arrows, αφού δεν μας ενδιαφέρει η σειρά αποθήκευσης. Η πολυπλοκότητα προσθήκης και αναζήτησης τους είναι O(logn).

Πίνακες Κατακερματισμού (Hash Map)

Οι πίνακες κατακερματισμού είναι η βασικότερη δομή του συστήματος, αφού χρησιμοποιείται για την αποθήκευση του Αναστρέψιμου δικτύου Πέτρι με πολλαπλές μάρκες ιδίου τύπου. Προτιμήθηκε αυτή η δομή δεδομένων αφού μπορεί να συσχετίζει τα ονόματα των συστατικών του δικτύου με το αντικείμενο τους. Αυτό επιτρέπει γρήγορη ανάκτηση του συστατικού.

Άκυκλος Γράφος

Αυτή τη δομή την κατασκευαστικέ ειδικά για την ιδιότητα της προσβασιμότητας στα αναστρέψιμα δίκτυα Πέτρι. Αποτελείται από τους κόμβους (Nodes) του γράφου και τις ακμές (Edges) τους. Αυτό επιτρέπει την αποθήκευση των μονοπατιών εκτελέσεις των

μεταβάσεων και στο τέλος της αναζήτησης τον υπολογισμό του συντομότερου μονοπατιού.

4.7 Αλγόριθμοι

Σε αυτή την υποενότητα θα εξεταστούν οι αλγόριθμοι που συμπεριλαμβάνει το σύστημα. Αυτοί οι αλγόριθμοι αποτελούν τις τέσσερεις κύριες λειτουργίες του συστήματος: έλεγχος αν το δίκτυο είναι καλά ορισμένο (well-formed), εμπρόσθια εκτέλεση, αντίστροφη εκτέλεση και έλεγχος προσβασιμότητας (reachability).

Συγκριτικά με τους αλγόριθμους των Αναστρέψιμων Δικτύων Πέτρι, υπάρχει αυξημένη πολυπλοκότητα λόγο των πολλαπλών μαρκών ιδίου τύπου. Απαιτούνται επιπλέων έλεγχοι για την εξακρίβωση των σωστών τύπων, την εύρεση όλων των δυνατών μαρκαρισμάτων για κάθε λειτουργία, καθώς και για την απαλοιφή ίδιων μαρκαρισμάτων που θα δημιουργούνται λόγο της δυνατότητας ιδίων τύπων.

4.7.1 Αλγόριθμος Καλά ορισμένο (Well-formed)

Ο αλγόριθμος αυτός, ελέγχει αν ένα Αναστρέψιμο δίκτυο Πέτρι με πολλαπλές μάρκες ιδίου τύπου είναι καλά ορισμένο. Χρειάζεται να επιβεβαιωθεί το καλά ορισμένο δίκτυο προτού το σύστημα προχωρήσει στις λειτουργείες της εκτέλεσης και των ιδιοτήτων του δικτύου στο simulator και properties tab. Στην περίπτωση όπου το δίκτυο δεν επιβεβαιωθεί σαν καλά ορισμένο, το σύστημα δεν μπορεί να προχωρήσει στις πιο πάνω λειτουργείες.

```
Algorithm 1: Well-Formed
Input: mrpn : MRPN;
Output: if mrpn is well formed, it returns true else returns false;

for each transition ∈ mrpn do
    //Definition 2.1
    preT = get incoming token variables of transition;
    postT = get outgoing token variables of transition;
    if (preT ≠ postT) then
        return false;
    end

    //Definition 2.2
    for each outgoing arrow (A) of transition do
        for each outgoing arrow (B = A + 1) of transition do
            tokensA = get token variables of arrow A;
            tokensB = get outgoing token variables of arrow B;
            if (tokensA ∩ tokensB ≠ ∅) then
                return false;
            end
        end
    end
end
return true;
```

Εικόνα 2: Ψευδοκώδικας well-formed

Συγκεκριμένα, ο αλγόριθμος Well-Formed για κάθε μετάβαση του δικτύου ελέγχει τα ακόλουθα:

1. Αν οι μεταβλητές στα εισερχόμενα τόξα της μετάβασης (preT) είναι οι ίδιες με τις μεταβλητές στα εξερχόμενα τόξα της (postT).
2. Αν οι μεταβλητές των μαρκών και των δεσμών εμφανίζονται μόνο σε ένα από τα εξερχόμενα τόξα της μετάβασης. Δηλαδή, δεν πρέπει να υπάρχουν κοινές μεταβλητές ανάμεσα στα εξερχόμενα τόξα της μετάβασης.

4.7.2 Αλγόριθμος εμπρόσθιας εκτέλεσης

Η λειτουργία της εμπρόσθιας εκτέλεσης σπάζει σε δύο κύριους αλγορίθμους. Τον αλγόριθμο 2, ο οποίος είναι υπεύθυνος για την εύρεση όλων των δυνατών μεταβάσεων που επιτρέπετε να εκτελεστούν και ταυτόχρονα όλα τα δυνατά διαφορετικά μαρκαρίσματα που μπορεί να έχει η κάθε μετάβαση. Ο δεύτερος είναι ο αλγόριθμος 4,

όπου πυροδοτεί την μετάβαση με το συγκεκριμένο μαρκάρισμα που επιλεκτικέ από τον χρήστη.

Πιο κάτω φαίνεται ο αλγόριθμος 2:

Algorithm 2: Get forward-enabled transitions

```
Input: mrpn : MRPN;
Output: return all forward-enabled markings for every transitions of
        mrpn;

Pair(Transition, Matching) [] enabled;
// matching is the match between tokens and variables
for each transition ∈ mrpn do
    boolean isEnabled = true;
    Matching [] allMatchings;
    for each incoming arrow ∈ transition do
        Place place = arrow.getSource();
        Matching [] matchingsOfArrow;
        Matching tempMatching;
        String[] variables = arrow.getTokens();
        isEnabled = call matchings for arrow algorithm;
        if (!isEnabled) then
            break;
        end
        allMatchings = merge of allMatchings and matchingsOfArrow;
    end
    if (isEnabled) then
        enabled.add(new Pair(transition, allMatchings));
    end
end
return enabled;
```

Εικόνα 20: Ψευδοκώδικας forward-enabled

Ο αλγόριθμος 2, βρίσκει για κάθε μετάβαση όλους τους δυνατούς συνδυασμούς μαρκών όπου μπορεί να πυροδοτήσει τη μετάβαση. Αυτό το καταφέρνει καλώντας για κάθε εισερχόμενα τόξο κάθε μετάβασης τον αλγόριθμο 3, όπου ταιριάζει τις μεταβλητές στην ετικέτα του τόξου με μάρκες που βρίσκονται στη θέση αφετηρίας του τόξου. Τότε ο αλγόριθμος 3, επιστρέφει όλα τα δυνατά ταιριάσματα για ένα τόξο και ο αλγόριθμος 2 αναλαμβάνει να συγχωνεύσει τα ταιριάσματα κάθε τόξου για να βρει το τελικό αποτέλεσμα.

```
Algorithm 3: Matchings for arrow algorithm
Input: matchingsOfArrow : Matching[];
       tempMatching: Matching;
       variables : String[];
       place : Place;
Output: if arrow has at least one matching, it returns true and add
        the matchings to the array else returns false;

if (all variables are matched) then
    matchingsOfArrow.add(tempMatching);
    return true;
end
Token [] previouslySelected;
Token [] tokens = place.getTokens();
String variable = variables[next not matched variable];
boolean hasMatching = false;
for each token ∈ tokens do
    //Definition 4.1
    if (token.sameType(variable) ∧ !tempMatching.contains(token))
     then
        boolean flag = true;
        for each bondVariable ∈ variable.getBonds() do
            if (tempMatching.contains(bondVariable) ∧
             !token.hasBond(bondVariable) then
                flag = false;
                break;
            end
        end
        //Definition 4.2
        if (!token.getBonds().equals(variable.getBonds())) then
            flag = false;
        end
        if (flag) then
            for each selectedToken ∈ previouslySelected do
                if (tokenComparison(token, selectedToken) then
                    flag = false;
                    break;
                end
            end
        end
        //Recursive call
        if (flag) then
            tempMatching.add(new Pair(token, variable));
            previouslySelected.add(token);
            if (call matchings for arrow algorithm) then
                hasMatching = true;
            end
            tempMatching.remove(last Pair);
        end
    end
end
return hasMatching;
```

Εικόνα 21: Ψευδοκώδικας εύρεσης ταιριάσματος

Πιο συγκεκριμένα, ο αλγόριθμος 3 λειτουργεί αναδρομικά. Κάθε αναδρομικό κάλεσμα ταιριάζει μια μεταβλητή της ετικέτας του τόξου με μία μάρκα. Για να γίνει επιτυχώς το ταίριασμα, πρέπει να ισχύουν οι δύο πρώτες συνθήκες του ορισμού για επιτρεπτή εκτέλεση προς τα εμπρός. Ακόμα, για κάθε αναδρομικό κάλεσμα κρατείται μια λίστα με μάρκες που έχουν ήδη ανατεθεί σε ένα προηγούμενο ταίριασμα και γίνεται έλεγχος αν η μάρκα που ελέγχεται τώρα δεν είναι ίδια με κάποια από αυτές (για παράδειγμα αν δύο μάρκες έχουν τύπο a και καθόλου δεσμούς, τότε θεωρείται ως ίδια ανάθεση. Αν όμως η

μία από τις δύο έχει κάποιο δεσμό τότε θεωρείτε διαφορετική ανάθεση). Ο έλεγχος αυτός γίνεται με σύγκριση των μαρκών και των δεσμών τους, θεωρώντας τα σαν γράφο. Με αυτό τον τρόπο, δεν δημιουργούνται ίδια μαρκαρίσματα στο τέλος του αλγορίθμου.

Ο αλγόριθμος 4 είναι υπεύθυνος για την εκτέλεση της μετάβασης και τη μεταφορά των επιλεγμένων μαρκών και των δεσμών τους στην εξερχόμενη θέση της μετάβασης καθώς και τη δημιουργία των νέων δεσμών. Ακόμα, υλοποιεί την τρίτη συνθήκη του ορισμού, όπου διαγράφει τους δεσμούς που καταστρέφονται λόγο της μετάβασης για αποφυγή κλωνοποίησης μαρκών σε πολλές εξερχόμενες θέσεις.

```
Algorithm 4: Fire forward-enabled transition
Input: mrpn : MRPN;
       transition : Transition;
       matching: Matching;
Output: execute transition forward;

//Definition 4.3
Arrow[] outgoingArrows = transition.getOutgoingArrows();
for each incomingArrow ∈ transition do
    for each bondVariable ∈ incomingArrow do
        if (bondVariable ∉ outgoingArrows) then
            Token t1 = matching.getToken(bondVariable.getVariable1());
            Token t2 = matching.getToken(bondVariable.getVariable2());
            delete connection between t1 and t2;
        end
    end
end

for each outgoingArrow ∈ transition do
    Place outgoingPlace = arrow.getDestination();
    for each variable ∈ outgoingArrow do
        Token token = matching.getToken(variable);
        move token and its connections from current place to
         outgoingPlace;
    end
    for each bondVariable ∈ outgoingArrow do
        Token t1 = matching.getToken(bondVariable.getVariable1());
        Token t2 = matching.getToken(bondVariable.getVariable2());
        if (!t1.hasBond(t2)) then
            create connection between t1 and t2;
        end
    end
end
transition.addToHistory();
return;
```

Εικόνα 22: Ψευδοκώδικας εμπρόσθιας εκτέλεσης

4.7.3 Αλγόριθμος αντίστροφης εκτέλεσης

Η λειτουργία της αντίστροφης εκτέλεσης σπάζει σε δύο κύριους αλγορίθμους. Τον αλγόριθμο 5, ο οποίος είναι υπεύθυνος για την εύρεση όλων των δυνατών μεταβάσεων

που επιτρέπετε να εκτελεστούν και ταυτόχρονα όλα τα δυνατά διαφορετικά μαρκαρίσματα που μπορεί να έχει η κάθε μετάβαση. Ο δεύτερος είναι ο αλγόριθμος 6, όπου πυροδοτεί την μετάβαση με το συγκεκριμένο μαρκάρισμα που επιλεκτικέ από τον χρήστη. Οι δύο αυτοί αλγόριθμοι είναι αρκετά παρόμοιοι με τους αλγορίθμους της εμπρόσθιας εκτέλεσης.

Πιο συγκεκριμένα, ο αλγόριθμος 5 λειτουργεί σαν τον αλγόριθμο 2 με τη διαφορά ότι για να εκλέξει μια μετάβαση, η ιστορία (history) της πρέπει να είναι μεγαλύτερη του μηδενός και αντί για τα εισερχόμενα τόξα παίρνει τα εξερχόμενα. Όπως και ο αλγόριθμος 2, καλεί τον αλγόριθμο 3 για την εύρεση των δυνατών ταιριασμάτων μαρκών και μεταβλητών για κάθε τόξο.

Algorithm 5: Get reverse-enabled transitions

```
Input: mrpn : MRPN;
Output: return all reverse-enabled markings for every transitions of
        mrpn;

Pair(Transition, Matching) [] enabled;
// matching is the match between tokens and variables
for each transition ∈ mrpn do
    if (transition.getHistory() == 0) then
        continue;
    end
    boolean isEnabled = true;
    Matching [] allMatchings;
    for each outgoing arrow ∈ transition do
        Place place = arrow.getDestination();
        Matching [] matchingsOfArrow;
        Matching tempMatching;
        String[] variables = arrow.getTokens();
        isEnabled = call matchings for arrow algorithm;
        if (!isEnabled) then
            break;
        end
        allMatchings = merge of allMatchings and matchingsOfArrow;
    end
    if (isEnabled) then
        enabled.add(new Pair(transition, allMatchings));
    end
end
return enabled;
```

Εικόνα 23: Ψευδοκώδικας reverse-enabled

Ο αλγόριθμος 6 λειτουργεί σαν τον αλγόριθμο 4, με τη διαφορά ότι μεταφέρει τις μάρκες και τους δεσμούς τους από τις εξερχόμενες θέσεις της μετάβασης στις εισερχόμενες.

```
Algorithm 6: Fire reverse-enabled transition
Input: mrpn : MRPN;
       transition : Transition;
       matching: Matching;
Output: execute transition reverse;

//Definition 6.3
Arrow[] incomingArrows = transition.getIncomingArrows();
for each outgoingArrow ∈ transition do
    for each bondVariable ∈ outgoingArrow do
        if (bondVariable ∉ incomingArrows) then
            Token t1 = matching.getToken(bondVariable.getVariable1());
            Token t2 = matching.getToken(bondVariable.getVariable2());
            delete connection between t1 and t2;
        end
    end
end

for each incomingArrow ∈ transition do
    Place incomingPlace = arrow.getSource();
    for each variable ∈ incomingArrow do
        Token token = matching.getToken(variable);
        move token and its connections from current place to
         incomingPlace;
    end
    for each bondVariable ∈ incomingArrow do
        Token t1 = matching.getToken(bondVariable.getVariable1());
        Token t2 = matching.getToken(bondVariable.getVariable2());
        if (!t1.hasBond(t2)) then
            create connection between t1 and t2;
        end
    end
end
transition.removeFromHistory();
return;
```

Εικόνα 24: Ψευδοκώδικας αντίστροφης εκτέλεσης

4.7.4 Αλγόριθμος Προσβασιμότητας

Ο αλγόριθμος προσβασιμότητας, υπολογίζει αν το μαρκάρισμα που ορίστηκε από τον χρήστη είναι προσβάσιμο από την αρχική κατάσταση του δικτύου. Στην περίπτωση το μαρκάρισμα είναι προσβάσιμο, τότε επιστρέφει την ακολουθία των μεταβάσεων που πρέπει να εκτελεστούν εμπρόσθια ή αντίστροφα για να φτάσει σε αυτό. Διαφορετικά, επιστρέφει μήνυμα ότι δεν υπάρχει μονοπάτι που να καταλήγει σε αυτό το μαρκάρισμα.

Στον αλγόριθμο 7, φαίνεται η δομή του αλγορίθμου όπου κατασκευάζεται ένας άκυκλος γράφος. Αυτός, αποτελείται από κόμβους (nodes) που περιέχουν ένα μαρκάρισμα και ακμές (edges) που περιέχουν την πληροφορία της μετάβασης και τη μέθοδο που θα εκτελεστεί προκειμένου από το ένα μαρκάρισμα να φτάσει στο άλλο. Ο αλγόριθμος 8, χρησιμοποιείται από τον 7 για να βρει όλους τους γείτονες (δηλαδή μαρκαρίσματα που

μπορεί να φτάσει με μόνο μια εκτέλεση από κάποιο μαρκάρισμα). Έπειτα, με τη χρήση μιας ουράς (queue) εφαρμόζεται κατά πλάτος αναζήτηση (BFS) στον γράφο. Όσο η ουρά δεν είναι άδεια, βγάζει ένα κόμβο από αυτή. Αν έχει ξανά επισκεφτεί το μαρκάρισμα αυτού του κόμβου το αγνοεί. Ο έλεγχος αυτός γίνεται με σύγκριση των μαρκών σε κάθε ίδια θέση των δύο δικτύων, με τον τρόπο που εξηγήθηκε στον αλγόριθμο 3. Αν όμως δεν έχει ξανά επισκεφτεί αυτό το μαρκάρισμα, τότε ελέγχει αν είναι το μαρκάρισμα που ψάχνει. Αυτός ο έλεγχος γίνεται με αναδρομική ανάθεση μαρκών στις μεταβλητές του ζητούμενου μαρκαρίσματος για κάθε θέση του δικτύου, παρόμοια με τον αλγόριθμο 3. Αν είναι το μαρκάρισμα που ψάχνει, επιστρέφει το μονοπάτι αντίστροφα μέχρι να φτάσει στον αρχικό κόμβο. Αν δεν είναι, καλεί τον αλγόριθμο 8 και βρίσκει όλους τους γείτονες του και τους προσθέτει στην ουρά.

Algorithm 7: Reachability algorithm

```
Input: mrpn : MRPN;
       forward : ForwardExecution;
       reverse: ReverseExecution;
       searchMarking: SearchMarking;
Output: The path (if any) that leads to searchMarking;

Set⟨Marking⟩ visited;
Queue queue;
Marking root = mrpn.getMarking();
Node rootNode = new Node(root);
queue.add(rootNode);
//BFS algorithm
while !queue.isEmpty() do
    Node node = queue.pop();
    if (contains(visited,node.marking)) then
        continue;
    end
    mrpn.setMarking(node.marking);
    if (checkTarker(searchMarking,node.marking)) then
        reversePath = calculatePath(rootNode, node);
        return reversePath;
    end
    Set⟨Node⟩ neighbors = getNeighbors(node);
    for each neighbor ∈ neighbors do
        queue.add(neighbor);
    end
    visited.add(node.marking);
end
return no path;
```

Εικόνα 25: Ψευδοκώδικας προσβασιμότητας

Πιο συγκεκριμένα για τον αλγόριθμο 8, για το μαρκάρισμα που θα υπολογίσει τους γείτονες του, οι αλγόριθμοι για εύρεση όλων των δυνατών μεταβάσεων όπου μπορεί να πυροδοτήσει και εμπρόσθια και αντίστροφα. Έπειτα, εκτελεί κάθε ένα από αυτά, με τον

αντίστοιχο αλγόριθμο πυροδότησης, και προσθέτει το μαρκάρισμα που δημιουργήθηκε στη λίστα των γειτόνων καθώς και το τρέχων μαρκάρισμα ως γονέα του νέου μαρκαρίσματος.

Algorithm 8: Get neighbours algorithm

```
Input: mrpn : MRPN;
       node : Node;
       forward : ForwardExecution;
       reverse: ReverseExecution;
Output: the neighbours as a list of nodes;

List⟨Node⟩ neighbors;
Marking marking = node.getMarking();
Matching [] allForwardMatchings = call get forward-enabled transitions;
Matching [] allReverseMatchings = call get reverse-enabled transitions;
for each matching ∈ allForwardMatchings do
    Transition transition = matching.getTransition();
    mrpn.setMarking(marking);
    call fire forward-enabled transition;
    Node neighbor = new Node(mrpn.getMarking());
    node.addChild(neighbor);
    neighbors.add(neighbor);
end
for each matching ∈ allReverseMatchings do
    Transition transition = matching.getTransition();
    mrpn.setMarking(marking);
    call fire reverse-enabled transition;
    Node neighbor = new Node(mrpn.getMarking());
    node.addChild(neighbor);
    neighbors.add(neighbor);
end
return neighbors;
```

Εικόνα 26: Ψευδοκώδικας εύρεσης γειτονικών markings

Κεφάλαιο 5

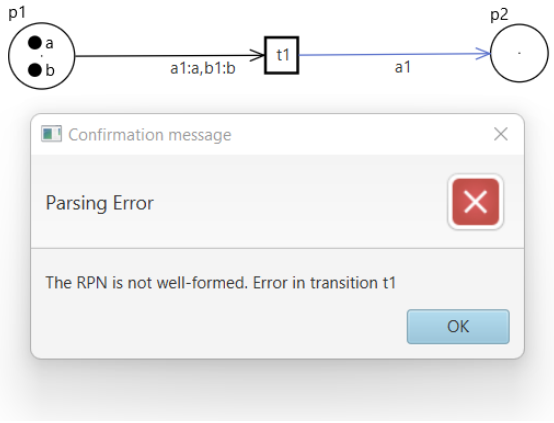
Αξιολόγηση Εφαρμογής

Στο παρόν κεφάλαιο θα αξιολογηθεί το λογισμικό με σκοπό να εξακριβωθεί αν ικανοποιούνται οι αποτίσεις που προκαθορίστηκαν. Αφού οι λειτουργικές απαιτήσεις έχουν υλοποιηθεί και παρουσιαστεί πιο πάνω, εδώ θα παρουσιαστούν οι μη λειτουργικές απαιτήσεις.

5.1 Ορθότητα

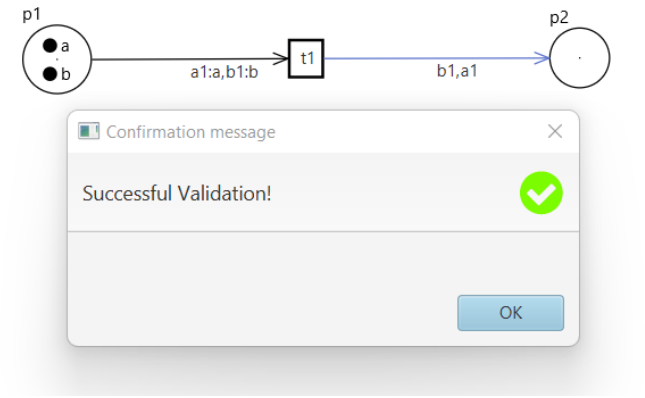
Ορθότητα σημαίνει ότι το σύστημα θα πρέπει να είναι αξιόπιστο και συνεπές, διεκπεραιώνοντας τις λειτουργίες ορθά. Για κάθε λειτουργεία του συστήματος έχουν ελεγχθεί τα διάφορα σενάρια, με σκοπό την εξακρίβωση της σωστής συμπεριφοράς του συστήματος. Παρακάτω, θα παρουσιαστεί ένα παράδειγμα κάθε λειτουργίας.

Αρχικά, παρουσιάζεται η λειτουργία ελέγχου για καλά ορισμένο (well-formed) Αναστρέψιμο δίκτυο Πέτρι με πολλαπλές μάρκες ιδίου τύπου, και στην περίπτωση που είναι και σε αυτή που δεν είναι καλά ορισμένο.



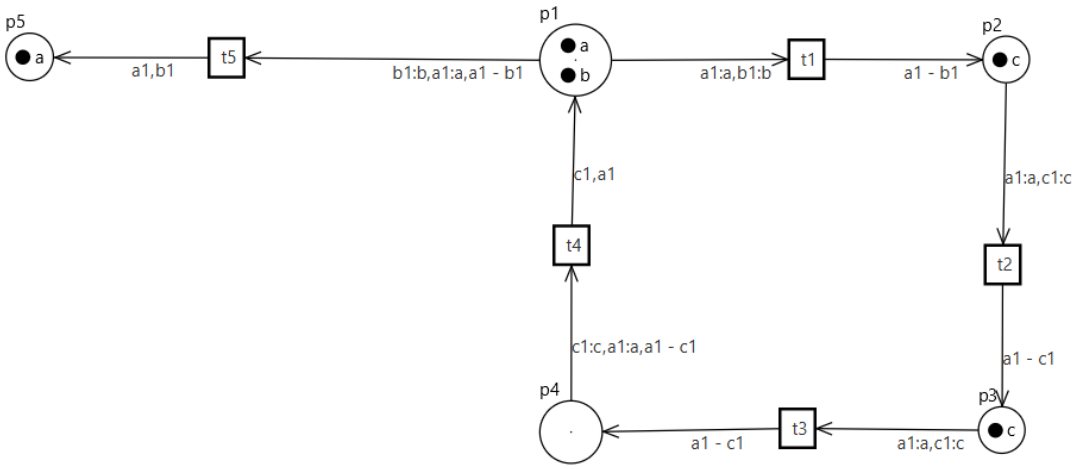
Εικόνα 27: Παράδειγμα μη καλά ορισμένου MRPN

Το παράδειγμα στην πιο πάνω εικόνα δεν είναι καλά ορισμένο, αφού οι μεταβλητές στα εισερχόμενα τόξα της μετάβασης t1 δεν είναι οι ίδιες με τις μεταβλητές στα εξερχόμενα τόξα (δεν υπάρχει η b1). Με την προσθήκη της μεταβλητής b1 στα εξερχόμενα τόξα της μετάβασης t1, το δίκτυο θα θεωρείται καλά ορισμένο.



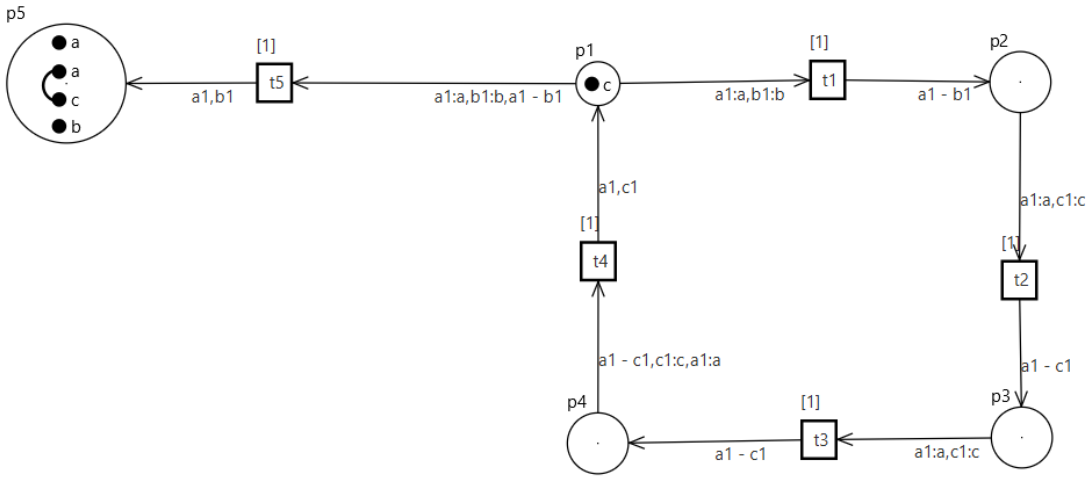
Εικόνα 28: Παράδειγμα καλά ορισμένου MRPN

Έπειτα, παρουσιάζεται η λειτουργία της εμπρόσθιας εκτέλεσης. Στην πιο κάτω εικόνα φαίνεται το δίκτυο που θα χρησιμοποιηθεί.



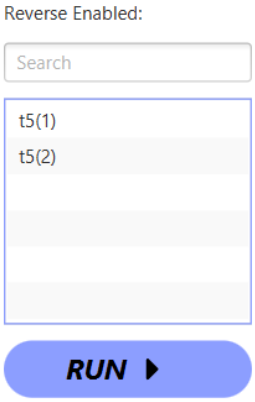
Εικόνα 29: Αρχική κατάσταση

Στην επόμενη εικόνα φαίνεται η κατάσταση του δικτύου μετά την εμπρόσθια εκτέλεση των μεταβάσεων t1, t2, t3, t4, t5 με αυτή τη σειρά. Η μετάβαση t1 δημιουργεί δεσμό ανάμεσα σε μάρκες τύπου a και b, η μεταβάσεις t2 και t3 δημιουργεί δεσμό ανάμεσα σε μάρκες τύπου a και c, η μετάβαση t4 καταστρέφει δεσμό ανάμεσα σε μάρκες τύπου a και c. Τέλος, η μετάβαση t5 καταστρέφει δεσμό ανάμεσα σε μάρκες τύπου a και b.



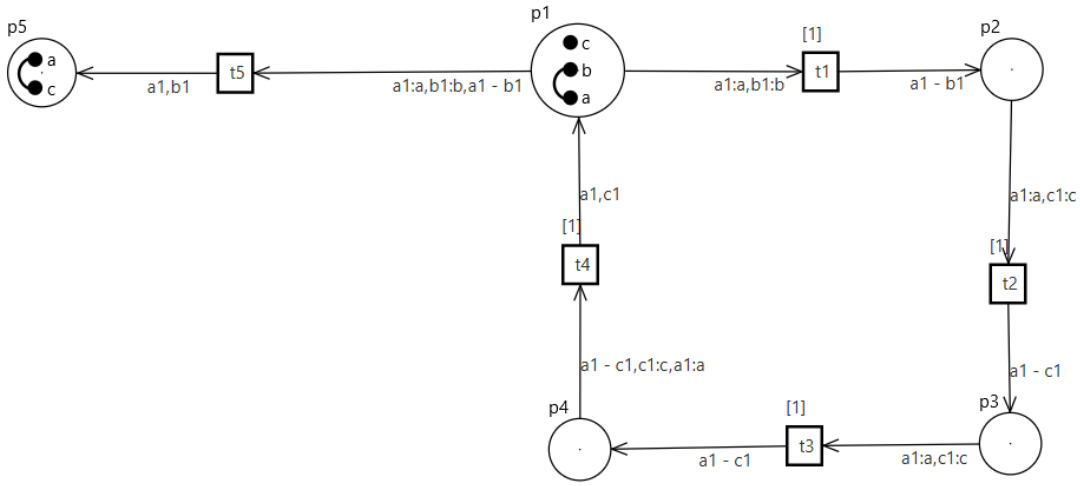
Εικόνα 30: Τελική κατάσταση

Στη συνέχεια, παρουσιάζεται η λειτουργία της αντίστροφης εκτέλεσης. Θα χρησιμοποιηθεί το προηγούμενο δίκτυο στην τελική του κατάσταση. Η μόνη μετάβαση που μπορεί να αντιστραφεί είναι η t5. Όμως, υπάρχου δύο δυνατοί τρόποι να αντιστραφεί η μετάβαση, οι οποίοι θα προκαλέσουν διαφορετικό αποτέλεσμα.



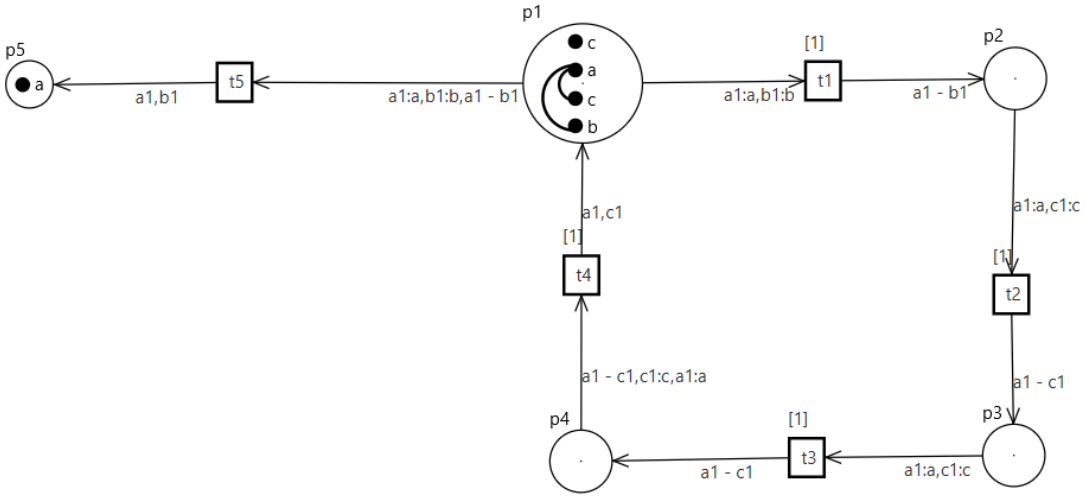
Εικόνα 31: Επιλογές αντιστρέψιμων μεταβάσεων

Ο πρώτος τρόπος είναι η επιλογή των μαρκών a (χωρίς το δεσμό με τη μάρκα c) και b και το αποτέλεσμα φαίνεται στην πιο κάτω εικόνα.



Εικόνα 32: Πρώτη κατάσταση αντιστροφής t5 μετάβασης

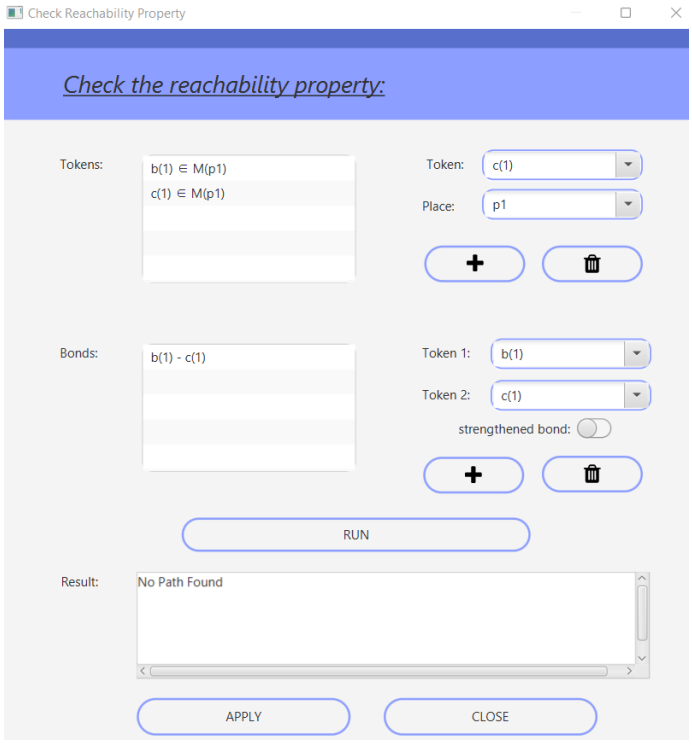
Ο δεύτερος τρόπος είναι η επιλογή των μαρκών a (με το δεσμό με τη μάρκα c) και b και το αποτέλεσμα φαίνεται στην πιο κάτω εικόνα.



Εικόνα 33: Δεύτερη κατάσταση αντιστροφής t5 μετάβασης

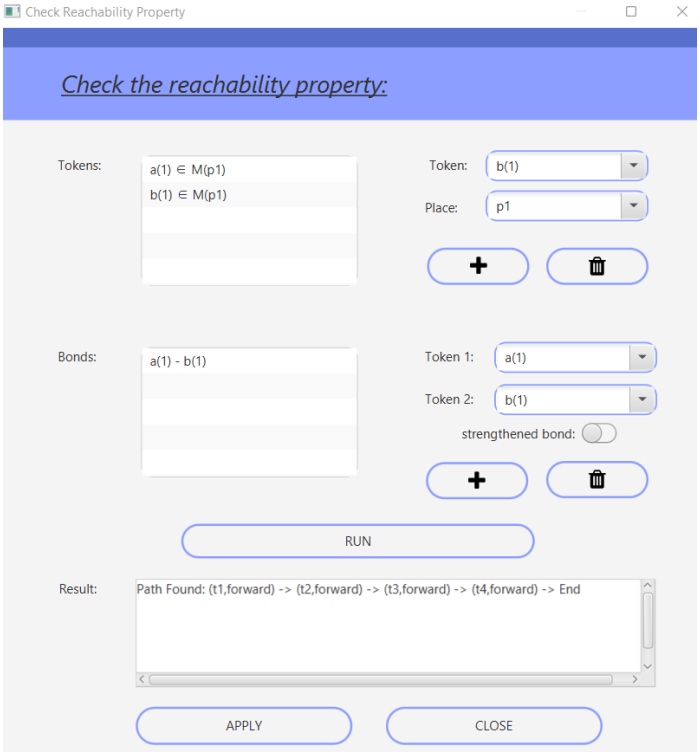
Τέλος, παρουσιάζεται η λειτουργία της προσβασιμότητας (reachable) σε Αναστρέψιμο δίκτυο Πέτρι με πολλαπλές μάρκες ιδίου τύπου, και στην περίπτωση που ένα μαρκάρισμα είναι προσβάσιμο και στην περίπτωση που δεν είναι. Θα χρησιμοποιεί το πιο πάνω δίκτυο.

Στην πιο κάτω εικόνα φαίνεται η περίπτωση όπου δεν ικανοποιείται η ιδιότητα της προσβασιμότητας. Το σύστημα παρουσιάζει μήνυμα, ότι δεν υπάρχει μονοπάτι που να καταλήγει στη ζητούμενη κατάσταση, όπου υπάρχει ένας δεσμός b-c στη θέση p1.



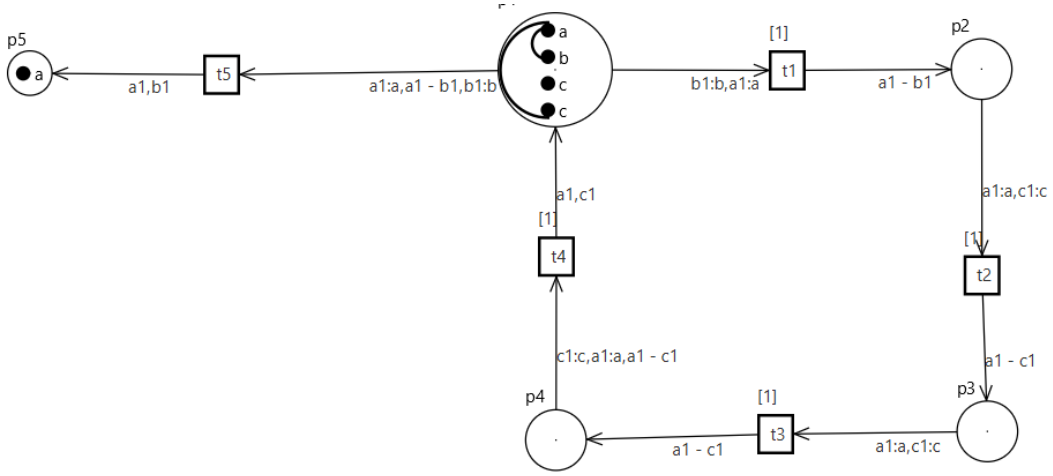
Εικόνα 34: Παράδειγμα μη προσιτής κατάστασης

Έπειτα, παρουσιάζεται η περίπτωση όπου ικανοποιείται η ιδιότητα της προσβασιμότητας. Το σύστημα παρουσιάζει μήνυμα με την ακολουθία εκτελέσεων των μεταβάσεων μέχρι να φτάσει στη ζητούμενη κατάσταση, όπου υπάρχει ένας δεσμός a-b στη θέση p1.



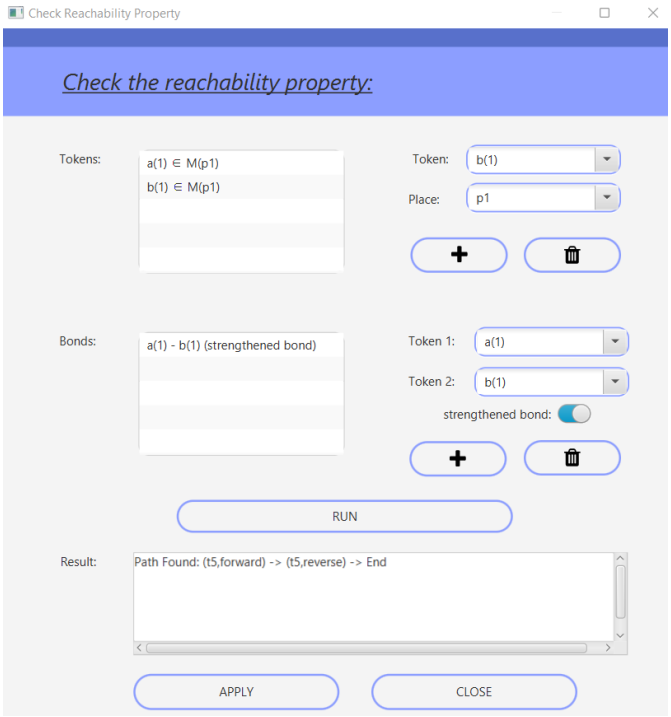
Εικόνα 35: Παράδειγμα 1 προσιτής κατάστασης

Στη συνέχεια, με το κουμπί APPLY το σύστημα μεταβαίνει σε αυτή την κατάσταση.



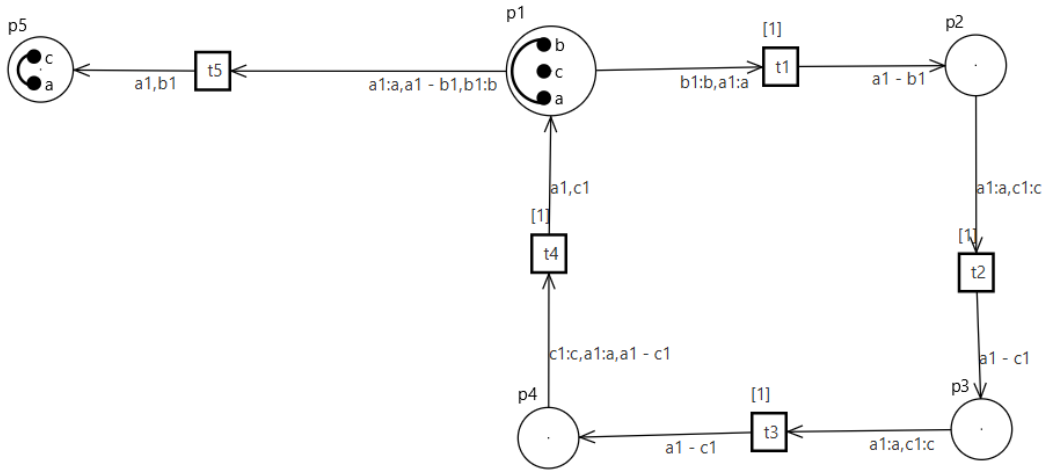
Εικόνα 36: Παράδειγμα 1 προσιτής κατάστασης και εκτελέσεις των μεταβάσεων

Τέλος, παρουσιάζεται η περίπτωση όπου ικανοποιείται η ιδιότητα της προσβασιμότητας με τη χρήση του ‘‘strengthened bond’’. Η κατάσταση που ζητείται περιέχει ένα ‘‘strengthened bond’’ (δηλαδή δεσμό όπου δεν υπάρχουν άλλοι δεσμοί στις μάρκες του) a-b στη θέση p1 και το δίκτυο που ελέγχεται είναι το πιο πάνω, μετά την εκτέλεση των καταστάσεων.



Εικόνα 37: Παράδειγμα 2 προσιτής κατάστασης

Στη συνέχεια, με το κουμπί APPLY το σύστημα μεταβαίνει σε αυτή την κατάσταση.

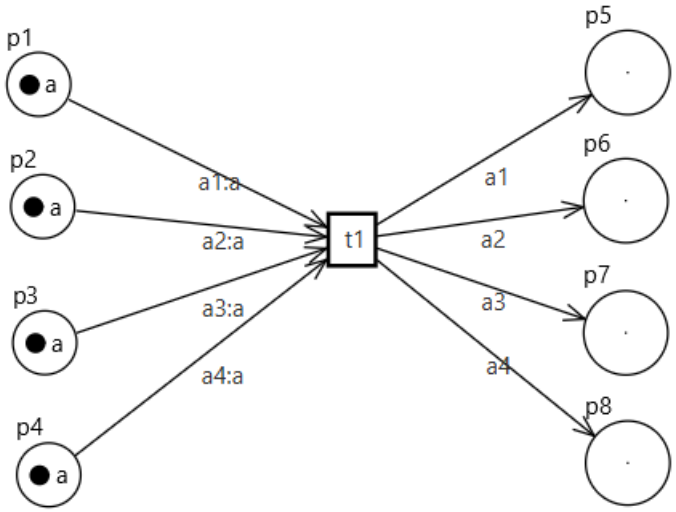


Εικόνα 38: Παράδειγμα 2 προσιτής κατάστασης και εκτελέσεις των μεταβάσεων

5.2 Επίδοση

Για την επίδοση του συστήματος θα μετρηθούν οι χρόνοι των βασικών λειτουργιών του, για αυξανόμενου μεγέθους δίκτυα. Οι λειτουργίες αυτές είναι η επικύρωση, οι μεταβάσεις στα simulator και properties tabs, ο υπολογισμός των επιτρεπτών εμπρόσθιων και αντίστροφων μεταβάσεων, η εκτέλεση μιας εμπρόσθιας και αντίστροφης μετάβασης και η ιδιότητα της προσβασιμότητας. Ο τρόπος μέτρησης του χρόνου θα γίνει με τη μέθοδο System.nanoTime() της java, με χρόνο εκκίνησης στην αρχή της κάθε λειτουργίας και χρόνο λήξης στο τέλος της.

Με μόνη εξαίρεση την ιδιότητα της προσβασιμότητας, τα δίκτυα που θα χρησιμοποιηθούν θα παράγονται αυτόματα σε XML μορφή από ένα πρόγραμμα που σχεδιαστικέ για αυτό το σκοπό. Το πρόγραμμα παίρνει ως είσοδο μια μεταβλητή N, η οποία καθορίζει τον αριθμό των θέσεων και τον μαρκών του δικτύου. Πιο συγκεκριμένα, το δίκτυο θα έχει 2N θέσεις, N θέσεις εισόδου και N θέσεις εξόδου, και N μάρκες ιδίου τύπου, μια σε κάθε θέση εισόδου. Θα υπάρχει πάντα μια μόνο μετάβαση όπου θα ενώνει τις θέσεις εισόδου με τις θέσεις εξόδου. Οι ετικέτες των τόξων θα έχουν μια μεταβλητή με τον τύπο των μαρκών. Στην πιο κάτω εικόνα φαίνεται ένα παράδειγμα για N = 4.



Εικόνα 39 : Αυτόματα παραγόμενο δίκτυο 1 με μέγεθος N=4.

Επιλέχθηκε να χρησιμοποιηθούν αυτού του τύπου δίκτυα αφού είναι εύκολο στα να δημιουργηθούν αυτόματα και είναι κατάλληλα για τον έλεγχο των λειτουργιών που θα εξυπηρετήσουν. Ο λόγος είναι ότι με την αύξηση του N, αυξάνονται τα στοιχεία διεπαφής και τα τόξα, όπου για τις λειτουργίες που θα χρησιμοποιηθούν είναι το βασικό στοιχείο που επιβαρύνει τους ελέγχους. Τα μεγέθη που θα χρησιμοποιηθούν είναι N = 100, 200, 500, 1000, 1500, 2000.

Για την περίπτωση της προσβασιμότητας θα χρησιμοποιηθεί ένα πιο δομικά περίπλοκο αλλά πιο μικρό δίκτυο που θα φτιαχτεί χειροκίνητα. Ο λόγος είναι ότι για τον αλγόριθμο της προσβασιμότητας, σημασία έχει ο αριθμός των διαφορετικών καταστάσεων στις οποίες μπορεί να βρεθεί το δίκτυο.

Χαρακτηριστικά υπολογιστή

Επεξεργαστής	AMD Ryzen 7 4700U with Radeon Graphics 2.00 GHz
RAM	16.0 GB
Τύπος συστήματος	64-bit operating system, x64-based processor
Λειτουργικό σύστημα	Windows 11

Πίνακας 1: Χαρακτηριστικά υπολογιστή

Μετάβαση στο Simulator tab

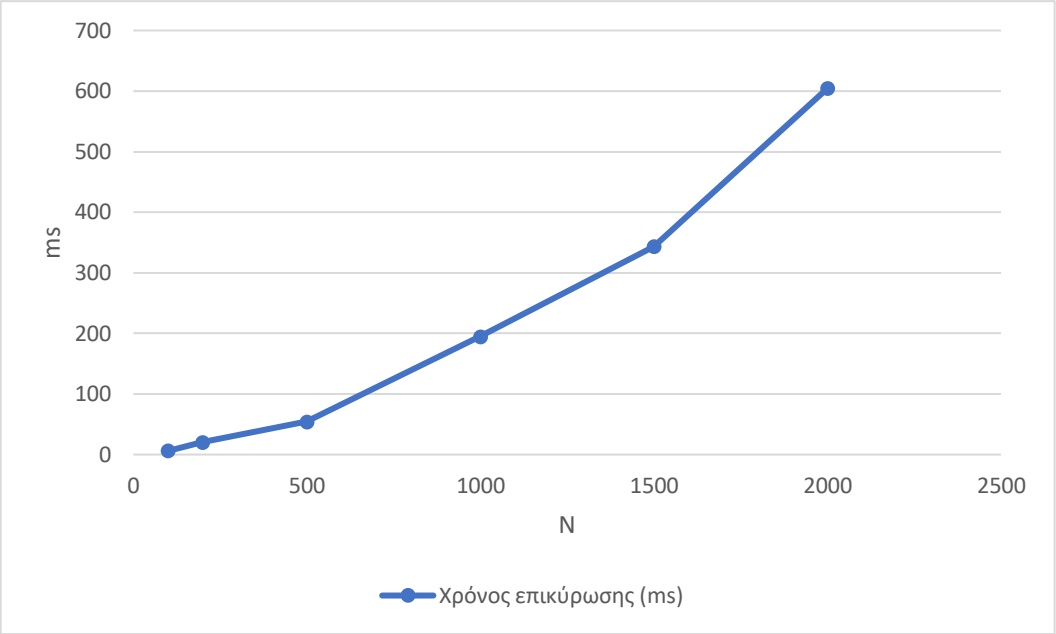
Με την επιλογή του Simulator tab από τον χρήστη, εκτός από τη μεταφορά στο περιβάλλον αυτό, εκτελούνται αρκετές λειτουργείες. Αρχικά, ελέγχεται ότι το δίκτυο είναι καλά ορισμένο και στην περίπτωση που είναι, γίνεται η μεταφορά του δικτύου και των γραφικών διεπαφών στο περιβάλλον της προσομοίωσης διαβάζοντας το από ένα προσωρινό αρχείο. Ακολούθως, εκτελείται ο αλγόριθμος της εύρεσης των επιτρεπτών εμπρόσθιων και αντιστρέψιμων μεταβάσεων.

Στον πιο κάτω πίνακα φαίνονται οι χρόνοι επικύρωσης, εύρεσης των εμπρόσθιων μεταβάσεων καθώς και ο συνολικός χρόνος μετάβασης στο tab. Στις λειτουργίες που λαμβάνουν μέρος είναι και η εύρεση των αντίστροφων μεταβάσεων, αλλά κατά τη μετάβαση δεν υπάρχουν μεταβάσεις που μπορούν να εκτελεστούν προς τα πίσω, λόγο του ότι οι μεταβάσεις έχουν history = 0, έτσι ο χρόνος είναι αμελητέος (ο χρόνος της λειτουργίας αυτής μελετάτε πιο κάτω, αφού εκτελέστηκε κάποια μετάβαση).

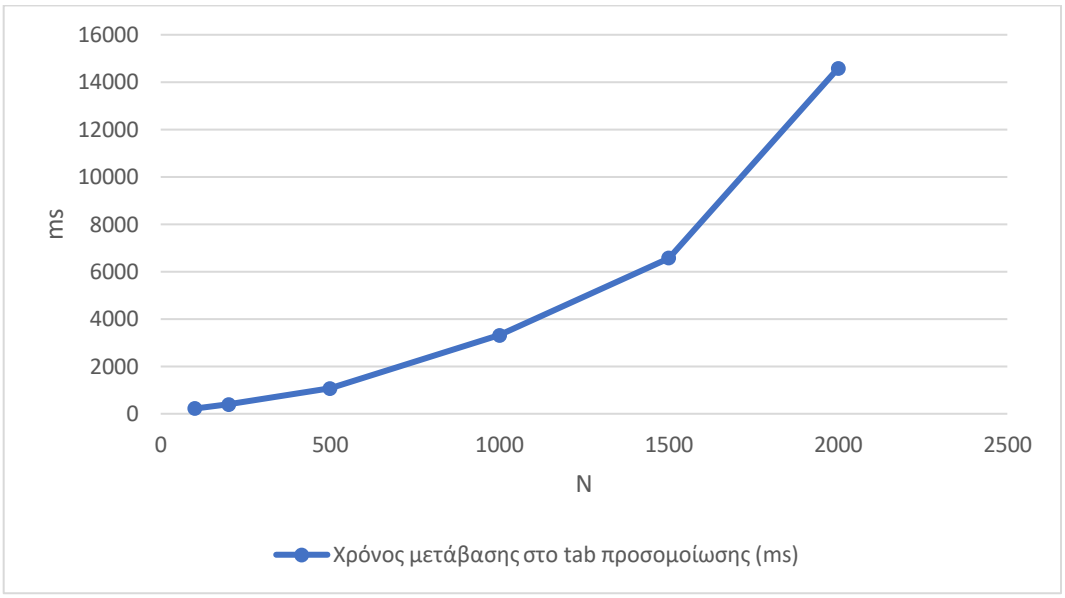
N	**100**	**200**	**500**	**1000**	**1500**	**2000**
Χρόνος μετάβασης στο tab προσομοίωσης (ms)	218.815	391.854	1075.242	3309.871	6573.255	14578.826
Χρόνος επικύρωσης (ms)	6.047	20.224	54.223	194.782	343.591	604.485
Χρόνος εύρεσης εμπρόσθιων μεταβάσεων (ms)	2.525	5.5861	30.031	43.638	55.682	120.139

Πίνακας 2: Χρόνοι για Simulator tab

Όπως φαίνεται και στις γραφικές πιο κάτω, παρατηρείται μια μικρή αύξηση στο χρόνο της επικύρωσης, η οποία αναμενόταν λόγο της αύξησης των ετικετών στα τόξα που ελέγχει. Ο χρόνος εύρεσης εμπρόσθιων μεταβάσεων αυξάνεται πιο ομαλά και μένει σε σχετικά μικρούς χρόνους. Η μεγαλύτερη καθυστέρηση προκαλείτε από το διάβασμα και την αρχικοποίηση των γραφικών διεπαφών. Αυτό οφείλεται στην αύξηση των στοιχείων του δικτύου καθώς και στο ότι το διάβασμα γίνεται από ένα αρχείο. Φαίνεται ότι ο χρόνος επικύρωσης και εύρεσης των μεταβάσεων συμβάλουν ελάχιστα στο συνολικό χρόνο μετάβασης στο tab προσομοίωσης.



Εικόνα 40 : Γραφική παράσταση χρόνου επικύρωσης ώς προς την παράμετρο μεγέθους N



Εικόνα 41 : Γραφική παράσταση χρόνου μετάβασης στο tab προσομοίωσης ώς προς την παράμετρο μεγέθους N

Στον πιο κάτω πίνακα φαίνονται οι χρόνοι εύρεσης των αντίστροφων μεταβάσεων και μετρήθηκαν αφού πρώτα εκτελεστικέ η εμπρόσθια μετάβαση. Παρατηρείται μια μεγάλη διαφορά στους χρόνους από εύρεσης των εμπρόσθιων μεταβάσεων. Αυτό το προσάπτουμε στον τρόπο που είναι δηλωμένες οι ετικέτες των εξερχόμενων τόξων, αφού δεν περιλαμβάνουν όλες τις πληροφορίες που περιλαμβάνουν αυτές των εισερχόμενων τόξων.

Χρόνος εύρεσης αντίστροφων μεταβάσεων (ms)	5.630	18.281	76.437	240.214	451.507	877.720

Πίνακας 3: Χρόνοι για εύρεσης αντίστροφων μεταβάσεων

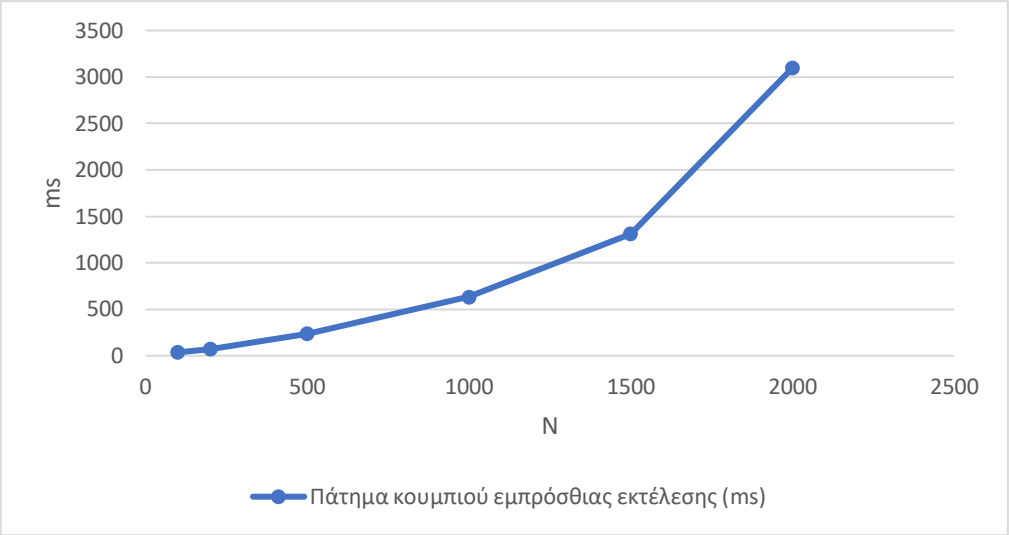
Εκτέλεση εμπρόσθιας/αντίστροφης εκτέλεσης

Με το πάτημα του κουμπιού για εκτέλεση της μετάβασης που επέλεξε ο χρήστης, ανανεώνονται τα στοιχεία και οι γραφικές διεπαφές του δικτύου και επαναϋπολογίζονται οι επιτρεπόμενες εμπρόσθιες και αντίστροφες μεταβάσεις.

N	**100**	**200**	**500**	**1000**	**1500**	**2000**
Πάτημα κουμπιού εμπρόσθιας εκτέλεσης (ms)	36.460	73.069	236.915	632.765	1311.523	3095.865
Χρόνος εκτέλεσης εμπρόσθιας μετάβασης (ms)	0.973	0.961	2.461	5.386	9.507	16.434
Πάτημα κουμπιού αντίστροφης εκτέλεσης (ms)	30.016	40.447	138.040	395.564	934.448	2018.358
Χρόνος εκτέλεσης αντίστροφης μετάβασης (ms)	0.666	1.122	4.434	5.039	8.507	17.951

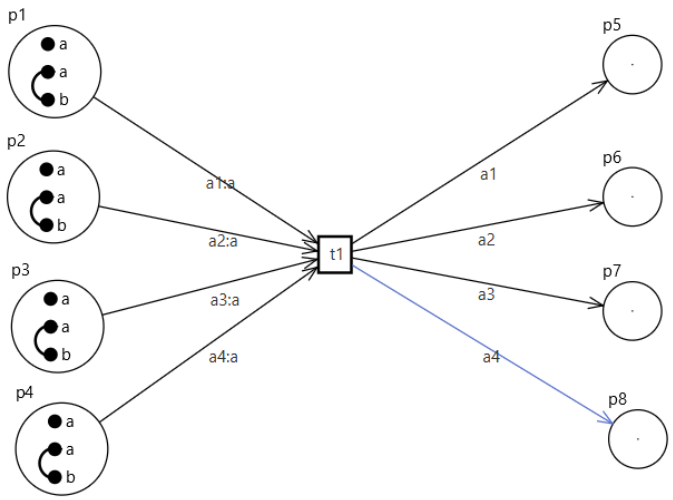
Πίνακας 4: Χρόνοι για εκτέλεση εμπρόσθιας/αντίστροφης εκτέλεσης

Όπως φαίνεται στον πιο κάτω πίνακα, οι χρόνοι για ολοκλήρωση της εμπρόσθιας και αντίθετης εκτέλεσης είναι εκθετικοί και μεταξύ τους είναι πολύ κοντά. Η διαφορά που υπάρχει είναι λόγο της διαφοράς υπολογισμού των αντίστροφων μεταβάσεων που φαίνεται πιο πάνω. Οι χρόνοι εκτέλεσης της μετάβασης είναι αμελητέοι και το μεγαλύτερο χρόνο τον ξοδεύει η αλλαγή των γραφικών διεπαφών. Στην πιο κάτω γραφική παράσταση, φαίνεται ο εκθετικός χρόνος πατήματος του κουμπιού εμπρόσθιας εκτέλεσης.



Εικόνα 42 : Γραφική παράσταση χρόνου εμπρόσθιας εκτέλεσης ώς προς την παράμετρο μεγέθους N

Εύρεση εμπρόσθιων μεταβάσεων για εκθετικά αυξανόμενο αριθμό



Εικόνα 43 : Αυτόματα παραγόμενο δίκτυο 2 με μέγεθος N=4.

Επιλέχθηκε να χρησιμοποιηθούν αυτού του τύπου δίκτυα αφού με την προσθήκη ενός δεσμού a-b σε κάθε θέση, οι επιτρεπτές μεταβάσεις που θα υπάρχουν είναι ίσες με 2^N.

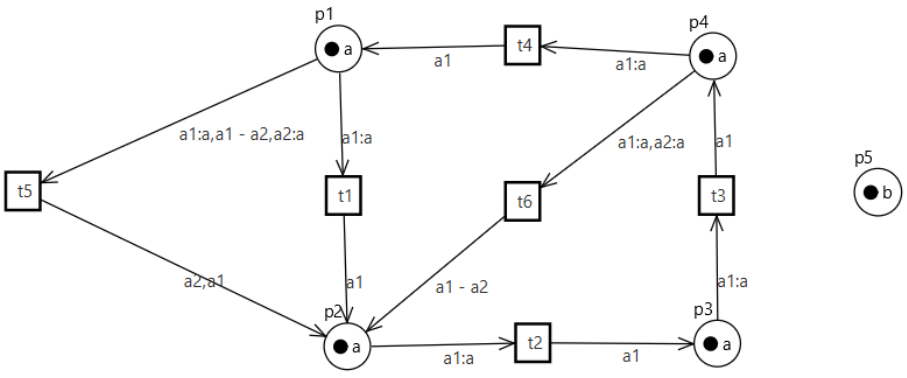
Αριθμός εμπρόσθιων μεταβάσεων (2^N)	**1024 (N=10)**	**32768 (N=15)**	**1048576 (N=20)**
Χρόνος εύρεσης εμπρόσθιων μεταβάσεων (ms)	1.594	37.971	600.006

Πίνακας 5: Χρόνοι για εκτέλεση εμπρόσθιας εκτέλεσης

Όπως φαίνεται στον πίνακα πιο πάνω, ο χρόνος εύρεσης των διαθέσιμων μεταβάσεων παραμένει μικρός ακόμα και σε πολύ μεγάλο αριθμό. Όμως, για N = 24, δηλαδή 16777216 εμπρόσθιες μεταβάσεις, το πρόγραμμα ξεμένει από heap space.

Προσβασιμότητα

Για την περίπτωση της προσβασιμότητας θα χρησιμοποιηθεί το πιο κάτω δίκτυο. Ο λόγος είναι ότι για τον αλγόριθμο της προσβασιμότητας, σημασία έχει ο αριθμός των διαφορετικών καταστάσεων στις οποίες μπορεί να βρεθεί το δίκτυο και έτσι τα προηγούμενα δίκτυα είναι ακατάλληλα. Το δίκτυο αυτό δημιουργήθηκε χειροκίνητα και η σκέψη πίσω του είναι να μπορούν οι μάρκες να κινούνται σε όλες τις θέσεις για να υπάρχουν πολλές καταστάσεις. Οι χρόνοι πάρθηκαν για τη χειρότερη δυνατή περίπτωση, δηλαδή όταν το μαρκάρισμα που δίδεται από τον χρήστη δεν είναι προσβάσιμο άρα ο αλγόριθμος θα περάσει από όλες τις δυνατές καταστάσεις (ο λόγος ύπαρξης της θέσης p5, με μάρκα τύπου b όπου δεν μπορεί να μετακινηθεί).



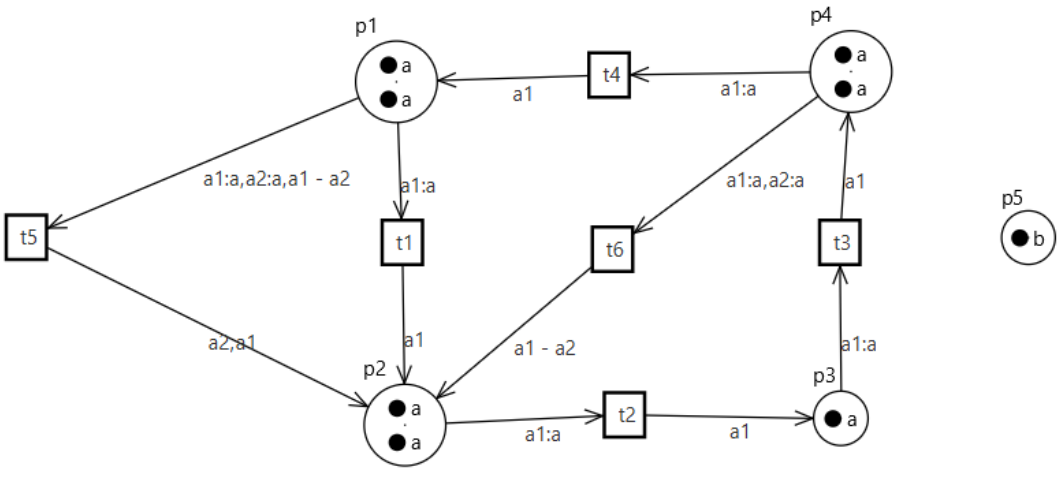
Εικόνα 44 : Δίκτυο 1 για έλεγχο προσβασιμότητας

Αριθμός διαφορετικών καταστάσεων	35
Αριθμός συνολικών καταστάσεων	154
Χρόνος εκτέλεσης αλγορίθμου (ms)	8.964
Χρόνος για έλεγχο καταστάσεων που ήδη επισκέφτηκε (ms)	2.652
Χρόνος για έλεγχο καταστάσεων που ήδη επισκέφτηκε (%)	29.5%

Πίνακας 6: Χρόνοι προσβασιμότητας 1

Όπως φαίνεται στον πίνακα, αρχικά το δίκτυο έχει 35 διαφορετικές καταστάσεις και 154 καταστάσεις όπου δημιουργήθηκαν και εκλέχθηκαν από το σύστημα (δηλαδή 119 καταστάσεις ήταν ίδιες με μια που είχε ήδη επεξεργαστεί και αγνοήθηκαν από το σύστημα). Ακόμα, φαίνονται οι χρόνοι για ολοκλήρωση του αλγορίθμου και έλεγχο καταστάσεων που ήδη επισκέφτηκε. Μας ενδιαφέρουν οι συνολικές καταστάσεις που δημιουργούνται γιατί, αντίθετα με την προηγούμενη έκδοση των αντιστρέψιμων δικτύων Πέτρι, με την προσθήκη πολλαπλών μαρκών ιδίου τύπου, ο έλεγχος για εξακρίβωση αν ο αλγόριθμος έχει ξανά επισκεφτεί μια κατάσταση δεν μπορεί να γίνει αυτόματα και πρέπει να ελέγχονται όλες οι προηγούμενες καταστάσεις. Αυτό έχει ως αποτέλεσμα να αυξάνεται ο χρόνος του αλγορίθμου.

Με την προσθήκη δύο μαρκών στο δίκτυο, οι καταστάσεις αυξάνονται αρκετά φτάνοντας στις 187 διαφορετικές και 1158 συνολικές. Το ποσοστό του χρόνου ελέγχου φτάνει στο 68.3%.

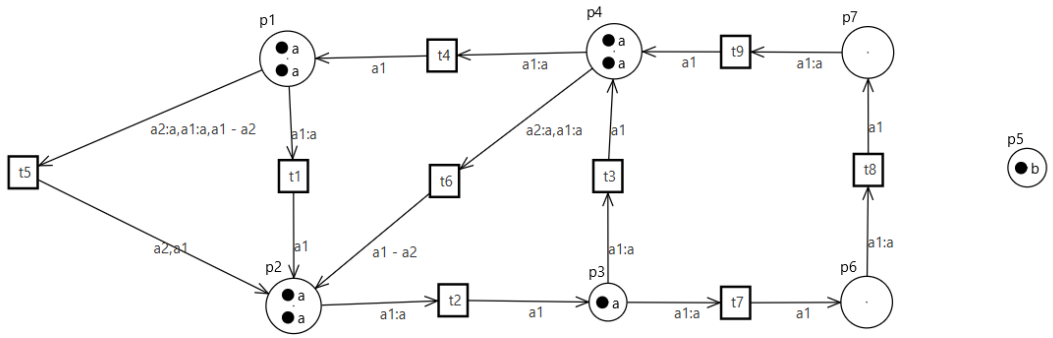


Εικόνα 45 : Δίκτυο 2 για έλεγχο προσβασιμότητας

Αριθμός διαφορετικών καταστάσεων	187
Αριθμός συνολικών καταστάσεων	1158
Χρόνος εκτέλεσης αλγορίθμου (ms)	49.224
Χρόνος για έλεγχο καταστάσεων που ήδη επισκέφτηκε (ms)	33.648
Χρόνος για έλεγχο καταστάσεων που ήδη επισκέφτηκε (%)	68.3%

Πίνακας 7: Χρόνοι προσβασιμότητας 2

Προσθέτοντας ακόμα δύο θέσει και τρείς μεταβάσεις, όπως φαίνεται στην πιο κάτω εικόνα, οι καταστάσεις αυξάνονται εκθετικά φτάνοντας τις 2888 διαφορετικές και 24569 συνολικές. Οι χρόνοι ακολουθούν και αυτοί εκθετική άνοδο με τον χρόνο ελέγχου να είναι υπεύθυνος για το μεγαλύτερο ποσοστό του συνολικού χρόνο, στο 91.6%.

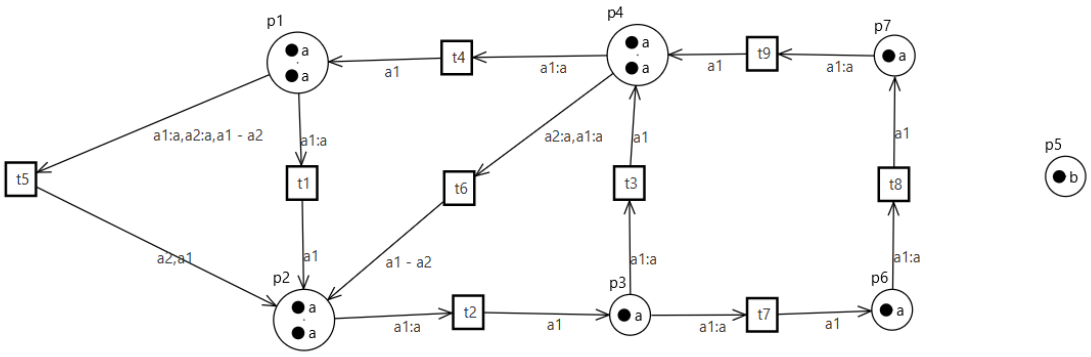


Εικόνα 46 : Δίκτυο 3 για έλεγχο προσβασιμότητας

Αριθμός διαφορετικών καταστάσεων	2888
Αριθμός συνολικών καταστάσεων	24569
Χρόνος εκτέλεσης αλγορίθμου (ms)	6847.027
Χρόνος για έλεγχο καταστάσεων που ήδη επισκέφτηκε (ms)	6271.617
Χρόνος για έλεγχο καταστάσεων που ήδη επισκέφτηκε (%)	91.6%

Πίνακας 8: Χρόνοι προσβασιμότητας 3

Τέλος, προσθέτοντας ακόμα δύο μάρκες, όπως φαίνεται στην πιο κάτω εικόνα, οι καταστάσεις και οι χρόνοι αυξάνονται σε σημείο που δεν είναι πλέον αποδεκτό.

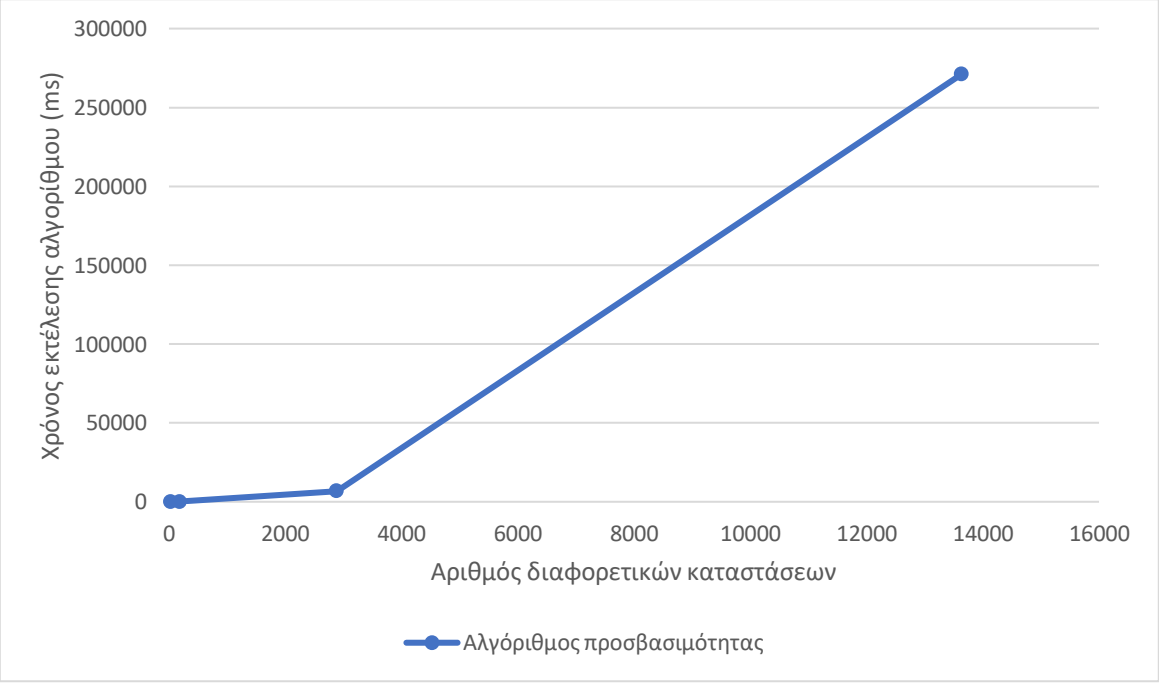


Εικόνα 47 : Δίκτυο 4 για έλεγχο προσβασιμότητας

Αριθμός διαφορετικών καταστάσεων	13637
Αριθμός συνολικών καταστάσεων	141627
Χρόνος εκτέλεσης αλγορίθμου (ms)	271099.12
Χρόνος για έλεγχο καταστάσεων που ήδη επισκέφτηκε (ms)	237164.88
Χρόνος για έλεγχο καταστάσεων που ήδη επισκέφτηκε (%)	87.5%

Πίνακας 9: Χρόνοι προσβασιμότητας 4

Συμπερασματικά, ο αλγόριθμος προσβασιμότητας έχει εκθετικό χρόνο και φτάνει πολύ νωρίς σε μη αποδεκτά όρια. Ακόμα, κάθε προσθήκη οποιουδήποτε νέου στοιχείου μπορεί να αυξήσει τις καταστάσεις σε πολύ μεγάλο βαθμό.



Εικόνα 48 : Γραφική παράσταση χρόνου προσβασιμότητας ώς προς τον αριθμό των καταστάσεων

Κεφάλαιο 6

Συμπεράσματα – Μελλοντική Εργασία

6.1 Συμπεράσματα

Η διπλωματική αυτή ασχολείται με τα Αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου (MRPN), τα οποία είναι ένα μαθηματικό μοντέλο που επεκτείνει τα Αναστρέψιμα δίκτυα Πέτρι (RPN) με την προσθήκη πολλαπλών μαρκών ιδίου τύπου. Δημιουργήθηκε ένα γραφικό εργαλείο, βασισμένο σε εργαλείο προηγούμενης εργασίας για τα Αναστρέψιμα δίκτυα Πέτρι, για τη δημιουργία, ανάλυση και προσομοίωση των δικτύων. Τα Αναστρέψιμα δίκτυα Πέτρι με πολλαπλές μάρκες ιδίου τύπου είναι πρόσφατα αναδυόμενο μοντέλο και έτσι δεν υπάρχουν εργαλεία για δημιουργία και προσομοίωσης τους. Σημαντικό να συμβιωθεί ότι το εργαλείο που δημιουργήθηκε χρίζει αρκετής βελτίωσης και υπάρχουν αρκετές λειτουργείες με τις οποίες μπορεί να επεκταθεί.

Υπήρχαν αρκετές προκλήσεις που έπρεπε να αντιμετωπιστούν κατά τη διάρκεια της εργασίας αυτής. Αρχικά, έπρεπε να κατανοηθούν οι έννοιες των Αναστρέψιμων δικτύων Πέτρι με πολλαπλές μάρκες ιδίου τύπου. Έπειτα, λόγο του ότι το εργαλείο είναι επέκταση προηγούμενου εργαλείου, μεγάλη προσπάθεια έπρεπε να καταβληθεί για κατανόηση της δομής και του κώδικα ενός αρκετά μεγάλου έργου. Επιπρόσθετα, μεγάλος βαθμός δυσκολίας υπήρχε στην μετατροπή της δομής για να ικανοποιεί της ανάγκες των Αναστρέψιμων δικτύων Πέτρι με πολλαπλές μάρκες ιδίου τύπου καθώς και η υλοποίηση αρκετά πιο περίπλοκων αλγορίθμων.

6.2 Μελλοντική Εργασία

Όπως προαναφέρθηκε, υπάρχουν αρκετά περιθώρια βελτίωσης του συστήματος. Κάποια από τα πιο βασικά είναι η βελτιστοποίηση του αλγορίθμου της προσβασιμότητας με σκοπό τη μείωση της πολυπλοκότητας του. Ακόμα, η εφαρμογή θα μπορούσε να γίνει πιο δια δραστική, με δυνατότητες τροποποίησης του δικτύου και στα άλλα δύο tabs εκτός από το editor tab. Επιπρόσθετα, θα μπορούσε να υλοποιηθεί αλγόριθμος ο οποίος να διαβάζει ένα XML αρχείο για τη δημιουργία του δικτύου, χωρίς να χρειάζεται τις θέσεις

των στοιχείων στον καμβά. Με αυτό τον τρόπο θα είναι πιο εύκολο η αυτόματη δημιουργία δικτύων από τρίτο πρόγραμμα.

Επίσης, η εφαρμογή μπορεί να επεκταθεί με την υλοποίηση όλων των ιδιοτήτων των Αναστρέψιμων δικτύων Πέτρι με πολλαπλές μάρκες ιδίου τύπου και την προσθήκη τους στο Properties tab, όπως τον έλεγχο για το αν δύο μεταβάσεις είναι ανεξάρτητες [1].

Τέλος, το μοντέλο αναστρέψιμων δικτύων Πέτρι με πολλαπλές μάρκες ιδίου τύπου μπορεί να επεκταθεί με την προσθήκη συνθηκών στις μεταβάσεις που θα ελέγχονται στην εμπρόσθια και αντίστροφη εκτέλεση [1].

Βιβλιογραφία

[1] A. Philippou and K. Psara, A collective interpretation semantics for reversing Petri nets. Theoretical Computer Science, In Press (2022)

[2] A. Philippou and K. Psara, Reversible computation in nets with bonds. Journal of Logical and Algebraic Methods in Programming 124: 100718 (2022)

[3] A. Philippou and K. Psara, "Reversible computation in petri nets," in International Conference on Reversible Computation, vol. 11106, pp. 84-101, 2018.

[4] I. Phillips, I. Ulidowski, and S. Yuen. A reversible process calculus and the modelling of the ERK signalling pathway. In Proceedings of RC 2012 Revised Papers, LNCS 7581, pages 218–232. Springer, 2012.

[5] S. Kuhn and I. Ulidowski. A calculus for local reversibility. In Proceedings of RC 2016, LNCS 9720, pages 20–35. Springer, 2016.

[6] J. Lee et al, "Reversible computation in asynchronous cellular automata," in International Conference on Unconventional Methods of Computation, 2002.

[7] T. Murata, "Petri nets: Properties, analysis and applications," Proc IEEE, vol. 77, (4), pp. 541-580, 1989.

[8] G. Callou, P. Maciel, D. Tutsch, J. Arajo, J. Ferreira and R. Souz, "A Petri Net-Based Approach to the Quantification of Data Center Dependability", Petri Nets - Manufacturing and Computer Science, 2012.

[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] Welcome to Apache Maven. Available: http://maven.apache.org/.

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

[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] Ε. Παναγή, "Ανάπτυξη γραφικού εργαλείου για την επεξεργασίας και προσομοίωσης αναστρέψιμων δικτύων Πέτρι", Διατριβή, Τμήμα Πληροφορικής Πανεπιστημίου Κύπρου, 2020.

Παράρτημα Α

Στην πιο κάτω εικόνα, παρουσιάζεται η δομή xml που αντιστοιχεί σε ένα αποθηκευμένο αναστρέψιμο δίκτυο Πέτρι με πολλαπλές μάρκες ιδίου τύπου.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<mrpn>
    <places>
        <place>
            <name>p1</name>
            <x>126.40000000000003</x>
            <y>144.00000000000006</y>
            <tokens>
                <token>
                    <id>b1</id>
                    <type>a</type>
                </token>
                <token>
                    <id>b2</id>
                    <type>a</type>
                </token>
            </tokens>
        </place>
        <place>
    </places>
    <transitions>
        <transition>
            <name>t1</name>
            <x>291.9999876022339</x>
            <y>88.79999160766602</y>
        </transition>
    </transitions>
```

Εικόνα 49: xml μορφή αποθηκευμένου MRPN

Ξεκινά με το στοιχείο ρίζα (mrpn), όπου χωρίζεται σε τέσσερα παιδία που δηλώνουν τα δομικά στοιχεία του δικτύου (places, transitions, arrows και totalBonds). Αρχικά, το στοιχείο παιδί places περιέχει ως παιδιά του όλες τις θέσεις του δικτύου. Κάθε θέσει αποτελείται από το όνομα (name), τη θέση της (x και y) στον καμβά και όλες τις μάρκες (tokens) που βρίσκονται σε αυτή τη θέση. Οι μάρκες με τη σειρά τους, περιέχουν το αναγνωριστικό (id) που χρησιμοποιείται από τις λειτουργίες του συστήματος και τον τύπο (type) τους. Το επόμενο στοιχείο παιδί της ρίζας είναι οι μεταβάσεις (transitions), όπου περιέχει όλες τις μεταβάσεις του δικτύου. Κάθε μετάβαση αποτελείται από το όνομα (name) και τη θέση της (x και y) στον καμβά.

```
    <arrows>
        <arrow>
            <source>p1</source>
            <destination>t1</destination>
            <label>
                <tokens>
                    <token>
                        <id>a1</id>
                        <type>a</type>
                    </token>
                    <token>
                        <id>a2</id>
                        <type>a</type>
                    </token>
                </tokens>
                <bonds>
                    <bond>
                        <token>a1</token>
                        <token>a2</token>
                    </bond>
                </bonds>
            </label>
        </arrow>
        <arrow>
    </arrows>
    <totalBonds>
        <bond>
            <token>b1</token>
            <token>b2</token>
        </bond>
    </totalBonds>
</mrpn>
```

Εικόνα 50: xml μορφή αποθηκευμένου MRPN

Στη συνέχεια, το επόμενο παιδί είναι τα τόξα (arrows) όπου περιέχουν όλα τα τόξα του δικτύου. Κάθε τόξο αποτελείται από τον κόμβο αφετηρίας (source), τον κόμβο προορισμού (destination) και την ετικέτα (label) του. Η ετικέτα περιέχει τις μεταβλητές των μάρκων και των δεσμών, όπου κάθε μάρκα αποτελείτε από το όνομα της μεταβλητής (id) και τον τύπο (type) της και κάθε δεσμός από τα δύο όνομα των μεταβλητών που τον αποτελούν. Το τελευταίο παιδί της ρίζας είναι οι δεσμοί του δικτύου (totalBonds) όπου περιέχει όλους τους δεσμούς. Κάθε ένας αποτελείται από τα αναγνωριστικά των δύο μαρκών που τον αποτελούν.

Παράρτημα Β

Ο κώδικας είναι αναρτημένος στη σελίδα github, στην ηλεκτρονική διεύθυνση:

https://github.com/riacov01/MRPN-UI