Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Μοντελοποίηση Ασαφών Συστημάτων με την χρήση Δικτύων Petri** **Modeling Fuzzy Systems using Petri Nets**
Ονοματεπώνυμο Φοιτητή	**Ιωάννης Σαπάκος**
Πατρώνυμο	**Γρηγόριος**
Αριθμός Μητρώου	**ΜΠΠΛ 13069**
Επιβλέπων	**Θέμης Παναγιωτόπουλος, Καθηγητής**

Ημερομηνία Παράδοσης **Νοέμβριος 2015**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

Περίληψη

Τα *ασαφή σύνολα* (π.χ. ψηλοί άνθρωποι, μεγάλοι αριθμοί) αποτελούν έναν αρκετά χρήσιμο τρόπο έκφρασης και μοντελοποίησης του πραγματικού κόσμου. Η *ασαφής λογική* αποτελεί μία κατηγορία λογικής στην οποία οι λογικές μεταβλητές δεν έχουν ως πεδίο ορισμού το σύνολο {0,1} αλλά μπορούν να πάρουν οποιαδήποτε τιμή στο πραγματικό διάστημα [0,1]. Η τιμή αληθείας ονομάζεται βαθμός συμμετοχής και εκφράζει τον βαθμό συμμετοχής της μεταβλητής σε ένα ασαφές σύνολο. Αν οι μεταβλητές αυτές συμμετέχουν σε κανόνες της μορφής if-then, τότε ονομάζουμε τους εν λόγω κανόνες ασαφείς κανόνες παραγωγής (fuzzyrules). Τα συστήματα τα οποία βασίζονται σε ένα σύνολο ασαφών κανόνων παραγωγής. Τα ασαφή συστήματα είναι ιδιαίτερα διαδεδομένα και βρίσκουν αρκετές εφαρμογές στον σύγχρονο κόσμο.

Τα δίκτυα Petri αποτελούν ένα πανίσχυρο εργαλείο μοντελοποίησης φορμαλισμών και δυναμικών συστημάτων. Τα δίκτυα αυτά συνδυάζουν μία καλά ορισμένη μαθηματική θεωρία μαζί με μία γραφική αναπαράσταση με την μορφή κατευθυνόμενου γραφήματος. Εκτός αυτού η κατάστασή τους είναι δυναμική και αλλάζει σε κάθε χρονική στιγμή.

Παρά το γεγονός του ότι τα δίκτυα Petri είναι κατάλληλα για την μοντελοποίηση δυναμικών συστημάτων, η πολυπλοκότητα ορισμένων δυναμικών συστημάτων ή οι ιδιάζουσες απαιτήσεις τους κατέστησαν αναγκαίο τον ορισμό περισσότερων ειδών δικτύων Petri. Για παράδειγμα, η μοντελοποίηση ενός ασαφούς συστήματος, το οποίο στηρίζεται σε ένα σύνολο ασαφών κανόνων είναι εξαιρετικά περίπλοκη. Έτσι λοιπόν, στόχος της παρούσας διπλωματικής εργασίας είναι η παρουσίαση και η μελέτη των Ασαφών Δικτύων Petri (FuzzyPetriNets – FPN) με τα οποία είναι δυνατή η μοντελοποίηση ασαφών συστημάτων. Πρακτικά, αυτό το οποίο θα επιτύχουμε είναι η αναπαράσταση και η μοντελοποίηση ενός συνόλου ασαφών κανόνων με την χρήση ενός FPN. Εκτός από την παρουσίαση και την επιστημονική τεκμηρίωση των FPN, αναπτύχθηκε επίσης λογισμικό (με την ονομασία PetriNetSim) το οποίο δίνει την δυνατότητες σχεδιασμού ενός FPN αλλά και προσομοίωσής του.

(Abstract)

Fuzzy sets (e.g. tall people, large numbers) are a quite useful way of expression and modeling of the real world. Fuzzy logic is a type of logic to which the logical variables do not have as domain the set {0,1} but can take any value on the real interval [0,1]. This value is called membership degree and expresses the level of participation of the variable in a fuzzy set. If these variables are involved in if-then rule, then we these rules are calledfuzzy production rules (fuzzy rules). Any system that is based on a set of fuzzy production rules is called Fuzzy System. Fuzzy systems are particularly widespread and find many applications in the modern world.

Petri nets are a powerful formalism modeling tool and dynamical systems. These networks combine a well-defined mathematical background along with a graphical representation in the form of directed graph. Besides that, their state is dynamic and can change during time.

Despite the fact that Petri nets are suitable for dynamic system modeling, the complexity of some dynamic systems is very big. Thus, modeling a fuzzy system, which is based on a set of fuzzy rules issome times extremely complicated. This thesis is focused on the presentation and the study of Fuzzy Networks Petri (Fuzzy Petri Nets - FPN) with which it is possible to modeling fuzzy systems. Moreover, we also developed a software (called PetriNetSim) which offrers design and simulation possibilities of a FPN.

Περιεχόμενα

Κεφάλαιο 1 – Εισαγωγή

1.1 Ασαφής λογική και δίκτυα Petri

Η ασαφής λογική αποτελεί μία κατηγορία λογικής στην οποία οι τιμές αληθείας των μεταβλητών που συμμετέχουν σε μία έκφραση δεν παίρνουν ως τιμές μόνο μέσα από το σύνολο τιμών $\{0,1\}$ αλλα μπορούν να πάρουν οποιαδήποτε τιμή στο διάστημα $[0,1]$. Η τιμή αυτή ονομάζεται *βαθμός συμμετοχής* ή *βαθμός αληθείας* και εκφράζει την βεβαιότητα συμμετοχής της μεταβλητής σε ένα *ασαφές σύνολο*, τα οποία είναι πολύ χρήσιμα για την μοντελοποίηση ομάδων αντικειμένων του πραγματικού κόσμου. Είναι αρκετά συχνό στον πραγματικό κόσμο οι κλάσεις (ομάδες ή σύνολα) διάφορων αντικειμένων δεν έχουν αυστηρά καθορισμένα κριτήρια συμμετοχής για τα μέλη τους. Ένα παράδειγμα ασαφούς συνόλου είναι η κλάση των ζώων, η οποία προφανώς περιέχει αντικείμενα όπως σκύλους, γάτες, άλογα, πουλιά κτλ. και προφανώς επίσης δεν περιέχει αντικείμενα όπως πέτρες, υγρά, φυτά, κτλ. Όπως υπάρχουν αντικείμενα όπως οι αστερίες, τα βακτήρια κτλ. των οποίων η συμμετοχή ή μη στην κλάση των ζώων είναι είτε σκιώδης είτε ασαφής. Ένα άλλο παράδειγμα μιας τέτοιας ασαφούς συμμετοχής ενός στοιχείου σε ένα σύνολο είναι η συμμετοχή του αριθμού 10 στο σύνολο των *«πραγματικών αριθμών που είναι πολύ μεγαλύτεροι από τον αριθμό 1»*.Τα ασαφή σύνολα και γενικότερα η ασαφής λογική προτάθηκαν από τον Zadeh το 1965 [1].

Τα συστήματα τα οποία βασίζονται στην ασαφή λογική, ονομάζονται *ασαφή συστήματα (fuzzysystems)*. Για την ακρίβεια ένα ασαφές σύστημα στηρίζεται σε ένα σύνολο *ασαφών κανόνων παραγωγής (fuzzy-setrules)*. Ένας ασαφής κανόνας παραγωγής είναι ένας κανόνας της μορφής if-then, ο οποίος έχει τόσο συνθήκες όσο και συμπεράσματα. Η διαφορά του με έναν συμβατικό κανόνα παραγωγής έγκειται στο ότι τόσο οι συνθήκες όσο και τα συμπεράσματά του χαρακτηρίζονται από έναν βαθμό αληθείας. Τα ασαφή συστήματα είναι ιδιαίτερα διαδεδομένα και βρίσκουν αρκετές εφαρμογές στον σύγχρονο κόσμο.

Τα *δίκτυα Petri (PetriNets)* προτάθηκαν για πρώτη φορά το 1962 από το CarlAdamPetri [3]. Τα δίκτυα Petriαποτελούν ένα πανίσχυρο εργαλείο μοντελοποίησης φορμαλισμών και συστημάτων όχι μόνο στην επιστήμη των υπολογιστών αλλά και γενικότερα. Τα δίκτυα αυτά συνδυάζουν μία καλά ορισμένη μαθηματική θεωρία μαζί με μία γραφική αναπαράσταση με την μορφή κατευθυνόμενου γραφήματος. Εκτός αυτού η κατάστασή τους είναι δυναμική και αλλάζει σε κάθε χρονική στιγμή. Η θεωρητική φύση των δικτύων Petrιεπιτρέπει μία λεπτομερή μοντελοποίηση και ανάλυση ενός συστήματος ενώ η γραφική τους φύση επιτρέπει την οπτικοποίησή του, μαζί με τις αλλαγές του στον χρόνο. Ο συνδυασμός αυτός αποτελεί τον κύριο λόγο επιτυχίας και διάδοσης των δικτύων Petri.

1.2 Εφαρμογές των ασαφών συστημάτων

Τα ασαφή συστήματα μπορούν να χρησιμοποιηθούν στην παραγωγή εκτιμήσεων πάνω στη λήψη αποφάσεων (decisionsupportsystems). Είναι επίσης πολύ δημοφιλή σε μηχανικά συστήματα ελέγχου, όπως για παράδειγμα σε συστήματα κλιματισμού, ελέγχου της ταχύτητας οχημάτων, συστήματα αεροσκαφών, αυτοματισμούς σε έξυπνα σπίτια, όπως επίσης και σε ελεγκτές βιομηχανικών διαδικασιών.

Η επιτυχία που γνώρισαν τα ασαφή συστήματα οφείλεται κυρίως σε δύο βασικές αδυναμίες των συμβατικών συστημάτων: (α) πολλές διαδικασίες είναι υπερβολικά περίπλοκες και είναι δύσκολο να μοντελοποιηθούν μαθηματικά. Τέτοια παραδείγματα διαδικασιών είναι συχνά σε συστήματα διοίκησης, επιχειρήσεων καθώς και τηλεπικοινωνιών και (β) ο όρος σταθερότητα δεν είναι εύκολο να περιγραφεί ακόμα και για τα παραδοσιακά συστήματα.

Σε γενικές γραμμές υπάρχουν πέντε διαφορετικοί τύποι συστημάτων όπου η ασαφής λογική είναι απαραίτητη, ή τουλάχιστον ωφέλιμη. Πιο συγκεκριμένα, αυτά είναι: (α) πολύπλοκα Συστήματα που είναι δύσκολο ή αδύνατο να μοντελοποιηθούν με συμβατικές μαθηματικές μεθόδους, (β) συστήματα με περίπλοκα και συνεχή δεδομένα τόσο εισόδου όσο και εξόδου, (γ) συστήματα τα οποία ελέγχονται από ειδικούς εμπειρογνώμονες και περιλαμβάνουν μία

πληθώρα κανόνων και διαδικασιών, (δ) συστήματα που κάνουν χρήση της ανθρώπινης παρατήρησης ως δεδομένα εισόδου ή ως βάση για διατύπωση κανόνων και (ε) συστήματα που εμπεριέχουν εξ' ορισμού βαθμό ασάφειας. Για παράδειγμα συστήματα που σχετίζονται με οικονομικές και κοινωνικές επιστήμες.

1.3 Στόχος της παρούσας εργασίας

Παρά το γεγονός του ότι τα δίκτυα Petrieίναι κατάλληλα για την μοντελοποίηση δυναμικών συστημάτων, η πολυπλοκότητα ορισμένων δυναμικών συστημάτων ή οι ιδιάζουσες απαιτήσεις τους κατέστησαν αναγκαίο τον ορισμό περισσότερων ειδών δικτύων Petri. Για παράδειγμα, η μοντελοποίηση ενός ασαφούς συστήματος, το οποίο στηρίζεται σε ένα σύνολο ασαφών κανόνων είναι εξαιρετικά περίπλοκη.

Έτσι λοιπόν, στόχος της παρούσας διπλωματικής εργασίας είναι η παρουσίαση και η μελέτη των *Ασαφών ΔικτύωνPetri (FuzzyPetriNets – FPN)* με τα οποία είναι δυνατή η μοντελοποίηση ασαφών συστημάτων. Πρακτικά, αυτό το οποίο θα επιτύχουμε είναι η αναπαράσταση και η μοντελοποίηση ενός συνόλου ασαφών κανόνων με την χρήση ενός FPN. Στο σημείο αυτό θα πρέπει να σημειωθεί ότι υπάρχουν πολλοί ορισμοί για την δομή και την λειτουργία των FPN [16] [17] [18] [19] [20]. Οι ορισμοί αυτοί διαφέρουν αρκετά μεταξύ τους και ενώ αρκετές φορές είναι και αντιφατικοί. Στην εν λόγω μελέτη θα αποσαφηνίσουμε πλήρως την έννοια των FPN και θα δώσουμε έναν αυστηρό μαθηματικό ορισμό για αυτά. Εκτός από την παρουσίαση και την επιστημονική τεκμηρίωση των FPN, θα αναπτυχθεί επίσης λογισμικό (με την ονομασία PetriNetSim) το οποίο θα δίνει την δυνατότητες σχεδιασμού ενός FPNαλλά και προσομοίωσής του. Πιο συγκεκριμένα, τα βήματα τα οποία θα ακολουθήσουμε για την επίτευξη του ζητούμενου στόχου είναι τα παρακάτω:

1. Βιβλιογραφική έρευνα στην ασαφή λογική και τα ασαφή συστήματα (Κεφάλαιο 2).
2. Βιβλιογραφική έρευνα στα δίκτυα Petri (Κεφάλαιο 3).
3. Ορισμός των ασαφών δικτύων Petri (FPN) (Κεφάλαιο 4).
4. Υλοποίηση λογισμικού στο οποίο είναι δυνατή η σχεδίαση και η προσομοίωση ασαφών δικτύων Petri (Κεφάλαια 5, 6 και 7).
5. Συμπεράσματα και μελλοντικές επεκτάσεις (Κεφάλαια 8 και 9).

Κεφάλαιο 2 – Ασαφή συστήματα, τι είναι και που τα εφαρμόζουμε

2.1 Η έννοια της ασαφούς λογικής

Είναι αρκετά συχνό στον πραγματικό κόσμο οι κλάσεις (ομάδες ή σύνολα) διάφορων αντικειμένων δεν έχουν αυστηρά καθορισμένα κριτήρια συμμετοχής για τα μέλη τους. Για παράδειγμα η κλάση των ζώων προφανώς περιέχει αντικείμενα όπως σκύλους, γάτες, άλογα, πουλιά κτλ. και προφανώς επίσης δεν περιέχει αντικείμενα όπως πέτρες, υγρά, φυτά, κτλ. Όπως υπάρχουν αντικείμενα όπως οι αστερίες, τα βακτήρια κτλ. των οποίων η συμμετοχή ή μη στην κλάση των ζώων είναι είτε σκιώδης είτε ασαφής. Ένα άλλο παράδειγμα μιας τέτοιας ασαφούς συμμετοχής ενός στοιχείου σε ένα σύνολο είναι η συμμετοχή του αριθμού 10 στο σύνολο των *«πραγματικών αριθμών που είναι πολύ μεγαλύτεροι από τον αριθμό 1»*.

Είναι προφανές λοιπόν ότι σύνολα όπως *«οι πραγματικοί αριθμοί που είναι πολύ μεγαλύτεροι από τον αριθμό 1»*, ή *«το σύνολο των όμορφων γυναικών»* ή *«το σύνολο των ανδρών που είναι ψηλοί»* δεν είναι σύνολα τα οποία έχουν έναν αυστηρό μαθηματικό ορισμό. Η απουσία του αυστηρού αυτού μαθηματικού ορισμού καθιστά δύσκολη την απόφαση για το αν ένα στοιχείο ανήκει ή όχι σε κάποιο από τα σύνολα αυτά. Παρά το γεγονός του ασαφούς ή πολλές φορές του ελλιπούς ορισμού των συνόλων αυτών, τέτοιου είδους σύνολα παίζουν έναν σημαντικό ρόλο στην ανθρώπινη καθημερινότητα και στις ανθρώπινες δραστηριότητες γενικότερα.

Στην συνέχεια του κεφαλαίου αυτού, θα ορίσουμε με μαθηματικό τρόπο την έννοια του *ασαφούς συνόλου (fuzzyset)*και στην συνέχεια θα ορίσουμε την έννοια της *ασαφούς λογικής (fuzzylogic)*και των *ασαφών συνόλων (fuzzysets)*.

2.2 Ασαφή σύνολα

Τα ασαφή σύνολα και γενικότερα η ασαφής λογική προτάθηκαν από τον Zadeh το 1965 [1]. Στην κλασική θεωρία συνόλων, λέμε ότι ένα σύνολο A αποτελεί *υποσύνολο* ενός συνόλου U όταν κάθε στοιχείο του συνόλου A εμπεριέχεται επίσης και στο σύνολο U. Το ότι το σύνολο A αποτελεί υποσύνολο του συνόλου U συμβολίζεται ως εξής: $A \subseteq U$. Στην ουσία δηλαδή ισχύει ότι:

$$A \subseteq U \Leftrightarrow (\forall x \in A \Rightarrow x \in U)$$

Υπάρχει επίσης η παραδοχή του ότι *το κενό σύνολο αποτελεί υποσύνολο κάθε συνόλου*U, δηλαδή ότι $\emptyset \subseteq U$.

Γνωρίζοντας το ότι ένα σύνολο A αποτελεί υποσύνολο ενός συνόλου U μπορούμε να ελέγξουμε ποια στοιχεία του A ανήκουν στο U, με την χρήση μιας συνάρτησης $X_A : U \to \{0,1\}$, η οποία ορίζεται ως εξής:

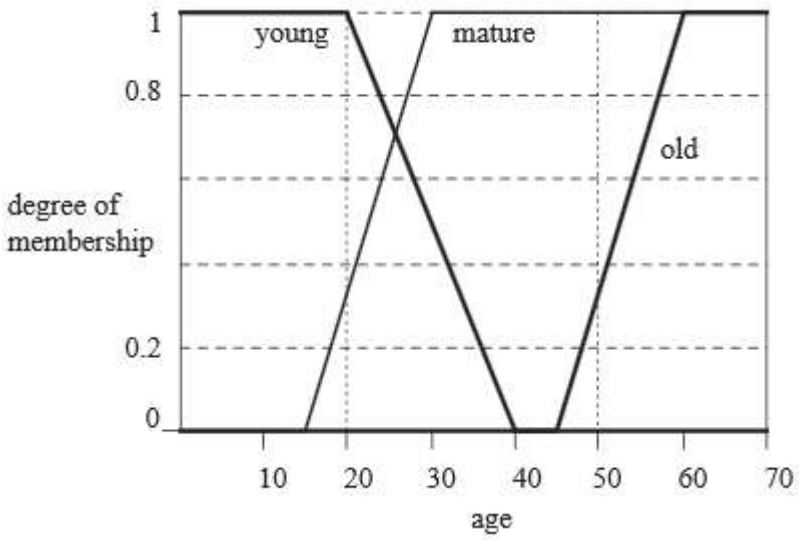
$$X_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$$

Ο παραπάνω ορισμός από την κλασική θεωρία των συνόλων αλλάζει στην θεωρία των ασαφών συνόλων. Πιο συγκεκριμένα, λέμε ότι ένα σύνολο A είναι *ασαφές υποσύνολο* του U αν μπορεί να οριστεί μία *συνάρτηση συμμετοχής (membershipfunction)*$\mu_A : U \to [0,1]$. Το $\mu_A(x)$ ονομάζεται βαθμός συμμετοχής του στοιχείου x στο σύνολο A. Η μηδενική τιμή αντιπροσωπεύει την πλήρη μη συμμετοχή του στοιχείου x ενώ η τιμή 1 την πλήρη συμμετοχή του. Είναι προφανές ότι οι ακραίες τιμές συμπίπτουν με τον ορισμό της $X_A(x)$ από την κλασική θεωρία συνόλων, όπως αυτή δόθηκε παραπάνω. Από εδώ και στο εξής, για να μπορέσουμε να διαχωρίσουμε τα σύνολα της κλασικής θεωρίας συνόλων από τα ασαφή σύνολα, θα αναφερόμαστε στα πρώτα ως *απλά σύνολα* ενώ στα δεύτερα ως *ασαφή σύνολα*.

Ένα παράδειγμα της χρησιμότητας του παραπάνω φορμαλισμού είναι η κατάταξη των ατόμων σε *«νέους»*, *«ώριμους»* και *«μεγάλους»* σε συνάρτηση με την ηλικία τους. Το αν ένας άνθρωπος είναι νέος, είναι μία έννοια η οποία δεν μπορεί να οριστεί εντελώς μαθηματικά, αφού ο κάθε άνθρωπος είναι διαφορετικός από τους άλλους. Το ίδιο μπορούμε να πούμε για το αν ο άνθρωπος είναι ώριμός ή αντίστοιχα μεγάλος. Έτσι λοιπόν αν θεωρήσουμε το υπερσύνολο των ανθρώπων, του οποίου υποσύνολα είναι οι νέοι, οι ώριμοι και οι μεγάλοι άνθρωποι, μπορούμε να ορίσουμε 3 συναρτήσεις συμμετοχής, ανάλογα με την ηλικία (προφανώς ο ορισμός των συναρτήσεων αυτών θα περιέχει ένα αρκετά μεγάλο βαθμό παραδοχών). Έτσι λοιπόν μπορούμε για κάθε υποσύνολο να ορίσουμε το εξής:

- $\mu_{young}(x)$: ένας άνθρωπος x είναι νέος (χωρίς καμία αμφιβολία) μέχρι την ηλικία των 20. Από την ηλικία των 20 μέχρι την ηλικία των 40 η νεότητά του αρχίζει να «φθίνει» ενώ από την ηλικία των 40 και μετά δεν θεωρείται νέος.
- $\mu_{mature}(x)$: ένας άνθρωπος x είναι ώριμος από την ηλικία των 30 και μετά. Πριν την ηλικία των 15 δεν θεωρείται καθόλου ώριμος ενώ από την ηλικία των 15 μέχρι την ηλικία των 30 ο βαθμός συμμετοχής του ανθρώπου στο σύνολό των ώριμων αυξάνει σταδιακά.
- $\mu_{old}(x)$: ένας άνθρωπος x είναι μεγάλος από την ηλικία των 60 και μετά. Πριν την ηλικία των 45 δεν θεωρείται μεγάλος ενώ από την ηλικία των 45 μέχρι την ηλικία των 60 ο βαθμός συμμετοχής του ανθρώπου στο σύνολο των μεγάλων αυξάνει σταδιακά.

Η γραφική αναπαράσταση των 3 συναρτήσεων συμμετοχής δίνεται παρακάτω στο Σχήμα 2.1 [2].



Σχήμα 2.1 [2]: η γραφική αναπαράσταση των συναρτήσεων συμμετοχής για τις έννοιες *«νέος»*, *«ώριμος»* και *«μεγάλος»*, σύμφωνα με τον ορισμό που δόθηκε παραπάνω.

Στο παραπάνω παράδειγμα βλέπουμε ότι ο βαθμός συμμετοχής αυξάνει σε κάθε συνάρτηση συμμετοχής από το 0 μέχρι το 1 με γραμμικό και συνεχή τρόπο. Φυσικά αυτό είναι κάτι το οποίο δεν μπορεί να ισχύει σε κάθε περίπτωση, αφού η συνάρτηση συμμετοχής δεν είναι απαραίτητα ούτε γραμμική ούτε συνεχής.

2.3 Ορισμοί πάνω στα ασαφή σύνολα

Για τους ορισμούς οι οποίοι θα ακολουθήσουν παρακάτω, υποθέτουμε ότι υπάρχουν δύο ασαφή υποσύνολα A και B του συνόλου U .

Ορισμός: το A ονομάζεται *κανονικό* αν $\exists\ x \in U$ τέτοιο ώστε $\mu_A(x) = 1$.

Στην ουσία δηλαδή αρκεί να υπάρχει έστω κι ένα στοιχείο του συνόλου U το οποίο να ανήκει στο ασαφές υποσύνολό του A με βαθμό συμμετοχής ακριβώς 1. Σε διαφορετική περίπτωση, το σύνολο A ονομάζεται μη κανονικό.

Ορισμός: το *σύνολο υποστήριξης (supportset)* ενός ασαφούς συνόλου A, το οποίο συμβολίζεται με $supp(A)$ είναι το υποσύνολο εκείνο του A για το οποίο ισχύει ότι $supp(A) = \{x \in U \mid \mu_A(x) > 0\}$.

Ορισμός: το ασαφές σύνολο A είναι κενό και συμβολίζεται με $\emptyset$ αν και μόνο αν ο βαθμός συμμετοχής του κάθε στοιχείου x του U, όσον αφορά την συνάρτηση συμμετοχής $\mu_A(x)$ είναι ίσος με μηδέν. Στην ουσία δηλαδή, ισχύει ότι: $A = \emptyset \Leftrightarrow (\forall x \in U \Rightarrow \mu_A(x) = 0)$.

Ορισμός: δύο ασαφή σύνολα A και B είναι *ίσα* αν και μόνο αν για κάθε στοιχείο x του U ισχύει ότι $\mu_A(x) = \mu_B(x)$.

Ορισμός: το *συμπληρωματικό ασαφές σύνολο* ενός ασαφούς συνόλου A συμβολίζεται με A' και έχει ως συνάρτηση συμμετοχής την $\mu_{A'}(x) = 1 - \mu_A(x)$.

Ο παραπάνω ορισμός διαφέρει αισθητά από τον ορισμό του συμπληρωματικού συνόλου από την κλασική θεωρία συνόλων, αφού αυτό το οποίο στην ουσία αλλάζει είναι η τιμή του βαθμού συμμετοχής.

Ορισμός: λέμε ότι ένα ασαφές σύνολο A *περιέχει* ένα ασαφές σύνολο B αν για κάθε στοιχείο x του U ισχύει ότι $\mu_A(x) \geq \mu_B(x)$. Αυτό συμβολίζεται ως $B \subset A$.

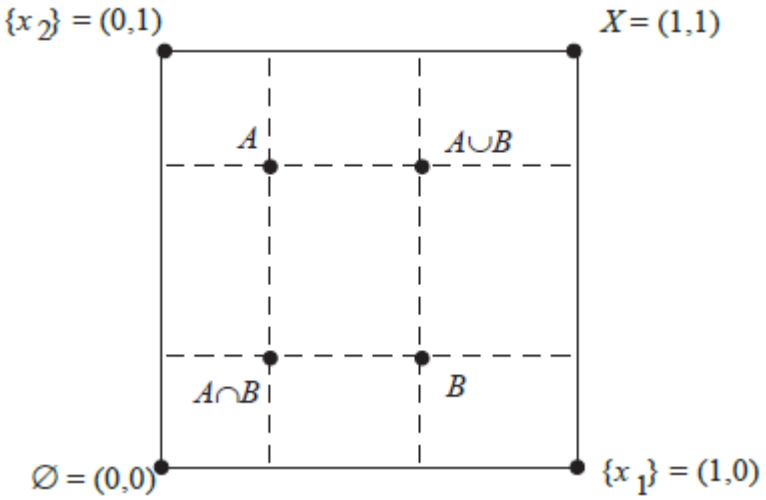
Ορισμός: η *ένωση (union)* δύο ασαφών συνόλων A και B συμβολίζεται ως $A \cup B$ και έχει ως συνάρτηση συμμετοχής την $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$.

Η ένωση, όπως και στην κλασική θεωρία συνόλων, είναι αντιμεταθετική $(A \cup B = B \cup A)$ καθώς επίσης και προσεταιριστική $(A \cup (B \cup C) = (A \cup B) \cup C)$.

Ορισμός: η *τομή (intersection)* δύο ασαφών συνόλων A και B συμβολίζεται ως $A \cap B$ και έχει ως συνάρτηση συμμετοχής την $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$.

Η τομή, όπως και στην κλασική θεωρία συνόλων, είναι αντιμεταθετική $(A \cap B = B \cap A)$ καθώς επίσης και προσεταιριστική $(A \cap (B \cap C) = (A \cap B) \cap C)$. Όπως και στην θεωρία των κλασικών συνόλων έτσι και στην θεωρία των ασαφών συνόλων, δύο ασαφή σύνολα καλούνται ξένα μεταξύ τους αν ισχύει ότι $A \cap B = \emptyset$.

Μία γραφική αναπαράσταση των πράξεων της ένωσης και της τομής δύο ασαφών συνόλων A και B παρουσιάζεται στο Σχήμα 2.2 [2].



Σχήμα 2.2 [2]: γραφική αναπαράσταση των πράξεων της ένωσης και της τομής δύο ασαφών συνόλων όπου το καθένα έχει μόλις ένα στοιχείο.

Ένας μη αυστηρός ορισμός των πράξεων της ένωσης και της τομής στα ασαφή σύνολα είναι ο εξής [1]:

- η ένωση δύο ασαφών συνόλων είναι το μικρότερο ασαφές σύνολο το οποίο περιέχει όλα τα στοιχεία των A και B, αφού $\max(\mu_A(x), \mu_B(x)) \geq \mu_A(x)$ και $\max(\mu_A(x), \mu_B(x)) \geq \mu_B(x)$. Με άλλα λόγια αν D είναι ένα ασαφές σύνολο το οποίο περιέχει τα ασαφή σύνολα A και B τότε σίγουρα περιέχει και την ένωσή τους.
- η τομή δύο ασαφών συνόλων είναι το μεγαλύτερο ασαφές σύνολο το οποίο περιέχεται τόσο στο A όσο και στο B, αφού $\min(\mu_A(x), \mu_B(x)) \leq \mu_A(x)$ και $\min(\mu_A(x), \mu_B(x)) \leq \mu_B(x)$.

Ορισμός: ο *πληθάριθμος (cardinality)* ενός ασαφούς συνόλου A συμβολίζεται $|A|$ ορίζεται ως το άθροισμα των βαθμών συμμετοχής $\mu_A(x)$ για κάθε στοιχείο x του U. Πιο συγκεκριμένα: $|A| = \sum_{x \in U} \mu_A(x)$.

2.4 Ασαφείς λογικοί τελεστές και ασαφής λογική

Είναι γνωστό από τα μαθηματικά ότι υπάρχει μία αναλογία μεταξύ της θεωρίας συνόλων και της κλασικής προτασιακής λογικής. Στην θεωρία συνόλων, οι τρεις τελεστές ένωση τομή και συμπλήρωμα ($\cup, \cap, '$) επιτρέπουν την κατασκευή νέων συνόλων από ήδη υπάρχοντα. Στην προτασιακή λογική, οι τελεστές OR, ANDκαι NOT ($\vee, \wedge, \neg$) χρησιμοποιούνται για την κατασκευή νέων λογικών εκφράσεων από ήδη υπάρχοντες.

Ο τελεστής της ένωσης στην κλασική θεωρία συνόλων μπορεί να οριστεί χρησιμοποιώντας τον λογικό τελεστή OR. Πιο συγκεκριμένα, έστω δύο απλά σύνολα A και B, οι συναρτήσεις συμμετοχής των οποίων είναι της μορφής $\mu_A, \mu_B : U \rightarrow \{0,1\}$. Η συνάρτηση συμμετοχής της ένωσής τους $\mu_{A \cup B}$ μπορεί να οριστεί με τον εξής τρόπο:

$$\mu_{A \cup B}(x) = \mu_A(x) \vee \mu_B(x), \forall x \in U$$

όπου η αριθμητική τιμή 0 της κάθε συνάρτησης συμμετοχής αντιστοιχεί στην λογική τιμή *ψευδές (false)* ενώ αντίστοιχα η αριθμητική τιμή 1 της κάθε συνάρτησης συμμετοχής αντιστοιχεί στην λογική τιμή *αληθές (true)*.

Με αντίστοιχο τρόπο μπορούμε να ορίσουμε τον τελεστή της τομής συνόλων κάνοντας χρήση του λογικού τελεστή AND. Πιο συγκεκριμένα, η συνάρτηση συμμετοχής της τομής των δύο παραπάνω συνόλων $\mu_{A \cap B}$ μπορεί να οριστεί με τον εξής τρόπο:

$$\mu_{A \cap B}(x) = \mu_A(x) \wedge \mu_B(x), \forall x \in U$$

Τέλος, το συμπλήρωμα A' ενός απλού συνόλου A έχει την παρακάτω συνάρτηση συμμετοχής:

$$\mu_{A'}(x) = \neg \mu_A(x)$$

Η αντιστοιχία μεταξύ των τελεστών της κλασικής θεωρίας συνόλων και της κλασικής προτασιακής λογικής μπορεί να επεκταθεί και στην εγκυρότητα κάποιων εξισώσεων και στα δύο συστήματα. Για παράδειγμα, οι κανόνες DeMorgan είναι έγκυροι τόσο στην κλασική θεωρία συνόλων όσο και στην κλασική προτασιακή λογική:

$$(A \cup B)' = A' \cap B' \text{ αντιστοιχεί στο } \neg(\mathbf{A} \vee \mathbf{B}) = \neg\mathbf{A} \wedge \neg\mathbf{B}$$
$$(A \cap B)' = A' \cup B' \text{ αντιστοιχεί στο } \neg(\mathbf{A} \wedge \mathbf{B}) = \neg\mathbf{A} \vee \neg\mathbf{B}$$

Από τα όσα ορίστηκαν στις προηγούμενες ενότητες για τα ασαφή σύνολα, μπορούμε να δημιουργήσουμε μία αντίστοιχη *ασαφή λογική (fuzzylogic)* η οποία ακολουθεί ανάλογες αντιστοιχίες με αυτές που παρουσιάσαμε παραπάνω. Φυσικά οι ασαφείς λογικοί τελεστές AND, ORκαι NOTθα πρέπει να οριστούν με τέτοιο τρόπο έτσι ώστε να υπάρχουν τα ίδια γενικά ήδη σχέσεων μεταξύ αυτών και των ισοδύναμών τους ασαφών τελεστών στην ασαφή θεωρία συνόλων. Η πιο απλή προσέγγιση είναι η παρακάτω αντιστοιχία:

- ασαφής λογικός τελεστής OR ($\tilde{\vee}$): $\mu_{A\cup B}(x) = \mu_A(x)\tilde{\vee}\mu_B(x) = \max(\mu_A(x), \mu_B(x))$.
- ασαφής λογικός τελεστής AND ($\tilde{\wedge}$): $\mu_{A\cap B}(x) = \mu_A(x)\,\tilde{\wedge}\,\mu_B(x) = \min(\mu_A(x), \mu_B(x))$.
- ασαφής λογικός τελεστής NOT ($\tilde{\neg}$):$\mu'_A(x) = \tilde{\neg}\,\mu_A(x) = 1 - \mu_A(x)$.

Οι αλγεβρικές ιδιότητες των παραπάνω τελεστών (αντίστοιχες των ιδιοτήτων της άλγεβρας Boole) παρουσιάζονται παρακάτω [2].

1. Αλγεβρικές ιδιότητες του ασαφούς λογικού τελεστή OR($\tilde{\vee}$)

- Αξίωμα O1: Πίνακας Αληθείας

- Αξίωμα O2: Αντιμεταθετικότητα

$$a\,\tilde{\vee}\,b = b\,\tilde{\vee}\,a$$

- Αξίωμα O3: Μονοτονία

Αν $a \le a'$ και $b \le b'$ τότε$a\tilde{\vee}b \le a'\tilde{\vee}b'$.

- Αξίωμα O4: Προσεταιριστικότητα

$$a\,\tilde{\vee}\,(b\,\tilde{\vee}\,c) = (a\,\tilde{\vee}\,b)\tilde{\vee}\,c$$

2. Αλγεβρικές ιδιότητες του ασαφούς λογικού τελεστή AND($\tilde{\wedge}$)

- Αξίωμα A1: Πίνακας Αληθείας

- Αξίωμα A2: Αντιμεταθετικότητα

$a \wedge b = b \wedge a$

- Αξίωμα Α3: Μονοτονία

Αν $a \leq a'$ και $b \leq b'$ τότε $a \wedge b \leq a' \wedge b'$.

- Αξίωμα Α4: Προσεταιριστικότητα

$a \wedge (b \wedge c) = (a \wedge b) \wedge c$

3. Αλγεβρικές ιδιότητες του ασαφούς λογικού τελεστή NOT($\neg$)

- Αξίωμα Ν1: Πίνακας Αληθείας

$\neg 0 = \mathbf{1}$
$\neg 1 = \mathbf{0}$

- Αξίωμα Ν2: Μονοτονία

Αν $a \leq b$ τότε $\neg b \leq \neg a$.

- Αξίωμα Ν3: Διπλή Άρνηση

$\neg\neg a = a$

Στο σημείο αυτό θα πρέπει να τονίσουμε ότι όσα παρουσιάστηκαν στις ενότητες 2.3 και 2.4 δεν είναι δεσμευτικά όσον αφορά τις ιδιότητες ασαφών συνόλων και τα αξιώματα των τελεστών της ασαφούς λογικής. Με άλλα λόγια οι ορισμοί για την ένωση, την τομή και το συμπλήρωμα θα μπορούσαν να είναι διαφορετικοί, κάνοντας χρήση διαφορετικών συναρτήσεων από τις min και max. Για παράδειγμα κάποιος θα μπορούσε να ορίσει την τομή συνόλων ως εξής: $\mu_{A \cap B}(x) = \min\left(1, \mu_A(x) + \mu_B(x)\right)$. Αυτό φυσικά θα είχε ως αποτέλεσμα να μην ισχύουν με τον ίδιο τρόπο οι μετέπειτα αλγεβρικές ιδιότητες των αντίστοιχων τελεστών και να ισχύουν κάποιες άλλες. Τα όσα παρουσιάστηκαν στις ενότητες 2.3 και 2.4 αποτελούν τους περισσότερο συνηθισμένους ορισμούς πάνω στα ασαφή σύνολα και την ασαφή λογική [2].

2.5 Ασαφή συστήματα ελέγχου

Οι ασαφείς λογικοί τελεστές μπορούν να χρησιμοποιηθούν σαν βάση για *ασαφή συστήματα ελέγχου*. Τα ασαφή συστήματα ελέγχου ανήκουν στην κατηγορία των *έμπειρων συστημάτων*. Ένα έμπειρο σύστημα είναι ένα υπολογιστικό σύστημα το οποίο μιμείται την ικανότητα ενός εμπειρογνώμονα στη λήψη αποφάσεων [5]. Τα έμπειρα συστήματα σχεδιάστηκαν για να λύνουν πολύπλοκα προβλήματα συλλογιζόμενα με βάση τη διαθέσιμη γνώση σε ένα πεδίο, όπως κάνει ένας εμπειρογνώμονας, και όχι εκτελώντας μία ακριβή διαδικασία επίλυσης την οποία έχει προδιαγράψει ένας προγραμματιστής, όπως στην περίπτωση του συμβατικού

προγραμματισμού υπολογιστών. Ο περισσότερο συνηθισμένος τρόπος λειτουργίας ενός έμπειρου συστήματος είναι η χρήση κανόνων IF-THEN, οι οποίοι ονομάζονται *κανόνες παραγωγής*. Οι κανόνες παραγωγής έχουν την παρακάτω μορφή:

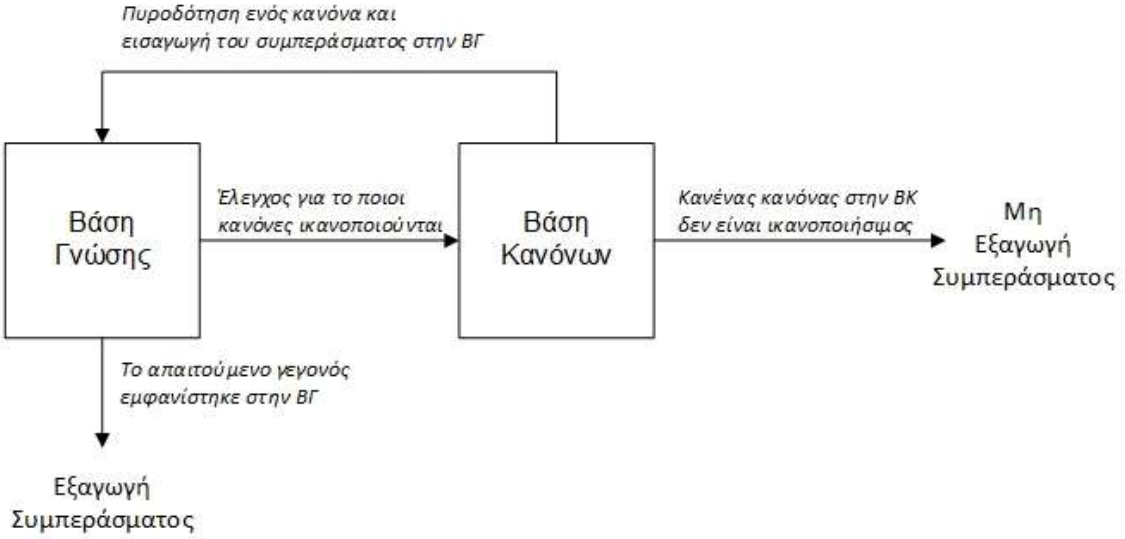
R1: If A then D

R2: If $B \wedge C$ then E

R3: If $E \vee \neg F$ then G

…

Ένα έμπειρο σύστημα, στην πιο απλή του μορφή, διαθέτει μία Βάση Κανόνων (ΒΚ) στην οποία αποθηκεύει τους κανόνες παραγωγής καθώς και μία Βάση Γνώσης (ΒΓ), στην οποία αποθηκεύει τα γεγονότα τα οποία είναι αληθή. Εκκινώντας από τα γεγονότα που υπάρχουν αρχικά στην ΒΓ, το έμπειρο σύστημα ελέγχει κάθε φορά τους κανόνες που ικανοποιούνται, παίρνει τα συμπεράσματά τους και τα εισάγει στην ΒΓ. Η διαδικασία αυτή επαναλαμβάνεται μέχρι ένα συγκεκριμένο συμπέρασμα εισαχθεί στην ΒΓ ή μέχρις ότου κανένας από τους κανόνες παραγωγής να μην είναι πλέον ικανοποιήσιμος (Σχήμα 2.3).



Σχήμα 2.3: οι βασικές αρχές λειτουργίας ενός έμπειρου συστήματος.

Οι παραπάνω κανόνες παραγωγής περιέχουν στις συνθήκες τους γεγονότα τα οποία είναι είτε αληθή (τιμή 1) είτε ψευδή (τιμή 0). Για τον λόγο αυτό για την σύνδεσή τους χρησιμοποιούνται οι κλασικοί λογικοί τελεστές AND, ORκαι NOT. Με βάση τα όσα ειπώθηκαν παραπάνω, μπορούμε να δομήσουμε κανόνες στους οποίους τα γεγονότα συνδέονται με ασαφείς λογικούς τελεστές. Για παράδειγμα:

R1: If $A \wedge \neg B$ then C

R2: If $A \vee \neg B$ then D

…

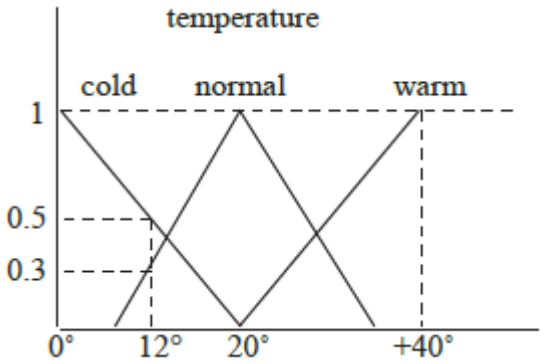
Η διαφορά σε σχέση με τους συμβατικούς κανόνες παραγωγής βρίσκεται στην σημασιολογία των ασαφών τελεστών. Ας υποθέσουμε ότι οι τιμές (των συναρτήσεων συμμετοχής) των γεγονότων A και B είναι 0.4 και 0.7 αντίστοιχα. Τις τιμές των συναρτήσεων συμμετοχής από εδώ και στο εξής θα τις ονομάζουμε επίσης και *τιμές βεβαιότητας* ή απλώς *βεβαιότητες* ή *βαθμούς αληθείας*. Έτσι λοιπόν έχουμε:

$$A \wedge \neg B = \min(0.4, 0.7) = 0.4$$
$$A \vee \neg B = \max(0.4, 0.7) = 0.7$$

Τα συμπεράσματα C και D δεν εξάγονται με βεβαιότητα 1 αλλά πλέον με βεβαιότητες 0.4 και 0.7 αντίστοιχα. Μπορούμε να δείξουμε την χρησιμότητα τέτοιων κανόνων παραγωγής με ένα παράδειγμα. Ας υποθέσουμε ένα σύστημα ελέγχου της ηλεκτρικής κατανάλωσης ενός αυτόματου ηλεκτρικού θερμοσίφωνα. Υποθέτουμε ότι το σύστημα αυτό λειτουργεί με βάση τους παρακάτω κανόνες:

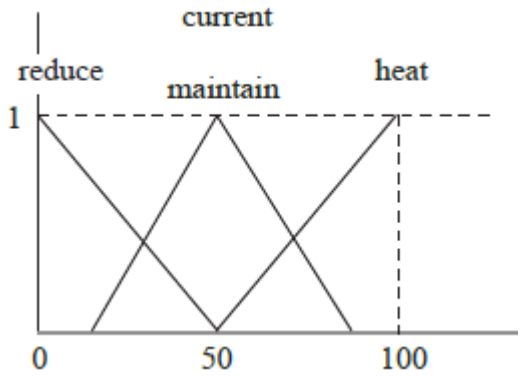
R1: If (θερμοκρασία νερού = κρύα) then ζέστανε
R2: If (θερμοκρασία νερού = κανονική) then διατήρησε
R3: If (θερμοκρασία νερού = ζεστή) then μείωσε την λειτουργία

Ας υποθέσουμε ότι οι τιμές της θερμοκρασίας του νερού κυμαίνονται στο διάστημα 0 C – 40 Cκαι ότι την τρέχουσα χρονική στιγμή η τιμή της θερμοκρασίας νερού είναι 12 C.Έστω ότι η θερμοκρασία αυτή έχει βαθμό συμμετοχής 0.5 στο ασαφές σύνολο «κρύα», 0.3 στο ασαφές σύνολο «κανονική» και 0 στο ασαφές σύνολο «ζεστή». Ας υποθέσουμε επίσης ότι η ηλεκτρική κατανάλωση του θερμοσίφωνα κυμαίνεται στην κλίμακα 0 έως 100 (προφανώς έγινε αναγωγή από κάποιες ηλεκτρικές μονάδες). Σκοπός του συστήματος ελέγχου είναι να μετατρέψει τις εισόδους που έχει λάβει από το περιβάλλον (0.5, 0.3 και 0) σε μία μοναδική τιμή ηλεκτρικής κατανάλωσης, έτσι ώστε να διατηρήσει την θερμοκρασία του νερού στα απαιτούμενα επίπεδα. Έστω ότι οι γραφικές παραστάσεις των συναρτήσεων συμμετοχής των 3 ασαφών συνόλων «κρύα», «κανονική» και «ζεστή» είναι αυτές οι οποίες δίνονται στο Σχήμα 2.4.



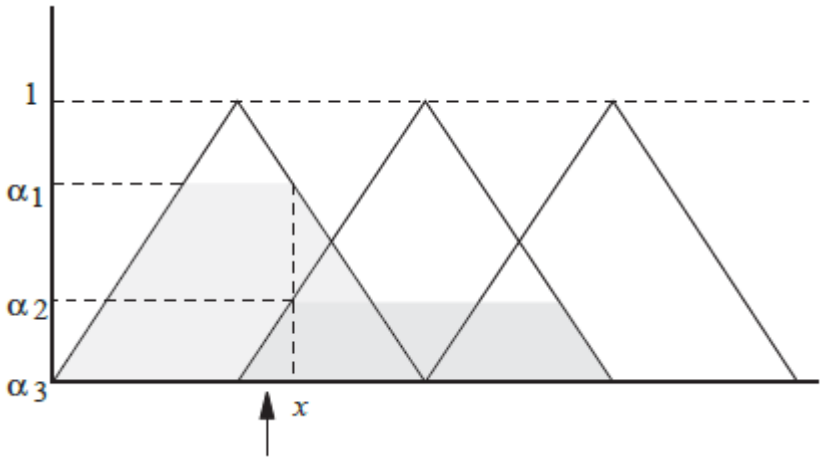
Σχήμα 2.4: οι γραφικές παραστάσεις των συναρτήσεων συμμετοχής των ασαφών συνόλων «κρύα» (cold), «κανονική» (normal) και «ζεστή» (warm).

Οι ηλεκτρικές καταναλώσεις των συμπερασμάτων «ζέστανε», «διατήρησε» και «μείωσε την λειτουργία» των κανόνων R1, R2 και R3 έχουν και αυτές τις δικές τους γραφικές παραστάσεις (Σχήμα 2.5). Κάνουμε την υπόθεση ότι οι γραφικές τους για λόγους απλότητας, είναι παραπλήσιες αυτών που παρουσιάζονται στο Σχήμα 2.5. Φυσικά αυτό δεν είναι κάτι απαραίτητο ή δεσμευτικό και οι εν λόγω γραφικές παραστάσεις θα μπορούσαν να έχουν οποιαδήποτε μορφή.



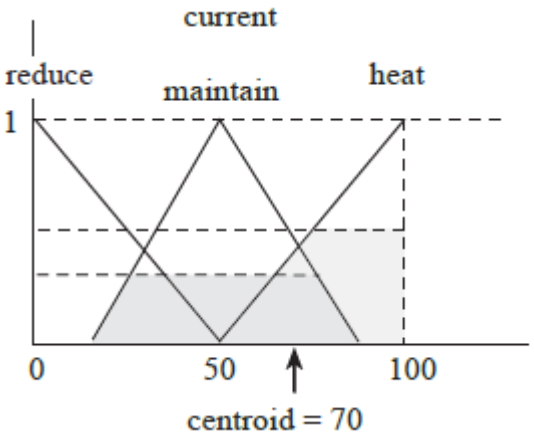
Σχήμα 2.5: οι αντίστοιχες γραφικές παραστάσεις των συμπερασμάτων «ζέστανε» (heat), «διατήρησε» (maintain) και «μείωσε την λειτουργία» (reduce).

Η μετατροπή των τιμών των συναρτήσεων συμμετοχής των εισόδων σε μία μόνο τιμή εξόδου (στο παράδειγμά μας αυτή αντιστοιχεί στην τιμή της ηλεκτρικής κατανάλωσης) ονομάζεται *αποσαφήνιση (defuzzification)* και μπορεί να γίνει με πολλούς τρόπους. Ένας από τους περισσότερο συνηθισμένους τρόπους είναι *η κεντροειδής μέθοδος (centroidmethod)* [2]. Σύμφωνα με την μέθοδο αυτή, υπολογίζουμε την βεβαιότητα του κάθε αποτελέσματος και σκιάζουμε την περιοχή που αυτό καταλαμβάνει, στην αντίστοιχη γραφική παράσταση. Στην συνέχεια βρίσκουμε το *κεντροειδές σημείο* (ή αλλιώς *κέντρο μάζας*) του σχήματος που σχηματίζεται στις κοινές σκιασμένες επιφάνειες των γραφικών παραστάσεων και καταγράφουμε την οριζόντια συντεταγμένη του(η οποία συμβολίζεται με $\overline{x}$). Ένα παράδειγμα εφαρμογής της μεθόδου παρουσιάζεται στο Σχήμα 2.6.



Σχήμα 2.6: το βέλος αναπαριστά την οριζόντια συντεταγμένη του κεντροειδούς σημείου των σκιασμένων επιφανειών των γραφικών παραστάσεων. Στο εν λόγω παράδειγμα η μέτρηση x της εισόδου αντιστοιχεί σε βεβαιότητες a_1, a_2 και a_3 αντίστοιχα. Το γεγονός του ότι η βεβαιότητα a_3 αντιστοιχεί στην τιμή 0 μπορεί να ερμηνευτεί ισοδύναμα και ως μη ενεργοποίηση του αντίστοιχου κανόνα.

Όσον αφορά το παράδειγμα το οποίο με το σύστημα ελέγχου για τον ηλεκτρικό θερμοσίφωνα, οι βεβαιότητες των συμπερασμάτων «ζέστανε», «διατήρησε» και «μείωσε την λειτουργία» των κανόνων R1, R2 και R3 έχουν ίδιες τιμές (0.5, 0.3 και 0) με τις τιμές των υποθέσεων. Αυτό συμβαίνει επειδή ο κάθε κανόνας έχει μόνο μία υπόθεση και έτσι απουσιάζει η χρήση των ασαφών λογικών τελεστών AND, ORκαι NOT. Φυσικά κάτι τέτοιο δεν ισχύει πάντοτε. Έτσι λοιπόν, με βάση την κεντροειδή μέθοδο, παίρνουμε το Σχήμα 2.7.



Σχήμα 2.7: εφαρμογή της κεντροειδούς μεθόδου στις γραφικές παραστάσεις του Σχήματος 2.5, για τιμές 0.5, 0.3 και 0 αντίστοιχα. Η οριζόντια συντεταγμένη του κεντροειδούς σημείου παίρνει την τιμή $\bar{x}$ =70.

Φυσικά η κεντροειδής μέθοδος δεν είναι ο μόνος τρόπος υπολογισμού της τελικής τιμής (αποσαφήνιση) η οποία θα δοθεί ως αποτέλεσμα από το σύστημα ελέγχου. Ο συνδυασμός των βεβαιοτήτων των συμπερασμάτων των κανόνων θα μπορούσε να εξαχθεί και με άλλους τρόπους, αλγεβρικούς ή γεωμετρικούς.

2.6 Συνήθεις μορφές συναρτήσεων συμμετοχής

Στις προηγούμενες ενότητες είδαμε παραδείγματα γραφικών παραστάσεων των συναρτήσεων συμμετοχής κάποιων ασαφών συνόλων. Στην συντριπτική πλειοψηφία των περιπτώσεων οι συναρτήσεις συμμετοχής χωρίζονται στις 4 παρακάτω κατηγορίες [6]:

1. Τριγωνικές συναρτήσεις συμμετοχής
2. Τραπεζοειδείς συναρτήσεις συμμετοχής
3. Γκαουσιανές συναρτήσεις συμμετοχής
4. Μη φραγμένες συναρτήσεις συμμετοχής

Θα αναλύσουμε την κάθε μία από τις κατηγορίες αυτές παρακάτω.

2.6.1 Τριγωνικές συναρτήσεις συμμετοχής

Οι τριγωνικές συναρτήσεις συμμετοχής έχουν γραφική παράσταση σχήματος τριγώνου και είναι της παρακάτω μορφής [6]:

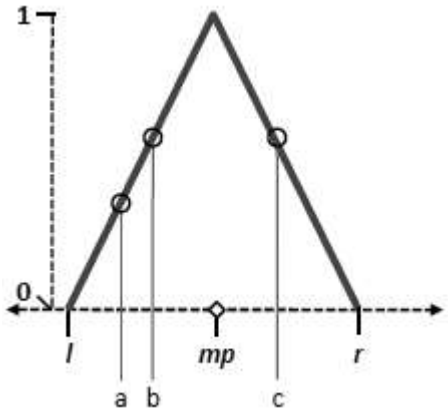
$$\mu_{tri}(x) = \begin{cases} 0, & \alpha v\ x \leq l \\ \dfrac{x-l}{mp-l} \cdot m, & \alpha v\ l < x < mp \\ m, & \alpha v\ x = mp \\ \dfrac{r-x}{r-mp} \cdot m, & \alpha v\ mp < x < r \\ 0, & \alpha v\ r \leq x \end{cases}$$

όπου:

- η παράμετρος $m \in [0,1]$ καθορίζει την μέγιστή τιμή της συνάρτησηςμ_{tri} , η οποία είναι συνήθως 1.
- η παράμετρος $l \in \mathbf{R}$ καθορίζει το αριστερό όριο του τριγώνου.

- η παράμετρος $r \in \mathbf{R}$ καθορίζει το δεξιό όριο του τριγώνου.
- η παράμετρος $mp \in \mathbf{R}$ είναι το σημείο του πεδίου ορισμού για το οποίο η τιμή της μ_{tri} είναι ίση με m , δηλαδή $\mu_{tri}(mp) = m$.

Η μ^{tri} παρουσιάζεται σχηματικά στο Σχήμα 2.8.



Σχήμα 2.8: η γραφική παράσταση μιας τριγωνικής συνάρτησης συμμετοχής. Τα σημεία a, bκαι cείναι διάφορες τιμές του συνόλου τιμών.

Σε ένα οποιοδήποτε τρίγωνο με μήκη πλευρών a , b και c , η οριζόντια συντεταγμένη$\overline{x}$ του κεντροειδούς σημείου είναι ίση με:

$$\overline{x} = \frac{a + b + c}{3}$$

Το κεντροειδές σημείο της τριγωνικής συνάρτησης συμμετοχής είναι το mp , αν το τρίγωνο είναι ισοσκελές, δηλαδή αν $mp - l = r - mp$. Ακόμα και αν το τρίγωνο δεν είναι ισοσκελές, είναι αρκετά συνηθισμένη η χρήση του σημείου mp ως κεντροειδές [6].

2.6.2 Τραπεζοειδείς συναρτήσεις συμμετοχής

Οι τραπεζοειδείς συναρτήσεις συμμετοχής έχουν γραφική παράσταση σχήματος τραπεζίου και είναι της παρακάτω μορφής [6]:

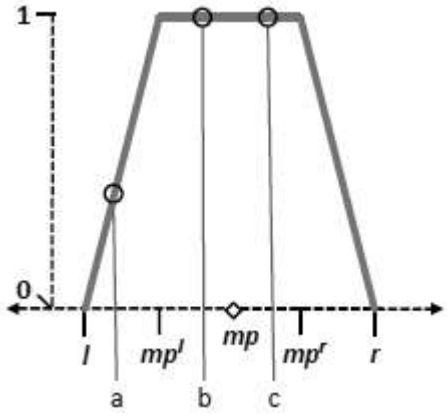
$$\mu_{tra}(x) = \begin{cases} 0, & \alpha v\ x \leq l \\ \dfrac{x - l}{mp^l - l} \cdot m, & \alpha v\ l < x < mp^l \\ m, & \alpha v\ mp^l \leq x \leq mp^r \\ \dfrac{r - x}{r - mp^r} \cdot m, & \alpha v\ mp^r < x < r \\ 0, & \alpha v\ r \leq x \end{cases}$$

όπου:

- η παράμετρος $m \in [0,1]$ καθορίζει την μέγιστή τιμή της συνάρτησηςμ_{tra} , η οποία είναι συνήθως 1.
- η παράμετρος $l \in \mathbf{R}$ καθορίζει το αριστερό όριο του τραπεζίου.
- η παράμετρος $r \in \mathbf{R}$ καθορίζει το δεξιό όριο του τραπεζίου.

- η παράμετρος $mp^l \in \mathbf{R}$ είναι η αριστερή κορυφή του τραπεζίου, δηλαδή το σημείο του πεδίου ορισμού για το οποίο η τιμή της μ_{tra} είναι ίση με m, δηλαδή $\mu_{tra}(mp^l) = m$.
- η παράμετρος $mp^r \in \mathbf{R}$ είναι η δεξιά κορυφή του τραπεζίου, δηλαδή το σημείο του πεδίου ορισμού για το οποίο η τιμή της μ_{tra} είναι ίση με m, δηλαδή $\mu_{tra}(mp^r) = m$.

Η μ^{tra} παρουσιάζεται σχηματικά στο Σχήμα 2.9.



Σχήμα 2.9: η γραφική παράσταση μιας τραπεζοειδούς συνάρτησης συμμετοχής. Τα σημεία a, b και c είναι διάφορες τιμές του συνόλου τιμών. Είναι προφανές ότι η μέγιστη τιμή της συνάρτησης μ_{tra} δίνεται από περισσότερα του ενός σημεία του πεδίου ορισμού.

Όσον αφορά το κεντροειδές σημείο, σε ένα οποιοδήποτε τραπεζοειδές με μήκη παράλληλων πλευρών a και b και ύψος h ,η οριζόντια συντεταγμένη$\overline{x}$ του κεντροειδούς σημείου είναι ίση με:

$$\overline{x} = \frac{b + 2a}{3(a + b)} \cdot h$$

2.6.3 Γκαουσιανές συναρτήσεις συμμετοχής

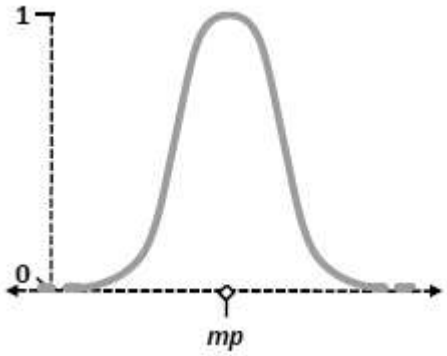
Οι γκαουσιανές συναρτήσεις συμμετοχής έχουν γραφική παράσταση σχήματος τριγώνου και είναι της παρακάτω μορφής [6]:

$$\mu_{gau}(x) = m \cdot e^{-\frac{(x-mp)^2}{2w^2}}$$

όπου:

- η παράμετρος $m \in [0,1]$ καθορίζει την μέγιστή τιμή της συνάρτησηςμ_{gau}, η οποία είναι συνήθως 1.
- η παράμετρος $w \in \mathbf{R}$ καθορίζει το μήκος («άνοιγμα») της γραφικής παράστασης (Σχήμα 2.10).
- η παράμετρος $mp \in \mathbf{R}$ είναι το σημείο του πεδίου ορισμού για το οποίο η τιμή της μ_{gau} είναι ίση με m, δηλαδή $\mu_{gau}(mp) = m$.

Η μ^{gau} παρουσιάζεται σχηματικά στο Σχήμα 2.10. Όπως είναι προφανές, μία γκαουσιανή συνάρτηση συμμετοχής είναι στην ουσία μία κανονική κατανομή.



Σχήμα 2.10: η γραφική παράσταση μιας γκαουσιανής συνάρτησης συμμετοχής.

Όσον αφορά το κεντροειδές σημείο, η οριζόντια συντεταγμένη του $\bar{x}$ του κεντροειδούς σημείου είναι ίση με mp (μέση τιμή της κανονικής κατανομής). Θα πρέπει να σημειώσουμε ότι στην γκαουσιανή συνάρτηση κατανομής $\forall x \in \mathbb{R}: \mu_{gau}(x) \neq 0$. Αυτό έχει σαν συνέπεια οι βαθμοί συμμετοχής όλων των στοιχείων ενός ασαφούς συνόλου με γκαουσιανή συνάρτηση συμμετοχής να είναι διάφοροι του μηδενός.

2.6.4 Μη φραγμένες συναρτήσεις συμμετοχής

Οι μη φραγμένες συναρτήσεις συμμετοχής έχουν το εξής χαρακτηριστικό: δίνουν την τιμή 1 σε όλα τα στοιχεία του πεδίου ορισμού τα οποία ξεπερνούν (ή δεν ξεπερνούν) ένα συγκεκριμένο κατώφλι (threshold) [6]. Υπάρχουν δύο κύριες μορφές μη φραγμένων συναρτήσεων συμμετοχής: οι τραπεζοειδής μορφής και οι γκαουσιανής μορφής. Θα αναλύσουμε τα δύο αυτά είδη παρακάτω.

<u>1. Τραπεζοειδείς μη φραγμένες συναρτήσεις συμμετοχής</u>

Οι τραπεζοειδείς συναρτήσεις συμμετοχής είναι της παρακάτω μορφής [6]:

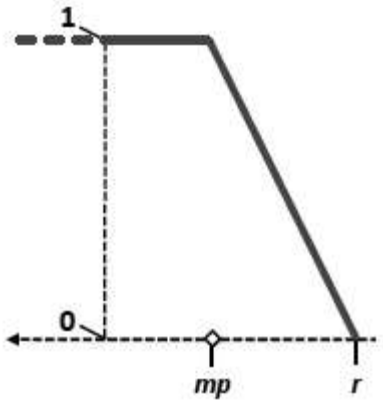
$$\mu_{left}^{tra}(x) = \begin{cases} m, & \alpha v\ x \leq mp \\ \dfrac{r-x}{r-mp}, & \alpha v\ mp < x < r \\ 0, & \alpha v\ r \leq x \end{cases}$$

όπου:

- η παράμετρος $m \in [0,1]$ καθορίζει την μέγιστή τιμή της συνάρτησης μ_{left}^{tra}, η οποία είναι συνήθως 1.
- η παράμετρος $r \in \mathbf{R}$ καθορίζει το δεξιό όριο του τραπεζίου.
- η παράμετρος $mp \in \mathbf{R}$ είναι το σημείο του πεδίου ορισμού για το οποίο η τιμή της μ_{left}^{tra} είναι ίση με m , δηλαδή $\mu_{left}^{tra}(mp) = m$.

Η μ_{left}^{tra} παρουσιάζεται σχηματικά στο Σχήμα 2.11. Οι συναρτήσεις συμμετοχής της παραπάνω μορφής είναι μη φραγμένες από την αριστερή τους πλευρά. Η συμμετρική γραφική παράσταση (μη φραγμένες από την δεξιά πλευρά) θα είχε την παρακάτω μορφή:

$$\mu_{right}^{tra}(x) = \begin{cases} m, & \alpha v\ x \leq l \\ \dfrac{x-l}{mp-l}, & \alpha v\ l < x < mp \\ 0, & \alpha v\ mp \leq x \end{cases}$$



Σχήμα 2.11: η γραφική παράσταση μιας τραπεζοειδούς μη φραγμένης από την αριστερή πλευρά συνάρτησης συμμετοχής.

Το κεντροειδές σημείο σε κάθε περίπτωση (μη φραγμένη είτε από τα αριστερά είτε από τα δεξιά) θεωρούμε ότι είναι το σημείο mp .

2. Γκαουσιανές μη φραγμένες συναρτήσεις συμμετοχής

Οι γκαουσιανές συναρτήσεις συμμετοχής είναι της παρακάτω μορφής [6]:

$$\mu_{left}^{gau}(x) = \begin{cases} m, & \alpha v\ x \leq mp \\ m \cdot e^{-\frac{(x-mp)^2}{2w^2}}, & \alpha v\ mp < x \end{cases}$$

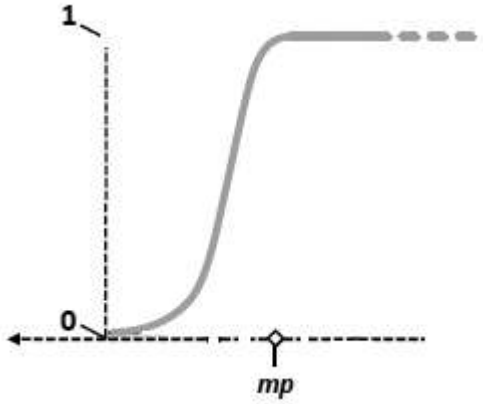
όπου:

- η παράμετρος $m \in [0,1]$ καθορίζει την μέγιστή τιμή της συνάρτησης μ_{left}^{gau}, η οποία είναι συνήθως 1.
- η παράμετρος $w \in \mathbf{R}$ καθορίζει το μήκος («άνοιγμα») της γραφικής παράστασης (Σχήμα 2.10).
- η παράμετρος $mp \in \mathbf{R}$ είναι το σημείο του πεδίου ορισμού για το οποίο η τιμή της μ_{left}^{gau} είναι ίση με m , δηλαδή $\mu_{left}^{tra}(mp) = m$.

Οι συναρτήσεις συμμετοχής της παραπάνω μορφής είναι μη φραγμένες από την αριστερή τους πλευρά. Η συμμετρική γραφική παράσταση (μη φραγμένες από την δεξιά πλευρά) θα είχε την παρακάτω μορφή:

$$\mu_{right}^{gau}(x) = \begin{cases} m \cdot e^{-\frac{(x-mp)^2}{2w^2}}, & \alpha v\ x < mp \\ m, & \alpha v\ mp \leq x \end{cases}$$

Η μ_{right}^{gau} παρουσιάζεται σχηματικά στο Σχήμα 2.12.



Σχήμα 2.12: η γραφική παράσταση μιας γκαουσιανής μη φραγμένης από την δεξιά πλευρά συνάρτησης συμμετοχής.

Και στις γκαουσιανές μη φραγμένες συναρτήσεις συμμετοχής, θεωρούμε ότι το κεντροειδές σημείο είναι το σημείο mp .

2.7 Κριτήρια ορθότητας των ασαφών κανόνων παραγωγής

Η μοντελοποίηση ενός συστήματος το οποίο στηρίζεται σε ασαφείς κανόνες παραγωγής στηρίζεται κατά κύριο βαθμό στην ορθότητα των κανόνων παραγωγής. Η ορθότητα των κανόνων παραγωγής αντιστοιχεί στην ικανοποίηση ενός συνόλου κριτηρίων, τα οποία αφορούν ένα σύνολο αντίστοιχων πιθανών λαθών. Τα πιθανά λάθη τα οποία μπορεί να προκύψουν από τον ορισμό των κανόνων είναι τα εξής [15]:

- Δομικά λάθη
  - Μη ολοκληρωμένοι κανόνες
  - Ασυνεπείς κανόνες
  - Κυκλικοί κανόνες
  - Πλεοναστικοί Κανόνες
- Εννοιολογικά λάθη
  - Μη ολοκληρωμένοι κανόνες
  - Μη έγκυροι κανόνες

Η κάθε κατηγορία λαθών θα αναλυθεί παρακάτω.

2.7.1 Δομικά λάθη

Τα *δομικά λάθη* αφορούν την μορφή (δομή) των κανόνων παραγωγής και είναι ανεξάρτητα της εννοιολογικής τους σημασιολογίας. Τα δομικά λάθη χωρίζονται σε 4 υποκατηγορίες (μη ολοκληρωμένοι, ασυνεπείς, κυκλικοί και πλεοναστικοί κανόνες).

i. Μη ολοκληρωμένοι κανόνες

Με τον όρο *μη ολοκληρωμένος κανόνας* αναφερόμαστε σε έναν κανόνα στον οποίο απουσιάζει είτε το τμήμα των υποθέσεων είτε το τμήμα των συμπερασμάτων.

Παράδειγμα Βάσης Κανόνων με μη ολοκληρωμένους κανόνες

R1: IfthenP_1

R2: If $P_1 \wedge P_3$ then P_2

R3: If P_1 then P_4

R4: If P_2 then

Είναι προφανές ότι στους κανόνες R1 και R4 απουσιάζουν η υπόθεση και το συμπέρασμα αντίστοιχα.

ii. Ασυνεπείς κανόνες

Με τον όρο *ασυνεπείς κανόνες* αναφερόμαστε σε μία Βάση Κανόνων στην οποία υπάρχουν κανόνες όπου τα ίδια συμπεράσματα μπορούν να εξάγουν αντικρουόμενα (αντιφατικά) αποτελέσματα.

Παράδειγμα Βάσης Κανόνων με ασυνεπείς κανόνες

R1: If $P_1 \wedge P_2$ then P_3

R2: If $P_3 \wedge P_4$ then P_5

R3: If $P_1 \wedge P_2 \wedge P_4$ then $\neg P_5$

Από την παραπάνω Βάση Κανόνων παρατηρούμε ότι αν ισχύουν οι υποθέσεις P_1, P_2 και P_4 τότε εξάγονται δύο αντιφατικά συμπεράσματα (P_5 και $\neg P_5$).

iii. Κυκλικοί κανόνες

Με τον όρο *κυκλικοί κανόνες* αναφερόμαστε σε κανόνες οι οποίοι έχουν κυκλική εξάρτηση στις υποθέσεις τους.

Παράδειγμα Βάσης Κανόνων με κυκλικούς κανόνες

R1: If P_1 then P_2

R2: If P_2 then P_3

R3: If P_3 then P_1

Στην παραπάνω Βάση Κανόνων ο κανόνας R2 έχει ως υπόθεση το συμπέρασμα του R1, ο R3 έχει ως υπόθεση το συμπέρασμα του R2 και ο R1 έχει ως υπόθεση το αποτέλεσμα του R3. Το εν λόγω γεγονός θα έχει ως αποτέλεσμα την κυκλική πυροδότηση των παραπάνω κανόνων και ουσιαστικά θα οδηγήσει σε μία ατέρμονη επανάληψη.

iv. Πλεοναστικοί κανόνες

Με τον όρο *πλεοναστικοί κανόνες* αναφερόμαστε σε μία Βάση Κανόνων η οποία εμπεριέχει κανόνες οι οποίοι στην πραγματικότητα είναι περιττοί. Ο πλεονασμός αυτός αναφορικά με τους κανόνες αυξάνει το μέγεθος της Βάσης Κανόνων και μπορεί να οδηγήσει επίσης και σε μη αναγκαίες παραγωγές συμπερασμάτων.

Παράδειγμα Βάσης Κανόνων με πλεοναστικούς κανόνες

R1: If $P_1 \wedge P_3$ then P_3

R2: If P_2 then P_4

R3: If P_2 then $P_4 \wedge P_5$

Στην παραπάνω Βάση Κανόνων ο κανόνας R2 θα μπορούσε να παραληφθεί, καθώς το συμπέρασμά του παράγεται επίσης και από τον κανόνα R3.

2.7.2 Εννοιολογικά λάθη

Τα εννοιολογικά λάθη αφορούν το περιεχόμενο των κανόνων παραγωγής και όχι την δομή τους. Τα εννοιολογικά λάθη χωρίζονται σε δύο υποκατηγορίες (μη ολοκληρωμένοι και μη έγκυροι κανόνες).

i. Μη ολοκληρωμένοι κανόνες

Οι μη ολοκληρωμένοι εννοιολογικά κανόνες είναι κανόνες οι οποίοι σε γενικές γραμμές δεν ικανοποιούν τις απαιτήσεις των χρηστών. Μερικά παραδείγματα τέτοιων λαθών είναι η απουσία κανόνων, η απουσία υποθέσεων ή συμπερασμάτων όχι από την δομή αλλά από την αντίληψη του χειριστή του συστήματος.

ii. Μη έγκυροι κανόνες

Οι μη έγκυροι εννοιολογικά κανόνες είναι κανόνες οι οποίοι οδηγούν το σύστημα να δώσει διαφορετικά αποτελέσματα σε διαφορετικές εκτελέσεις για πανομοιότυπη είσοδο.

2.8 Χρήσεις των ασαφών συστημάτων

Τα ασαφή συστήματα μπορούν να χρησιμοποιηθούν στην παραγωγή εκτιμήσεων πάνω στη λήψη αποφάσεων (decisionsupportsystems). Είναι επίσης πολύ δημοφιλή σε μηχανικά συστήματα ελέγχου, όπως για παράδειγμα σε συστήματα κλιματισμού, ελέγχου της ταχύτητας οχημάτων, συστήματα αεροσκαφών, αυτοματισμούς σε έξυπνα σπίτια, όπως επίσης και σε ελεγκτές βιομηχανικών διαδικασιών.

Η επιτυχία που γνώρισαν τα ασαφή συστήματα οφείλεται κυρίως σε δύο βασικές αδυναμίες των συμβατικών συστημάτων:

1. Πολλές διαδικασίες είναι υπερβολικά περίπλοκες και είναι δύσκολο να μοντελοποιηθούν μαθηματικά. Τέτοια παραδείγματα διαδικασιών είναι συχνά σε συστήματα διοίκησης, επιχειρήσεων καθώς και τηλεπικοινωνιών.
2. Ο όρος σταθερότητα δεν είναι εύκολο να περιγραφεί ακόμα και για τα παραδοσιακά συστήματα.

Σε γενικές γραμμές υπάρχουν πέντε διαφορετικοί τύποι συστημάτων όπου η ασαφής λογική είναι απαραίτητη, ή τουλάχιστον ωφέλιμη:

1. Πολύπλοκα Συστήματα που είναι δύσκολο ή αδύνατο να μοντελοποιηθούν με συμβατικές μαθηματικές μεθόδους.
2. Συστήματα με περίπλοκα και συνεχή δεδομένα τόσο εισόδου όσο και εξόδου.
3. Συστήματα τα οποία ελέγχονται από ειδικούς εμπειρογνώμονες και περιλαμβάνουν μία πληθώρα κανόνων και διαδικασιών.
4. Συστήματα που κάνουν χρήση της ανθρώπινης παρατήρησης ως δεδομένα εισόδου ή ως βάση για διατύπωση κανόνων.
5. Συστήματα που εμπεριέχουν εξ' ορισμού βαθμό ασάφειας. Για παράδειγμα συστήματα που σχετίζονται με οικονομικές και κοινωνικές επιστήμες.

Όσον αφορά τα συστήματα υποστήριξης αποφάσεων και τα έμπειρα συστήματα, τα κύρια πλεονεκτήματα της χρήσης ασαφών μοντέλων έναντι συμβατικών μεθόδων λειτουργίας είναι τα ακόλουθα:

1. **Ικανότητα μοντελοποίησης ιδιαίτερα πολύπλοκων επιχειρηματικών προβλημάτων:** τα ασαφή συστήματα προσφέρουν γενικευμένες προσεγγίσεις και είναι κατάλληλα για τη μοντελοποίηση πολύπλοκων επιχειρηματικών προβλημάτων. Πιο συγκεκριμένα, έχουν την ικανότητα να προσεγγίσουν τη συμπεριφορά συστημάτων που διαθέτουν έναν αριθμό ελάχιστα γνωστών χαρακτηριστικών. Τα ασαφή συστήματα

βασισμένα σε κανόνες είναι πιο αποτελεσματικά από τα παραδοσιακά συστήματα βασισμένα σε κανόνες γιατί απαιτούν λιγότερους κανόνες. Η ικανότητά τους να επεξηγούν τη συλλογιστική τους, προσφέρει έναν ιδανικό τρόπο αντιμετώπισης αυτών των προβλημάτων.

2. **Βελτιωμένη γνωστική μοντελοποίηση έμπειρων συστημάτων**: για πολλούς μηχανικούς γνώσης, ένα σημαντικό πλεονέκτημα των ασαφών συστημάτων είναι η ικανότητα άμεσης κωδικοποίησης της γνώσης. Το γεγονός αυτό αποτελεί μία σημαντική αποτυχία των παραδοσιακών έμπειρων συστημάτων και συστημάτων υποστήριξης αποφάσεων, που αναγκάζει τους χειριστές τους να αλλάξουν την μορφή των κανόνων και να την φέρουν στα μέτρα και σταθμά του εκάστοτε συστήματος.. Η διαδικασία αυτή όχι μόνο κάνει πιο πολύπλοκους τους κανόνες, αλλά στερεί από τον χειριστή την ικανότητα να διατυπώσει μια λύση σε ένα πολύπλοκο πρόβλημα.
3. **Μειωμένη Πολυπλοκότητα Μοντέλου:** τα ασαφή συστήματα απαιτούν την χρήση λιγότερων κανόνων από τα συμβατικά συστήματα και επίσης οι κανόνες αυτοί βρίσκονται πιο κοντά στον τρόπο που έκφρασης της γνώσης στη φυσική γλώσσα. Το γεγονός αυτό έχει δύο επιπλέον πλεονεκτήματα: i) το μοντέλο γνώσης μπορεί να τροποποιηθεί με λιγότερα λάθη και ii) η απλότητα που χαρακτηρίζει ένα ασαφές μοντέλο σημαίνει πρακτικά ότι λογικά ή δομικά προβλήματα μπορούν να εντοπισθούν και να επιλυθούν σε μικρό σχετικά χρόνο.
4. Ικανότητα μοντελοποίησης συστημάτων που εμπλέκουν πολλούς διαφορετικούς χρήστες: τα ασαφή συστήματα είναι κατάλληλα να υποστηρίξουν πολλούς συνεργαζόμενους χρήστες, ακόμα και αν αυτοί διαφωνούν μεταξύ τους. Έτσι, οι χρήστες του ασαφούς συστήματος είναι δυνατό να διαφωνούν παντελώς ή να υπάρχουν απόψεις διαφορετικές μεταξύ τους. Οι απόψεις αυτές, αν και αντικρουόμενες, μπορούν με τη μορφή κανόνων να μοντελοποιηθούν στα ασαφή συστήματα.

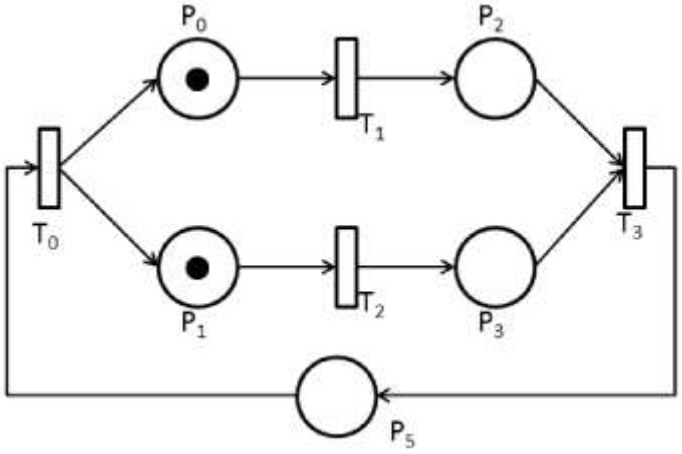
Κεφάλαιο 3 – Τα δίκτυα Petri (PetriNets)

3.1 Εισαγωγή

Τα *δίκτυα Petri (PetriNets)*προτάθηκαν για πρώτη φορά το 1962 από το CarlAdamPetri [3]. Τα δίκτυα Petriαποτελούν ένα πανίσχυρο εργαλείο μοντελοποίησης φορμαλισμών και συστημάτων όχι μόνο στην επιστήμη των υπολογιστών αλλά και γενικότερα. Τα δίκτυα αυτά συνδυάζουν μία καλά ορισμένη μαθηματική θεωρία μαζί με μία γραφική αναπαράσταση με την μορφή κατευθυνόμενου γραφήματος. Εκτός αυτού η κατάστασή τους είναι δυναμική και αλλάζει σε κάθε χρονική στιγμή. Η θεωρητική φύση των δικτύων Petri επιτρέπει μία λεπτομερή μοντελοποίηση και ανάλυση ενός συστήματος ενώ η γραφική τους φύση επιτρέπει την οπτικοποίησή του, μαζί με τις αλλαγές του στον χρόνο. Ο συνδυασμός αυτός αποτελεί τον κύριο λόγο επιτυχίας και διάδοσης των δικτύων Petri.

3.2 Χαρακτηριστικά των δικτύων Petri

Ένα δίκτυο Petrieίναι ένα *κατευθυνόμενο βεβαρημένο γράφημα* (Σχήμα 3.1) το οποίο αποτελείται από δύο ειδών κόμβους: τις *θέσεις (places)* και τις *μεταβάσεις (transitions)*.Οι θέσεις συνηθίζεται να αναπαριστούν στατικές έννοιες του συστήματος προς μοντελοποίηση ενώ αντίθετα οι μεταβάσεις αναπαριστούν γεγονότα ή ενέργειες. Οι μόνες επιτρεπτές συνδέσεις είναι μεταξύ κόμβων διαφορετικού είδους. Έτσι είναι λάθος κάποιος να συνδέσει μεταξύ τους δύο θέσεις ή δύο μεταβάσεις.



Σχήμα 3.1: παράδειγμα δικτύου Petri με 4 θέσεις και 4 μεταβάσεις.

Μέσα στις θέσεις μπορούν να τοποθετηθούν *τεκμήρια (tokens)*, τα οποία αναπαρίστανται με έναν μαύρο κύκλο. Ο μέγιστος αριθμός τεκμηρίων που μπορούν να τοποθετηθούν σε μία θέση ονομάζεται *μέγιστη χωρητικότητα* της θέσης. Ο αριθμός των τεκμηρίων που βρίσκονται σε μία θέση μία δεδομένη χρονική στιγμή ονομάζεται *τρέχουσα χωρητικότητα* της θέσης. Ο αριθμός των τεκμηρίων που βρίσκονται σε κάθε θέση στην αρχική κατάσταση του δικτύου Petrionομάζεται *αρχική χωρητικότητα*. Τα τεκμήρια μεταφέρονται από την μία θέση στην άλλη με την βοήθεια των μεταβάσεων και είναι αυτά που στην ουσία προσδίδουν την δυναμική συμπεριφορά των δικτύων Petri. Η μεταφορά ενός ή περισσοτέρων τεκμηρίων μέσω μιας μετάβασης ονομάζεται ενεργοποίηση της μετάβασης. Τα βάρη στις ακμές στην ουσία δείχνουν το πόσα τεκμήρια μπορούν να μεταφερθούν από μία θέση σε μία άλλη με την χρήση μιας μετάβασης. Είναι αρκετά σύνηθες η απουσία του βάρους σε μία ακμή να ισοδυναμεί με την τιμή 1.

Η συνδεσμολογία μεταξύ των θέσεων και των μεταβάσεων μεταβάλλει την συμπεριφορά του δικτύου. Για παράδειγμα, αν συνδέσουμε μία θέση με δύο μεταβάσεις, τότε το δίκτυο γίνεται μη ντετερμινιστικό, αφού η ενεργοποίηση των μεταβάσεων γίνεται ισοπίθανη. Φυσικά μπορεί ένα δίκτυο να είναι πλήρως ντετερμινιστικό ανάλογα πάλι με την συνδεσμολογία. Οι ακριβείς κανόνες λειτουργίας θα παρουσιαστούν αναλυτικά στις παρακάτω ενότητες.

3.3Τυπικός Ορισμός των δικτύων Petri

Ένα δίκτυο Petriμπορεί να οριστεί τυπικά [4] ως μία διατεταγμένη εξάδα στοιχείων $N = (P, T, I, O, M_{max}, M(t))$, όπου:

(1) $P = \{p_1, p_2, \ldots, p_m\}$ είναι ένα σύνολο από θέσεις.

(2) $T = \{t_1, t_2, \ldots, t_n\}$ είναι ένα σύνολο από μεταβάσεις όπου $P \cup T \neq \emptyset$ και $P \cap T = \emptyset$.

(3) $I: P \times T \rightarrow N$ είναι μία *συνάρτηση εισόδου (inputfunction)* η οποία ορίζει τις κατευθυνόμενες ακμές από τις θέσεις προς τις μεταβάσεις. Στην προκείμενη

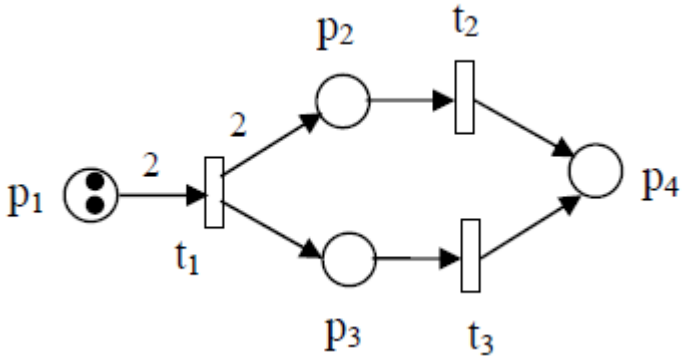
περίπτωση, το σύνολο N είναι ένα σύνολο μη αρνητικών ακεραίων. Η τιμή 0 της συνάρτησης ισοδυναμεί με την μη ύπαρξη κατευθυνόμενης ακμής μεταξύ της θέσης και της μετάβασης.

(4) $O: T \times P \rightarrow N$ είναι μία *συνάρτηση εξόδου (outputfunction)* η οποία ορίζει τις κατευθυνόμενες ακμές από τις μεταβάσεις προς τις θέσεις. Και στην περίπτωση αυτή, το σύνολο N είναι ένα σύνολο μη αρνητικών ακεραίων. Επίσης και εδώ η τιμή 0 της συνάρτησης ισοδυναμεί με την μη ύπαρξη κατευθυνόμενης ακμής μεταξύ της μετάβασης και της θέσης.

(5) $M_{max}: P \rightarrow N$ είναι μία συνάρτηση η οποία δείχνει την μέγιστη χωρητικότητα της κάθε θέσης.

(6) $M(t): P \rightarrow N$ είναι μία συνάρτηση η οποία δείχνει την τρέχουσα χωρητικότητα της κάθε θέσης την χρονική στιγμή t .Για t = 0 αναφερόμαστε στην αρχική κατάσταση του δικτύου.

Ένα παράδειγμα έκφρασης ενός δικτύου Petri με βάση τον παραπάνω ορισμό παρουσιάζεται στο Σχήμα 3.2.



Σχήμα 3.2: ένα απλό δίκτυο Petri με 4 θέσεις και 3 μεταβάσεις το οποίο βρίσκεται στην αρχική του κατάσταση.

Στο παραπάνω σχήμα ισχύουν τα εξής:

(1) $P = \{p_1, p_2, p_3, p_4\}$

(2) $T = \{t_1, t_2, t_3\}$

(3) $I(p_1, t_1) = 2$, $I(p_1, t_2) = 0$, $I(p_1, t_3) = 0$
$I(p_2, t_1) = 0$, $I(p_2, t_2) = 1$, $I(p_2, t_3) = 0$
$I(p_3, t_1) = 0$, $I(p_3, t_2) = 0$, $I(p_3, t_3) = 1$
$I(p_4, t_1) = 0$, $I(p_4, t_2) = 0$, $I(p_4, t_3) = 0$

(4) $O(t_1, p_1) = 0$, $O(t_1, p_2) = 2$, $O(t_1, p_3) = 1$, $O(t_1, p_4) = 0$
$O(t_2, p_1) = 0$, $O(t_2, p_2) = 0$, $O(t_2, p_3) = 0$, $O(t_2, p_4) = 1$
$O(t_3, p_1) = 0$, $O(t_3, p_2) = 0$, $O(t_3, p_3) = 0$, $O(t_3, p_4) = 1$

(5) $M_{\max(p_1)} = 2$
$M_{\max(p_2)} = 2$
$M_{\max(p_3)} = 1$
$M_{\max(p_4)} = 2$

(6) $M(t = 0, p_1) = 2$
$M(t = 0, p_2) = 0$
$M(t = 0, p_3) = 0$
$M(t = 0, p_4) = 0$

Είναι προφανές επίσης ότι:

i. Η συνάρτηση $I: P \times T \rightarrow N$ μπορεί επίσης να αναπαρασταθεί με την χρήση ενός δισδιάστατου πίνακα διαστάσεων $P \times T$. Για παράδειγμα, για το δίκτυο Petrito οποίο παρουσιάζεται στο Σχήμα 3.2 ο πίνακας θα ήταν ο παρακάτω:

$$I = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

ii. Η συνάρτηση $O: T \times P \rightarrow N$ μπορεί επίσης να αναπαρασταθεί με την χρήση ενός δισδιάστατου πίνακα διαστάσεων $T \times P$. Για παράδειγμα, για το δίκτυο Petrito οποίο παρουσιάζεται στο Σχήμα 3.2 ο πίνακας θα ήταν ο παρακάτω:

$$O = \begin{bmatrix} 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

iii. Η συνάρτηση $M_{max}: P \rightarrow N$ μπορεί να αναπαρασταθεί με ένα διάνυσμα διάστασης P . Για παράδειγμα, για το δίκτυο Petrito οποίο παρουσιάζεται στο Σχήμα 3.2 το διάνυσμα θα ήταν το παρακάτω:

$$M_{max} = [2 \quad 2 \quad 1 \quad 2]^T$$

iv. Η συνάρτηση $M(t): P \rightarrow N$ μπορεί να αναπαρασταθεί με ένα διάνυσμα διάστασης P . Για παράδειγμα, για το δίκτυο Petrito οποίο παρουσιάζεται στο Σχήμα 3.2, για t = 0, το διάνυσμα θα ήταν το παρακάτω:

$$M(0) = [2 \quad 0 \quad 0 \quad 0]^T$$

3.4 Κανόνες πυροδότησης

Στην ενότητα αυτή θα παρουσιάσουμε τους κανόνες πυροδότησης οι οποίοι ισχύουν σε ένα οποιοδήποτε δίκτυο Petri. Πιο συγκεκριμένα, με τον όρο *πυροδότηση (firing)*αναφερόμαστε στην ενεργοποίηση μιας εκ των μεταβάσεων του δικτύου. Κάθε χρονική στιγμή σε ένα δίκτυο Petrικάποιες από τις μεταβάσεις είναι σε θέση να πυροδοτηθούν ενώ κάποιες άλλες όχι. Θα αναφερόμαστε στις μεταβάσεις οι οποίες μπορούν να πυροδοτηθούν ως ικανοποιήσιμες. Η πυροδότηση μιας ικανοποιήσιμης μετάβασης έχει ως αποτέλεσμα την μεταφορά τεκμηρίων στις θέσεις του δικτύου. Αν σε μία δεδομένη χρονική στιγμή δεν υπάρχει ούτε μία μετάβαση η οποία είναι ικανοποιήσιμη τότε λέμε ότι το δίκτυο Petriβρίσκεται σε *αδιέξοδο (deadlock)*. Οι κανόνες πυροδότησης παρουσιάζονται παρακάτω. Για να μπορέσουμε να ορίσουμε τους κανόνες με τυπικό τρόπο, θα πρέπει αρχικά να δώσουμε κάποιους ορισμούς.

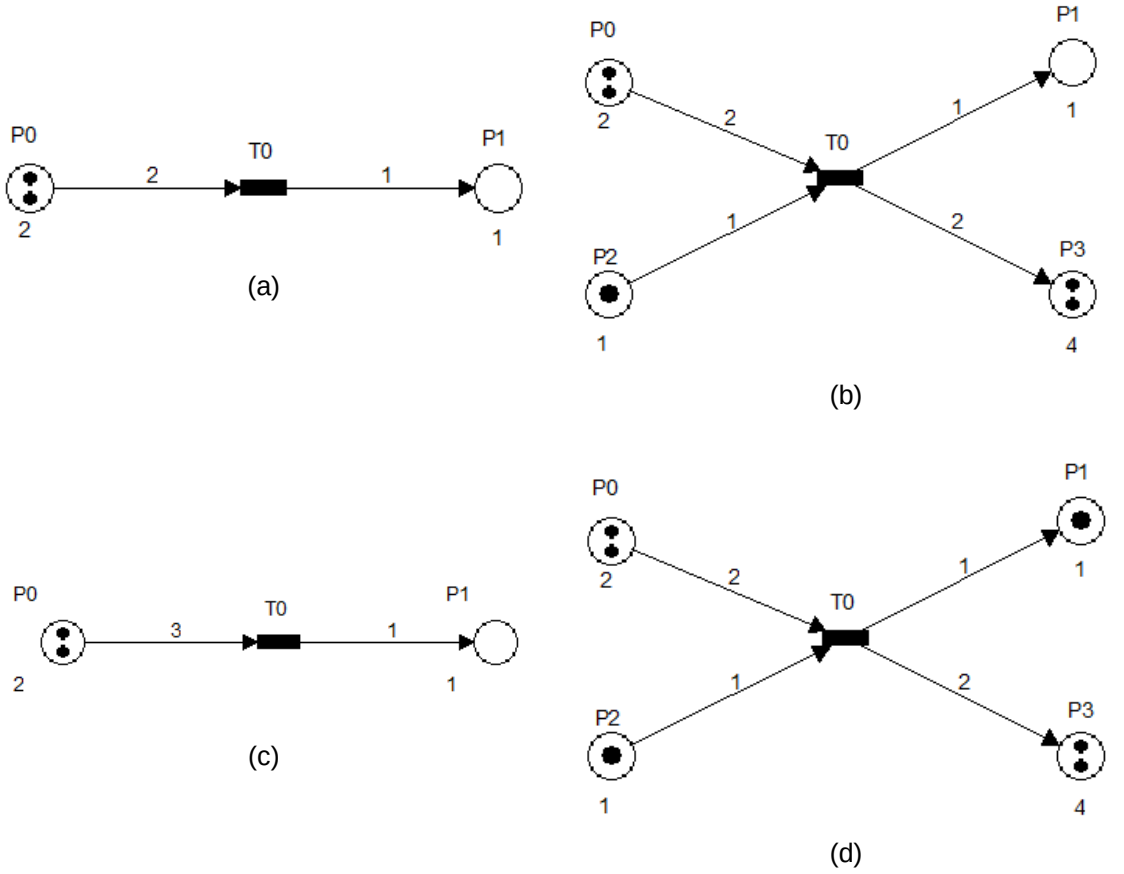
Ορισμός: με τον *όρο εισερχόμενη ακμή* μιας μετάβασης t_i αναφερόμαστε στις ακμές εκείνες για τις οποίες ισχύει ότι $I(p_j, t_i) > 0$. Από εδώ και στο εξής θα συμβολίζουμε την εισερχόμενη ακμή αυτή ως e_{ji} .

Ορισμός: με τον όρο *εξερχόμενη ακμή* μιας μετάβασης t_i αναφερόμαστε στις ακμές εκείνες για τις οποίες ισχύει ότι $O(t_i, p_j) > 0$.Από εδώ και στο εξής θα συμβολίζουμε την εισερχόμενη ακμή αυτή ως e_{ij} .

Ορισμός: μία μετάβαση t_i είναι *ικανοποιήσιμη* την χρονική στιγμή t όταν:

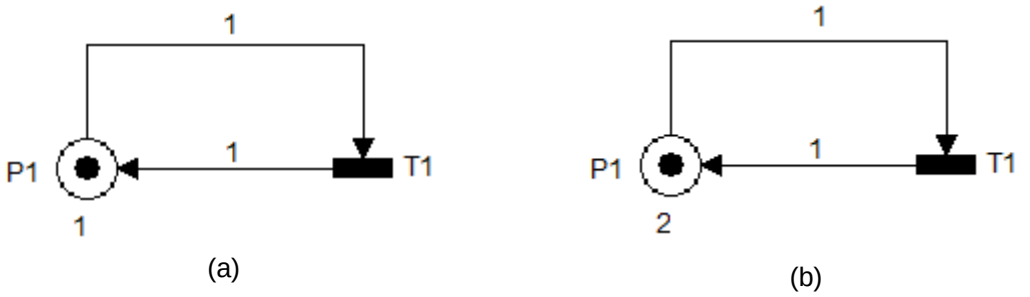
1. για κάθε εισερχόμενη ακμή της e_{ji} ισχύει ότι $M(t,p_j) \geq I(p_j,t_i)$ και
2. για κάθε εξερχόμενη ακμή της e_{ij} ισχύει ότι $M(t,p_j) + O(t_i,p_j) \leq M_{\max(p_j)}$

Στην ουσία δηλαδή θα πρέπει η μετάβαση t_i να είναι σε θέση να «τραβήξει» τεκμήρια από την συνδεδεμένη θέση κάθε εισερχόμενης ακμής της και να «δώσει» τεκμήρια σε κάθε συνδεδεμένη θέση κάθε εξερχόμενης ακμής της. Μερικά παραδείγματα ικανοποιήσιμων και μη ικανοποιήσιμων μεταβάσεων παρουσιάζονται στο Σχήμα 3.3.



Σχήμα 3.3: (a) και (b): η μετάβαση T0 είναι ικανοποιήσιμη. (c) και (d): η μετάβαση T0 δεν είναι ικανοποιήσιμη.

Μία ειδική περίπτωση ικανοποιησιμότητας είναι όταν μία μετάβαση έχει και εισερχόμενη και εξερχόμενη ακμή από και προς την ίδια θέση (ανακύκλωση). Ένα τέτοιο παράδειγμα παρουσιάζεται στο Σχήμα 3.4. Στην περίπτωση αυτή ι μετάβαση εξετάζεται την χρονική στιγμή t ξεχωριστά για την εισερχόμενη και για την εξερχόμενη ακμή της. Τέτοιες μεταβάσεις συνήθως δημιουργούν σύγχιση στον αναγνώστη ενός δικτύου Petriγιατί ενώ αναμένει σε κάποιες περιπτώσεις να είναι ικανοποιήσιμη, σύμφωνα με τον παραπάνω ορισμό δεν είναι.



Σχήμα 3.4: (α) Η T1 δεν είναι ικανοποιήσιμη αφού $M(t,p_1) + O(t_1,p_1) > M_{max}(p_1)$. (β) Η T1 είναι ικανοποιήσιμη αφού με την αύξηση της μέγιστης χωρητικότητας της P1 σε $M_{max}(p_1) = 2$ ισχύει πλέον ότι $M(t,p_1) + O(t_1,p_1) = M_{max}(p_1)$.

Με την χρήση των παραπάνω ορισμών, μπορούμε να ορίσουμε τους κανόνες πυροδότησης:

Κανόνας πυροδότησης 1:όταν μία ικανοποιήσιμη μετάβαση t_i πυροδοτείται την χρονική στιγμή t τότε εκτελεί και τις δύο παρακάτω ενέργειες:

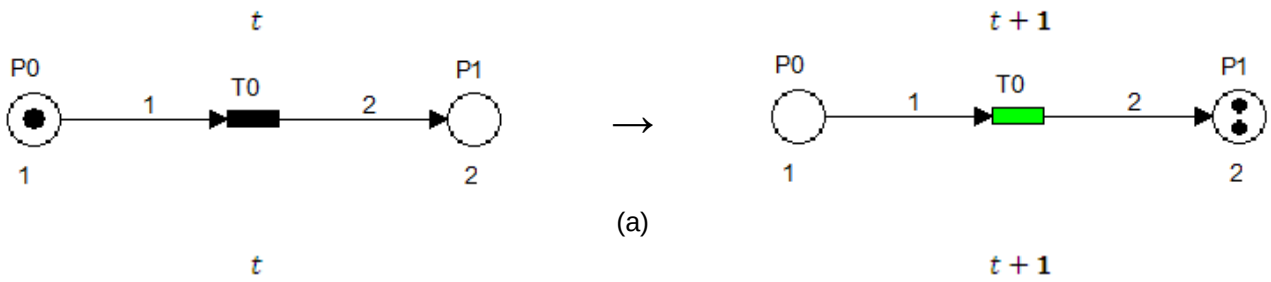
1. Για κάθε θέση p_j για την οποία ισχύει ότι $I(p_j,t_i) > 0$, $M(t+1,p_j) = M(t,p_j) - I(p_j,t_i)$.
2. Για κάθε θέση p_j για την οποία ισχύει ότι $O(t_i,p_j) > 0$, $M(t+1,p_j) = M(t,p_j) + O(t_i,p_j)$.

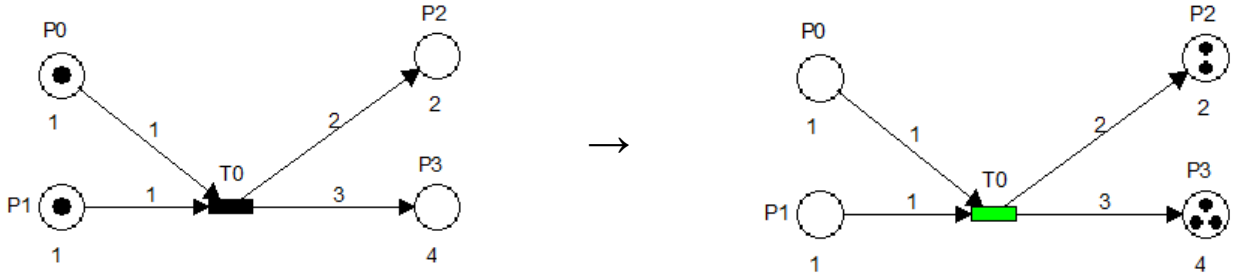
Με λιγότερο τυπικό τρόπο αυτό το οποίο συμβαίνει είναι το εξής:

1. Αφαιρεί τεκμήρια από κάθε θέση με την οποία είναι συνδεδεμένη με εισερχόμενη ακμή. Ο αριθμός των τεκμηρίων τα οποία καταναλώνει ορίζονται από το βάρος της εισερχόμενης ακμής.
2. Προσθέτει τεκμήρια σε κάθε θέση με την οποία είναι συνδεδεμένη με εξερχόμενη ακμή. Ο αριθμός των τεκμηρίων τα οποία προσθέτει ορίζονται από το βάρος της εξερχόμενης ακμής.

Στο σημείο αυτό θα πρέπει να γίνει σαφές το εξής: δεν είναι απαραίτητο ο συνολικός αριθμός των τεκμηρίων που εισέρχονται σε μία μετάβαση να είναι ίσος με τον συνολικό αριθμό των τεκμηρίων που εξέρχονται από αυτή.

Παραδείγματα πυροδοτήσεων με βάση τον Κανόνα 1 παρουσιάζονται στο Σχήμα 3.5.



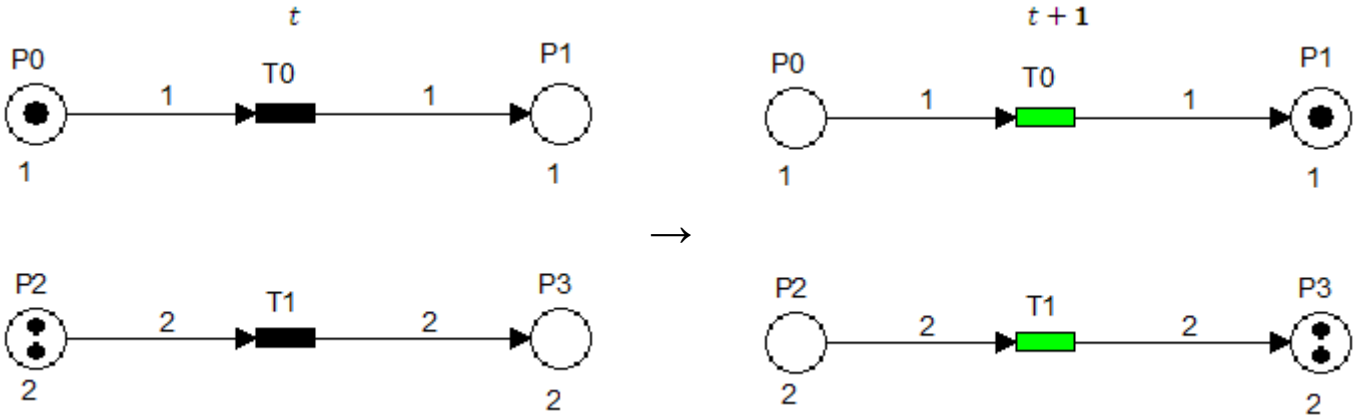


(b)

Σχήμα 3.5: (a) και (b) παραδείγματα πυροδοτήσεων μεταβάσεων.

Κανόνας πυροδότησης 2: όταν δύο ή περισσότερες ικανοποιήσιμες μεταβάσεις την χρονική στιγμή t δεν έχουν καμία κοινή θέση, τόσο στις εισερχόμενες όσο και στις εξερχόμενες ακμές τους, τότε πυροδοτούνται ταυτόχρονα.

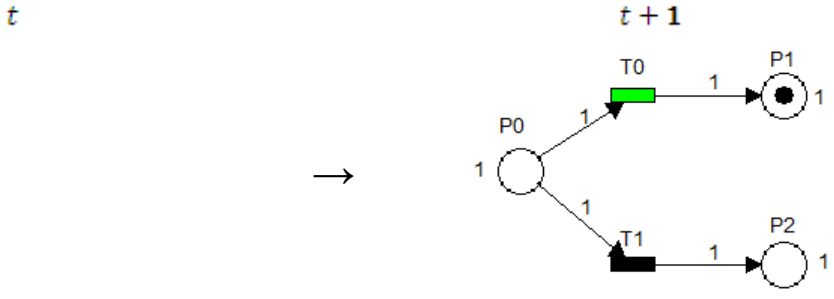
Ένα παράδειγμα πυροδότησης με βάση τον Κανόνα 2 παρουσιάζεται στο Σχήμα 3.6.

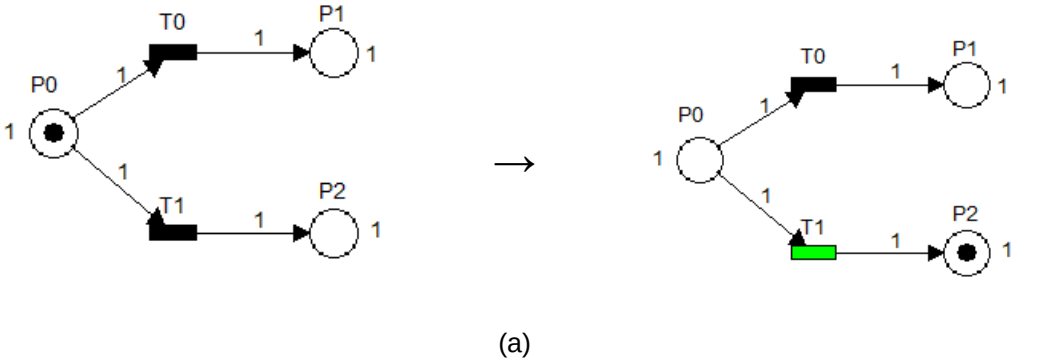


Σχήμα 3.6: παράδειγμα ταυτόχρονης πυροδότησης μεταβάσεων.

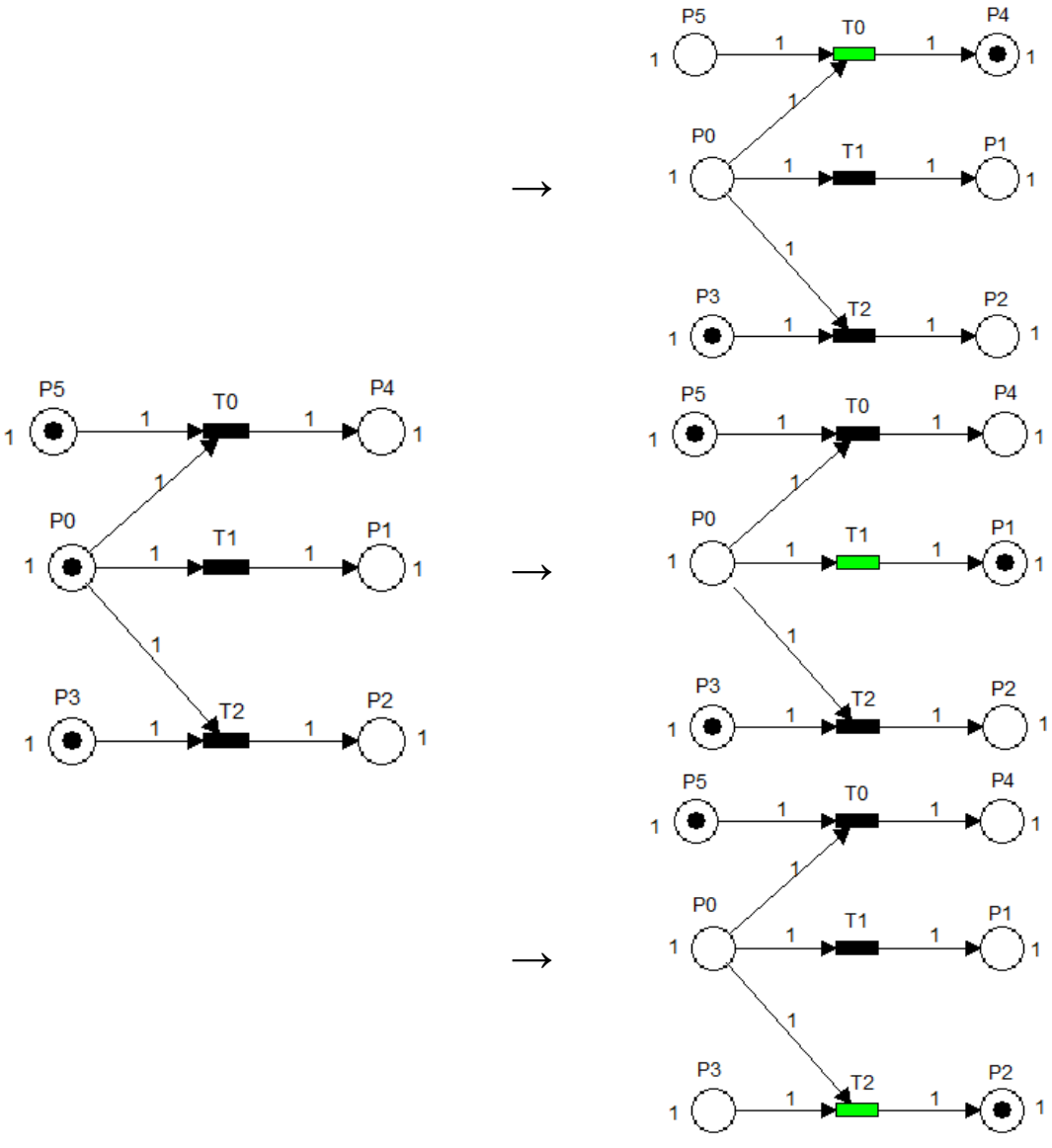
Κανόνας πυροδότησης 3: αν m στον αριθμό ικανοποιήσιμες μεταβάσεις μεταβάσεις την χρονική στιγμή t έχουν τουλάχιστον μία κοινή θέση είτε αυτή αφορά εισερχόμενη είτε εξερχόμενη ακμή τότε είναι ισοπίθανες προς πυροδότηση με πιθανότητα $\frac{1}{m}$.

Παραδείγματα πυροδοτήσεων με βάση τον Κανόνα 1 παρουσιάζονται στο Σχήμα 3.7.





(a)

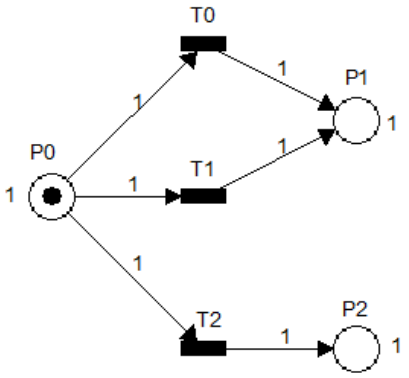


(b)

Σχήμα 3.7: (a) και (b) παραδείγματα πυροδοτήσεων ισοπίθανων μεταβάσεων.

Ως απόρια του Κανόνα 3, η πιθανότητα εκτέλεσης ενός γεγονότος μπορεί να μην είναι κατ' ανάγκη ισοπίθανη με όλα τα υπόλοιπα ενδεχόμενα, αν ο σχεδιασμός του δικτύου Petri είναι κατάλληλος. Πιο συγκεκριμένα, είναι προφανές ότι στο Σχήμα 3.7 (a) η πιθανότητα μεταφοράς

του τεκμηρίου από την θέση P0 στην θέση P1 είναι 50%, αφού οι δύο μεταβάσεις T0και T1 είναι ισοπίθανες. Ας υποθέσουμε ότι είναι επιθυμητό η πιθανότητα αυτή να αυξηθεί σε 2/3 ενώ αντίστοιχα η πιθανότητα μεταφοράς του τεκμηρίου από την θέση P0 στην θέση P2 να μειωθεί σε 1/3. Αν ο σχεδιασμός του δικτύου Petrieίναι αυτός ο οποίος αναπαρίσταται στο Σχήμα 3.8, τότε αυτό γίνεται πραγματικότητα.



Σχήμα 3.8: αλλαγή των προκαθορισμένων πιθανοτήτων μέσω επανασχεδιασμού του δικτύου Petri.

3.4. Δυνατότητες μοντελοποίησης των δικτύων Petri

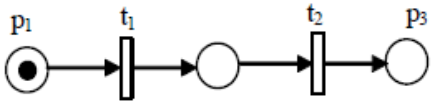
Τα τυπικά χαρακτηριστικά τα οποία διέπουν ένα οποιοδήποτε δυναμικό σύστημα είναι τα ακόλουθα [4]:

1. Σειριακή εκτέλεση
2. Επίλυση συγκρούσεων
3. Ταυτόχρονη εκτέλεση
4. Συγχρονισμός
5. Αμοιβαίος αποκλεισμός

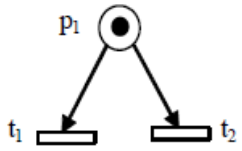
Τα χαρακτηριστικά αυτά μπορούν να μοντελοποιηθούν αποδοτικά με την χρήση δικτύων Petri. Πιο συγκεκριμένα, και με την βοήθεια του Σχήματος 3.9:

1. *Σειριακή εκτέλεση:* στο Σχήμα 3.9(a), η μετάβαση t_2 μπορεί να πυροδοτηθεί μόνο έπειτα από την πυροδότηση της μετάβασης t_1. Τέτοιοι περιορισμοί προτεραιότητας είναι συνηθισμένοι στην εκτέλεση ανάμεσα στα διάφορα συστατικά μέρη που απαρτίζουν ένα δυναμικό σύστημα.
2. *Επίλυση συγκρούσεων:* στο Σχήμα 3.9(b) οι μεταβάσεις t_1 και t_2 βρίσκονται σε καθεστώς σύγκρουσης. Πιο συγκεκριμένα και οι δύο είναι ικανοποιήσιμες αλλά λόγω του Κανόνα πυροδότησης 3 μόνο μία από τις δύο μεταβάσεις θα πυροδοτηθεί, με έναν μη ντετερμινιστικό, πιθανοτικό τρόπο. Στο εν λόγω παράδειγμα οι μεταβάσεις t_1 και t_2 είναι ισοπίθανες. Αν ένα γεγονός έχει μεγαλύτερες πιθανότητες εκτέλεσης έναντι ενός άλλου, τότε με κατάλληλη κατασκευή του δικτύου (για παράδειγμα όπως στο Σχήμα 3.8) μπορούμε να επιτύχουμε το επιθυμητό αποτέλεσμα.
3. *Ταυτόχρονη εκτέλεση:* στο Σχήμα 3.9 (c)οι μεταβάσεις t_1 και t_2 θα πυροδοτηθούν ταυτόχρονα (Κανόνας πυροδότησης 2).Η ταυτόχρονη εκτέλεση είναι ένα σημαντικό χαρακτηριστικό των συστατικών μερών ενός δυναμικού συστήματος.
4. *Συγχρονισμός:* είναι αρκετά σύνηθες σε ένα δυναμικό σύστημα ένα γεγονός να απαιτεί πολλούς πόρους για να μπορεί να εκτελεστεί. Ο απαιτούμενος συγχρονισμός για την συγκέντρωση των πόρων αυτών μπορεί να μπορεί να μοντελοποιηθεί με την χρήση μεταβάσεων σαν αυτή που παρουσιάζεται στο Σχήμα 3.9 (d).Στο παράδειγμα αυτό η μετάβαση t_1 πυροδοτείται μόνο όταν υπάρχει τεκμήριο τόσο στην θέση p_1 όσο και στην θέση p_2. Στην ουσία η μετάβαση t_1 συγχρονίζει τις διαδικασίες οι οποίες απαιτούνται έτσι ώστε οι θέσεις p_1 και p_2 να λάβουν τα απαιτούμενα κριτήρια.

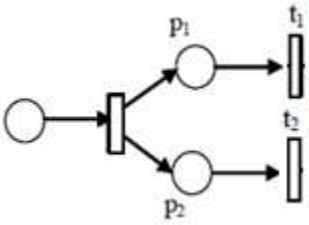
5. *Αμοιβαίος αποκλεισμός:* ο αμοιβαίος αποκλεισμός αποτελεί ένα από τα πλέον συνηθέστερα προβλήματα συγχρονισμού διεργασιών στα λειτουργικά συστήματα. Το πρόβλημα αυτό εμφανίζεται όταν δύο ή περισσότερες παράλληλες διεργασίες πρέπει να αποκτήσουν ατομικά η κάθε μία πρόσβαση σε έναν κοινόχρηστο πόρο. Στο Σχήμα 3.9 (f) παρουσιάζεται ένα παράδειγμα υλοποίησης του αμοιβαίου αποκλεισμού με την χρήση δικτύου Petriγια δύο διεργασίες p_2 και p_3. Επειδή οι μεταβάσεις t_1 και t_3 είναι ισοπίθανες, μόνο μία από τις δύο θα εκτελείται κάθε φορά με αποτέλεσμα το τεκμήριο από την θέσηp_1 να μεταβαίνει είτε στην θέση p_2 είτε στην θέση p_3.



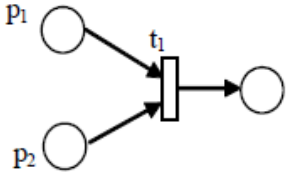
(a) σειριακή εκτέλεση



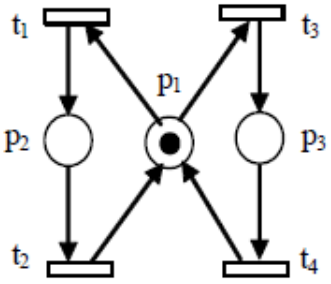
(b) επίλυση συγκρούσεων



(c) ταυτόχρονη εκτέλεση



(d) συγχρονισμός



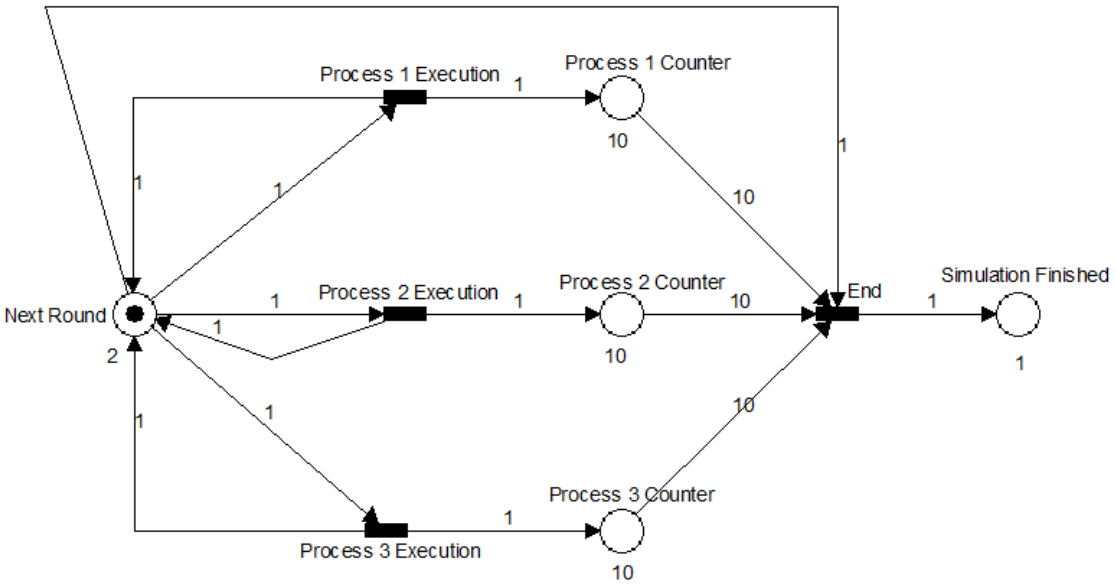
(f) αμοιβαίος αποκλεισμός

Σχήμα 3.9: βασικά παραδείγματα μοντελοποίησης των δικτύων Petri.

Στο σημείο αυτό κρίνεται σκόπιμο να δοθεί ένα παράδειγμα μοντελοποίησης ενός δυναμικού συστήματος με την χρήση ενός δικτύου Petri. Ας θεωρήσουμε ένα σύστημα το οποίο κάθε χρονική στιγμή μπορεί να εκτελέσει με τυχαίο τρόπο οποιαδήποτε από 3 διεργασίες (Process 1, 2 και 3). Κάθε μία από τις διεργασίες αυτές μπορεί να εκτελεστεί το πολύ 10 φορές. Όταν και οι 3 διεργασίες εκτελεστούν από 10 φορές η καθεμία, τότε το σύστημα τερματίζει την λειτουργία του.

Ένα δίκτυο Petri(όχι απαραίτητα το μοναδικό) το οποίο μπορεί να μοντελοποιήσει την παραπάνω περιγραφή παρουσιάζεται στο Σχήμα 3.10 το οποίο ακολουθεί. Στο σχήμα αυτό, οι εκτελέσεις των διεργασιών μοντελοποιούνται ως μεταβάσεις με ονομασίες *Process 1 Execution*, *Process 2 Execution*και *Process 3 Execution*. Οι μεταβάσεις αυτές είναι ισοπίθανες αρχικά, καθώς και οι 3 λαμβάνουν εισερχόμενη ακμή από την κοινή θέση *NextRound*, η οποία έχει ένα τεκμήριο. Η θέση *NextRound* αναπαριστά στην ουσία την μετάβαση στην επόμενη χρονική στιγμή και έχει χωρητικότητα 2, αφού είναι επιθυμητό μετά την εκτέλεση οποιασδήποτε εκ των 3 ισοπίθανων μεταβάσεων το τεκμήριο να επιστρέφει πίσω άμεσα (παρατηρούμε ότι η κάθε μετάβαση έχει και μία εξερχόμενη ακμή προς την θέση *NextRound*). Η απαίτηση του άνω ορίου

εκτελέσεων (10) κάθε μετάβασης ρυθμίζεται με την χρήση των θέσεων *Process 1 Counter*, *Process 2 Counter*και *Process 3 Counter*οι οποίες παίζουν τον ρόλο των μετρητών. Πιο συγκεκριμένα, κάθε φορά που μια από τις 3 μεταβάσεις πυροδοτείται στέλνει ένα τεκμήριο στον αντίστοιχο μετρητή με τον οποίον είναι συνδεδεμένη. Όταν ο μετρητής γεμίσει με 10 τεκμήρια, η αντίστοιχη μετάβαση δεν θα είναι σε θέση να πυροδοτηθεί ξανά, αφού δεν θα είναι ικανοποιήσιμη. Όταν και οι 3 θέσεις *Process 1 Counter*, *Process 2 Counter*και*Process 3 Counter* έχουν τρέχουσα χωρητικότητα ίση με 10, τότε η μετάβαση *End* γίνεται ικανοποιήσιμη. Η μετάβαση αυτή καταναλώνει τα 10 τεκμήρια από κάθε έναν από τους 3 μετρητές αλλά επίσης καταναλώνει και το τεκμήριο της θέσης *NextRound*, έτσι ώστε το δίκτυο να μην μπορεί να εκτελεστεί ξανά και να καταλήξει σε αδιέξοδο. Όταν η μετάβαση *End*πυροδοτηθεί, στέλνει ένα τεκμήριο στην θέση *SimulationFinished*, η παρουσία του οποίου στην θέση αυτή σημαίνει ότι το δίκτυο Petri βρίσκεται πλέον σε αδιέξοδο.



Σχήμα 3.10: το δίκτυο Petri της παραπάνω περιγραφής την χρονική στιγμή t = 0.

3.5 Ταξινόμηση των δικτύων Petri

Τα δίκτυα Petri μπορούν να ταξινομηθούν σε κατηγορίες, ανάλογα με την αρχιτεκτονική τους [8]. Με τον όρο αρχιτεκτονική ενός δικτύου Petriαναφερόμαστε στα σύνολα τιμών των συναρτήσεων I , O , $M_{\max}$ και $M(t)$, έτσι όπως αυτές ορίστηκαν παραπάνω. Οι κατηγορίες των δικτύων Petri παρουσιάζονται παρακάτω [8].

Για τον μαθηματικό ορισμό των περισσότερων από τις κατηγορίες που ακολουθούν, θα πρέπει να ορίσουμε τους παρακάτω συμβολισμούς.

Ορισμός: το σύνολο των θέσεων από τις οποίες δέχεται εισερχόμενη ακμή μια μετάβαση$t_j \in T$ συμβολίζεται με $\rightarrow t_j$. Πιο συγκεκριμένα: $\rightarrow t_j = \{p \in P : I(p,t_j) > 0\}, t_j \in T$.

Ορισμός: το σύνολο των θέσεων στις οποίες στέλνει εξερχόμενη ακμή μια μετάβαση$t_j \in T$ συμβολίζεται με $t_j \rightarrow$. Πιο συγκεκριμένα: $t_j \rightarrow = \{p \in P : O(t_j,p) > 0\}, t_j \in T$.

Ορισμός: το σύνολο των μεταβάσεων από τις οποίες δέχεται εισερχόμενη ακμή μια θέση $p_i \in P$ συμβολίζεται με $\rightarrow p_i$. Πιο συγκεκριμένα: $\rightarrow p_i = \{t \in T : I(p_i,t) > 0\}, p_i \in P$.

Ορισμός: το σύνολο των μεταβάσεων στις οποίες στέλνει εξερχόμενη ακμή μια θέση$p_i \in P$ συμβολίζεται με $p_i \rightarrow$. Πιο συγκεκριμένα: $p_i \rightarrow = \{t \in T : O(t,p_i) > 0\}, p_i \in P$.

Με βάση τους παραπάνω συμβολισμούς, οι κατηγορίες των δικτύων Petri παρουσιάζονται στην συνέχεια.

3.5.1 ΔίκτυαPetri Θέσης/Μετάβασης (Place/Transition Petri Nets)

Σε αυτή την κατηγορία ανήκουν όλα τα δίκτυα Petri, ανεξαρτήτως αρχιτεκτονικής. Η κάθε θέση μπορεί να έχει οποιαδήποτε μέγιστη χωρητικότητα ενώ δεν υπάρχει κανένας περιορισμός για τα βάρη των ακμών(Σχήμα 3.11).

3.5.2 Συνήθη ΔίκτυαPetri (OrdinaryPetriNets)

Σε αυτή την κατηγορία ανήκουν τα δίκτυα Petriισχύει ότι:

(1) $I(p_i,t_j) \in \{0,1\}, \quad \forall p_i \in P \; και \; \forall t_j \in T$

(2) $O(t_j,p_i) \in \{0,1\}, \forall p_i \in P \; και \; \forall t_j \in T$

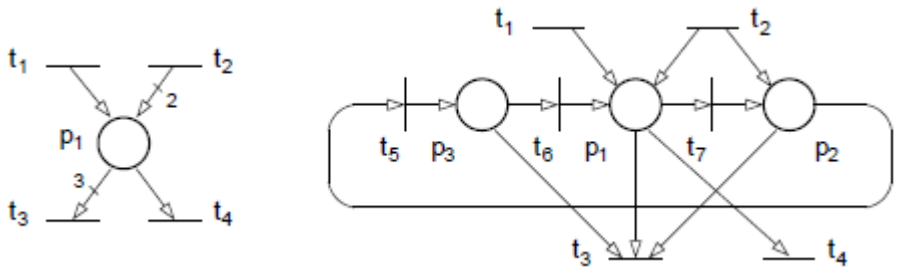
Με άλλα λόγια στην κατηγορία αυτή ανήκουν όλα εκείνα τα δίκτυα Petriστα οποία κάθε υπαρκτή ακμή έχει βάρος ακριβώς 1 (Σχήμα 3.11).

3.5.3 ΔίκτυαPetri Συνθήκης/Γεγονότος (Condition/Event Petri Nets)

Σε αυτή την κατηγορία ανήκουν τα συνήθη δίκτυα Petriγια τα οποία επιπρόσθετα ισχύει ότι:

$$M_{\max(p_i)} = 1, \qquad \forall\, p_i \in P$$

Με άλλα λόγια στην κατηγορία αυτή ανήκουν όλα εκείνα τα δίκτυα Petriστα οποία η μέγιστη χωρητικότητα της κάθε θέσης είναι 1 (μπορεί δηλαδή να λάβει το πολύ 1 τεκμήριο) ενώ κάθε υπαρκτή ακμή έχει βάρος ακριβώς 1. Όπως είναι προφανές, στον ορισμό των συνήθων δικτύων προστίθεται ο περιορισμός για την μέγιστη χωρητικότητα των θέσεων. Με την χρήση των δικτύων PetriΣυνθήκης/Γεγονότος μπορούμε να μοντελοποιήσουμε συστήματα με την χρήση λογικών τιμών (οι οποίες μπορούν να λάβουν μόνο τις τιμές 0 και 1) (Σχήμα 3.11).



(a) (b)

Σχήμα 3.11 [8]: (a) δίκτυο Petri θέσης μετάβασης και όχι σύνηθες. (b) δίκτυο Petri θέσης μετάβασης, σύνηθες και συνθήκης γεγονότος.

3.5.4 Μηχανές Καταστάσεων (StateMachines)

Σε αυτή την κατηγορία ανήκουν τα συνήθη δίκτυα Petriγια τα οποία επιπρόσθετα ισχύει ότι:

(1) $|\rightarrow t_j| = 1, \forall t_j \in T$

(2)

Με άλλα λόγια στην κατηγορία αυτή ανήκουν τα συνήθη δίκτυα Petriγια τα οποία ισχύει ότι κάθε μετάβαση έχει ακριβώς μία εισερχόμενη ακμή και ακριβώς μία εξερχόμενη (Σχήμα 3.12).

(a) (b)

Σχήμα 3.12 [8]: (a) σύνηθες δίκτυο Petri το οποίο είναι και μηχανή καταστάσεων. (b) σύνηθες δίκτυο Petri το οποίο δεν είναι μηχανή καταστάσεων.

3.5.4 Μαρκαρισμένα γραφήματα (MarkedGraphs)

Σε αυτή την κατηγορία ανήκουν τα συνήθη δίκτυα Petriγια τα οποία επιπρόσθετα ισχύει ότι:

(1) $|\rightarrow p_i| = 1, \forall p_i \in P$
(2)

Με άλλα λόγια στην κατηγορία αυτή ανήκουν τα συνήθη δίκτυα Petriγια τα οποία ισχύει ότι κάθε θέση έχει ακριβώς μία εισερχόμενη ακμή και ακριβώς μία εξερχόμενη (Σχήμα 3.13).

(a) (b)

Σχήμα 3.13 [8]: (a) σύνηθες δίκτυο Petri το οποίο είναι και μαρκαρισμένο γράφημα. (b) σύνηθες δίκτυο Petri το οποίο δεν είναι μαρκαρισμένο γράφημα.

3.5.5 Δίκτυα Petriελεύθερης επιλογής (FreeChoicePetriNets)

Σε αυτή την κατηγορία ανήκουν τα συνήθη δίκτυα Petriγια τα οποία επιπρόσθετα ισχύει ότι:

$$\forall p_i \in P : |p_i \rightarrow| > 1 \Rightarrow |\rightarrow t_k| = 1 \; \forall t_k \in p_i \rightarrow$$

Με άλλα λόγια στην κατηγορία αυτή ανήκουν τα συνήθη δίκτυα Petriγια τα οποία ισχύει ότι αν μία μετάβαση t_k λαμβάνει εισερχόμενη ακμή από την μία θέση p_i, τότε δεν μπορεί να λάβει εισερχόμενη ακμή από καμία άλλη θέση (Σχήμα 3.14).

(a) (b)

Σχήμα 3.14 [8]: (a) σύνηθες δίκτυο Petri το οποίο είναι και ελεύθερης επιλογής. (b) σύνηθες δίκτυο Petri το οποίο δεν είναι ελεύθερης επιλογής.

3.5.6 Απλά ΔίκτυαPetri (Simple Petri Nets)

Σε αυτή την κατηγορία ανήκουν τα συνήθη δίκτυα Petriγια τα οποία επιπρόσθετα ισχύει ότι:

$$\forall t_j \in T : |\{p_k \in \rightarrow t_j\} : |p_k \rightarrow| > 1\}| \leq 1$$

Με άλλα λόγια στην κατηγορία αυτή ανήκουν τα συνήθη δίκτυα Petriγια τα οποία ισχύει ότι κάθε μετάβαση t_j έχει το πολύ μία θέση p_k που δίνει εισερχόμενη ακμή τόσο στην ίδια όσο και σε άλλη μετάβαση (Σχήμα 3.15).

(a) (b)

Σχήμα 3.15 [8]: (a) σύνηθες δίκτυο Petri το οποίο είναι και απλό. (b) σύνηθες δίκτυο Petri το οποίο δεν είναι απλό.

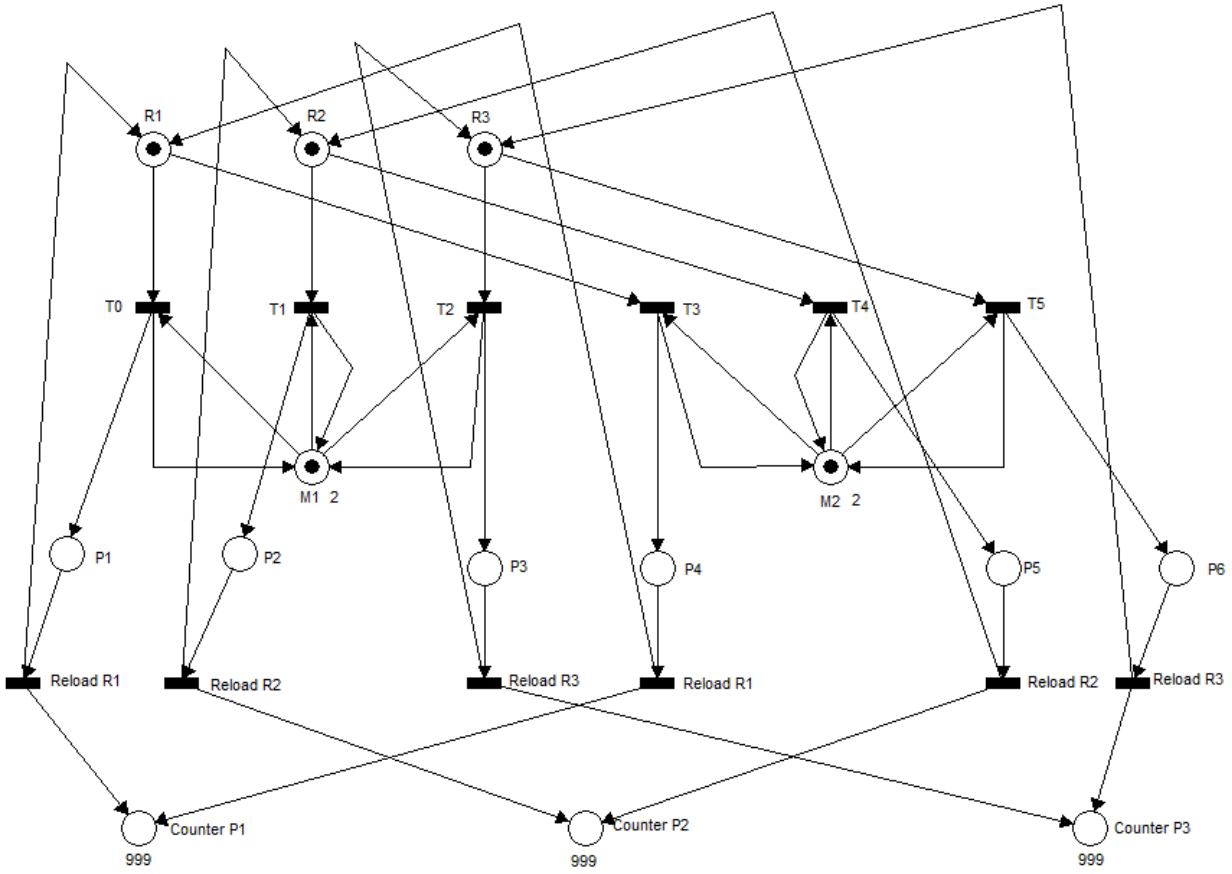
3.6 Παραλλαγές των δικτύων Petri

Στην ενότητα αυτή θα παρουσιαστούν ορισμένες παραλλαγές των δικτύων Petri, οι οποίες κυρίως χρησιμοποιούνται για την μοντελοποίηση περισσότερο πολύπλοκων δυναμικών συστημάτων. Η κυριότερη παραλλαγή των κλασικών δικτύων Petrieίναι τα *δίκτυα Petri υψηλού επιπέδου (HighLevelPetriNets - HLPN)* [4]. Σε αντίθεση με τα κλασικά δίκτυα Petri, όπου τα τεκμήρια είναι όλα ίδια μεταξύ τους και δεν υπάρχει κάποιος τρόπος διαχωρισμού, στα HLPNο διαχωρισμός αυτός είναι εφικτός. Τα HLPNμε την σειρά τους έχουν αρκετές υποκατηγορίες. Κριτήριο διαχωρισμού των υποκατηγοριών αυτών είναι το κριτήριο διαχωρισμού των τεκμηρίων. Οι υποκατηγορίες αυτές παρουσιάζονται αναλυτικά παρακάτω. Η κατανόηση και η μελέτη των παραλλαγών αυτών των δικτύων Petrieίναι σημαντική, καθώς η μοντελοποίηση ενός δυναμικού συστήματος με την χρήση δικτύων Petriαποτελεί στην ουσία τον ορισμό μίας νέας παραλλαγής.

3.6.1 Χρωματισμένα ΔίκτυαPetri (Colored Petri Nets)

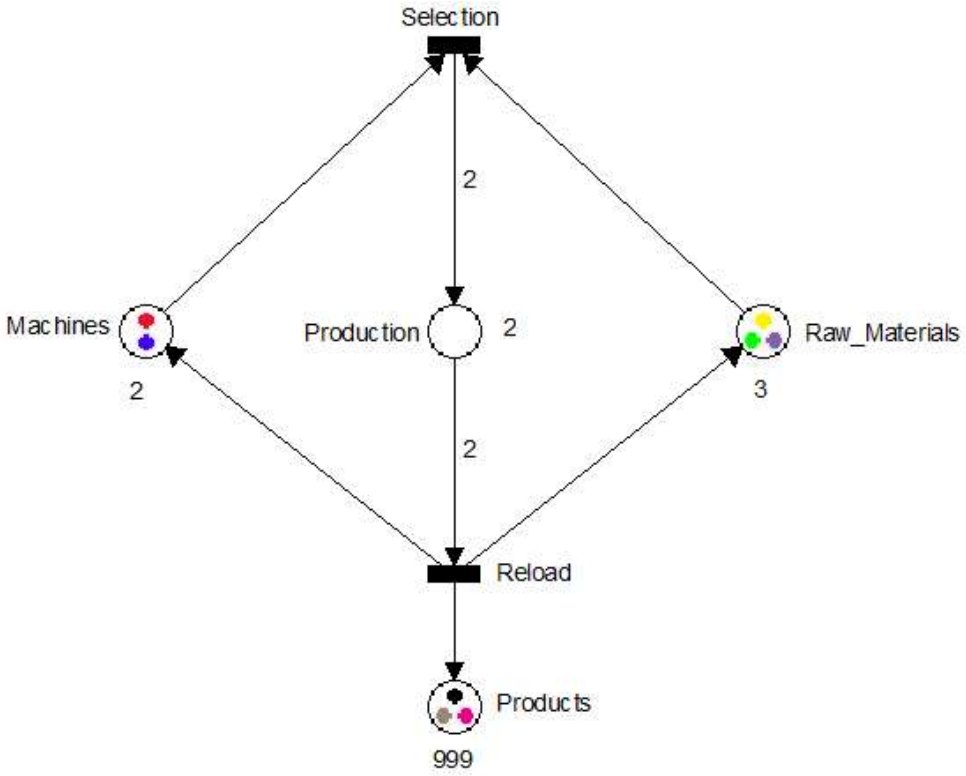
Τα *Χρωματισμένα Δίκτυα Petri (ColoredPetriNets – CPN)*προτάθηκαν αρχικά από τον KurtJensen το 1981 [9]. Σε ένα CPNτο κάθε τεκμήριο χαρακτηρίζεται από ένα χρώμα, το οποίο στην ουσία αποτελεί και την ταυτότητά του. Επιπρόσθετα, τόσο οι θέσεις όσο και οι μεταβάσεις χαρακτηρίζονται από ένα σύνολο χρωμάτων (όχι απαραίτητα από ένα). Οι κανόνες πυροδότησης των μεταβάσεων δεν έχουν πλέον ως κριτήριο μόνο τον αριθμό των τεκμηρίων και των θέσεων αλλά και το χρώμα των τεκμηρίων. Έτσι για παράδειγμα μία μετάβαση μπορεί να πυροδοτηθεί μόνο αν λάβει δύο κόκκινα τεκμήρια και να δώσει ως έξοδο ένα μπλε κι ένα κίτρινο τεκμήριο.

Για να κάνουμε περισσότερο κατανοητή την λειτουργία των CPN ας δούμε ένα παράδειγμα χρήσης τους. Ας υποθέσουμε ότι σε ένα σύστημα παραγωγής ενός εργοστασίου υπάρχουν δύο μηχανές M_1 και M_2, οι οποίες είναι ταυτόσημες και δουλεύουν με ακριβώς τον ίδιο τρόπο. Η κάθε μηχανή λαμβάνει ως είσοδο ένα πρωτογενές υλικό και παράγει το αντίστοιχο προϊόν. Υπάρχουν 3 διαφορετικά πρωτογενή υλικά R_1, R_2 και R_3 και συνεπώς μπορούν να παραχθούν 3 διαφορετικά προϊόντα , P_2 και P_3. Κάθε φορά που μία μηχανή παράγει ένα προϊόν, γεμίζει και πάλι το πρωτογενές υλικό το οποίο χρησιμοποίησε. Κάθε χρονική στιγμή μας ενδιαφέρει να γνωρίζουμε τον αριθμό των προϊόντων που παρήχθησαν από το κάθε είδος. Ένα κλασικό δίκτυο Petrito οποίο μοντελοποιεί την παραπάνω συμπεριφορά παρουσιάζεται στο Σχήμα 3.16.



Σχήμα 3.16: ένα κλασικό δίκτυο Petri το οποίο μοντελοποιεί την παραπάνω περιγραφή. Η μέγιστη χωρητικότητα 999 των θέσεων CounterP1, CounterP2 και CounterP3 αντιστοιχεί σε πρακτικά άπειρη χωρητικότητα.

Από το παραπάνω σχήμα γίνεται σαφές ότι η πολυπλοκότητα του κλασικού δικτύου Petri, για το εν λόγω πρόβλημα είναι μεγάλη. Το γεγονός αυτό οφείλεται στο ότι παρά το γεγονός ότι η διαδικασία και για τα 3 προϊόντα είναι ακριβώς η ίδια, αυτή θα πρέπει να σχεδιαστεί με πανομοιότυπο τρόπο και για τις δύο μηχανές M_1 και M_2. Τα CPNείναι χρήσιμα για την επίλυση τέτοιου είδους προβλημάτων. Πιο συγκεκριμένα, ας δούμε το CPN το οποίο παρουσιάζεται στο Σχήμα 3.17.



Σχήμα 3.17: το CPN το οποίο μοντελοποιεί την λειτουργία του συστήματος παραγωγής.

Στο CPN που παρουσιάζεται στο Σχήμα 3.17 τα τεκμήρια έχουν πλέον χρώματα. Πιο συγκεκριμένα:

- Το τεκμήριο που αντιστοιχεί στην μηχανή M_1 αναπαρίσταται με *μπλε* χρώμα.
- Το τεκμήριο που αντιστοιχεί στην μηχανή M_2 αναπαρίσταται με *κόκκινο* χρώμα.
- Το τεκμήριο που αντιστοιχεί στο πρωτογενές υλικό R_1 αναπαρίσταται με *κίτρινο* χρώμα.
- Το τεκμήριο που αντιστοιχεί στο πρωτογενές υλικό R_2 αναπαρίσταται με *πράσινο* χρώμα.
- Το τεκμήριο που αντιστοιχεί στο πρωτογενές υλικό R_3 αναπαρίσταται με *μοβ* χρώμα.
- Τα τεκμήρια που αντιστοιχούν στα προϊόντα P_1 αναπαρίστανται με *μαύρο* χρώμα.
- Τα τεκμήρια που αντιστοιχούν στα προϊόντα P_2 αναπαρίστανται με *γκρι* χρώμα.

- Τα τεκμήρια που αντιστοιχούν στα προϊόντα P_3 αναπαρίστανται με *ροζ* χρώμα.

Αυτό το οποίο κάνει το CPN να λειτουργεί διαφορετικά, είναι οι κανόνες πυροδότησης των μεταβάσεών του. Όσον αφορά την μετάβαση Selection, αυτή δεν διαφέρει σε τίποτα από μία μετάβαση ενός κλασικού δικτύου Petri. Αντίθετα, η μετάβαση *Reload*έχει περισσότερο περίπλοκους κανόνες πυροδότησης, οι οποίοι δίνονται λεκτικά (και όχι με αυστηρά μαθηματικό τρόπο) παρακάτω. Οι κανόνες αυτοί στην ουσία ελέγχουν το χρώμα των 2 τεκμηρίων που βρίσκονται στην θέση *Production*και ανάλογα με αυτό δίνουν τα σωστά τεκμήρια στις θέσεις *Products*, *Machines* και *Raw_Materials*. Αν συμβολίσουμε με <*Machine,Raw_Material*> την δυάδα τεκμηρίων που βρίσκονται στην θέση *Production*, τότε:

1. Αν <*Machine,Raw_Material*> = <*μπλε,κίτρινο*> τότε i) βάλε στην *Products* ένα *μαύρο* τεκμήριο, ii) βάλε στην *Machines* ένα *μπλε* τεκμήριο και iii) βάλε στην *Raw_Materials* ένα *κίτρινο* τεκμήριο.
2. Αν <*Machine,Raw_Material*> = <*μπλε,πράσινο*> τότε i) βάλε στην *Products* ένα *γκρι* τεκμήριο, ii) βάλε στην *Machines*ένα *μπλε* τεκμήριο και iii) βάλε στην *Raw_Materials* ένα *πράσινο* τεκμήριο.
3. Αν <*Machine,Raw_Material*> = <*μπλε,μωβ*> τότε i) βάλε στην *Products* ένα *ροζ* τεκμήριο, ii) βάλε στην *Machines*ένα *μπλε* τεκμήριο και iii) βάλε στην *Raw_Materials* ένα *μωβ* τεκμήριο.
4. Αν <*Machine,Raw_Material*> = <*κόκκινο,κίτρινο*> τότε i) βάλε στην *Products* ένα *μαύρο* τεκμήριο, ii) βάλε στην *Machines*ένα *κόκκινο* τεκμήριο και iii) βάλε στην *Raw_Materials* ένα *κίτρινο* τεκμήριο.
5. Αν <*Machine,Raw_Material*> = <*κόκκινο,πράσινο*> τότε i) βάλε στην *Products* ένα *γκρι* τεκμήριο, ii) βάλε στην *Machines*ένα *κόκκινο* τεκμήριο και iii) βάλε στην *Raw_Materials* ένα *πράσινο* τεκμήριο.
6. Αν <*Machine,Raw_Material*> = <*κόκκινο,μωβ*> τότε i) βάλε στην*Products*ένα *ροζ* τεκμήριο, ii) βάλε στην *Machines*ένα *κόκκινο* τεκμήριο και iii) βάλε στην *Raw_Materials* ένα *μωβ* τεκμήριο.

Είναι εμφανές ότι η μοντελοποίηση του συστήματος με το CPNμειώνει σημαντικά τον όγκο και την πολυπλοκότητα του δικτύου.

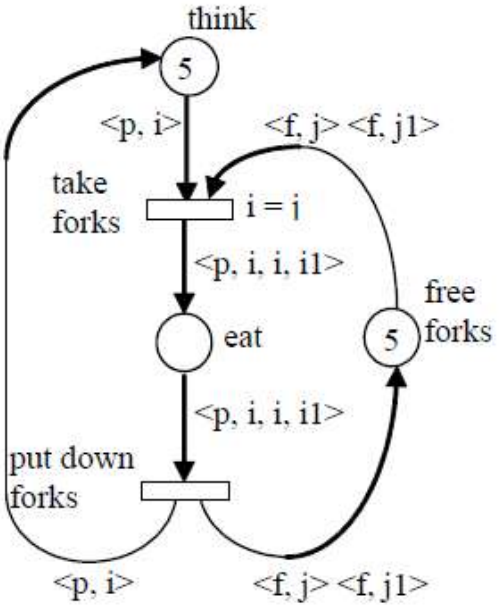
3.6.2 ΔίκτυαPetri Δηλώσεων/Μεταβάσεων (Predicate/Transition Petri Nets)

Τα Δίκτυα Petri *Δηλώσεων/Μεταβάσεων (Predicate/TransitionPetriNets – Pr/T)* προτάθηκαν αρχικά από τους Genrich και Lautenbachto 1981 [10] και αποτελούν για την ακρίβεια την πρώτη κατηγορία HLPN. Τα Pr/T έχουν τρεις κύριες διαφορές σε σχέση με τα κλασικά δίκτυα Petri:

1. Το κάθε τεκμήριο μπορεί να έχει ένα σύνολο από χαρακτηριστικά, ένα τουλάχιστον από τα οποία το διαχωρίζει από τα υπόλοιπα τεκμήρια.
2. Για τις μεταβάσεις ισχύουν οι κανόνες πυροδότησης οι οποίοι παρουσιάστηκαν στην Ενότητα 3.3 αλλά μπορούν να φέρουν επιπρόσθετα και *δηλώσεις (predicates)*. Οι δηλώσεις στις μεταβάσεις είναι *λογικές συνθήκες*, οι οποίες αναφέρονται στα χαρακτηριστικά των τεκμηρίων που εισέρχονται στην μετάβαση. Μία μετάβαση είναι πλέον ικανοποιήσιμη αν ισχύουν τα κριτήρια της Ενότητας 3.4 και επιπρόσθετα αν οι δηλώσεις της είναι αληθείς.
3. Οι ακμές αντί για βάρη φέρουν και αυτές δηλώσεις. Οι δηλώσεις αυτές περιγράφουν τα τεκμήρια τα οποία μεταφέρονται μέσω των ακμών. Οι δηλώσεις πάνω στις ακμές είναι αναγκαίες αφού τα τεκμήρια είναι διαφορετικά μεταξύ τους.

Για να γίνει περισσότερο κατανοητή η λειτουργία των Pr/T ας δούμε ένα παράδειγμα. Ας θεωρήσουμε το γνωστό πρόβλημα των 5 δειπνούντων φιλοσόφων (diningphilosophers). Στο πρόβλημα αυτό υπάρχουν σε ένα κυκλικό τραπέζι 5 φιλόσοφοι και στο κέντρο του τραπεζιού υπάρχει το φαγητό. Ανάμεσα σε δύο φιλοσόφους υπάρχει ένα πιρούνι. Οι φιλόσοφοι σκέφτονται και συζητούν και με τυχαίο τρόπο τρώνε το φαγητό. Για να φάει ο κάθε φιλόσοφος το φαγητό χρειάζεται τα δύο πιρούνια τα οποία βρίσκονται αριστερά και δεξιά του. Έτσι για παράδειγμα, αν

συμβολίσουμε τους φιλόσοφους ως $P = [p_1, p_2, p_3, p_4, p_5]$ και τα πηρούνια ως $F = [f_1, f_2, f_3, f_4, f_5]$, ο φιλόσοφος p_3 χρειάζεται τα πηρούνια f_3 και f_4, ο φιλόσοφος p_4 χρειάζεται τα πηρούνια f_3 και f_4, ο φιλόσοφος p_5 χρειάζεται τα πηρούνια f_5 και f_1 κοκ. Όπως είναι προφανές, ο κάθε φιλόσοφος για να μπορέσει να δειπνήσει απαιτεί την ύπαρξη των αντίστοιχων πηρουνιών. Αν αυτά δεν είναι διαθέσιμα, τότε θα πρέπει να περιμένει μέχρι αυτά να γίνουν. Το πρόβλημα των δειπνούντων φιλοσόφων αποτελεί ένα από τα πιο γνωστά προβλήματα συγχρονισμού διεργασιών και χρησιμοποιείται ευρέως ως παράδειγμα για την επίλυση με την χρήση σημαφόρων στα Λειτουργικά Συστήματα. Η μοντελοποίηση της παραπάνω περιγραφής με την χρήση ενός κλασικού δικτύου Petri αποτελεί πρόκληση, καθώς το μέγεθος και η πολυπλοκότητα του δικτύου θα είναι μεγάλα. Η παραπάνω περιγραφή μπορεί να αναπαρασταθεί με το Pr/Ττο οποίο παρουσιάζεται στο Σχήμα 3.18 που ακολουθεί.



Σχήμα 3.18 [4]: μοντελοποίηση των δειπνούντων φιλοσόφων με την χρήση ενός Pr/T.

Στο παραπάνω σχήμα η θέση *think* περιέχει τους 5 δειπνούντες φιλοσόφους. Θεωρούμε ότι το κάθε τεκμήριο στην θέση αυτή, το οποίο αναπαριστά έναν φιλόσοφο, αναπαρίσταται ως <*p, i*>, όπου *i* είναι ο αριθμός του κάθε φιλοσόφου. Αντίστοιχη αναπαράσταση <*f, j*> έχουν και τα πιρούνια, τα οποία βρίσκονται στην θέση *freeforks*. Η μετάβαση *takeforks* λαμβάνει κάθε χρονική στιγμή έναν φιλόσοφο και δύο πιρούνια. Πιο συγκεκριμένα, η μετάβαση έχει δύο εισερχόμενες ακμές από τις δύο θέσεις *think* και *freeforks*. Η δήλωση στην πρώτη ακμή είναι <*p, i*> ενώ στην δεύτερη <*f, j*> <*f, j1*> (με *j1* αναπαριστούμε την πράξη *j+1*). Για να μπορέσει να πυροδοτηθεί η μετάβαση, θα πρέπει να ικανοποιείται η δήλωση *i = j*. Έτσι για παράδειγμα αν *i = 3*, τότε το *j* θα πρέπει να έχει την τιμή 3, έτσι ώστε ο φιλόσοφος <*p, 3*> να πάρει τα πιρούνια <*f, 3*> και <*f, 4*>. Μετά την πυροδότησή της, η μετάβαση *takeforks* δίνει ένα τεκμήριο στην θέση eat. Το τεκμήριο αυτό έχει την δομή <*p, i, i, i1*> και περιέχει τόσο τον κωδικό του φιλοσόφου όσο και τους κωδικούς των πιρουνιών που χρησιμοποιεί για να δειπνήσει. Τέλος, η λειτουργία της μετάβασης *putdownforks* δουλεύει με καθαρά αντίστροφη λογική, έτσι ώστε να επιστρέφει τον φιλόσοφο και τα πιρούνια στις θέσεις *think* και *freeforks* αντίστοιχα.

3.6.3 Χρονικά Δίκτυα Petri (Timed Petri Nets)

Όπως ήδη είδαμε, η ανάγκη διαχωρισμού και ταυτοποίησης των τεκμηρίων ενός δικτύου Petri απαιτεί την εισαγωγή μεταβλητών οι οποίες συνοδεύουν τα κριτήρια. Για παράδειγμα στα CPN οι μεταβλητές αυτές είναι χρώματα. Αν οι μεταβλητές αναπαριστούν χρόνο (είτε χρονική στιγμή

είτε χρονική διάρκεια) τότε το δίκτυο ονομάζεται *Χρονικό Δίκτυο Petri (TimedPetriNet – TPN)* [11]. Ο ορισμός ενός TPN σε σχέση με τα κλασικά δίκτυα Petriεστιάζει σε 3 διαφορετικές κατευθύνσεις:

- Την αρχιτεκτονική δομή του δικτύου.
- Στο μαρκάρισμα των στοιχείων του δικτύου.
- Στους κανόνες πυροδότησης των μεταβάσεων του δικτύου.

Όσον αφορά την αρχιτεκτονική δομή του δικτύου, αυτή τοπολογικά μοιάζει με την δομή ενός κλασικού δικτύου Petri. Το μαρκάρισμα των στοιχείων στην ουσία αντιστοιχεί στην απόδοση χρονικών τιμών σε ένα ή περισσότερα από τα παρακάτω:

- Μεταβάσεις
- Θέσεις
- Ακμές

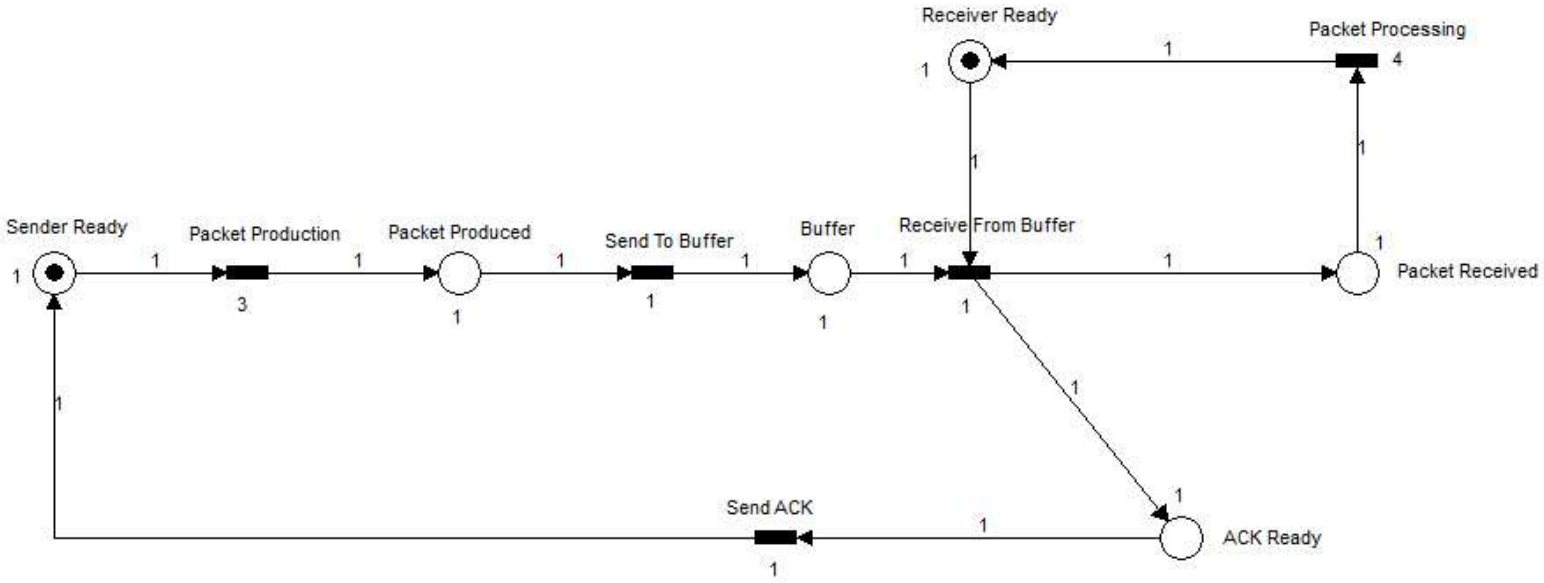
Αυτό το οποίο όμως διαφοροποιεί περισσότερα τα TPN από τα κλασικά δίκτυα Petrieίναι οι κανόνες πυροδότησης, αφού πλέον εξαρτώνται από τις χρονικές τιμές οι οποίες χαρακτηρίζουν τα στοιχεία του δικτύου.

Όπως είναι φυσικό, τα παραπάνω οδηγούν σε μία πληθώρα διαφορετικών τύπων TPN. Από τους τύπους αυτούς οι σημαντικότεροι είναι τα *Χρονικά Δίκτυα Petri Ντετερμινιστικού Χρόνου* [12] και τα *Χρονικά Δίκτυα Petri Στοχαστικού Χρόνου* [13] [14]. Και οι δύο αυτές κατηγορίες TPN παρουσιάζονται στην συνέχεια.

3.6.4 Χρονικά Δίκτυα Petri Ντετερμινιστικού Χρόνου (DeterministicTimed Petri nets)

Τα *Χρονικά Δίκτυα Petri Ντετερμινιστικού Χρόνου (Deterministic timed Petri nets - DTPN)* προτάθηκαν αρχικά από τον Ramchandani το 1974 [12]. Στα DTPN κάθε μετάβαση χαρακτηρίζεται από μία χρονική ετικέτα. Η ετικέτα αυτή μοντελοποιεί το γεγονός του ότι οι μεταβάσεις αντιστοιχούν σε ενέργειες και οι ενέργειες απαιτούν κάποιον χρόνο για να εκτελεστούν.

Περισσότερο τυπικά, εκτός από την διατεταγμένη εξάδα στοιχείων $N = (P, T, I, O, M_{max}, M(t))$ που παρουσιάστηκε στην Ενότητα 3.3, ένα DTPN χαρακτηρίζεται επιπλέον και από μία συνάρτηση $\tau\colon T \to \mathbf{R}^+$ η οποία αντιστοιχίζει κάθε μετάβαση σε μία θετική χρονική τιμή. Η χρονική τιμή αυτή ονομάζεται *καθυστέρηση (delay)*. Πλέον, μία μετάβαση $t_i \in T$ είναι *ικανοποιήσιμη* την χρονική στιγμή x αν και μόνο αν τα απαιτούμενα τεκμήρια από τις θέσεις που στέλνουν εισερχόμενες ακμές στην μετάβαση βρίσκονται μέσα στις θέσεις αυτές για χρόνο τ_i, όπου τ_i η τιμή της συνάρτησης τ για την μετάβαση t_i. Στην ουσία αυτό το οποίο συμβαίνει είναι ότι όταν τα εμφανιστούν τα απαιτούμενα τεκμήρια στις αντίστοιχες θέσεις, τότε η μετάβαση t_i δεν πυροδοτείται ακαριαία αλλά υπάρχει μία χρονική καθυστέρηση ίση με τ_i. Έτσι λοιπόν αν η μετάβαση πυροδοτηθεί την χρονική στιγμή x, αυτό πρακτικά σημαίνει ότι τα απαιτούμενα κριτήρια βρίσκονταν στις αντίστοιχες θέσεις από την χρονική στιγμή $x - \tau_i$. Ένα παράδειγμα DTPN, το οποίο περιγράφει ένα πρωτόκολλο επικοινωνίας, παρουσιάζεται στο Σχήμα 3.19.



Σχήμα 3.19: το πρωτόκολλο επικοινωνίας το οποίο περιγράφηκε παραπάνω.

Στο Σχήμα 3.19 μοντελοποιείται ένα απλό πρωτόκολλο ανταλλαγής πακέτων μεταξύ ενός αποστολέα κι ενός παραλήπτη. Ο αποστολέας παράγει και στέλνει ένα πακέτο, το οποίο ο παραλήπτης λαμβάνει και επεξεργάζεται. Το μέσο επικοινωνίας μεταξύ αποστολέα και παραλήπτη είναι ένας ενδιάμεσος χώρος (buffer). Όταν ο παραλήπτης λάβει το πακέτο στέλνει μία επιβεβαίωση (ACK) στον αποστολέα, έτσι ώστε αυτός να αρχίσει την παραγωγή του επόμενου πακέτου. Η διαδικασία αυτή συνεχίζεται επ' άπειρον και δεν τερματίζει ποτέ. Υπάρχουν οι ακόλουθοι χρονικοί περιορισμοί:

- Η παραγωγή του πακέτου από τον αποστολέα απαιτεί 3 χρονικές μονάδες.
- Η τοποθέτηση του πακέτου από τον αποστολέα στον ενδιάμεσο χώρο απαιτεί 1 χρονική μονάδα.
- Η παραλαβή του πακέτου από τον παραλήπτη από τον ενδιάμεσο χώρο απαιτεί 1 χρονική μονάδα.
- Η αποστολή της επιβεβαίωσης (ACK) από τον παραλήπτη στον αποστολέα απαιτεί 1 χρονική μονάδα.
- Η επεξεργασία του πακέτου από τον παραλήπτη απαιτεί 4 χρονικές μονάδες.

Οι χρονικές απαιτήσεις της προσομοίωσης αναπαρίστανται με την χρήση των χρονικών μονάδων πάνω στις μεταβάσεις του DTPN. Αν το πρόβλημα έπρεπε να μοντελοποιηθεί με την χρήση ενός κλασικού δικτύου Petri, τότε ο αριθμός των απαιτούμενων στοιχείων θα ήταν αρκετά μεγαλύτερος. Αυτό οφείλεται στο γεγονός ότι σε ένα κλασικό δίκτυο Petri η κάθε μετάβαση εκτελείται πάντα σε σταθερό χρόνο τ οπότε ένα γεγονός το οποίο απαιτεί χρόνο ίσο με $k \cdot t$ απαιτεί την σειριακή τοποθέτηση k μεταβάσεων. Έτσι λοιπόν στο συγκεκριμένο παράδειγμα θα ήταν απαραίτητη η χρήση 10 συνολικά μεταβάσεων έναντι 5 στην παρούσα φάση.

3.6.5 Χρονικά Δίκτυα Petri Στοχαστικού Χρόνου (StochasticTimed Petrinets)

Τα *Χρονικά Δίκτυα Petri Στοχαστικού Χρόνου (StochasticTimedPetrinets - STPN)* [13] [14] είναι μία κατηγορίαTPN τα οποία είναι σε θέση να μοντελοποιούν στοχαστικές διαδικασίες. Με τον όρο στοχαστική διαδικασία αναφερόμαστε σε μία συλλογή τιμών τυχαίων μεταβλητών σε ένα χρονικό διάστημα. Οι τυχαίες μεταβλητές παίρνουν τιμές συνήθως με την χρήση μιας πιθανοτικής κατανομής. Σε αντίθεση με τα DTPN, στα STPN η καθυστέρηση της κάθε μετάβασης δεν είναι συγκεκριμένη και σταθερή αλλά παράγεται με βάση μία πιθανοτική κατανομή, η οποία συνήθως είναι η *αρνητική εκθετική κατανομή*, της οποίας ο τύπος είναι ο εξής:

$$f(\lambda; x) = \begin{cases} \lambda e^{-\lambda x}, x \geq 0 \\ 0, \quad x < 0 \end{cases}$$

Η παράμετρος λ ονομάζεται *ρυθμός* της κατανομής. Φυσικά η εν λόγω πιθανοτική κατανομή δεν είναι η μόνη η οποία μπορεί να χρησιμοποιηθεί αλλά είναι η πλέον συνηθέστερη. Τα STPN είναι ιδιαίτερα χρήσιμα στην μοντελοποίηση συστημάτων του πραγματικού κόσμου, αφού συνήθως οι ενέργειες που λαμβάνουν χώρα σε αυτά δεν διαρκούν απαραίτητα ένα συγκεκριμένο χρονικό διάστημα. Σε όλα τα δίκτυα Petrita οποία είδαμε ως τώρα, η πυροδότηση μιας μετάβασης, η οποία αντιστοιχεί σε μία ενέργεια, διαρκεί πάντοτε συγκεκριμένο χρονικό διάστημα, κάτι το οποίο είναι ουτοπικό σε ένα πραγματικό δυναμικό σύστημα. Έτσι λοιπόν είναι περισσότερο ρεαλιστικό η πυροδότηση μιας μετάβασης να διαρκεί ένα τυχαίο (αν και φραγμένο) χρονικό διάστημα, γεγονός το οποίο γίνεται πλέον εφικτό με την χρήση ενός STPN.

Αν η κατανομή η οποία χρησιμοποιείται είναι η αρνητική εκθετική, τότε ένα STPN εκτός από την διατεταγμένη εξάδα στοιχείων $N = (P, T, I, O, M_{max}, M(t))$ που παρουσιάστηκε στην Ενότητα 3.3 περιλαμβάνει και μία συνάρτηση $\Lambda: T \to \mathbf{R}$, η οποία παράγει τον ρυθμό για κάθε μετάβαση $t_i \in T$. Πιο συγκεκριμένα, θεωρούμε ότι κάθε μετάβαση χαρακτηρίζεται από έναν διαφορετικό ρυθμό λ_i . Οι παράμετροι της εκθετικής κατανομής για κάθε μετάβαση t_i είναι η μετάβαση (t_i) και ο ρυθμός της (λ_i). Η πλήρης περιγραφή της λειτουργίας των STPN απαιτεί την ανάλυση στοχαστικών μοντέλων καθώς και κεφαλαίων της θεωρίας πιθανοτήτων, τα οποία εκτείνονται εκτός των ορίων της παρούσας μελέτης.

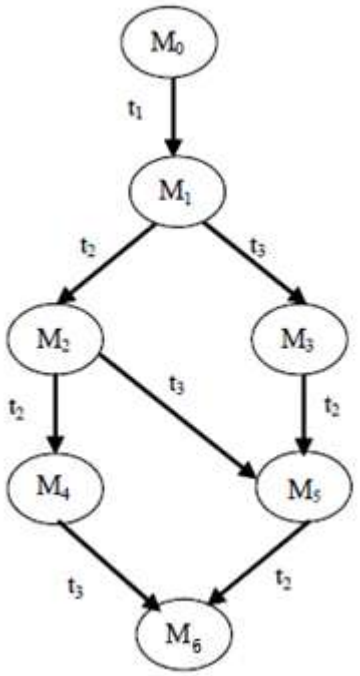
3.7 Ανάλυση Προσβασιμότητας

Δεδομένου ενός δικτύου Petri μπορούμε να κατασκευάσουμε τον γράφο προσβασιμότητας (reachabilitygraph) πάνω στον οποίον αναπαριστούμε όλες τις δυνατές καταστάσεις στις οποίες μπορεί να περιέλθει το δίκτυο Petri. Πιο συγκεκριμένα, όπως είδαμε κατά τον τυπικό ορισμό ενός δικτύου Petri, η συνάρτηση $M(t)$ μας επιστρέφει την τρέχουσα χωρητικότητα των θέσεων του δικτύου την χρονική στιγμή t . Ξεκινώντας από την αρχική κατάσταση $M(0)$ μπορούμε να δείξουμε με την βοήθεια ενός κατευθυνόμενου γραφήματος τις καταστάσεις στις οποίες μπορεί να περιέλθει το δίκτυο Petri την χρονική στιγμή $t + 1$. Ο κάθε κόμβος του γράφου συμβολίζεται ως M_i με $i \geq 0$ ενώ η κάθε ακμή του γράφου φέρει πάνω της τις μεταβάσεις εκείνες οι οποίες πυροδοτήθηκαν για να είναι δυνατή η σύνδεση της μίας κατάστασης με την επόμενή της. Ο αλγόριθμος κατασκευής του γράφου προσβασιμότητας παρουσιάζεται παρακάτω:

1. Θέσε ως αρχικό κόμβο M_0 την κατάσταση $M(0)$. Κάθε κατάσταση M_i διαθέτει μία λογική τιμή $M_i.expanded$ η οποία δείχνει αν έχει επεκταθεί ή όχι. Ισχύει ότι $M_0.expanded := false$.
2. Αρχικοποίησε έναν μετρητή $i := 0$.
3. Επέλεξε τυχαία μία κατάσταση M_k την οποία δεν έχουμε ακόμα επεκτείνει (δηλαδή $M_k.expanded = false$).
4. Για κάθε κατάσταση η οποία είναι δυνατόν να προκύψει από την M_k :
   a. Αν η κατάσταση δεν υπάρχει ήδη ως κόμβος του γράφου, τότε θέσε $i := i + 1$ και ονόμασε τη νέα κατάσταση M_i .

b. Σύνδεσε την προηγούμενη κατάσταση M_k με την επόμενη κατάσταση (νέα ή υπάρχουσα) με μία κατευθυνόμενη ακμή και βάλε ως σχόλιο πάνω στην ακμή τις μεταβάσεις οι οποίες πυροδοτήθηκαν.

5. Θέσε $M_k.expanded := true$.
6. Αν δεν υπάρχει καμία κατάσταση M_i για την οποία ισχύει ότι $M_i.expanded = false$ τερμάτισε τον αλγόριθμο. Σε διαφορετική περίπτωση πήγαινε πίσω στο βήμα 3.



Σχήμα 3.20: ο γράφος προσβασιμότητας για το δίκτυο Petri που παρουσιάζεται στο Σχήμα 3.2.

Ο γράφος προσβασιμότητας ο οποίος παρουσιάζεται παραπάνω αφορά το δίκτυο Petri του Σχήματος 3.2 (4 θέσεων και 3 μεταβάσεων).Οι δυνατές καταστάσεις του εν λόγω δικτύου είναι 6. Πιο συγκεκριμένα:

$M_0 = [2 \quad 0 \quad 0 \quad 0]^T$
$M_1 = [0 \quad 2 \quad 1 \quad 0]^T$
$M_2 = [0 \quad 1 \quad 1 \quad 1]^T$
$M_3 = [0 \quad 2 \quad 0 \quad 1]^T$
$M_4 = [0 \quad 0 \quad 1 \quad 2]^T$
$M_5 = [0 \quad 1 \quad 0 \quad 2]^T$
$M_6 = [0 \quad 0 \quad 0 \quad 3]^T$

Κεφάλαιο 4 – Μοντελοποίηση Ασαφών Συστημάτων με την χρήση Δικτύων Petri

4.1 Εισαγωγή

Όπως είδαμε στο Κεφάλαιο 2, τα κλασικά δίκτυα Petrieίναι σε θέση να μοντελοποιήσουν με αρκετά ικανοποιητικό τρόπο ένα οποιοδήποτε δυναμικό σύστημα. Παρά το γεγονός αυτό, η πολυπλοκότητα ορισμένων δυναμικών συστημάτων ή οι ιδιάζουσες απαιτήσεις τους κατέστησαν αναγκαίο τον ορισμό περισσότερων ειδών δικτύων Petri, τα σημαντικότερα από τα οποία παρουσιάστηκαν στο προηγούμενο κεφάλαιο. Στο εν λόγω κεφάλαιο θα παρουσιάσουμε τα *Ασαφή Δίκτυα Petri (FuzzyPetriNets – FPN)* με τα οποία είναι δυνατή η μοντελοποίηση ασαφών συστημάτων. Πρακτικά, αυτό το οποίο θα επιτύχουμε είναι η αναπαράσταση και η προσομοίωση μίας ασαφούς Βάσης Κανόνων με ασαφείς κανόνες με την χρήση ενός FPN. Στο σημείο αυτό θα πρέπει να σημειωθεί ότι υπάρχουν πολλοί ορισμοί για την δομή και την λειτουργία των FPN [16] [17] [18] [19] [20]. Στην εν λόγω μελέτη παρατίθεται η πλέον γενικότερη μορφή των ορισμών αυτών, τόσο όσον αφορά την δομή των FPN όσο και των κανόνων πυροδότησής τους.

4.2 Ορισμός των Ασαφών Δικτύων Petri

Ένα *Ασαφές Δίκτυο Petri (FuzzyPetriNet – FPN)* μπορεί να χρησιμοποιηθεί για να αναπαραστήσει ένα ασαφές σύστημα το οποίο βασίζεται σε κανόνες. Ένα FPN [16] [17] είναι

ένα κατευθυνόμενο γράφημα το οποίο περιέχει δύο ειδών κόμβους: τις θέσεις και τις μεταβάσεις. Κάθε θέση αναπαριστά μία υπόθεση ή ένα συμπέρασμα ενός ασαφούς κανόνα. Οι θέσεις περιέχουν τεκμήρια ενώ επιπρόσθετα χαρακτηρίζονται από έναν *βαθμό βεβαιότητας* (στο πραγματικό διάστημα $[0,1]$), ο οποίος αντιστοιχεί στην τιμή της συνάρτησης συμμετοχής για το αντίστοιχο γεγονός (υπόθεση ή συμπέρασμα). Κάθε μετάβαση αναπαριστά έναν κανόνα και επιπρόσθετα χαρακτηρίζεται από έναν βαθμό βεβαιότητας. Ο βαθμός βεβαιότητας μιας μετάβασης αναπαριστά την βεβαιότητα με την οποία εξάγονται τα συμπεράσματα του κανόνα, στην περίπτωση που ισχύουν όλες οι υποθέσεις του κανόνα και έχουν βεβαιότητα ίση με 1. Αν οι υποθέσεις του κανόνα έχουν βεβαιότητες μικρότερες του 1, τότε τα συμπεράσματα εξάγονται με βαθμό βεβαιότητας μικρότερο από αυτόν της μετάβασης. Ο βαθμός βεβαιότητας μιας μετάβασης ονομάζεται επίσης και *κατώφλι (threshold)*. Οι συνδέσεις μεταξύ θέσεων και μεταβάσεων (υποθέσεων/συμπερασμάτων και κανόνων) γίνονται με την χρήση ακμών, οι οποίες φέρουν θετικά ακέραια βάρη.

Με βάση τα παραπάνω, είναι πλέον σε θέση να δώσουμε και τυπικά τον ορισμό ενός FPN. Ο ορισμός ενός FPN αποτελεί ουσιαστικά μία επέκταση του ορισμού ενός κλασικού δικτύου Petri, ο οποίος παρουσιάστηκε στην ενότητα 3.3. Ένα FPN μπορεί να οριστεί τυπικά ως μία διατεταγμένη δεκάδα στοιχείων $N = (P, T, D, I, O, M_{max}, M(t), \mu, \alpha, \beta)$, όπου:

(1) $P = \{p_1, p_2, \dots, p_m\}$ είναι ένα σύνολο από θέσεις.

(2) $T = \{t_1, t_2, \dots, t_n\}$ είναι ένα σύνολο από μεταβάσεις όπου $P \cup T \neq \emptyset$ και $P \cap T = \emptyset$.

(3) $D = \{d_1, d_2, \dots, d_m\}$ είναι ένα σύνολο από γεγονότα (υποθέσεις ή συμπεράσματα κανόνων) όπου $P \cup T \cup D \neq \emptyset$ και $P \cap T \cap D = \emptyset$.

(4) $I: P \times T \rightarrow N$ είναι μία *συνάρτηση εισόδου (inputfunction)* η οποία ορίζει τις κατευθυνόμενες ακμές από τις θέσεις προς τις μεταβάσεις. Στην προκείμενη περίπτωση, το σύνολο N είναι ένα σύνολο μη αρνητικών ακεραίων. Η τιμή 0 της συνάρτησης ισοδυναμεί με την μη ύπαρξη κατευθυνόμενης ακμής μεταξύ της θέσης και της μετάβασης.

(5) $O: T \times P \rightarrow N$ είναι μία *συνάρτηση εξόδου (outputfunction)* η οποία ορίζει τις κατευθυνόμενες ακμές από τις μεταβάσεις προς τις θέσεις. Και στην περίπτωση αυτή, το σύνολο N είναι ένα σύνολο μη αρνητικών ακεραίων. Επίσης και εδώ η τιμή 0 της συνάρτησης ισοδυναμεί με την μη υπάρξη κατευθυνόμενης ακμής μεταξύ της μετάβασης και της θέσης.

(6) $M_{max}: P \rightarrow N$ είναι μία συνάρτηση η οποία δείχνει την μέγιστη χωρητικότητα της κάθε θέσης.

(7) $M(t): P \rightarrow N$ είναι μία συνάρτηση η οποία δείχνει την τρέχουσα χωρητικότητα της κάθε θέσης την χρονική στιγμή t .Για t = 0 αναφερόμαστε στην αρχική κατάσταση του δικτύου.

(8) $\mu: T \rightarrow [0,1]$ είναι μία συνάρτηση η οποία αντιστοιχίζει κάθε μετάβαση σε έναν βαθμό βεβαιότητας (κατώφλι).

(9) $\alpha: P \rightarrow [0,1]$ είναι μία συνάρτηση η οποία αντιστοιχίζει κάθε θέση σε έναν βαθμό βεβαιότητας (κατώφλι).

(10) $\beta: P \rightarrow D$ είναι μία συνάρτηση η οποία αντιστοιχίζει κάθε θέση σε ένα γεγονός.

Από τον παραπάνω ορισμό είναι προφανές ότι τα κλασικά δίκτυα Petriαποτελούν στην ουσία υποκατηγορία των ασαφών δικτύων Petri για $\mu: T \rightarrow \{1\}$ και $\alpha: P \rightarrow \{1\}$.

4.3 Κανόνες πυροδότησης στα Ασαφή Δίκτυα Petri

Οι κανόνες πυροδότησης στα FPN έχουν επεκταθεί έτσι ώστε να είναι σε θέση να υποστηρίζουν τις παραπάνω ιδιότητες που αυτά έχουν. Παρακάτω, παρουσιάζουμε τους ορισμούς οι οποίοι διαφέρουν σε σχέση με αυτούς που παρουσιάστηκαν στην Ενότητα 3.4. Οι έννοιες οι οποίες

δεν ορίζονται (π.χ. εισερχόμενες/εξερχόμενες ακμές) είναι πανομοιότυπες και δεν έχουν παραλλαχθεί.

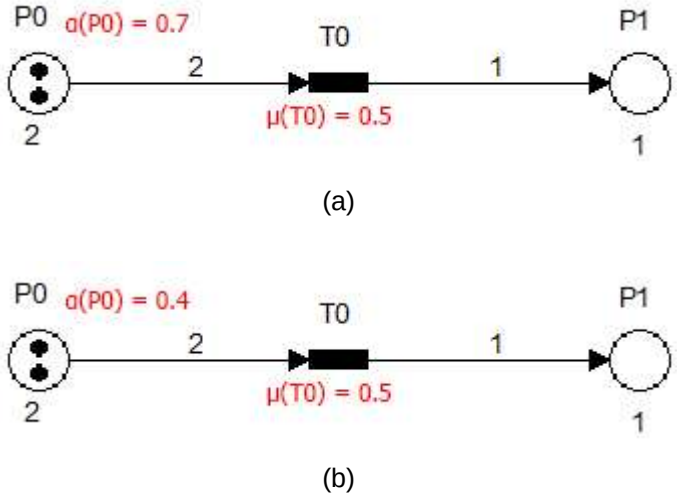
Ορισμός: μία μετάβαση t_i είναι *ικανοποιήσιμη* την χρονική στιγμή t όταν:

1. για κάθε εισερχόμενη ακμή της e_{ji} ισχύει ότι $M(t,p_j) \geq I(p_j,t_i)$ και
2. για κάθε εξερχόμενη ακμή της e_{ij} ισχύει ότι $M(t,p_j) + O(t_i,p_j) \leq M_{\max(p_j)}$ και
3. για κάθε θέση p_j που αντιστοιχεί σε μία εισερχόμενη ακμή e_{ji} ισχύει ότι $\alpha(p_j) \geq \mu(t_i)$.

Με λιγότερο τυπικό τρόπο, τα κριτήρια ικανοποιησιμότητας μίας ασαφούς μετάβασης είναι:

1. κάθε θέση η οποία στέλνει εισερχόμενη προς την μετάβαση ακμή έχει αριθμό κριτηρίων τουλάχιστον ίσο με το βάρος της ακμής και
2. κάθε θέση η οποία λαμβάνει εξερχόμενη από την μετάβαση ακμή έχει «χώρο» έτσι ώστε να μπορέσει να λάβει αριθμό τεκμηρίων τουλάχιστον ίσο με το βάρος της ακμής και
3. κάθε θέση η οποία στέλνει εισερχόμενη προς την μετάβαση ακμή θα πρέπει να έχει βαθμό βεβαιότητας τουλάχιστον ίσο με τον βαθμό βεβαιότητας της μετάβασης.

Από τον παραπάνω ορισμό παρατηρούμε ότι μία ασαφής μετάβαση έχει ένα παραπάνω κριτήριο για να χαρακτηριστεί ικανοποιήσιμη σε σχέση με μία κλασική μετάβαση. Ένα παράδειγμα ικανοποιήσιμων και μη ικανοποιήσιμων μεταβάσεων παρουσιάζεται στο Σχήμα 4.1.



Σχήμα 4.1: (a) η μετάβαση T0 είναι ικανοποιήσιμη. (b) η μετάβαση T0 δεν είναι ικανοποιήσιμη, αφού παρά το γεγονός του ότι η θέση P0 έχει τα απαιτούμενα τεκμήρια, ο βαθμός βεβαιότητάς της είναι μικρότερος του βαθμού βεβαιότητας της μετάβασης T0.

Όσον αφορά τους κανόνες πυροδότησης, αυτός που στην ουσία πλέον αλλάζει είναι ο Κανόνας 1 ενώ οι Κανόνες 2 και 3 ισχύουν με ακριβώς τον ίδιο τρόπο. Πιο συγκεκριμένα:

Κανόνας πυροδότησης 1: όταν μία ικανοποιήσιμη ασαφής μετάβαση t_i πυροδοτείται την χρονική στιγμή t τότε εκτελεί και τις τρείς παρακάτω ενέργειες:

1. Για κάθε θέση p_j για την οποία ισχύει ότι $I(p_j,t_i) > \mathbf{0}$, $M(t+1,p_j) = M(t,p_j) - I(p_j,t_i)$.
2. Για κάθε θέση p_j για την οποία ισχύει ότι $O(t_i,p_j) > \mathbf{0}$, $M(t+1,p_j) = M(t,p_j) + O(t_i,p_j)$.

3. Για κάθε θέση p_j για την οποία ισχύει ότι $O(t_i, p_j) > \mathbf{0}$, $\alpha(t+1, p_j) = \max\{\mu(t_i) \cdot min(\{\alpha(p_k) | I(p_k, t_i) > 0\}), \alpha(t)\}$.

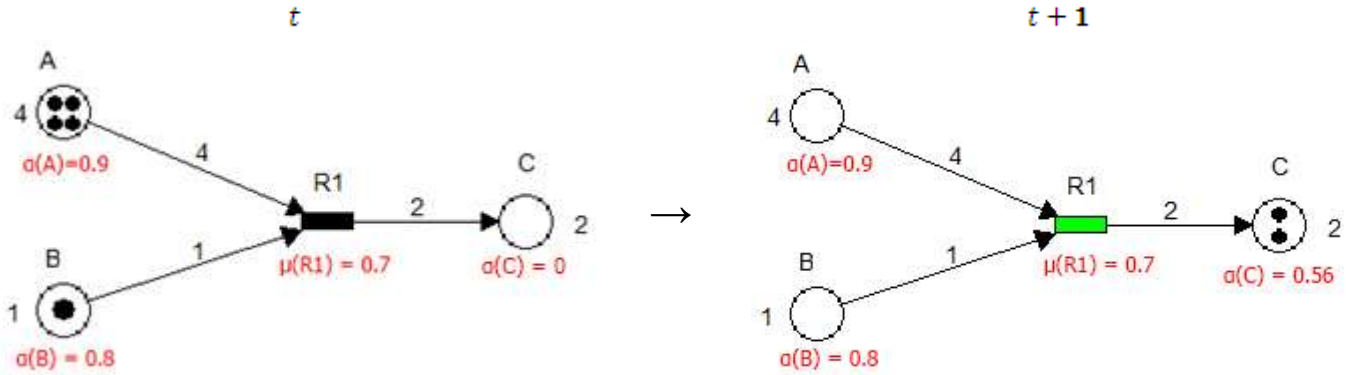
Με λιγότερο τυπικό τρόπο αυτό το οποίο συμβαίνει είναι το εξής:

1. Αφαιρεί τεκμήρια από κάθε θέση με την οποία είναι συνδεδεμένη με εισερχόμενη ακμή. Ο αριθμός των τεκμηρίων τα οποία καταναλώνει ορίζονται από το βάρος της εισερχόμενης ακμής.
2. Προσθέτει τεκμήρια σε κάθε θέση με την οποία είναι συνδεδεμένη με εξερχόμενη ακμή. Ο αριθμός των τεκμηρίων τα οποία προσθέτει ορίζονται από το βάρος της εξερχόμενης ακμής.
3. Μεταβάλλει τον βαθμό βεβαιότητας κάθε θέσης με την οποία είναι συνδεδεμένη με εξερχόμενη ακμή. Πιο συγκεκριμένα, ο νέος βαθμός βεβαιότητας ισούται με το γινόμενο του βαθμού βεβαιότητας της μετάβασης με τον μικρότερο βαθμό βεβαιότητας των θέσεων από τις οποίες η μετάβαση δέχεται εισερχόμενη ακμή. Στην περίπτωση που η θέση-στόχος έχει μεγαλύτερο βαθμό βεβαιότητας, τότε ο βαθμός βεβαιότητάς της δεν μεταβάλλεται.

Για να γίνει περισσότερο κατανοητός, ο παραπάνω κανόνας πυροδότησης, ας δούμε ένα παράδειγμα. Αν υποθέσουμε ότι έχουμε τον παρακάτω ασαφή κανόνα:

R1: If $A \wedge B$ then C

με $\alpha(A) = 0.9$, $\alpha(B) = 0.8$, $\mu(R1) = 0.7$ και $\alpha(C) = \mathbf{0}$. Ο κανόνας μπορεί να πυροδοτηθεί αφού $\alpha(A) \geq \mu(R1)$ και $\alpha(B) \geq \mu(R1)$. Έτσι λοιπόν το συμπέρασμα C εξάγεται με βεβαιότητα $\alpha(C) = \max\{\mu(R1) \cdot \min[(\{\alpha(A), \alpha(B)\}), 0]\} = \max\{0.7 \cdot 0.8, 0\} = \mathbf{0.56}$. ToFPN το οποίο αντιστοιχεί στον παραπάνω κανόνα παρουσιάζεται στο Σχήμα 4.2.



Σχήμα 4.2: παράδειγμα πυροδότησης ασαφούς μετάβασης.

Στο σημείο αυτό θα πρέπει να τονισθεί ότι ο αριθμός των τεκμηριών στις θέσεις A και B δεν έχει καμία σημασία ως προς την σωστή μοντελοποίηση του κανόνα. Φυσικά τα βάρη των αντίστοιχων ακμών θα πρέπει να ικανοποιούνται έτσι ώστε η μετάβαση να μπορεί να πυροδοτηθεί. Για τον λόγο αυτό είναι πολύ συχνό στα FPN να χρησιμοποιείται ένα και μόνο τεκμήριο στις μεταβάσεις.

4.4 Δομή των ασαφών κανόνων

Στην ενότητα 2.4 είδαμε ότι ο ασαφής λογικός τελεστής AND ($\wedge$) ορίζεται ως εξής:

$A \wedge B = \min(A, B)$

Συνεπώς οι κανόνες της ασαφούς Βάσης Κανόνων θα πρέπει να έχουν υποθέσεις χωρίς άρνηση, οι οποίες συνδέονται μόνο μέσω του ασαφούς λογικού τελεστή AND. Η μορφή των κανόνων παραπέμπει στις προτάσεις Horn της προτασιακής λογικής. Μία *πρόταση Horn*είναι μία λογική πρόταση της παρακάτω μορφής:

$$A_1 \wedge A_2 \wedge \cdots \wedge A_n \Rightarrow B$$

Πιο συγκεκριμένα, για τις προτάσεις Horn ισχύει το εξής:

1. Δεν υπάρχει καθόλου άρνηση (με τον τρόπο αυτό η Βάση Κανόνων δεν μπορεί ποτέ να εξάγει αντιφατικά αποτελέσματα, δηλαδή B και $\neg B$ ταυτόχρονα.
2. Όλες οι υποθέσεις συνδέονται μεταξύ τους με τον τελεστή AND ($\wedge$).
3. Υπάρχει το πολύ ένα συμπέρασμα σε κάθε πρόταση (μία πρόταση Horn θα μπορούσε να μην οδηγεί σε κάποιο συμπέρασμα).

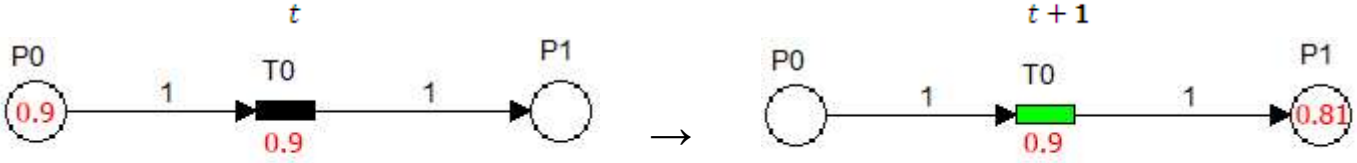
Όσον αφορά τον ορισμό των FPN που δώσαμε προηγουμένως, αρκεί από τα παραπάνω να ισχύουν τα κριτήρια 1 και 2, αφού στην περίπτωση που ο κανόνας έχει πολλά συμπεράσματα τότε η αντίστοιχη μετάβαση που πυροδοτείται καθορίζει τα τεκμήρια και τους βαθμούς βεβαιότητας σε καθένα από τα συμπεράσματα (θέσεις) αυτά.

4.5 Διαφορετικές προσεγγίσεις στον ορισμό των Ασαφών Δικτύων Petri

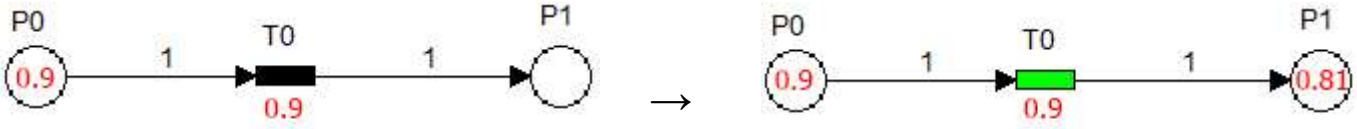
Εκτός από τον γενικευμένο ορισμό για ένα FPN που δόθηκε παραπάνω, υπάρχει μία πληθώρα διαφορετικών προσεγγίσεων. Οι προσεγγίσεις αυτές έχουν να κάνουν κυρίως με την χρήση και την λειτουργία των τεκμηρίων που υπάρχουν στις θέσεις. Επίσης κάποιες άλλες προσεγγίσεις ασχολούνται με το πως θα πρέπει να υπολογιστεί ο βαθμός βεβαιότητας μιας θέσης η οποία δέχεται εισερχόμενες ακμές από πολλές μεταβάσεις.

Μία πρώτη εναλλακτική προσέγγιση είναι αυτή στην οποία απουσιάζουν παντελώς τα τεκμήρια από τις θέσεις [18]. Έτσι λοιπόν, αντί των τεκμηρίων μέσα στις θέσεις υπάρχουν μόνο οι βαθμοί βεβαιότητας αυτών. Η προσέγγιση αυτή είναι αρκετά λογική, αφού όπως είδαμε στον προηγούμενο γενικευμένο ορισμό, οι τιμές των βαθμών βεβαιότητας είναι ανεξάρτητες από τον αριθμό των τεκμηρίων που μεταφέρονται.

Η προσέγγιση αυτή έχει με την σειρά της δύο περαιτέρω παραλλαγές. Σύμφωνα με την πρώτη παραλλαγή, κατά την πυροδότηση ενός κανόνα οι βαθμοί βεβαιότητας των θέσεων που στέλνουν εισερχόμενες ακμές σε μία μετάβαση απορροφώνται από την μετάβαση και μετασχηματίζονται σε νέους βαθμούς βεβαιότητας προς τις θέσεις όπου η μετάβαση στέλνει εξερχόμενες ακμές. Στην δεύτερη παραλλαγή, κατά την πυροδότηση ενός κανόνα οι βαθμοί βεβαιότητας συνεχίζουν και υφίστανται στις θέσεις που στέλνουν εισερχόμενες ακμές προς την μετάβαση που πυροδοτείται. Η παραλλαγή αυτή στηρίζεται στην λογική του ότι σε ένα έμπειρο σύστημα, όταν ένα γεγονός αποδεικνύεται ότι ισχύει, τότε συνεχίζει και ισχύει καθ' όλη την διαδικασία εξαγωγής ενός συμπεράσματος. Οι δύο διαφορετικές παραλλαγές παρουσιάζονται στο Σχήμα 4.3.



(a)



(b)

Σχήμα 4.3: (a) πυροδότηση με βάση την πρώτη παραλλαγή. Ο βαθμός βεβαιότητας δεν υπάρχει πλέον στην θέση P0. (b) πυροδότηση με βάση την δεύτερη παραλλαγή. Ο βαθμός βεβαιότητας συνεχίζει και υπάρχει στην θέση P0.

Μία δεύτερη εναλλακτική προσέγγιση έχει να κάνει με τον τρόπο υπολογισμού του βαθμού βεβαιότητας μία θέσης η οποία δέχεται εξερχόμενη ακμή από πολλές μεταβάσεις. Στον ορισμό του Κανόνα Πυροδότησης 1 που δόθηκε στην Ενότητα 4.3, όταν μία μετάβαση πυροδοτείται, ο βαθμός βεβαιότητας που προκύπτει από αυτή αντικαθιστά τον ήδη υπάρχον βαθμό βεβαιότητας της θέσης μόνο αν είναι μεγαλύτερος. Φυσικά αυτός δεν είναι ο μοναδικός τρόπος υπολογισμού του βαθμού βεβαιότητας. Άλλοι τρόποι υπολογισμού είναι επίσης οι $t-norms$ και $s-norms$ [18], οι οποίοι συνδυάζουν τον παλαιό με το νέο βαθμό βεβαιότητας μέσω κάποιου μαθηματικού τύπου.

Ανεξαρτήτως των διαφορετικών παραλλαγών, όλες οι δυνατές προσεγγίσεις για τα FPN θα πρέπει να πληρούν δυο πολύ σημαντικά κριτήρια:

1. Να είναι σε θέση να μοντελοποιήσουν μία ασαφή Βάση Κανόνων.
2. Να πληρούν τους βασικούς κανόνες λειτουργίας των δικτύων Petri.

Κεφάλαιο 5 – Ανάλυση και Σχεδίαση του PetriNetSim

5.1 Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιάσουμε την σχεδίαση ενός λογισμικού το οποίο θα υποστηρίζει τον σχεδιασμό και την προσομοίωση της εκτέλεσης Ασαφών Δικτύων Petri. Το λογισμικό θα

υλοποιηθεί στην γλώσσα προγραμματισμού Java. Το όνομα του λογισμικού θα είναι PetriNetSim (PetriNetSimulator). Οι προδιαγραφές του λογισμικού καθώς και το εννοιολογικό μοντέλο σχεδίασής του παρουσιάζονται παρακάτω.

5.2 Απαιτήσεις του PetriNetSim

Το λογισμικό το οποίο θα υλοποιηθεί θα πρέπει να πληρεί τουλάχιστον τις απαιτήσεις που παρουσιάζονται παρακάτω. Οι απαιτήσεις του λογισμικού έχουν χωριστεί σε λειτουργικές και μη λειτουργικές.

Οι απαιτήσεις παρουσιάζονται στον Πίνακα 5.1

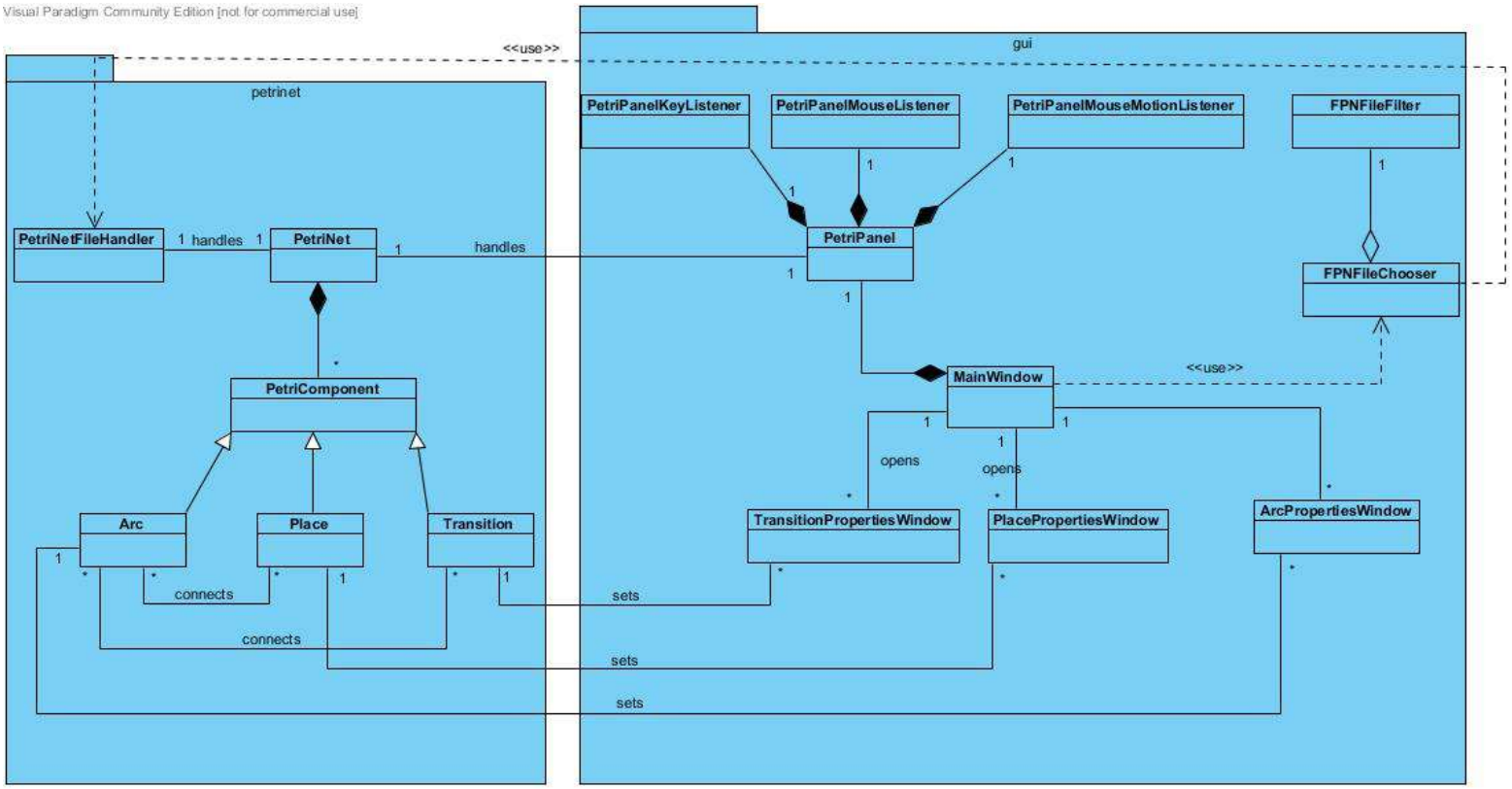
Λειτουργικές Απαιτήσεις	
#	**Περιγραφή**
#1	Το λογισμικό θα πρέπει να υποστηρίζει τον σχεδιασμό FPN μέσω γραφικού περιβάλλοντος. Το γραφικό περιβάλλον θα πρέπει να περιλαμβάνει κουμπιά για τον σχεδιασμό θέσεων, μεταβάσεων και ακμών.
#2	Ο χρήστης θα πρέπει να είναι σε θέση να θέτει τα χαρακτηριστικά της κάθε θέσης (ονομασία, μέγιστη χωρητικότητα, αρχική χωρητικότητα και βαθμός βεβαιότητας) μέσω του γραφικού περιβάλλοντος.
#3	Ο χρήστης θα πρέπει να είναι σε θέση να θέτει τα χαρακτηριστικά της κάθε μετάβασης (ονομασία, βαθμός βεβαιότητας) μέσω του γραφικού περιβάλλοντος.
#4	Ο χρήστης θα πρέπει να είναι σε θέση να θέτει τα χαρακτηριστικά της κάθε ακμής (βάρος) μέσω του γραφικού περιβάλλοντος.
#5	Το λογισμικό θα πρέπει να υποστηρίζει την προσομοίωση της λειτουργίας του FPN, μέχρι κουμπιού με την ονομασία Simulate.
#6	Το λογισμικό θα πρέπει να περιλαμβάνει κουμπί με την ονομασία Step. Με την χρήση του εν λόγω κουμπιού ο χρήστης θα μπορεί να εκτελεί την πυροδότηση των ικανοποιήσιμων μεταβάσεων την τρέχουσα χρονική στιγμή.
#7	Το λογισμικό θα πρέπει να περιλαμβάνει κουμπί με την ονομασία Play. Με την χρήση του εν λόγω κουμπιού ο χρήστης θα μπορεί να εκτελεί την πυροδότηση των ικανοποιήσιμων μεταβάσεων για κάθε χρονική στιγμή, μέχρι το FPN να μεταβεί σε αδιέξοδο. Μεταξύ των πυροδοτήσεων θα πρέπει να μεσολαβεί κάποιο μικρό χρονικό διάστημα, έτσι ώστε να είναι δυνατή η παρακολούθηση της διαδικασίας από τον χρήστη.
#8	Το λογισμικό θα πρέπει να περιλαμβάνει κουμπί με την ονομασία Pause. Με την χρήση του εν λόγω κουμπιού ο χρήστης θα μπορεί να σταματά την προσομοίωση στην περίπτωση που ο χρήστης είχα πατήσει το πλήκτρο Play.
#9	Το λογισμικό θα πρέπει να περιλαμβάνει κουμπί με την ονομασία Reset. Με την χρήση του εν λόγω κουμπιού όλες οι θέσεις παίρνουν την αρχική τους χωρητικότητα.
#10	Στην περίπτωση που η προσομοίωση εκτελείται, ο χρήστης δεν μπορεί να αλλάξει την δομή του FPN, μέχρι να απενεργοποιήσει την προσομοίωση πατώντας ξανά το πλήκτρο Simulate.
#11	Το λογισμικό θα πρέπει να εμφανίζει μηνύματα λάθους στην περίπτωση απόπειρας σύνδεσης δύο θέσεων ή δύο μεταβάσεων από τον χρήστη.
#12	Θα πρέπει να είναι δυνατή η αποθήκευση και η ανάκτηση αρχείων που αναπαριστούν FPN.
Μη Λειτουργικές Απαιτήσεις	
#	**Περιγραφή**
#1	Το λογισμικό θα πρέπει να υλοποιηθεί στην γλώσσα προγραμματισμού Java και ο χειρισμός του να γίνεται μέσω της χρήστης γραφικού περιβάλλοντος.

#2	Το αρχικό παράθυρο του λογισμικού θα πρέπει να είναι διαστάσεων 745x410 και να ανοίγει στο κέντρο της οθόνης.
#3	Ο χρήστης θα πρέπει να μπορεί να αλλάζει τις διαστάσεις του κεντρικού παραθύρου του λογισμικού ανά πάσα χρονική στιγμή.
#4	Το λογισμικό θα πρέπει να παρέχει τις επιλογές διαχείρισης των αρχείων μέσω μενού (επιλογές Νέο αρχείο, άνοιγμα αρχείου, αποθήκευση αρχείου, αποθήκευση αρχείου ως... και έξοδος).
#5	Τα αρχεία του λογισμικού θα πρέπει να έχουν κατάληξη .fpn.
#6	Τα αρχεία του λογισμικού θα πρέπει να αποθηκεύουν την πληροφορία σε μορφή κειμένου.
#7	Τα κουμπιά του λογισμικού θα πρέπει να συμπεριφέρονται με τρόπο τέτοιο έτσι ώστε ο χρήστης να γνωρίζει ανά πάσα χρονική στιγμή την τελευταία επιλογή η οποία ενεργοποιήθηκε.
#8	Τα χαρακτηριστικά της κάθε θέσης/μετάβασης/ακμής θα πρέπει να ρυθμίζονται με την χρήση αναδυόμενου διαλόγου το οποίο θα έχει τα κουμπιά Save και Cancel.
#9	Η φόρμα σχεδίασης του FPN θα πρέπει να έχει άσπρο φόντο.

Πίνακας 5.1: οι λειτουργικές και οι μη λειτουργικές απαιτήσεις του λογισμικού.

5.3 Διάγραμμα Κλάσεων

Το Διάγραμμα Κλάσεων (εννοιολογικό μοντέλο) πάνω στο οποίο στηρίζεται η εφαρμογή μας παρουσιάζεται στην Εικόνα 5.1 της επόμενης σελίδας.



Εικόνα 5.1: το διάγραμμα κλάσεων (εννοιολογικό μοντέλο) του συστήματος προς σχεδίαση. Στο παραπάνω διάγραμμα παρατηρούμε ότι οι κλάσεις χωρίζονται σε δύο πακέτα: (a) στο πακέτο petrinet περιλαμβάνονται όλες οι κλάσεις που αφορούν την αποθήκευση των πληροφοριών σχετικά με τα στοιχεία ενός FPN και (b) στο πακέτο gui περιλαμβάνονται όλες οι κλάσεις που αφορούν τον σχεδιασμό του γραφικού περιβάλλοντος.

ΚΕΦΑΛΑΙΟ 6 – Υλοποίηση του PetriNetSim

6.1 - Εισαγωγή

Στο εν λόγω κεφάλαιο θα παρουσιάσουμε όλες τις τεχνικές λεπτομέρειες που αφορούν την υλοποίηση του PetriNetSim. Πιο συγκεκριμένα, θα αναφέρουμε την δομή του προγράμματος, τις κλάσεις τις οποίες υλοποιήθηκαν, τον σκοπό και τις μεθόδους της κάθε μίας από αυτές. Επίσης, στα σημεία στα οποία κρίνεται απαραίτητο, θα παρουσιαστεί η λογική του αλγορίθμου που υλοποιείται πίσω από κάθε μέθοδο.

6.2 Αρχιτεκτονική Δομή της εφαρμογής

Η εφαρμογή PetriNetSim υλοποιήθηκε στην γλώσσα προγραμματισμού Java. Πιο συγκεκριμένα, έγινε χρήση του IDENetBeans [22]. Η εφαρμογή αποτελείται από 3 πακέτα (packages), τα οποία είναι τα παρακάτω:

Πακέτο petrinet

Στο πακέτο αυτό περιέχονται όλες οι κλάσεις οι οποίες είναι απαραίτητες για την αναπαράσταση στην γλώσσα προγραμματισμού ενός FPN. Οι κλάσεις είναι 6 στον αριθμό και τα ονόματά τους δίνονται στην συνέχεια:

- Arc
- PetriComponent
- PetriNet
- PetriNetFileHandler
- Place
- Transition

Πακέτο gui

Στο πακέτο αυτό περιέχονται όλες οι κλάσεις οι οποίες είναι απαραίτητες για την υλοποίηση του γραφικού περιβάλλοντος. Οι κλάσεις είναι 10 στον αριθμό και τα ονόματά τους δίνονται στην συνέχεια:

- ArcPropertiesWindow
- FPNFileChooser
- FPNFileFilter
- MainWindow
- PetriPanel
- PetriPanelKeyListener
- PetriPanelMouseListener
- PetriPanelMouseMotionListener
- PlacePropertiesWindow
- TransitionPropertiesWindow

Πακέτο petrinetsim

Στο πακέτο αυτό περιέχεται μόνο η κλάση PetriNetSim, η οποία περιέχει την μέθοδο main. Η μέθοδο main απλά δημιουργεί και εμφανίζει ένα αντικείμενο της κλάσης gui.MainWindow, οπότε η συνεισφορά του εν λόγω πακέτου στο λογισμικό συνολικά είναι μηδαμινή.

Στις ακόλουθες ενότητες θα αναλύσουμε την λειτουργία των κλάσεων που περιέχονται στα πακέτα petrinetκαι gui. Η περιγραφή της κάθε κλάσης θα χωριστεί σε 3 επιμέρους μέρη:

1. Σκοπός της κλάσης
2. Πεδία
3. Μέθοδοι
4. Περιγραφή των αλγορίθμων (όπου αυτό απαιτείται)

6.3 Περιγραφή του πακέτου petrinet

6.3.1 Κλάση PetriComponent

1. Σκοπός: η κλάση PetriComponentείναι η υπερκλάση των στοιχείων που απαρτίζουν ένα FPN. Πιο συγκεκριμένα, τα στοιχεία αυτά είναι οι θέσεις, οι μεταβάσεις και οι ακμές, οι οποίες αποτελούν και υποκλάσεις της PetriComponent. Η κλάση PetriComponentείναι αφηρημένη (abstract), καθώς περιέχει μεθόδους που οι υποκλάσεις της υλοποιούν με διαφορετικό τρόπο.

2. Πεδία της κλάσης PetriComponent

Τα πεδία της κλάσης PetriComponentπαρουσιάζονται παρακάτω στον Πίνακα 6.1. Για κάθε πεδίο παρουσιάζεται το όνομά του, ο τύπος του καθώς και μία σύντομη περιγραφή της χρησιμότητάς του.

Όνομα Πεδίου	Τύπος	Περιγραφή
id	int	Κάθε PetriComponent έχει έναν μοναδικό ακέραιο αριθμό ως ταυτότητα.
nextId	static int	Ο επόμενος ακέραιος αριθμός που θα χρησιμοποιηθεί ως ταυτότητα (αρχική τιμή 0).
name	String	Το όνομα του PetriComponent.
isSelected	boolean	Flag το οποίο δείχνει αν το εν λόγω PetriComponent έχει επιλεχθεί.
drawOn	boolean	Flag το οποίο δείχνει αν το εν λόγω PetriComponent πρέπει να απεικονισθεί άμεσα στο γραφικό περιβάλλον. Είναι χρήσιμο κατά την ανάκτηση ενός PetriComponent από ένα αρχείο.

Πίνακας 6.1: τα πεδία της κλάσης petrinet.PetriComponent.

3. Μέθοδοι της κλάσης PetriComponent

Οι μέθοδοι της κλάσης PetriComponentπαρουσιάζονται παρακάτω στον Πίνακα 6.2. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	Περιγραφή
public PetriComponent()	Constructor κατά την δημιουργία του, το PetriComponent παίρνει αυτόματα τον επόμενο αύξοντα αριθμό ως ταυτότητα.
public int getId()	Getter για το πεδίο id.
public void setId(int id)	Setter για το πεδίο id.
public String getName()	Getter για το πεδίο name.
public void setName(String name)	Setter για το πεδίο name.
public boolean isSelected()	Getter για το πεδίο isSelected.
public void setIsSelected(boolean isSelected)	Setter για το πεδίο isSelected.
public abstract void draw(Graphics g)	Αφηρημένη (abstract) μέθοδος την οποία κάθε υποκλάση υλοποιεί με την δικό της τρόπο. Στην μέθοδο αυτή το κάθε στοιχείο ζωγραφίζει τον εαυτό του στο PetriPanel.
public abstract boolean contains(Point p)	Αφηρημένη (abstract) μέθοδος την οποία κάθε υποκλάση υλοποιεί με την δικό της τρόπο. Στην μέθοδο αυτή το κάθε στοιχείο ελέγχει αν μέσα στα όριά του περιέχεται κάποιο σημείο.

public abstract Rectangle2D getBounds()	Αφηρημένη (abstract) μέθοδος την οποία κάθε υποκλάση υλοποιεί με την δικό της τρόπο. Η μέθοδος επιστρέφει ένα ορθογώνιο παραλληλόγραμμο το οποίο περιλαμβάνει μέσα στα όριά του το στοιχείο στο γραφικό περιβάλλον.
public abstract void updatePosition(int x, int y, Graphics g)	Αφηρημένη (abstract) μέθοδος την οποία κάθε υποκλάση υλοποιεί με την δικό της τρόπο. Η μέθοδος ανανεώνει την θέση ενός στοιχείου στον χώρο, όταν ο χρήστης το μετακινεί με την χρήση του ποντικιού.

Πίνακας 6.2: οι μέθοδοι της κλάσης petrinet.PetriComponent.

6.3.2 Η κλάση Place

1. Σκοπός: η κλάση Place υλοποιεί την κλάση PetriComponent και αναπαριστά μία θέση του FPN.

2. Πεδία της κλάσης Place

Τα πεδία της κλάσης Place παρουσιάζονται παρακάτω στον Πίνακα 6.3. Για κάθε πεδίο παρουσιάζεται το όνομά του, ο τύπος του καθώς και μία σύντομη περιγραφή της χρησιμότητάς του.

Όνομα Πεδίου	**Τύπος**	**Περιγραφή**
maxCapacity	int	Η μέγιστη χωρητικότητα της θέσης.
initialCapacity	int	Η αρχική χωρητικότητα της θέσης.
tokensNo	int	Ο αριθμός των τεκμηρίων της θέσης (τρέχουσα χωρητικότητα).
truthFactor	double	Ο βαθμός βεβαιότητας της θέσης.
placeShape	Shape	Το σχήμα της θέσης στο γραφικό περιβάλλον.
outgoingTransitions	ArrayList<Transition>	Οι μεταβάσεις στις οποίες η θέση στέλνει εξερχόμενη ακμή.
incomingTransitions	ArrayList<Transition>	Οι μεταβάσεις από τις οποίες η θέση δέχεται εισερχόμενη ακμή.
placeCounter	static int	Αύξων αριθμός δημιουργίας κάθε θέσης (αρχική τιμή 0).
petriNet	PetriNet	Το δίκτυο Petri στο οποίο η θέση ανήκει.
WIDTH	static final int	Το μήκος μιας θέσης στο γραφικό περιβάλλον (αρχική τιμή 35).
HEIGHT	static final int	Το ύψος μιας θέσης στο γραφικό περιβάλλον (αρχική τιμή 35).

Πίνακας 6.3: τα πεδία της κλάσης petrinet.Place.

3. Μέθοδοι της κλάσης Place

Οι μέθοδοι της κλάσης Place παρουσιάζονται παρακάτω στον Πίνακα 6.4. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	**Περιγραφή**

public Place(int x, int y, Graphics g)	Constructor
public int getMaxCapacity()	Getter για το πεδίο maxCapacity.
public void setMaxCapacity(int maxCapacity)	Setter για το πεδίο maxCapacity.
public int getInitialCapacity()	Getter για το πεδίο initialCapacity.
public void setInitialCapacity(int initialCapacity)	Setter για το πεδίο initialCapacity.
public int getTokensNo()	Getter για το πεδίο tokensNo.
public void setTokensNo(int tokensNo)	Setter για το πεδίο tokensNo.
public double getTruthFactor()	Getter για το πεδίο truthFactor.
public void setTruthFactor(double truthFactor)	Setter για το πεδίο truthFactor.
public Shape getPlaceShape()	Getter για το πεδίο placeShape.
public void setPlaceShape(Shape placeShape)	Setter για το πεδίο placeShape.
public PetriNet getPetriNet()	Getter για το πεδίο petriNet.
public void setPetriNet(PetriNet petriNet)	Setter για το πεδίο petriNet.
public ArrayList<Transition> getOutgoingTransitions()	Getter για το πεδίο outgoingTransitions.
public ArrayList<Transition> getIncomingTransitions()	Setter για το πεδίο outgoingTransitions.
public void addOutgoingTransition(Transition t)	Προσθέτει μία μετάβαση στην λίστα με τις εξερχόμενες μεταβάσεις.
public void removeOutgoingTransition(Transition t)	Διαγράφει μία μετάβαση από την λίστα με τις εξερχόμενες μεταβάσεις.
public void addIncomingTransition(Transition t)	Προσθέτει μία μετάβαση στην λίστα με τις εισερχόμενες μεταβάσεις.
public void removeIncomingTransition(Transition t)	Διαγράφει μία μετάβαση από την λίστα με τις εισερχόμενες μεταβάσεις.
public void draw(Graphics g)	Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Η εν λόγω μέθοδος ζωγραφίζει μία θέση στο γραφικό περιβάλλον.
public boolean contains(Point p)	Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Η εν λόγω μέθοδος ελέγχει αν μία θέση περιέχει ένα συγκεκριμένο σημείο, το οποίο δίνεται ως όρισμα.
public Rectangle2D getBounds()	Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Η εν λόγω μέθοδος επιστρέφει ένα τετράγωνο το οποίο περιλαμβάνει πλήρως την θέση μέσα στα όριά του.
public void updatePosition(int x, int y, Graphics g)	Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Η εν λόγω μέθοδος αλλάζει την θέση μίας θέσης στον χώρο, όταν ο χρήστης την μετακινεί με την χρήση του ποντικιού.

Πίνακας 6.4: οι μέθοδοι της κλάσης petrinet.Place.

6.3.3 Η κλάση Transition

1. Σκοπός: η κλάση Transitionεπεκτείνει την κλάση PetriComponent και αναπαριστά μία μετάβαση του FPN.

2. Πεδία της κλάσης Transition

Τα πεδία της κλάσης Transition παρουσιάζονται παρακάτω στον Πίνακα 6.5. Για κάθε πεδίο παρουσιάζεται το όνομά του, ο τύπος του καθώς και μία σύντομη περιγραφή της χρησιμότητάς του.

Όνομα Πεδίου	Τύπος	Περιγραφή
threshold	double	Το κατώφλι (βαθμός βεβαιότητας) της μετάβασης.
transitionShape	Shape	Το σχήμα της μετάβασης στο γραφικό περιβάλλον.
outgoingPlaces	ArrayList<Place>	Οι θέσεις στις οποίες η μετάβαση στέλνει εξερχόμενη ακμή.
incomingPlaces	ArrayList<Place>	Οι θέσεις από τις οποίες η μετάβαση δέχεται εισερχόμενη ακμή.
transitionCounter	static int	Αύξων αριθμός δημιουργίας κάθε μετάβασης (αρχική τιμή 0).
petriNet	PetriNet	Το δίκτυο Petri στο οποίο η μετάβαση ανήκει.
WIDTH	static final int	Το μήκος μιας θέσης στο γραφικό περιβάλλον (αρχική τιμή 30).
HEIGHT	static final int	Το ύψος μιας θέσης στο γραφικό περιβάλλον (αρχική τιμή 15).

Πίνακας 6.5: τα πεδία της κλάσης petrinet.Transition.

3. Μέθοδοι της κλάσης Transition

Οι μέθοδοι της κλάσης Transition παρουσιάζονται παρακάτω στον Πίνακα 6.6. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	Περιγραφή
public Transition(int x, int y, Graphics g)	Constructor
public double getThreshold()	Getter για το πεδίο threshold.
public void setThreshold(double threshold)	Setter για το πεδίο threshold.
public Shape getTransitionShape()	Getter για το πεδίο transitionShape.
public void setTransitionShape(Shape transitionShape)	Setter για το πεδίο transitionShape.
public ArrayList<Place> getOutgoingPlaces()	Getter για το πεδίο outgoingPlaces.
public ArrayList<Place> getIncomingPlaces()	Setter για το πεδίο outgoingPlaces.
public PetriNet getPetriNet()	Getter για το πεδίο petriNet.
public void setPetriNet(PetriNet petriNet)	Setter για το πεδίο petriNet.
public void addOutgoingPlace(Place p)	Προσθέτει μία θέση στην λίστα με τις εξερχόμενες θέσεις.
public void removeOutgoingPlace(Place p)	Διαγράφει μία θέση από την λίστα με τις εξερχόμενες θέσεις.
public void addIncomingPlaces(Place p)	Προσθέτει μία θέση στην λίστα με τις εισερχόμενες θέσεις.
public void removeIncomingPlace(Place p)	Διαγράφει μία θέση από την λίστα με τις εισερχόμενες θέσεις.

public void draw(Graphics g)	Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Η εν λόγω μέθοδος ζωγραφίζει μία μετάβαση στο γραφικό περιβάλλον.
public boolean contains(Point p)	Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Η εν λόγω μέθοδος ελέγχει αν μία μετάβαση περιέχει ένα συγκεκριμένο σημείο, το οποίο δίνεται ως όρισμα.
public Rectangle2D getBounds()	Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Η εν λόγω μέθοδος επιστρέφει ένα τετράγωνο το οποίο περιλαμβάνει πλήρως την μετάβαση μέσα στα όριά του.
public void updatePosition(int x, int y, Graphics g)	Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Η εν λόγω μέθοδος αλλάζει την θέση μίας μετάβασης στον χώρο, όταν ο χρήστης την μετακινεί με την χρήση του ποντικιού.
public void fire()	Πυροδοτεί την μετάβαση. Ο ακριβής αλγόριθμος της μετάβασης περιγράφεται παρακάτω.

Πίνακας 6.6: οι μέθοδοι της κλάσης petrinet.Transition.

4. Περιγραφή αλγορίθμων

4.1 Αλγόριθμος πυροδότησης της μετάβασης, ο οποίος υλοποιείται στην μέθοδο publicvoidfire().

```
// Ανανέωσε τις τρέχουσες χωρητικότητες των εισερχόμενων θέσεων
Για κάθε θέση p που ανήκει στις εισερχόμενες θέσεις της μετάβασης
   Βρες την ακμή a που συνδέει την θέση με την μετάβαση;
   Θέσε τον αριθμό των τεκμηρίων της θέσης p ίσο p.τρέχουσα_χωρητικότητα – a.βάρος;
// Ανανέωσε τις τρέχουσες χωρητικότητες και τους βαθμούς
// βεβαιότητας των εξερχόμενων θέσεων
// Αρχικά βρες τον μικρότερο βαθμό βεβαιότητας από τις
// εισερχόμενες θέσεις
minTruthFactor = 1.0;
Για κάθε θέση p που ανήκει στις εισερχόμενες θέσεις της μετάβασης
     Εάνp.truthFactor<minTruthFactor
truthFactor = p.truthFactor;
// Υπολόγισε το νέο βαθμό βεβαιότητας των εξερχόμενων θέσεων
newTruthFactor = threshold * minTruthFactor;
// Για κάθε θέση με την οποία η μετάβαση συνδέεται με εξερχόμενη ακμή
Για κάθε θέση p που ανήκει στις εξερχόμενες θέσεις της μετάβασης
   Βρες την ακμή a που συνδέει την θέση με την μετάβαση;
   Θέσε τον αριθμό των τεκμηρίων της θέσης p ίσο p.τρέχουσα_χωρητικότητα + a.βάρος;
   Εάν minTruthFactor > p.truthFactor
      p.truthFactor = minTruthFactor;
```

6.3.4 Η κλάση Arc

1. Σκοπός: η κλάση Arc επεκτείνει την κλάση PetriComponent και αναπαριστά μία ακμή του FPN.

2. Πεδία της κλάσης Arc

Τα πεδία της κλάσης Arc παρουσιάζονται παρακάτω στον Πίνακα 6.7. Για κάθε πεδίο παρουσιάζεται το όνομά του, ο τύπος του καθώς και μία σύντομη περιγραφή της χρησιμότητάς του.

Όνομα Πεδίου	Τύπος	Περιγραφή
weight	int	Το βάρος της ακμής.
line	Line2D.Double	Η γραμμή στο γραφικό περιβάλλον.
arrow	Shape	Το βέλος στο γραφικό περιβάλλον.
startsFrom	PetriComponent	Το στοιχείο του δικτύου Petri από το οποίο η ακμή εξέρχεται.
endsTo	PetriComponent	Το στοιχείο του δικτύου Petri στο οποίο η ακμή καταλήγει.

Πίνακας 6.7: τα πεδία της κλάσης petrinet.Arc.

3. Μέθοδοι της κλάσης Arc

Οι μέθοδοι της κλάσης Transition παρουσιάζονται παρακάτω στον Πίνακα 6.8. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	Περιγραφή
public Arc(PetriComponent startsFrom, PetriComponent endsTo, Graphics g)	Constructor
public int getWeight()	Getterγια το πεδίο weight.
public void setWeight(int weight)	Setterγια το πεδίο weight.
public PetriComponent getStartsFrom()	Getter για το πεδίο startsFrom.
public PetriComponent getEndsTo()	Setterγια το πεδίο endsTo.
private int sign(double x1, double x2)	Ελέγχει το πρόσημο της διαφοράς των x1, x2. Η μέθοδος αυτή είναι απαραίτητη για τον σχεδιασμό της ακμής στο γραφικό περιβάλλον.
public static Shape createArrowShape(Point2D.Double fromPt, Point2D.Double toPt)	Ζωγραφίζει ένα βέλος στο γραφικό περιβάλλον.
private Point2D.Double midpoint(Point2D.Double p1, Point2D.Double p2)	Επιστρέφει το μέσο της απόστασης δύο σημείων.
public void draw(Graphics g)	Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Η εν λόγω μέθοδος ζωγραφίζει μία ακμή στο γραφικό περιβάλλον.
public boolean contains(Point p)	Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Η εν λόγω μέθοδος ελέγχει αν μία ακμή περιέχει ένα συγκεκριμένο σημείο, το οποίο δίνεται ως όρισμα.
public Rectangle2D getBounds()	Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Η εν λόγω μέθοδος επιστρέφει ένα τετράγωνο το οποίο περιλαμβάνει πλήρως την ακμή μέσα στα όριά του.

public void updatePosition(int x, int y, Graphics g)	Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Στην περίπτωση της ακμής αυτή δεν μπορεί να μετακινηθεί, καθώς εξαρτάται από την θέση των στοιχείων που ενώνει.

Πίνακας 6.8: οι μέθοδοι της κλάσης petrinet.Arc.

6.3.5 Η κλάση PetriNet

1. Σκοπός: η κλάση PetriNet αναπαριστά ένα FPN.

2. Πεδία της κλάσης PetriNet

Τα πεδία της κλάσης PetriNet παρουσιάζονται παρακάτω στον Πίνακα 6.9. Για κάθε πεδίο παρουσιάζεται το όνομά του, ο τύπος του καθώς και μία σύντομη περιγραφή της χρησιμότητάς του.

Όνομα Πεδίου	Τύπος	Περιγραφή
places	ArrayList<Place>	Λίστα με τις θέσεις του δικτύου.
transitions	ArrayList<Transition>	Λίστα με τις μεταβάσεις του δικτύου.
arcs	ArrayList<Arc>	Λίστα με τις ακμές του δικτύου.
selectedComponentOn	boolean	Flag το οποίο αποθηκεύει την πληροφορία για το αν κάποιο στοιχείο του PetriNet έχει επιλεγεί στο γραφικό περιβάλλον

Πίνακας 6.9: τα πεδία της κλάσης petrinet.PetriNet.

3. Μέθοδοι της κλάσης PetriNet

Οι μέθοδοι της κλάσης PetriNet παρουσιάζονται παρακάτω στον Πίνακα 6.10. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	Περιγραφή
public PetriNet()	Constructor
public ArrayList<Place> getPlaces()	Getterγια το πεδίο places.
public void setPlaces(ArrayList<Place> places)	Setterγια το πεδίο places.
public ArrayList<Transition> getTransitions()	Getter για το πεδίο transitions.
public ArrayList<Arc> getArcs()	Getter για το πεδίο arcs.
public void setTransitions(ArrayList<Transition> transitions)	Setter για το πεδίο transitions.
public void setSelectedComponentOn()	Setter για το πεδίο selectedComponent.
public boolean hasSelectedComponent()	Getter για το πεδίο selectedComponent.
public void addPlace(Place p)	Προσθέτει μία θέση στην λίστα με τις θέσεις.
public void removePlace(Place p)	Διαγράφει μία θέση από την λίστα με τις θέσεις.
public void addTransition(Transition t)	Προσθέτει μία μετάβαση στην λίστα με τις μεταβάσεις.
public void removeTransition(Transition t)	Διαγράφει μία μετάβαση από την λίστα με τις

	μεταβάσεις.
public void addArc(Arc a)	Προσθέτει μία ακμή στην λίστα με τις ακμές.
public void removeArc(Arc a)	Διαγράφει μία ακμή από την λίστα με τις ακμές.
public void draw(Graphics g)	Ζωγραφίζει το FPN στο γραφικό περιβάλλον, καλώντας τις επιμέρους μεθόδους draw των στοιχείων που το απαρτίζουν.
public PetriComponent getSelected(Point point)	Επιστρέφει το στοιχείο του FPN στο οποίο περιέχεται το σημείο το οποίο η μέθοδος δέχεται σαν όρισμα.
public PetriComponent getSelected()	Επιστρέφει το στοιχείο του FPN το οποίο είναι επιλεγμένο. Αν κανένα στοιχείο δεν είναι επιλεγμένο, τότε επιστρέφει null.
public void removeComponent(PetriComponent c)	Διαγράφει ένα στοιχείο από το FPN.
public void deselectAll()	Απο-επιλέγει όλα τα επιλεγμένα στοιχεία του FPN.
public boolean subGroup(ArrayList<Transition> group1, ArrayList<Transition> group2)	Ελέγχει αν ένα σύνολο μεταβάσεων αποτελεί υποσύνολο ενός άλλου. Η μέθοδος αυτή χρησιμοποιείται από την μέθοδο stepSimulation().
public int stepSimulation()	Η μέθοδος αυτή βρίσκει όλες τις ικανοποιήσιμες μεταβάσεις του FPN σε μία δεδομένη χρονική στιγμή και τις πυροδοτεί, σύμφωνα πάντα με τους κανόνες πυροδότησης. Η μέθοδος επιστρέφει τον αριθμό των μεταβάσεων που εν τέλει πυροδοτήθηκαν.
public void reset()	Θέτει σε όλες τις θέσεις την αρχική τους χωρητικότητα.
public PetriComponent getComponentById(int id)	Επιστρέφει το στοιχείο του FPN συναρτήσει της ταυτότητάς του, την οποία δέχεται ως όρισμα.
public int findMaxId()	Βρίσκει και επιστρέφει την μέγιστη ταυτότητα που έχει δοθεί σε κάποιο στοιχείο του FPN.

Πίνακας 6.10: οι μέθοδοι της κλάσης petrinet.PetriNet.

4. Περιγραφή αλγορίθμων

4.1 Αλγόριθμος πυροδότησης των ικανοποιήσιμων μεταβάσεων, ο οποίος υλοποιείται στην μέθοδο public int stepSimulation().

```
// Στην εν λόγω λίστα αποθηκεύονται οι ικανοποιήσιμες μεταβάσεις
satisfiable = { };

// Βήμα 1: βρες αρχικά τις ικανοποιήσιμες μεταβάσεις
Για κάθε μετάβαση t
   boolean flag = true;
   Για κάθε εισερχόμενη θέση p
      Βρες την ακμή a που συνδέει την p με την t;
// Έλεγξε τα τεκμήρια της θέσης σε σχέση με το
      // βάρος της ακμής καθώς και τον βαθμό αληθείας της
      // θέσης σε σχέση με το κατώφλι της μετάβασης
Αν p.τρέχουσα_χωρητικότητα < a.βάρος OR p.βαθμός αληθείας < t.κατώφλι
         flag = false;
   Για κάθε εισερχόμενη θέση p
      Βρες την ακμή a που συνδέει την t με την p;
```

```
      // Έλεγξε τα τεκμήρια της θέσης σε σχέση με το βάρος της ακμής
      Αν p.τρέχουσα_χωρητικότητα + a.βάρος > p.μέγιστυη_χωρητικότητα
         flag = false;
   // Αν η μετάβαση πληρεί τα κριτήρια τότε κάνε εισαγωγή αυτής
// στην λίστα satisfiable
Αν flag = true
satisfiable = satisfiable U t;

// Βήμα 2: βρες ποιες από τις ικανοποιήσιμες μεταβάσεις είναι ισοπίθανες
equalProbableGroups = { };

Για κάθε θέση p
tGroup1 = { };
Για κάθε εξερχόμενη μετάβαση t της p
      Ανη t ∈ satisfiable
         tGroup1 = tGroup1 U { t };
Αν tGroup1 ≠ { }
      equalProbableGroups = equalProbableGroups ΕΝΩΣΗ tGroup1;
   tGroup2 = { };
Για κάθε εισερχόμενη μετάβαση t της p
Ανη t ∈ satisfiable
         tGroup2 = tGroup2 U { t };
Αν tGroup2 ≠ { }
      equalProbableGroups = equalProbableGroups U tGroup2;

// Ειδική μέριμνα για τις μεταβάσεις που δεν έχουν εισόδους
// ή εξόδους άρα είναι ικανοποιήσιμες
Για κάθε μετάβαση t
Αν t.εισερχόμενες_θέσεις = { } ANDt.εξερχόμενες_θέσεις = { }
tGroup = { t };
   equalProbableGroups = equalProbableGroups U tGroup;

// Βήμα 3: Πυροδότηση
fireCounter = 0;
ΓιακάθεgroupισοπίθανωνμεταβάσεωνtGroup ∈ satisfiable
Επέλεξε με τυχαίο τρόπο μία μετάβαση selected ∈ tGroup;
   Πυροδότησε την selected;
fireCounter++;
   Για κάθε μετάβαση t ∈ tGroup
      Διέγραψε την t αν τυχόν ανήκει στα υπόλοιπα groups του satisfiable
// Επέστρεψε τον αριθμό των μεταβάσεων που πυροδοτήθηκαν
return fireCounter;
```

6.3.6 Η κλάση PetriNetFileHandler

1. Σκοπός :η κλάση PetriNetFileHandler χρησιμοποιείται την αποθήκευση και την ανάκτηση FPN σε αντίστοιχα αρχεία (με κατάληξη .fpn).

2. Πεδία της κλάσης PetriNetFileHandler

Η κλάσηPetriNetFileHandler δεν έχει πεδία.

3. Μέθοδοι της κλάσης PetriNetFileHandler

Οι μέθοδοι της κλάσης PetriNetFileHandle ιπαρουσιάζονται παρακάτω στον Πίνακα 6.11. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	Περιγραφή
public void savePetriNet(PetriNet petriNet, String filename)	Αποθηκεύει ένα FPN στο αρχείο με ονομασία filename.
public void readPetriNet(PetriPanel petriPanel, String filename)	Ανακτά ένα FPN από το αρχείο με ονομασία filename.

Πίνακας 6.11: οι μέθοδοι της κλάσης petrinet.PetriNetFileHandler.

4. Περιγραφή αλγορίθμων

4.1 Περιγραφή της δομής των αρχείο .fpn

Τα αρχεία .fpnέχουν την παρακάτω δομή:

```
Αριθμόςθέσεων
Id Name x y maxCapacity initialCapacity tokensNo truthFactor
…
Αριθμός μεταβάσεων
IdNamexythreshold
...
Αριθμός ακμών
IdId-στοιχείου-1 Id-στοιχείου-2
…
```

Πιο συγκεκριμένα, αρχικά αποθηκεύεται ο αριθμός των θέσεων που περιέχονται στο FPN. Κάθε θέση καταλαμβάνει στην συνέχεια μία γραμμή στο αρχείο κειμένου με τις ακόλουθες πληροφορίες:

- **Id**: η ταυτότητα της θέσης.
- **Name**: το όνομα της θέσης.
- **x**: η συντεταγμένη χτου κέντρου της θέσης στο γραφικό περιβάλλον.
- **y**: η συντεταγμένη υτου κέντρου της θέσης στο γραφικό περιβάλλον.
- **maxCapacity**: η μέγιστη χωρητικότητα της θέσης.
- **initialCapacity**: η αρχική χωρητικότητα της θέσης.
- **tokensNo**: η τρέχουσα χωρητικότητα της θέσης.
- **truthFactor**: ο βαθμός αληθείας της θέσης.

Στην συνέχεια αποθηκεύεται ο αριθμός των μεταβάσεων που περιέχονται στο FPN. Κάθε μετάβαση καταλαμβάνει στην συνέχεια μία γραμμή στο αρχείο κειμένου με τις ακόλουθες πληροφορίες:

- **Id**: η ταυτότητα της μετάβασης.
- **Name**: το όνομα της μετάβασης.
- **x**: η συντεταγμένη χτου κέντρου της μετάβασης στο γραφικό περιβάλλον.
- **y**: η συντεταγμένη υτου κέντρου της μετάβασης στο γραφικό περιβάλλον.
- **threshold**: το κατώφλι (βαθμός αληθείας) της μετάβασης.

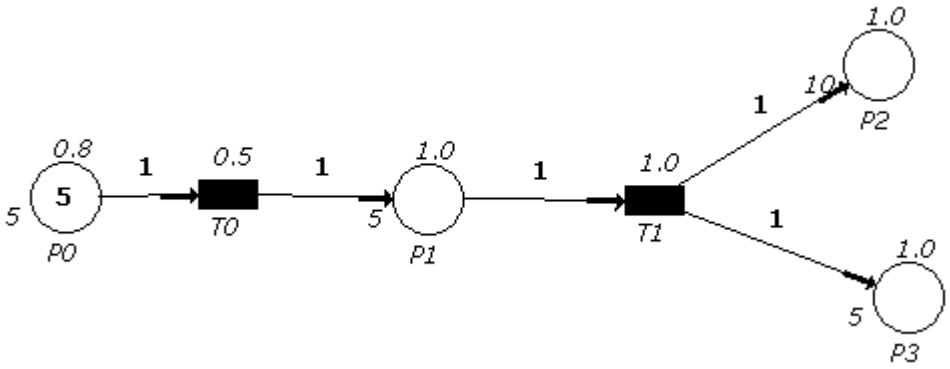
Τέλος, στην συνέχεια αποθηκεύεται ο αριθμός των ακμών που συνδέουν τα στοιχεία του FPN. Κάθε ακμή καταλαμβάνει στην συνέχεια μία γραμμή στο αρχείο κειμένου με τις ακόλουθες πληροφορίες:

- **Id**: η ταυτότητα της ακμής
- **Id-στοιχείου-1**:το Idτου στοιχείου (θέση ή μετάβαση) από το οποίο η ακμή ξεκινά.
- **Id-στοιχείου-2**:το Idτου στοιχείου (θέση ή μετάβαση) στο οποίο η ακμή καταλήγει.

Ένα παράδειγμα ενός αρχείου .fpn το οποίο αντιστοιχεί στο FPNπου παρουσιάζεται στην Εικόνα 6.1, ακολουθεί παρακάτω:

Αρχείο .fpn

```
4
1 P0 93 156 5 5 5 0.8
2 P1 274 157 5 0 0 1.0
3 P2 499 90 10 0 0 1.0
4 P3 514 206 5 0 0 1.0
2
5 T0 177 165 0.5
6 T1 390 168 1.0
5
7 1 5
8 5 2
9 2 6
10 6 3
11 6 4
```



Εικόνα 6.1: το FPN το οποίο αντιστοιχεί στο παραπάνω αρχείο.

6.4 Περιγραφή του πακέτου gui

6.4.1 Η κλάση PetriPanel

1. Σκοπός: η κλάση PetriPanel αποτελεί τον καμβά στον οποίον ο χρήστης σχεδιάζει το FPN.

2. Πεδία της κλάσης PetriPanel

Τα πεδία της κλάσης PetriPanel παρουσιάζονται παρακάτω στον Πίνακα 6.12. Για κάθε πεδίο παρουσιάζεται το όνομά του, ο τύπος του καθώς και μία σύντομη περιγραφή της χρησιμότητάς του.

Όνομα Πεδίου	Τύπος	Περιγραφή
petriNet	PetriNet	Το FPN το οποίο σχεδιάζεται.
parentWindow	MainWindow	Αναφορά στο JFrame το οποίο περιέχει το PetriPanel.
simulationOn	boolean	Flag το οποίο δείχνει αν η προσομοίωση είναι ενεργή.
changed	boolean	Flag το οποίο δείχνει αν πραγματοποιήθηκαν αλλαγές στο FPN.
saved	boolean	Flag το οποίο δείχνει αν το FPN έχει αποθηκευθεί.

Πίνακας 6.12: τα πεδία της κλάσης gui.PetriPanel.

3. Μέθοδοι της κλάσης PetriPanel

Οι μέθοδοι της κλάσης PetriPanel παρουσιάζονται παρακάτω στον Πίνακα 6.13. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	Περιγραφή
public PetriPanel(MainWindow parentWindow)	Constructor
public MainWindow getParentWindow()	Gette ργια το πεδίο parentWindow.
public PetriNet getPetriNet()	Getter για το πεδίο petriNet.
public void setPetriNet(PetriNet petriNet)	Setter για το πεδίο petriNet.
public boolean isSimulationOn()	Getter γιατοπεδίοsimulationOn.
public void setSimulationOn(boolean simulationOn)	Setter γιατοπεδίοsimulationOn.
public boolean hasChanged()	Getter για το πεδίο changed.
public void setChanged(boolean changed)	Setter για το πεδίο changed.
public boolean isSaved()	Getter για το πεδίο saved.
public void setSaved(boolean saved)	Setter για το πεδίο saved.
public void paintComponent(Graphics g)	Ζωγραφίζει το περιεχόμενο του PetriPanel, καλώντας την μέθοδο paintBackGround αλλά και την μέθοδο draw του PetriNet.
public void paintBackGround(Graphics g)	Ζωγραφίζει το background του PetriPanel.
public int stepSimulation()	Καλεί την μέθοδο stepSimulation() του PetriNet.

public void resetPetriNet()	Καλεί την μέθοδο reset() του PetriNet.

Πίνακας 6.13: οι μέθοδοι της κλάσης gui.PetriPanel.

6.4.2 Η κλάση PetriPanelMouseListener

1. Σκοπός: η κλάση PetriPanelMouseListener διαχειρίζεται τα γεγονότα κατά τα οποία ο χρήστης επιλέγει κάτι με το ποντίκι μέσα στο PetriPanel.Η κλάση υλοποιεί το interfacejava.awt.event.MouseListener.

2. Πεδία της κλάσης PetriPanelMouseListener

Τα πεδία της κλάσης PetriPanelMouseListener παρουσιάζονται παρακάτω στον Πίνακα 6.14. Για κάθε πεδίο παρουσιάζεται το όνομά του, ο τύπος του καθώς και μία σύντομη περιγραφή της χρησιμότητάς του.

Όνομα Πεδίου	Τύπος	Περιγραφή
placeCreationOn	boolean	Flag το οποίο αποθηκεύει την πληροφορία για το αν μία θέση έχει επιλεγεί.
transitionCreationOn	boolean	Flag το οποίο αποθηκεύει την πληροφορία για το αν μία μετάβαση έχει επιλεγεί.
arcCreationOn	boolean	Flag το οποίο αποθηκεύει την πληροφορία για το αν μία ακμή έχει επιλεγεί.
petriPanel	PetriPanel	Αναφορά προς το PetriPanel.
petriNet	PetriNet	Αναφορά προς το FPN.
startingPoint	Point	Σε περίπτωση που ο χρήστης αρχίζει να σύρει το ποντίκι, τότε θα πρέπει να γνωρίζουμε το σημείο από όπου το γεγονός αυτό αρχίζει.
endingPoint	Point	Σε περίπτωση που ο χρήστης αρχίζει να σύρει το ποντίκι, τότε θα πρέπει να γνωρίζουμε το σημείο από όπου το γεγονός αυτό τερματίζει.

Πίνακας 6.14: τα πεδία της κλάσης gui.PetriPanelMouseListener.

3. Μέθοδοι της κλάσης PetriPanelMouseListener

Οι μέθοδοι της κλάσης PetriPanelMouseListener παρουσιάζονται παρακάτω στον Πίνακα 6.15. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	Περιγραφή
public PetriPanelMouseListener(PetriPanel panel)	Constructor
public void setPetriNet(PetriNet petriNet)	Getter για το πεδίο petriNet.
public boolean isPlaceCreationOn()	Getter για το πεδίο placeCreationOn.
public void setPlaceCreationOn(boolean placeCreationOn)	Setter για το πεδίο placeCreationOn.
public boolean isTransitionCreationOn()	Getter για το πεδίο transitionCreationOn.
public void setTransitionCreationOn(boolean transitionCreationOn)	Setter για το πεδίο transitionCreationOn.

public boolean isArcCreationOn()	Getter για το πεδίο arcCreationOn.
public void setArcCreationOn(boolean arcCreationOn)	Setter για το πεδίο arcCreationOn.
public void mouseClicked(MouseEvent e)	Εκτελείται όταν ο χρήστης κάνει κλικ στον καμβά (πίεση και αποπίεση).
public void clear()	Ζωγραφίζει εξ'αρχής το PetriNet, καθαρίζοντας τις όποιες επιλογές στοιχείων σε αυτό.
public void mousePressed(MouseEvent e)	Εκτελείται όταν ο χρήστης κάνει κλικ στον καμβά και κρατά το πλήκτρο του ποντικού πατημένο.
public void mouseReleased(MouseEvent e)	Εκτελείται όταν ο χρήστης έχει κάνει ήδη κρατήσει πατημένο το πλήκτρο του ποντικιού και στην συνέχεια το απελευθερώνει.
public void mouseEntered(MouseEvent e)	Εκτελείται όταν ο δείκτης του ποντικιού βρίσκεται μέσα στα όρια του καμβά. Η μέθοδος αυτή δεν έχει καμία πρακτική χρησιμότητα στο συγκεκριμένο λογισμικό.
public void mouseExited(MouseEvent e)	Εκτελείται όταν ο δείκτης του ποντικιού βγαίνει έξω από τα όρια του καμβά. Η μέθοδος αυτή δεν έχει καμία πρακτική χρησιμότητα στο συγκεκριμένο λογισμικό.

Πίνακας 6.15: οι μέθοδοι της κλάσης gui.PetriPanelMouseListener.

6.4.3 Η κλάση PetriPanelMouseMotionListener

1Σκοπός: η κλάση PetriPanelMouseMotionListener διαχειρίζεταιτα γεγονότα κατά τα οποία ο χρήστης σύρει τον δείκτη του ποντικιού μέσα στο PetriPanel. Η κλάση υλοποιεί το interfacejava.awt.event.MouseMotionListener.

2. Πεδία της κλάσης PetriPanelMouseMotionListener

Τα πεδία της κλάσης PetriPanelMouseListener παρουσιάζονται παρακάτω στον Πίνακα 6.16. Για κάθε πεδίο παρουσιάζεται το όνομά του, ο τύπος του καθώς και μία σύντομη περιγραφή της χρησιμότητάς του.

Όνομα Πεδίου	**Τύπος**	**Περιγραφή**
drawLineOn	boolean	Flag το οποίο αποθηκεύει την πληροφορία για το αν ο χρήστης σύρει το ποντίκι για να ζωγραφίσει μία ακμή ή για να μεταφέρει ένα στοιχείο του FPN από ένα σημείο σε ένα άλλο.
petriPanel	PetriPanel	Αναφορά προς το PetriPanel.
petriNet	PetriNet	Αναφορά προς το FPN.
startingPoint	Point	Σε περίπτωση που ο χρήστης αρχίζει να σύρει το ποντίκι, τότε θα πρέπει να γνωρίζουμε το σημείο από όπου το γεγονός αυτό αρχίζει.

Πίνακας 6.16: τα πεδία της κλάσης gui.PetriPanelMouseMotionListener.

3. Μέθοδοι της κλάσης PetriPanelMouseMotionListener

Οι μέθοδοι της κλάσης PetriPanelMouseMotionListener παρουσιάζονται παρακάτω στον Πίνακα 6.17. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	Περιγραφή
public PetriPanelMouseMotionListener(PetriPanel petriPanel)	Constructor
public boolean isDrawLineOn()	Getter για το πεδίο drawLineOn.
public void setDrawLineOn(boolean drawLineOn)	Setter για το πεδίο drawLineOn.
public Point getStartingPoint()	Getter για το πεδίο startingPoint.
public void setStartingPoint(Point startingPoint)	Setter για το πεδίο startingPoint.
public void setPetriNet(PetriNet petriNet)	Setter για το πεδίο petriNet.
public void mouseDragged(MouseEvent e)	Εκτελείται όταν ο χρήστης σύρει τον δείκτη του ποντικιού ενώ πρώτα έχει πατήσει και έχει κρατήσει το αριστερό πλήκτρο.
public void mouseMoved(MouseEvent e)	Εκτελείται όταν ο χρήστης απλά μετακινεί το ποντίκι μέσα στον καμβά. Η μέθοδος δεν έχει καμία πρακτική σημασία στο εν λόγω λογισμικό.

Πίνακας 6.17: οι μέθοδοι της κλάσης gui.PetriPanelMouseMotionListener.

6.4.4 Η κλάση PetriPanelKeyListener

1. Σκοπός: η κλάση PetriPanelKeyListener διαχειρίζεται τα γεγονότα κατά τα οποία ο χρήστης πληκτρολογεί κάτι μέσα στο PetriPanel. Η κλάση υλοποιεί το interfacejava.awt.event.KeyListener.

2. Πεδίατηςκλάσης PetriPanelKeyListener

Τα πεδία της κλάσης PetriPanelKeyListener παρουσιάζονται παρακάτω στον Πίνακα 6.18. Για κάθε πεδίο παρουσιάζεται το όνομά του, ο τύπος του καθώς και μία σύντομη περιγραφή της χρησιμότητάς του.

Όνομα Πεδίου	Τύπος	Περιγραφή
petriPanel	PetriPanel	Αναφορά προς το PetriPanel.
petriNet	PetriNet	Αναφορά προς το FPN.

Πίνακας 6.18: τα πεδία της κλάσης gui.PetriPanelKeyListener.

3. Μέθοδοι της κλάσης PetriPanelMouseMotionListener

Οι μέθοδοι της κλάσης PetriPanelKeyListener παρουσιάζονται παρακάτω στον Πίνακα 6.19. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	Περιγραφή
public PetriPanelKeyListener(PetriPanel petriPanel)	Constructor
public void setPetriNet(PetriNet petriNet)	Setter για το πεδίο petriNet.
public void keyTyped(KeyEvent e)	Εκτελείται όταν ο χρήστης κρατά πατημένο ένα πλήκτρο. Η μέθοδος αυτή δεν έχει καμία πρακτική σημασία στο συγκεκριμένο λογισμικό.

public void keyPressed(KeyEvent e)	Εκτελείται όταν ο χρήστης παρά ένα πλήκτρο (πίεση και ελευθέρωση). Η μέθοδος ελέγχει αν το πλήκτρο ήταν το Delete. Αν ναι και ο χρήστης έχει ήδη επιλέξει ένα στοιχείο το στοιχείο αυτό διαγράφεται.
public void keyReleased(KeyEvent e)	Εκτελείται όταν ο χρήστης αφήνει ένα πλήκτρο το οποίο είχε πατήσει. Η μέθοδος αυτή δεν έχει καμία πρακτική σημασία στο συγκεκριμένο λογισμικό.

Πίνακας 6.19: ο ιμέθοδοι της κλάσης gui.PetriPanelKeyListener.

6.4.5 Η κλάση FPNFileFlter

1. Σκοπός: η κλάση FPNFileFilter φιλτράρει τα αρχεία τα οποία έχουν κατάληξη .fpn, σε έναν επιλογέα αρχείων. Η κλάση αυτή επεκτείνει την κλάση javax.swing.filechooser.FileFilter.

2. Πεδία της κλάσης FPNFileFilter

Η κλάση FPNFileFilter δεν έχει πεδία.

3. Μέθοδοι της κλάσης FPNFileFilter

Οι μέθοδοι της κλάσης FPNFileFilter παρουσιάζονται παρακάτω στον Πίνακα 6.20. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	**Περιγραφή**
public boolean accept(File f)	Ελέγχει αν ένα αρχείο, το οποίο δέχεται σαν όρισμα, είναι ένα αρχείο .fpn και επιστρέφει true ή false ανάλογα.
public String getDescription()	Επιστρέφει την περιγραφεί (κείμενο) ενός αρχείου .fpn.

Πίνακας 6.20: οι μέθοδοι της κλάσης gui.FPNFileFilter.

6.4.6 Η κλάση FPNFileChooser

1. Σκοπός: η κλάση FPNFileChooser αποτελεί έναν επιλογέα αρχείων για αρχεία τα οποία έχουν κατάληξη .fpn. Η κλάση αυτή επεκτείνει την κλάση javax.swing.JfileChooser ενώ επίσης χρησιμοποιεί και την κλάση gui.FPNFileFilter.

2. Πεδία τη σκλάσης FPNFileChooser

Η κλάσηFPNFileChooser δεν έχει πεδία.

3. Μέθοδοι της κλάσης FPNFileChooser

Οι μέθοδοι της κλάσης FPNFileChooser παρουσιάζονται παρακάτω στον Πίνακα 6.21. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	**Περιγραφή**
public FPNFileChooser()	Constructor
public void approveSelection()	Στην περίπτωση που το αρχείο ήδη υπάρχει, τότε θα

	πρέπει να εμφανίζεται διάλογος επιβεβαίωσης για την αντικατάστασή του.

Πίνακας 6.21: οι μέθοδοι της κλάσης gui.FPNFileChooser.

6.4.7 Η κλάση PlacePropertiesWindow

1. Σκοπός: η κλάση PlacePropertiesWindow αντιστοιχεί στο παράθυρο του γραφικού περιβάλλοντος στο οποίο ο χρήστης μπορεί να θέσει τις παραμέτρους μίας θέσης. Η κλάση αυτή επεκτείνει την κλάση javax.swing.Jframe.

2. Πεδία της κλάσης PlacePropertiesWindow

Τα πεδία της κλάσης PlacePropertiesWindow παρουσιάζονται παρακάτω στον Πίνακα 6.22. Για κάθε πεδίο παρουσιάζεται το όνομά του, ο τύπος του καθώς και μία σύντομη περιγραφή της χρησιμότητάς του.

Όνομα Πεδίου	**Τύπος**	**Περιγραφή**
place	Place	Η θέση για την οποία θέτουμε τις παραμέτρους.
parent	MainWindow	Αναφορά στο γονικό παράθυρο του γραφικού περιβάλλοντος.

Πίνακας 6.22: τα πεδία της κλάσης gui.PlacePropertiesWindow.

3. Μέθοδοι της κλάσης PlacePropertiesWindow

Οι μέθοδοι της κλάσης PlacePropertiesWindow παρουσιάζονται παρακάτω στον Πίνακα 6.23. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	**Περιγραφή**
public PlacePropertiesWindow(Place place, MainWindow parent)	Constructor
public void setInitialData()	Αρχικοποιεί τις πληροφορίες που παρουσιάζονται στο παράθυρο ανάλογα με τα δεδομένα της θέσης.
private void initComponents()	Παράγεται αυτόματα από το IDENetBeans.
private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης πατά το κουμπί cancel.
private void saveButtonActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης πατά το κουμπί save.
private void formWindowClosing(java.awt.event.WindowEvent evt)	Εκτελείται όταν ο χρήστης κλείνει το παράθυρο.

Πίνακας 6.23: οιμέθοδο της κλάσης gui.PlacePropertiesWindow.

6.4.8 Η κλάσηTransitionPropertiesWindow

1. Σκοπός: η κλάσηTransitionPropertiesWindow αντιστοιχεί στο παράθυρο του γραφικού περιβάλλοντος στο οποίο ο χρήστης μπορεί να θέσει τις παραμέτρους μίας μετάβασης. Η κλάση αυτή επεκτείνει την κλάση javax.swing.Jframe.

2. Πεδία της κλάσης TransitionPropertiesWindow

Τα πεδία της κλάσης TransitionPropertiesWindow παρουσιάζονται παρακάτω στον Πίνακα 6.24. Για κάθε πεδίο παρουσιάζεται το όνομά του, ο τύπος του καθώς και μία σύντομη περιγραφή της χρησιμότητάς του.

Όνομα Πεδίου	Τύπος	Περιγραφή
transition	Transition	Η μετάβαση για την οποία θέτουμε τις παραμέτρους.
parent	MainWindow	Αναφορά στο γονικό παράθυρο του γραφικού περιβάλλοντος.

Πίνακας 6.24: τα πεδία της κλάσης gui.TransitionPropertiesWindow.

3. Μέθοδοι της κλάσης TransitionPropertiesWindow

Οι μέθοδοι της κλάσης TransitionPropertiesWindow παρουσιάζονται παρακάτω στον Πίνακα 6.25. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	Περιγραφή
public TransitionPropertiesWindow(Transition transition, MainWindow parent)	Constructor
public void setInitialData()	Αρχικοποιεί τις πληροφορίες που παρουσιάζονται στο παράθυρο ανάλογα με τα δεδομένα της μετάβασης.
private void initComponents()	Παράγεται αυτόματα από το IDENetBeans.
private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης πατά το κουμπί cancel.
private void saveButtonActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης πατά το κουμπί save.
private void formWindowClosing(java.awt.event.WindowEvent evt)	Εκτελείται όταν ο χρήστης κλείνει το παράθυρο.

Πίνακας 6.25: οιμέθοδο της κλάσηςgui.TransitionPropertiesWindow.

6.4.9 Η κλάση ArcPropertiesWindow

1.Σκοπός: η κλάση ArcPropertiesWindow αντιστοιχεί στο παράθυρο του γραφικού περιβάλλοντος στο οποίο ο χρήστης μπορεί να θέσει τις παραμέτρους μίας ακμής. Η κλάση αυτή επεκτείνει την κλάση javax.swing.Jframe.

2. Πεδία της κλάσης ArcPropertiesWindow

Τα πεδία της κλάσης ArcPropertiesWindow παρουσιάζονται παρακάτω στον Πίνακα 6.26. Για κάθε πεδίο παρουσιάζεται το όνομά του, ο τύπος του καθώς και μία σύντομη περιγραφή της χρησιμότητάς του.

Όνομα Πεδίου	Τύπος	Περιγραφή
arc	Arc	Η ακμή για την οποία θέτουμε τις παραμέτρους.
parent	MainWindow	Αναφορά στο γονικό παράθυρο του γραφικού περιβάλλοντος.

Πίνακας 6.26: τα πεδία της κλάσης gui.ArcPropertiesWindow.

3. Μέθοδοι της κλάσης ArcPropertiesWindow

Οι μέθοδοι της κλάσης ArcPropertiesWindow παρουσιάζονται παρακάτω στον Πίνακα 6.27. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	Περιγραφή
public ArcPropertiesWindow(Arc arc, MainWindow parent)	Constructor
private void initComponents()	Παράγεται αυτόματα από το IDENetBeans.
private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης πατά το κουμπί cancel.
private void saveButtonActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης πατά το κουμπί save.
private void formWindowClosing(java.awt.event.WindowEvent evt)	Εκτελείται όταν ο χρήστης κλείνει το παράθυρο.

Πίνακας 6.27: οι μέθοδο ιτης κλάσης gui.ArcPropertiesWindow.

6.4.10 Η κλάση MainWindow

1. Σκοπός:η κλάση MainWindow αντιστοιχεί στο κύριο παράθυρο του γραφικού περιβάλλοντος, το οποίο περιλαμβάνει τον καμβά σχεδίασης καθώς και όλες τις επιλογές σχεδίασης και διαχείρισης των αρχείων. Η κλάση αυτή επεκτείνει την κλάση javax.swing.Jframe.

2. Πεδία της κλάσης MainWindow

Τα πεδία της κλάσης MainWindow παρουσιάζονται παρακάτω στον Πίνακα 6.28. Για κάθε πεδίο παρουσιάζεται το όνομά του, ο τύπος του καθώς και μία σύντομη περιγραφή της χρησιμότητάς του.

Όνομα Πεδίου	Τύπος	Περιγραφή
placeCreationOn	boolean	Flag to οποίο αποθηκεύει την πληροφορία για το αν ο χρήστης σχεδιάζει μία θέση.
transitionCreationOn	boolean	Flag to οποίο αποθηκεύει την πληροφορία για το αν ο χρήστης σχεδιάζει μία μετάβαση.
arcCreationOn	boolean	Flag to οποίο αποθηκεύει την πληροφορία για το αν ο χρήστης σχεδιάζει μία ακμή.
simulationOn	boolean	Flagto οποίο αποθηκεύει την πληροφορία για το αν η προσομοίωση είναι ενεργή.

timer	javax.swing.Timer	Timero οποίος προσθέτει καθυστέρηση ανάμεσα στα βήματα της προσομοίωσης.
DEFAULT_FILE_NAME	final String	Σταθερά η οποία αποθηκεύει το πρόθεμα (prefix) κάθε νέου αρχείου.
filename	String	Το όνομα του αρχείου .fpn.
fpnFile	java.io.File	Αναφορά προς το αρχείο .fpn.
fileCounter	int	Αύξων αριθμός για τον αριθμό του κάθε νέου αρχείου που δημιουργείται.
defaultBorder	javax.swing.border.Border	Ο border σχεδίασης ενός κουμπιού στην περίπτωση που αυτό είναι μη επιλεγμένο.

Πίνακας 6.28: τα πεδία της κλάσης gui.MainWindow.

3. Μέθοδοι της κλάσης MainWindow

Οι μέθοδοι της κλάσης MainWindow παρουσιάζονται παρακάτω στον Πίνακα 6.27. Για κάθε μέθοδο παρουσιάζεται η πλήρης υπογραφή της καθώς και μια σύντομη περιγραφή της λειτουργίας της.

Υπογραφή Μεθόδου	**Περιγραφή**
public MainWindow()	Constructor
public void updateTitle()	Ανανεώνει τον τίτλο του παραθύρου.
private void initComponents()	Παράγεται αυτόματα από το IDENetBeans.
public void unpressAll()	Απενεργοποιεί τις επιλογές όλων των κουμπιών.
private void placeButtonActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης πατά το πλήκτρο δημιουργίας μίας θέσης.
private void transitionButtonActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης πατά το πλήκτρο δημιουργίας μίας μετάβασης.
private void arcButtonActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης πατά το πλήκτρο δημιουργίας μίας ακμής.
private void cursorButtonActionPerformed(java.awt.event.ActionEvent evt)	εκτελείται όταν ο χρήστης πατά το πλήκτρο επιλογής ενός στοιχείου.
private void simulationButtonActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης πατά το πλήκτρο της προσομοίωσης.
private void stepButtonActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης πατά το πλήκτρο εκτέλεσης ενός βήματος της προσομοίωσης.
private void playButtonActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης πατά το πλήκτρο εκτέλεσης όλης της προσομοίωσης συνολικά.
private void resetButtonActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης πατά το πλήκτρο reset, έτσι ώστε τοFPN να επανέλθει στην αρχική του κατάσταση.
private void	Εκτελείται όταν ο χρήστης πατά το πλήκτρο

pauseButtonActionPerformed(java.awt.event.ActionEvent evt)	pause.
private void newFileMenuItemActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης επιλέγει την δημιουργία ενός νέου αρχείου.
private void saveMenuItemActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης επιλέγει την αποθήκευση του τρέχοντος αρχείου.
private void openFileMenuItemActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης επιλέγει το άνοιγμα ενός αρχείου.
private void saveAsMenuItemActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης επιλέγει την αποθήκευση του τρέχοντος αρχείου ως...
private void formWindowClosing(java.awt.event.WindowEvent evt)	Εκτελείται όταν ο χρήστης κλείσει το παράθυρο.
private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt)	Εκτελείται όταν ο χρήστης επιλέγει την επιλογή εξόδου από το menu.
public void refresh()	Ανανεώνει τον καμβά.
private void readFileName()	Εξάγει το όνομα του αρχείου από το απόλυτο μονοπάτι αυτού.

Πίνακας 6.29: οι μέθοδοι της κλάσης gui.MainWindow.

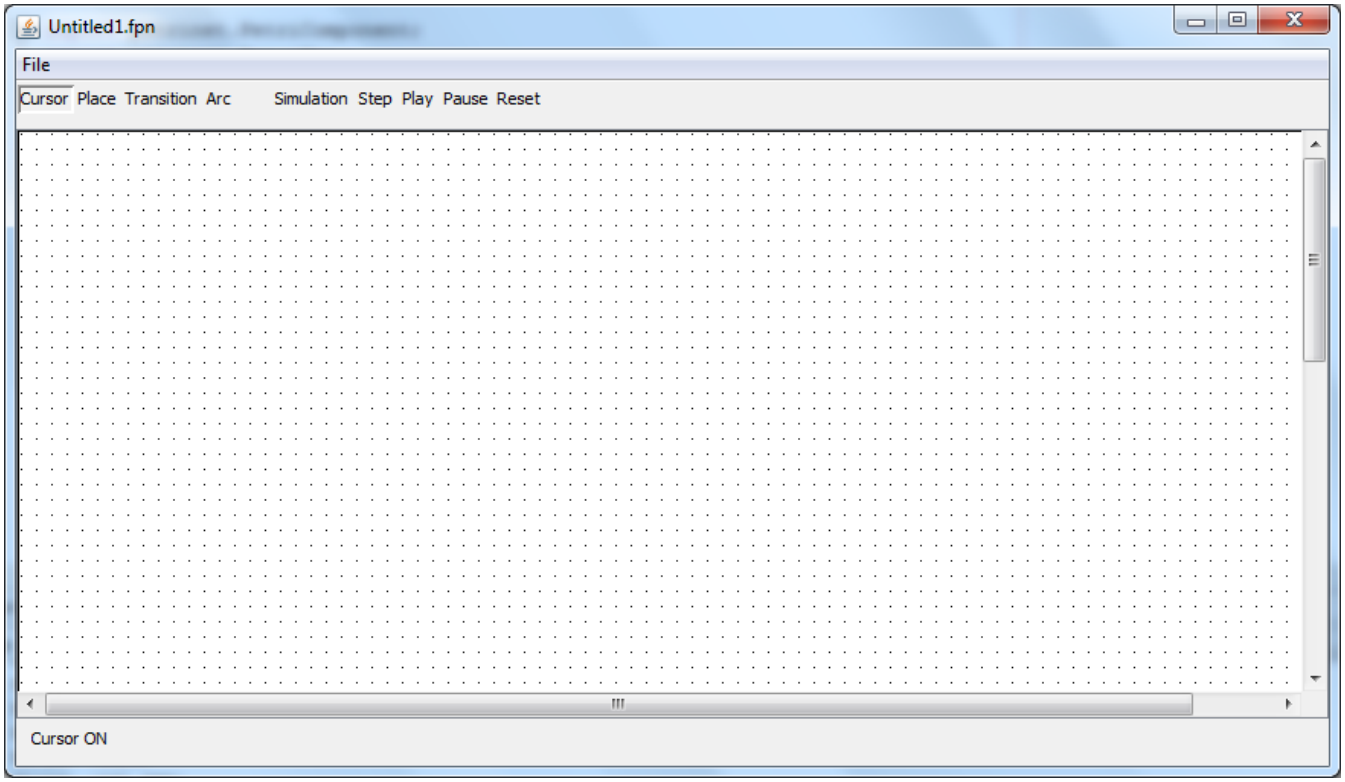
ΚΕΦΑΛΑΙΟ 7 – Περιγραφή της εφαρμογής PetriNetSim

7.1 – Εισαγωγή

Στο εν λόγω κεφάλαιο θα περιγράψουμε την δομή και την χρήση της εφαρμογήςPetriNetSim. Οι λειτουργίες της εφαρμογής θα δοθούν με την χρήση εικόνων. Για κάθε εικόνα ακολουθεί μία σύντομη περιγραφή.

7.2 Κεντρικό παράθυρο της εφαρμογής

Το κεντρικό παράθυρο της εφαρμογής παρουσιάζεται στην Εικόνα 7.1.



Εικόνα 7.1: το κεντρικό παράθυρο της εφαρμογής.

Στην παραπάνω εικόνα μπορούμε να δούμε ότι εμφανίζονται τα εξής κουμπιά:

- Cursor
- Place
- Transition

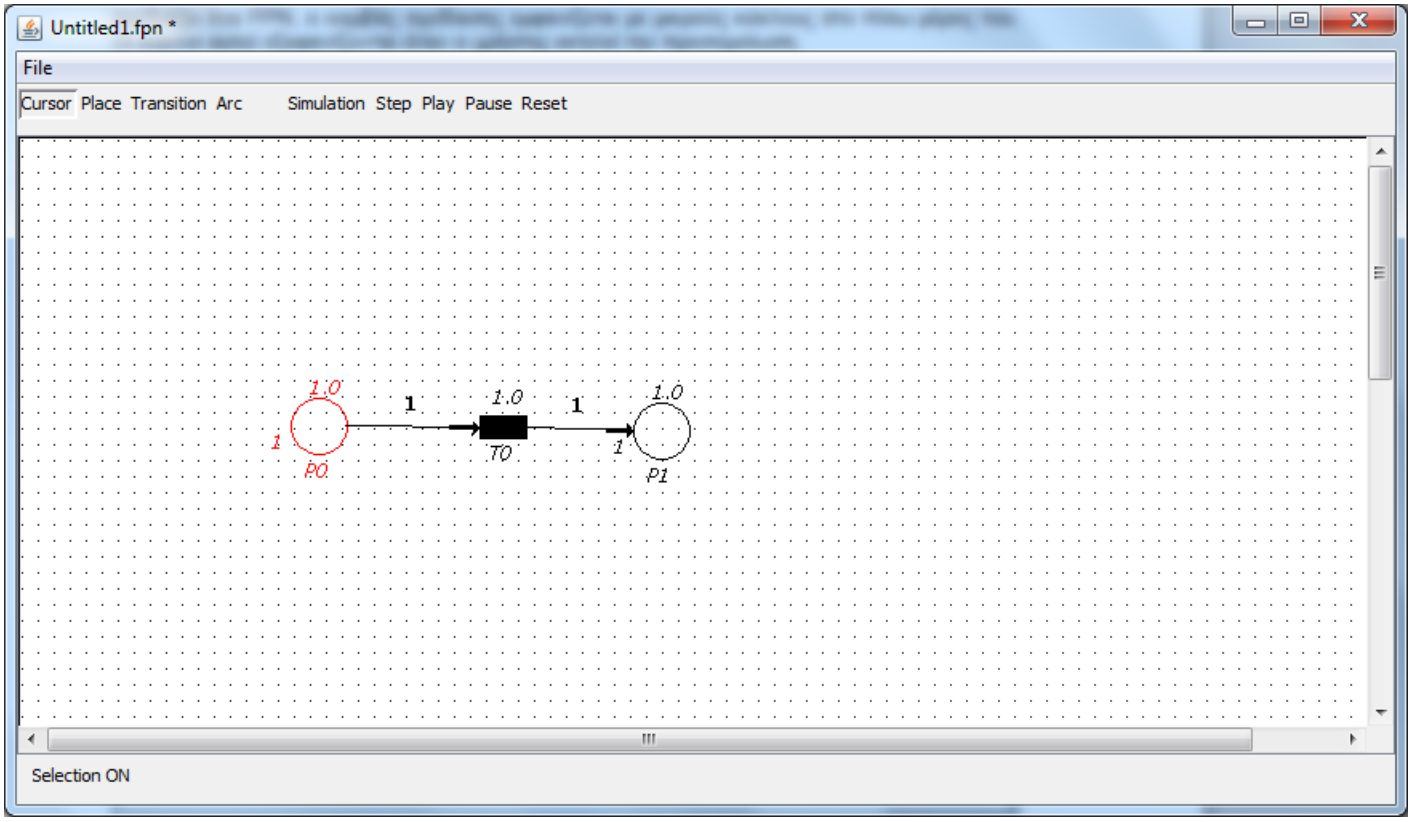
- Arc
- Simulation
- Step
- Play
- Pause
- Reset

καθώς ένα μενού επιλογών (File). Η χρήση του καθενός από τα παραπάνω περιγράφεται στην συνέχεια. Η σχεδίαση του FPN λαμβάνει χώρα μέσα στον καμβά σχεδίασης. Όταν ο χρήστης σχεδιάζει ένα FPN, ο καμβάς σχεδίασης εμφανίζεται με μικρούς κύκλους στο πίσω μέρος του. Οι κύκλοι αυτοί εξαφανίζονται όταν ο χρήστης εκτελεί την προσομοίωση.

7.3 Χρήση των κουμπιών του κεντρικού παραθύρου

7.3.1 Κουμπί Cursor

Το κουμπί αυτό χρησιμοποιείται για την επιλογή στοιχείων του FPN (θέσεις, μετάβασεις, ακμές). Στην περίπτωση που ένα στοιχείο επιλεγεί, τότε το χρώμα του αλλάζει σε κόκκινο (Εικόνα 7.2).



Εικόνα 7.2: επιλογή μίας θέσης.

7.3.2 Κουμπί Place

Το κουμπί αυτό χρησιμοποιείται για την δημιουργία μίας θέσης. Ο χρήστης επιλέγοντας το κουμπί Place έχει στην συνέχεια να δημιουργήσει μία θέση σε οποιοδήποτε σημείο του καμβά. Κάθε νέα θέση, παίρνει αυτόματα την ονομασία P [αύξων ακέραιος αριθμός], μέγιστη

χωρητικότητα ίση με 1, τρέχουσα χωρητικότητα ίση με 0 και βαθμό βεβαιότητας ίσο με 1.0. Ο χρήστης έχει την δυνατότητα στην συνέχεια να μεταβάλλει τις παραπάνω τιμές.

7.3.3 Κουμπί Transition

Το κουμπί αυτό χρησιμοποιείται για την δημιουργία μίας μετάβασης. Ο χρήστης επιλέγοντας το κουμπί Transition έχει στην συνέχεια να δημιουργήσει μία μετάβαση σε οποιοδήποτε σημείο του καμβά. Κάθε νέα θέση, παίρνει αυτόματα την ονομασία T [αύξων ακέραιος αριθμός] και κατώφλι ίσο με 1.0. Ο χρήστης έχει την δυνατότητα στην συνέχεια να μεταβάλλει τις παραπάνω τιμές.

7.3.4 Κουμπί Arc

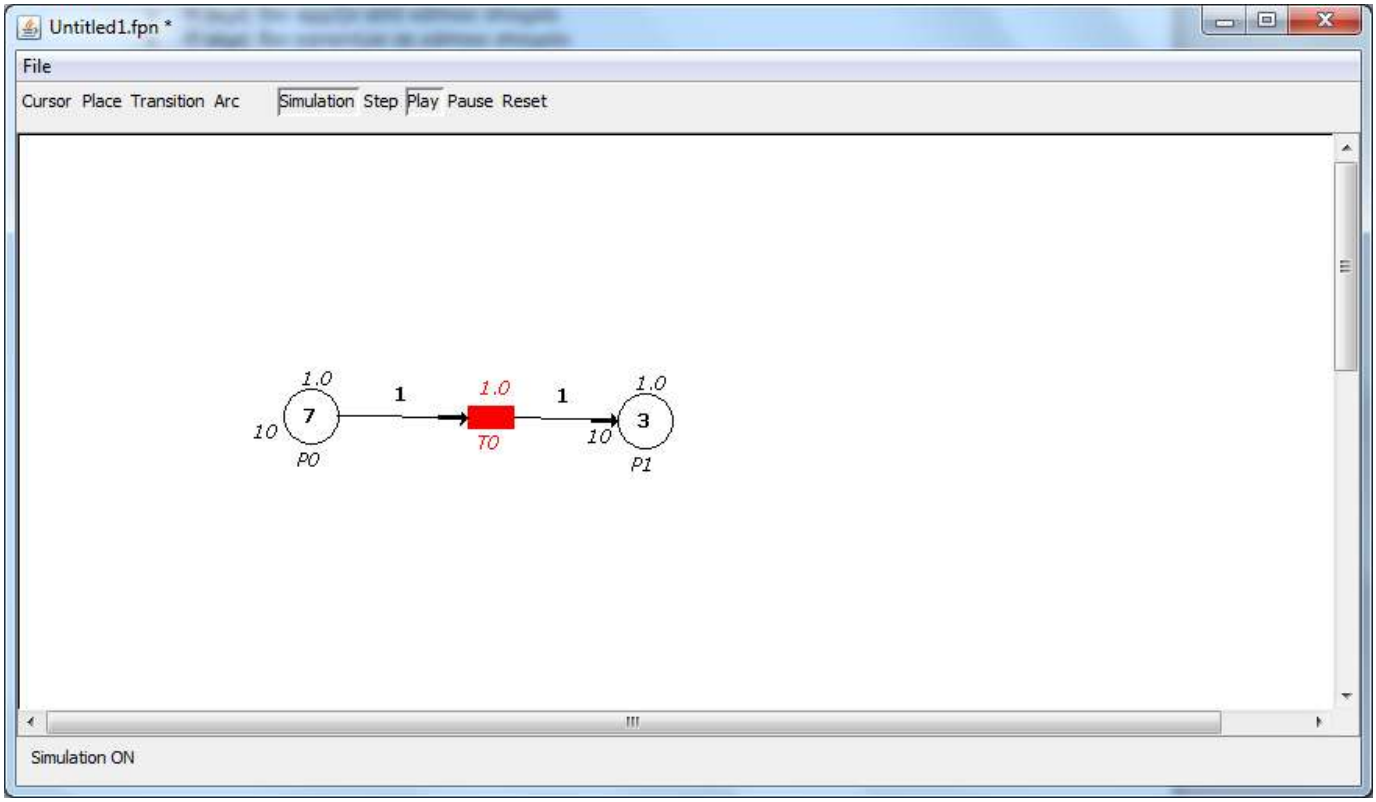
Το κουμπί αυτό χρησιμοποιείται για την δημιουργία μίας θέσης. Ο χρήστης επιλέγοντας το κουμπί Arc έχει στην συνέχεια να δημιουργήσει μία θέση μεταξύ δύο οποιωνδήποτε διαφορετικών στοιχείων του FPN. Για να το κάνει αυτό, επιλέγει το πρώτο στοιχείο με το αριστερό πλήκτρο του ποντικιού και κρατώντας πατημένο το πλήκτρο το μεταφέρει στο δεύτερο στοιχείο. Η δημιουργία μιας ακμής μπορεί να αποτύχει για κάποιον από τους εξής λόγους:

- Η ακμή δεν αρχίζει από κάποιο στοιχείο.
- Η ακμή δεν καταλήγει σε κάποιο στοιχείο.
- Η ακμή συνδέει στοιχεία διαφορετικού τύπου.

Σε κάθε περίπτωση η εφαρμογή εμφανίζει ανάλογο μήνυμα. Κάθε νέα ακμή παίρνει αυτόματα ως βάρος την τιμή 1. Στην συνέχεια ο χρήστης έχει την δυνατότητα να μεταβάλλει την τιμή αυτή.

7.3.5 Κουμπί Simulation

Το κουμπί αυτό χρησιμοποιείται για την έναρξη αλλά και την λήξη της προσομοίωσης. Όταν η προσομοίωση αρχίζει να εκτελείται, τότε το φόντο του καμβά αλλάζει σε άσπρο χρώμα (Εικόνα 7.3) και ο χρήστης δεν έχει την δυνατότητα οποιαδήποτε αλλαγής στο FPN. Σε οποιοδήποτε βήμα της προσομοίωσης, ο χρήστης μπορεί να την τερματίσει, πατώντας και πάλι το κουμπί Simulation.



Εικόνα 7.3: η εικόνα της εφαρμογής κατά την προσομοίωση.

7.3.6 Τοκουμπί Step

Ο χρήστης έχει την δυνατότητα επιλογής του κουμπιού Step μόνο κατά την διάρκεια μιας προσομοίωσης. Το κουμπί Step μεταβιβάζει το FPN στην επόμενη κατάστασή του, πυροδοτώντας τις ικανοποιήσιμες μεταβάσεις, σύμφωνα πάντα με τους κανόνες πυροδότησης.

7.3.7 Το κουμπί Play

Ο χρήστης έχει την δυνατότητα επιλογής του κουμπιού Play μόνο κατά την διάρκεια μιας προσομοίωσης. Το κουμπί Play μεταβιβάζει το FPN σε κάθε επόμενη κατάσταση είτε μέχρι ο χρήστης να πατήσει το πλήκτρο Pause είτε μέχρι να τερματίσει την προσομοίωση.

7.3.8 Το κουμπί Pause

Ο χρήστης έχει την δυνατότητα επιλογής του κουμπιού Pause μόνο εφόσον έχει προηγουμένως επιλέξει το κουμπί Play. Το κουμπί Pause σταματά την λειτουργία του κουμπιού Play, χωρίς όμως να τερματίζει την προσομοίωση.

7.3.9 Το κουμπί Reset

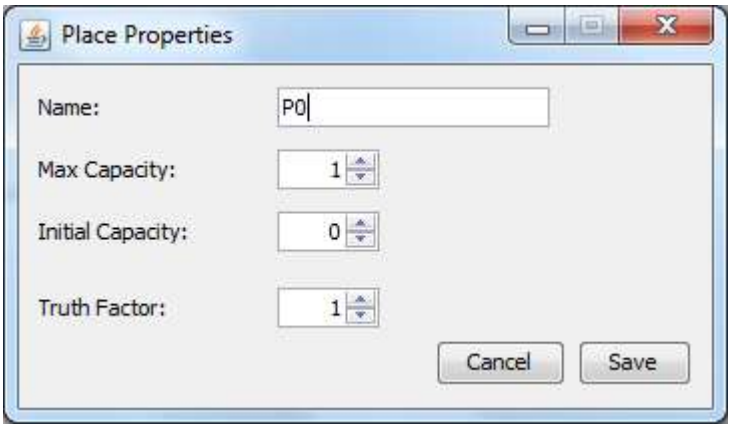
Με το κουμπί αυτό, το FPN επανέρχεται στην αρχική του κατάσταση. Πιο συγκεκριμένα, αυτό σημαίνει ότι κάθε θέση παίρνει ως τρέχουσα χωρητικότητα την αρχική της. Ο χρήστης μπορεί να

πατήσει το πλήκτρο Reset ανεξαρτήτως αν η προσομοίωση είναι ενεργή ή όχι, αρκεί να μην έχει προηγουμένως πατηθεί το πλήκτρο Play.

7.4 Αλλαγή στις τιμές των χαρακτηριστικών των στοιχείων του FPN

Όσο το κουμπί Cursor είναι επιλεγμένο, ο χρήστης μπορεί να επιλέξει οποιοδήποτε στοιχείο του FPNκαι να κάνει διπλό κλικ πάνω του. Ανάλογα με το είδος του στοιχείου (θέση, μετάβαση ή ακμή) εμφανίζεται ανάλογο παράθυρο.

7.4.1 Αλλαγή των χαρακτηριστικών μιας θέσης

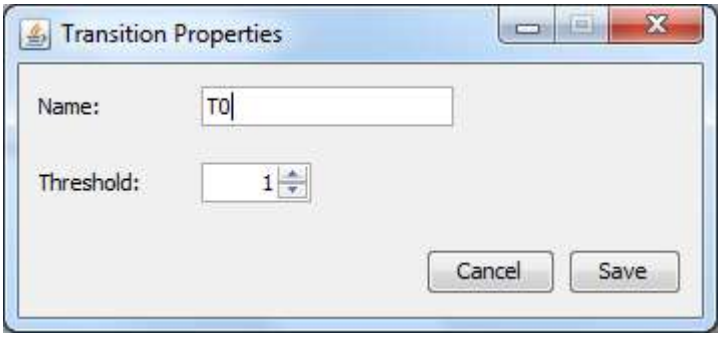


Εικόνα 7.4: παράθυρο αλλαγής των χαρακτηριστικών μιας θέσης.

Ο χρήστης μπορεί να αλλάξει τα χαρακτηριστικά μιας θέσης μέσω του παραθύρου που εμφανίζεται στην Εικόνα 7.4. Πιο συγκεκριμένα, μπορεί να αλλάξει το όνομα της θέσης, την μέγιστη χωρητικότητα, την αρχική χωρητικότητα και τον βαθμό αληθείας της.

7.4.2 Αλλαγή των χαρακτηριστικών μιας μετάβασης

Ο χρήστης μπορεί να αλλάξει τα χαρακτηριστικά μιας μετάβασης μέσω του παραθύρου που εμφανίζεται στην Εικόνα 7.5. Πιο συγκεκριμένα, μπορεί να αλλάξει το όνομα της μετάβασης καθώς και το κατώφλι της.



Εικόνα 7.5: παράθυρο αλλαγής των χαρακτηριστικών μιας μετάβασης.

7.4.3 Αλλαγή των χαρακτηριστικών μιας ακμής

Ο χρήστης μπορεί να αλλάξει τα χαρακτηριστικά μιας ακμής μέσω του παραθύρου που εμφανίζεται στην Εικόνα 7.6. Πιο συγκεκριμένα, μπορεί να αλλάξει την τιμή του βάρους της ακμής.

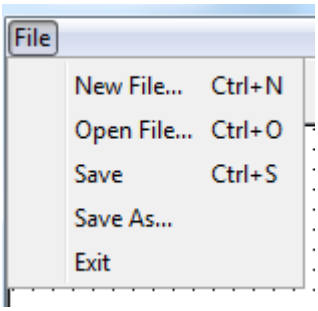
Εικόνα 7.6: παράθυρο αλλαγής των χαρακτηριστικών μιας ακμής.

7.5 Επιλογές αρχείων

Οι επιλογές όσον αφορά τον χειρισμό των αρχείων .fpn υπάρχουν στο μενού με την ονομασία File. Πιο συγκεκριμένα, οι επιλογές είναι οι παρακάτω:

- **NewFile…**: δημιουργία ενός νέου αρχείου
- **OpenFile…**: άνοιγμα ενός ήδη υπάρχοντος αρχείου
- **Save**: αποθήκευση του τρέχοντος αρχείου
- **SaveAs…**: αποθήκευση του τρέχοντος αρχείου ως...
- **Exit**: έξοδος από το πρόγραμμα

Το μενού επιλογών παρουσιάζεται στην Εικόνα 7.7.



Εικόνα 7.7: το μενού επιλογών με τις επιλογές για τα αρχεία.

ΚΕΦΑΛΑΙΟ 8 – Συμπεράσματα

Στην παρούσα εργασία καταδείξαμε την χρησιμότητα των δικτύων Petri στην μοντελοποίηση ασαφών συστημάτων. Πιο συγκεκριμένα, τα συμπεράσματα τα οποία μπορούν να εξαχθούν από την συνολική έρευνα είναι τα εξής:

1. Τα ασαφή συστήματα, τα οποία στηρίζονται στην ασαφή λογική, είναι κατάλληλα για την μοντελοποίηση δυναμικών συστημάτων του πραγματικού κόσμου.
2. Ένα ασαφές σύστημα μπορεί να αναπαρασταθεί με την χρήση ασαφών κανόνων παραγωγής.
3. Οι ασαφείς κανόνες παραγωγής εκτός από τις συνθήκες και τα συμπεράσματά τους περιλαμβάνουν και τους λεγόμενους βαθμούς βεβαιότητας ή βαθμούς αληθείας, οι οποίοι αντιστοιχούν στον βαθμό συμμετοχής στο αντίστοιχο ασαφές σύνολο της συνθήκης ή του συμπεράσματος.
4. Τα δίκτυα Petri αποτελούν ένα καλό εργαλείο για την μοντελοποίηση δυναμικών συστημάτων, αφού διαθέτουν δυνατότητες αναπαράστασης τόσο ντετερμινιστικών όσο και μη ντετερμινιστικών διαδικασιών.
5. Μία παραλλαγή των δικτύων Petri, τα οποία ονομάζονται ασαφή δίκτυα Petri (FuzzyPetriNets – FPN) είναι σε θέση να αναπαραστήσουν με ισοδύναμο τρόπο ασαφείς κανόνες παραγωγής αλλά και να προσομοιώσουν την εκτέλεσή τους. Ως συνέπεια, ένα οποιοδήποτε ασαφές σύστημα το οποίο λειτουργεί με την χρήση κανόνων παραγωγής μπορεί να αναπαρασταθεί με την χρήση ενός FPN.
6. Ως την τρέχουσα χρονική στιγμή συγγραφής της παρούσας εργασίας, δεν υπάρχει διαθέσιμο κάποιο λογισμικό το οποίο να είναι σε θέση να επιτρέπει στον χρήστη να σχεδιάζει FPN και να τα προσομοιώνει. Για τον λόγο αυτό υλοποιήθηκε το PetriNetSim, το οποίο παρέχει αυτές τις δυνατότητες.

ΚΕΦΑΛΑΙΟ 9 – Επεκτάσεις

Με δεδομένο το ότι η έρευνα πάνω στα ασαφή συστήματα εξελίσσεται συνεχώς, η παρούσα πτυχιακή εργασία μπορεί να επεκταθεί σε περισσότερους τομείς από αυτούς που ήδη μελετά στην παρούσα έκδοση.

Μερικές δυνατές επεκτάσεις είναι οι παρακάτω:

- Αυτόματη παραγωγή του FPN με βάση τους ασαφείς κανόνες παραγωγής (Ασαφής Βάση Κανόνων). Αυτό θα μπορούσε να γίνει με δύο τρόπους: είτε το πρόγραμμα να διαβάζει σαν είσοδο ένα αρχείο κειμένου, το οποίο να περιέχει τους ασαφείς κανόνες παραγωγής γραμμένους με κάποια συγκεκριμένη σύνταξη είτε ο χρήστης να έχει την δυνατότητα συγγραφής των κανόνων την εκάστοτε χρονική στιγμή.
- Έλεγχος συνέπειας της Βάσης Κανόνων έτσι ώστε να ελεγχθεί αν υπάρχουν αντιφάσεις, έμμεσες ή άμεσες.
- Κατασκευή του λογισμικού και σε έκδοση για το Διαδίκτυο (web εφαρμογή).
- Προσθήκη δυνατότητας ανάγνωσης και παραγωγής περισσότερων τύπων αρχείων από το λογισμικό (π.χ. αρχεία XML ή JSON).
- Μετάφραση του λογισμικού και σε άλλες γλώσσες. Αυτό φυσικά απαιτεί αλλαγή της δομής του, έτσι ώστε να είναι εφικτή η μετάφραση και από τρίτους.
- Εξαγωγή του σχεδιασμένου FPN σε μορφή εικόνας.
- Συγγραφή εγχειριδίου οδηγιών.
- Συγγραφή Javadoc.

ΠΗΓΕΣ – ΑΝΑΦΟΡΕΣ

[1] Zadeh, L.A. 1965. Fuzzy Sets. Department of Electrical Engineering and Electronics Research Laborary, University of California, Berkeley, California.

[2] Rojas, R. 1996. Neural Networks. Springer-Verlag, Berlin.

[3] Petri, C.A. 1962. Kommunikation mit Automaten. English Translation, 1966: Communication with Automata, Technical Report RADC-TR-65-377, Rome Air Dev. Center, New York.

[4] Wang, J. 2005. Petri Nets for Dynamic Event-Driver System Modeling. Department of Software Engineering, Monmouth University.

[5] Jackson, P. 1998. Introduction To Expert Systems ($3^{η}$ έκδοση), Addison Wesley.

[6] Windhager L. 2013. Modeling of Dynamic Systems with Petri Nets and Fuzzy Logic. Master Thesis.

[7] Virtanen H. E. 2004. A Study in Fuzzy Petri Nets and the Relationship to Fuzzy Logic Programming. Department of Computer Science, Abo Akademi University, Finland.

[8] Giglio D. 2009. Petri Nets. University of Genova, Italy.

[9] Jensen, K. 1981. Colored Petri nets and the invariant-method, Theoretical Computer Science 14: 317-336.

[10] Genrich, J.H., and K. Lautenbach, 1981. System modeling with high-level Petri nets. Theoretical Computer Science 13: 109-136.

[11] Wang, J. 1998. Timed Petri Nets, Theory and Application. Boston: Kluwer Academic Publishers.

[12] Ramchandani, C. 1974. Analysis of Asynchronous Concurrent Systems by Petri Nets. Project MAC, TR-120, M.I.T., Cambridge, MA.

[13] Molloy, M. 1982. Performance analysis using stochastic Petri nets. IEEE Transactions on Computers 31(9): 913-917.

[14] Bause, F., and P. Kritzinger. 2002. Stochastic Petri Nets -- An Introduction to the Theory. Germany: Vieweg Verlag.

[15] Kouzeghar, M., Badamchizadeh, M. A., Khanmohammadi S. 2011. Fuzzy Petri Nets for Human Behavior Verification and Validation, International Journal of Advanced Computer Science and Applications, Vol. 2, No. 12.

[16] He, X., Chu, W. C., Yang, H., Yang, S. J.H. "A New Approach to Verify Rule-Based Systems Using Petri Nets". In: IEEE proceeding of 23th Annual International Computer Software and Applications Conference (COMPSAC'99). (1999) 462-467.

[17] Shen, V. R. L.: Knowledge Representation using High-Level Fuzzy Petri Nets. IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems And Humans. Vol. 36, No. 6. (2006) 1220-1227.

[18] Zhigniew, S. 2013. Generalised fuzzy Petri nets for approximate reasoning in decision support systems. Institute of Computer Sciense, University of Rzeszow, Poland.

[19] Ribaric, S., Pavesic, N., Zadrija, V. 2009. Intersection Search for a Fuzzy Petri Net-Based Knowledge Representation Scheme, J.D. Velásquez et al. (Eds.): KES 2009, Part I, LNAI 5711, pp. 1–10, 2009.

[20] Yousefzadeh, S., Khodabakhshi, S., Pouyan, Ali A., 2013. Designing a Control Vision System for FMS Cells Based on Fuzzy Petri Nets, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 8, pp. 16-22.

[21] S. J. H. Yang, J. J. P. Tsai, and C. Chen. 2003. Fuzzy rule base systems verification using high-level Petri nets. IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 2, pp. 457-473, Mar./Apr. 2003.

[22] NetBeans IDE, http://www.netbeans.org/

ΠΑΡΑΡΤΗΜΑ Α – ΚΩΔΙΚΑΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ PetriNetSim

Κλάση petrinet.PetriComponent

```
// Η κλάση PetriComponent είναι η υπερκλάση των στοιχείων που απαρτίζουν ένα FPN.
// Πιο συγκεκριμένα, τα στοιχεία αυτά είναι οι θέσεις, οι μεταβάσεις και οι ακμές,
// οι οποίες αποτελούν και υποκλάσεις της PetriComponent.
// Η κλάση PetriComponent είναι αφηρημένη (abstract), καθώς περιέχει μεθόδους
// που οι υποκλάσεις της υλοποιούν με διαφορετικό τρόπο.

package petrinet;

import java.awt.Graphics;
import java.awt.Point;
import java.awt.geom.Rectangle2D;

public abstract class PetriComponent {

    // Κάθε PetriComponent έχει έναν μοναδικό ακέραιο αριθμό ως ταυτότητα
    privateintid;
    // Ο επόμενος ακέραιος αριθμός που θα χρησιμοποιηθεί ως ταυτότητα
public static int nextId;
```

```
    // Το όνομα του PetriComponent
    protectedStringname;
    // Flag το οποίο δείχνει αν το εν λόγω PetriComponent έχει επιλεχθεί
protectedbooleanisSelected;
    // Flag το οποίο δείχνει αν το εν λόγω PetriComponent πρέπει να
    // απεικονισθεί άμεσα στο γραφικό περιβάλλον. Είναι χρήσιμο κατά την
    // ανάκτηση ενός PetriComponent από ένα αρχείο.
public static boolean drawOn = true;

// Constructor
    // Κατά την δημιουργία του, το PetriComponent παίρνει αυτόματα τον
    // επόμενο αύξοντα αριθμό ως ταυτότητα.
public PetriComponent() {
        id = ++nextId;
    }

    // Getters/Setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public boolean isSelected() {
        return isSelected;
    }

    public void setIsSelected(boolean isSelected) {
        this.isSelected = isSelected;
    }

    // Αφηρημένη (abstract) μέθοδος την οποία κάθε υποκλάση υλοποιεί με την δικό της τρόπο.
    // Στην μέθοδο αυτή το κάθε στοιχείο ζωγραφίζει τον εαυτό του στο PetriPanel.
public abstract void draw(Graphics g);

// Αφηρημένη (abstract) μέθοδος την οποία κάθε υποκλάση υλοποιεί με την δικό της τρόπο.
    // Στην μέθοδο αυτή το κάθε στοιχείο ελέγχει αν μέσα στα όριά του περιέχεται κάποιο
σημείο.
public abstract boolean contains(Point p);

// Αφηρημένη (abstract) μέθοδος την οποία κάθε υποκλάση υλοποιεί με την δικό της τρόπο.
```

```
    // Η μέθοδος επιστρέφει ένα ορθογώνιο παραλληλόγραμμο το οποίο περιλαμβάνει
    // μέσα στα όριά του το στοιχείο στο γραφικό περιβάλλον.
publicabstractRectangle2DgetBounds();


    // Αφηρημένη (abstract) μέθοδος την οποία κάθε υποκλάση υλοποιεί με την δικό της τρόπο.
    // Η μέθοδος ανανεώνει την θέση ενός στοιχείου στον χώρο, όταν ο χρήστης το
    // μετακινεί με την χρήση του ποντικιού.
public abstract void updatePosition(int x, int y, Graphics g);

}
```

Κλάσηpetrinet.Place

```
// ΗκλάσηPlaceυλοποιείτηνκλάσηPetriComponentκαιαναπαριστά
// μία θέση του δικτύου Petri.

package petrinet;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.Shape;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Rectangle2D;
import java.util.ArrayList;

public class Place extends PetriComponent {

// Η μέγιστη χωρητικότητα της θέσης
privateintmaxCapacity;
    // Η αρχική χωρητικότητα της θέσης
privateintinitialCapacity;
    // Ο αριθμός των τεκμηρίων της θέσης (τρέχουσα χωρητικότητα)
privateinttokensNo;
    // Ο βαθμός βεβαιότητας της θέσης
privatedoubletruthFactor;
    // Το σχήμα της θέσης στο γραφικό περιβάλλον
privateShapeplaceShape;
    // Οι μεταβάσεις στις οποίες η θέση στέλνει εξερχόμενη ακμή
privateArrayList<Transition>outgoingTransitions;
    // Οι μεταβάσεις από τις οποίες η θέση δέχεται εισερχόμενη ακμή
private ArrayList<Transition> incomingTransitions;
    // Αύξων αριθμός δημιουργίας κάθε θέσης
    public static int placeCounter = 0;
    // Το δίκτυο Petri στο οποίο η θέση ανήκει
    private PetriNet petriNet;
// Οι προκαθορισμένες διαστάσεις μίας θέσης στο γραφικό περιβάλλον
private static final int WIDTH = 35;
```

```
    private static final int HEIGHT = 35;

    // Constructor
    public Place(int x, int y, Graphics g) {
        outgoingTransitions = new ArrayList<>();
        incomingTransitions = new ArrayList<>();
        placeShape = new Ellipse2D.Double(x,y,WIDTH,HEIGHT);
        maxCapacity = 1;
        truthFactor = 1.0;
        name = "P" + placeCounter++;
        if (drawOn)
            this.draw(g);
    }

    // Getters/Setters
    public int getMaxCapacity() {
        return maxCapacity;
    }

    public void setMaxCapacity(int maxCapacity) {
        this.maxCapacity = maxCapacity;
    }

    public int getInitialCapacity() {
        return initialCapacity;
    }

    public void setInitialCapacity(int initialCapacity) {
        this.initialCapacity = initialCapacity;
    }

    public int getTokensNo() {
        return tokensNo;
    }

    public void setTokensNo(int tokensNo) {
        this.tokensNo = tokensNo;
    }

    public double getTruthFactor() {
        return truthFactor;
    }

    public void setTruthFactor(double truthFactor) {
        this.truthFactor = truthFactor;
    }

    public Shape getPlaceShape() {
        return placeShape;
    }
```

```
    public void setPlaceShape(Shape placeShape) {
        this.placeShape = placeShape;
    }

    public PetriNet getPetriNet() {
        return petriNet;
    }

    public void setPetriNet(PetriNet petriNet) {
        this.petriNet = petriNet;
    }

    // Μέθοδοι χειρισμού των ArrayLists

    public ArrayList<Transition> getOutgoingTransitions() {
        return outgoingTransitions;
    }

    public ArrayList<Transition> getIncomingTransitions() {
        return incomingTransitions;
    }

    public void addOutgoingTransition(Transition t) {
        outgoingTransitions.add(t);
    }

    public void removeOutgoingTransition(Transition t) {
        outgoingTransitions.remove(t);
    }

    public void addIncomingTransition(Transition t) {
        incomingTransitions.add(t);
    }

    public void removeIncomingTransition(Transition t) {
        incomingTransitions.remove(t);
    }

    // Ζωγραφίζει μία θέση στο γραφικό περιβάλλον.
@Override
    public void draw(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;
        if (isSelected)
            g2d.setColor(Color.RED);
        g2d.draw(placeShape);
        if (tokensNo > 0) {
            Font tokensFont = new Font("Verdana",Font.BOLD,12);
            g2d.setFont(tokensFont);
            int offset;
            if (tokensNo < 10) offset = 6;
            else if (tokensNo < 100) offset = 10;
```

```
            else offset = 13;
            g2d.drawString(Integer.toString(tokensNo),
                    (float) placeShape.getBounds2D().getCenterX()-offset,
                    (float) placeShape.getBounds2D().getCenterY()+4);
        }

        int offset;
        if (maxCapacity < 10) offset = 30;
        else if (maxCapacity < 100) offset = 37;
        else offset = 45;
        Font maxCapacityFont = new Font("Verdana",Font.ITALIC,12);
        g2d.setFont(maxCapacityFont);
        g2d.drawString(Integer.toString(maxCapacity),
                    (float) placeShape.getBounds2D().getCenterX()-offset,
                    (float) placeShape.getBounds2D().getCenterY()+14);

        Font tFactorFont = new Font("Verdana",Font.ITALIC,12);
        g2d.setFont(tFactorFont);
        g2d.drawString(Double.toString(truthFactor),
                    (float) placeShape.getBounds2D().getCenterX()-6,
                    (float) placeShape.getBounds2D().getCenterY()-20);

        Font nameFont = new Font("Verdana",Font.ITALIC,12);
        g2d.setFont(nameFont);
        g2d.drawString(name,
                    (float) placeShape.getBounds2D().getCenterX()-(5+name.length()*2),
                    (float) placeShape.getBounds2D().getCenterY()+32);

        if (isSelected)
            g2d.setColor(Color.BLACK);
}

    // Ελεγχει αν μία θέση περιέχει ένα συγκεκριμένο σημείο,
    // το οποίο δίνεται ως όρισμα.
@Override
    public boolean contains(Point p) {
        return placeShape.getBounds2D().contains(p);
}

    // Επιστρέφει ένα τετράγωνο το οποίο περιλαμβάνει πλήρως την θέση
// μέσα στα όριά του.
    @Override
    public Rectangle2D getBounds() {
        returnplaceShape.getBounds();
    }

    // Αλλάζει την θέση μίας θέσης στον χώρο, όταν ο χρήστης την
    // μετακινεί με την χρήση του ποντικιού.
@Override
    public void updatePosition(int x, int y, Graphics g) {
        placeShape = new Ellipse2D.Double(x, y, WIDTH, HEIGHT);
```

```
    }

}
```

Κλάσηpetrinet.Transition

```
// ΗκλάσηTransitionυλοποιείτηνκλάσηPetriComponentκαιαναπαριστά
// μία μετάβαση του δικτύου Petri.

package petrinet;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.Shape;
import java.awt.geom.Rectangle2D;
import java.util.ArrayList;

public class Transition extends PetriComponent {

    // Το κατώφλι της μετάβασης
    private double threshold;
// Το σχήμα της μετάβασης στο γραφικό περιβάλλον
privateShapetransitionShape;
    // Οι θέσεις στις οποίες η μετάβαση στέλνει εξερχόμενη ακμή
privateArrayList<Place>outgoingPlaces;
    // Οι θέσεις από τις οποίες η μετάβαση δέχεται εισερχόμενη ακμή
privateArrayList<Place>incomingPlaces;
    // Αύξων αριθμός δημιουργίας κάθε μετάβασης
public static int transitionCounter = 0;
    // Το δίκτυο Petri στο οποίο η μετάβαση ανήκει
    privatePetriNetpetriNet;
    // Οι προκαθορισμένες διαστάσεις της μετάβασης στο γραφικό περιβάλλον
private static final int WIDTH = 30;
    private static final int HEIGHT = 15;

    // Constructor
    public Transition(int x, int y, Graphics g) {
        outgoingPlaces = new ArrayList<>();
        incomingPlaces = new ArrayList<>();
        transitionShape = new Rectangle(x,y,WIDTH,HEIGHT);
        threshold = 1.0;
        name = "T" + transitionCounter++;
        if (drawOn)
            this.draw(g);
    }

    // Getters/Setters
```

```
    public double getThreshold() {
        return threshold;
    }

    public void setThreshold(double threshold) {
        this.threshold = threshold;
    }

    public Shape getTransitionShape() {
        return transitionShape;
    }

    public void setTransitionShape(Shape transitionShape) {
        this.transitionShape = transitionShape;
    }

    public ArrayList<Place> getOutgoingPlaces() {
        return outgoingPlaces;
    }

    public ArrayList<Place> getIncomingPlaces() {
        return incomingPlaces;
    }

    public PetriNet getPetriNet() {
        return petriNet;
    }

    public void setPetriNet(PetriNet petriNet) {
        this.petriNet = petriNet;
    }

    // Μέθοδοι χειρισμού των ArrayLists

    public void addOutgoingPlace(Place p) {
        outgoingPlaces.add(p);
    }

    public void removeOutgoingPlace(Place p) {
        outgoingPlaces.remove(p);
    }

    public void addIncomingPlaces(Place p) {
        incomingPlaces.add(p);
    }

    public void removeIncomingPlace(Place p) {
        incomingPlaces.remove(p);
    }

    // Ζωγραφίζει μία μετάβαση στο γραφικό περιβάλλον.
```

```
@Override
    public void draw(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;
        if (isSelected)
            g2d.setColor(Color.RED);
        g2d.fill(transitionShape);

        Font thresholdFont = new Font("Verdana",Font.ITALIC,12);
        g2d.setFont(thresholdFont);
        g2d.drawString(Double.toString(threshold),
                    (float) transitionShape.getBounds2D().getCenterX()-7,
                    (float) transitionShape.getBounds2D().getCenterY()-15);

        Font nameFont = new Font("Verdana",Font.ITALIC,12);
        g2d.setFont(nameFont);
        g2d.drawString(name,
                    (float) transitionShape.getBounds2D().getCenterX()-(5+name.length()*2),
                    (float) transitionShape.getBounds2D().getCenterY()+20);

        if (isSelected)
            g2d.setColor(Color.BLACK);
    }

    // Ελεγχει αν μία μετάβαση περιέχει ένα συγκεκριμένο σημείο,
    // το οποίο δίνεται ως όρισμα.
@Override
    public boolean contains(Point p) {
        returntransitionShape.contains(p);
    }

    // Επιστρέφει ένα τετράγωνο το οποίο περιλαμβάνει πλήρως την μετάβαση
// μέσα στα όριά του.
    @Override
    public Rectangle2D getBounds() {
        returntransitionShape.getBounds();
    }

    // Αλλάζει την θέση μίας μετάβασης στον χώρο, όταν ο χρήστης την
    // μετακινεί με την χρήση του ποντικιού.
@Override
    public void updatePosition(int x, int y, Graphics g) {
        ((Rectangle) transitionShape).setLocation(x,y);
}

    // Πυροδοτεί την μετάβαση.
publicvoidfire() {
        // Ανανέωσε τις τρέχουσες χωρητικότητες των εισερχόμενων θέσεων
for (Place p : incomingPlaces) {
            // Βρες την ακμή σύνδεσης
            for (Arc a : petriNet.getArcs()) {
                if (a.getStartsFrom() instanceof Place &&
```

```
                        a.getStartsFrom() == p &&
                        a.getEndsTo() instanceof Transition &&
                        a.getEndsTo() == this) {
                        // Ανανέωσε τον αριθμό τεκμηρίων της θέσης
int currentCapacity = p.getTokensNo();
                        int newCapacity = currentCapacity - a.getWeight();
                        p.setTokensNo(newCapacity);
break;
                    }
                }
        }
        // Ανανέωσε τις τρέχουσες χωρητικότητες και τους βαθμούς
        // βεβαιότητας των εξερχόμενων θέσεων
        // Αρχικά βρες τον μικρότερο βαθμό βεβαιότητας από τις
// εισερχόμενες θέσεις
        double minTruthFactor = 1.0;
        for (Place p : incomingPlaces) {
             if (p.getTruthFactor() < minTruthFactor)
                 minTruthFactor = p.getTruthFactor();
}
doublenewTruthFactor = threshold * minTruthFactor;
        // Για κάθε θέση με την οποία η μετάβαση συνδέεται με εξερχόμενη ακμή
for (Placep : outgoingPlaces) {
             // Βρες την ακμή σύνδεσης
for (Arc a : petriNet.getArcs()) {
                  if (a.getStartsFrom() instanceof Transition &&
                        a.getStartsFrom() == this &&
                        a.getEndsTo() instanceof Place &&
                        a.getEndsTo() == p) {
                        // Ανανέωσε τον αριθμό τεκμηρίων της θέσης
int currentCapacity = p.getTokensNo();
                        int newCapacity = currentCapacity + a.getWeight();
                        p.setTokensNo(newCapacity);
                        // Ανανέωσε τον βαθμό βεβαιότητας
if (minTruthFactor > p.getTruthFactor())
                             p.setTruthFactor(newTruthFactor);
                        break;
                  }
             }
        }
    }

}
```

Κλάσηpetrinet.Arc

```
// ΗκλάσηTransitionεπεκτείνειτηνκλάσηPetriComponentκαιαναπαριστά
// μία ακμή του δικτύου Petri.

package petrinet;
```

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.Polygon;
import java.awt.Shape;
import java.awt.geom.AffineTransform;
import java.awt.geom.Line2D;
import java.awt.geom.Point2D;
import java.awt.geom.Rectangle2D;

public class Arc extends PetriComponent {

// Το βάρος της ακμής
privateintweight;
    // Η γραμμή στο γραφικό περιβάλλον
privateLine2D.Doubleline;
    // Το βέλος στο γραφικό περιβάλλον
privateShapearrow;
    // Το στοιχείο του δικτύου Petri από το οποίο η ακμή εξέρχεται
privatePetriComponentstartsFrom;
    // Το στοιχείο του δικτύου Petri στο οποίο η ακμή καταλήγει
private PetriComponent endsTo;

    // Constructor
    public Arc(PetriComponent startsFrom, PetriComponent endsTo, Graphics g) {
        this.startsFrom = startsFrom;
        this.endsTo = endsTo;
        weight = 1;
        if (drawOn)
            this.draw(g);
    }

    // Getters/Setters
    public int getWeight() {
        return weight;
    }

    public void setWeight(int weight) {
        this.weight = weight;
    }

    public PetriComponent getStartsFrom() {
        return startsFrom;
    }

    public PetriComponent getEndsTo() {
        returnendsTo;
    }
```

```
    // Ελέγχει το πρόσημο της διαφοράς των x1, x2. Η μέθοδος αυτή είναι
    // απαραίτητη για τον σχεδιασμό της ακμής στο γραφικό περιβάλλον.
private int sign(double x1, double x2) {

        if (x1 < x2)
            return 1;
        else if (x1 > x2)
            return -1;
else
return 0;

    }

    // Ζωγραφίζει ένα βέλος στο γραφικό περιβάλλον
public Shape createArrowShape(Point2D.Double fromPt, Point2D.Double toPt) {
        Polygon arrowPolygon = new Polygon();
        arrowPolygon.addPoint(-6, 1);
        arrowPolygon.addPoint(3, 1);
        arrowPolygon.addPoint(3, 3);
        arrowPolygon.addPoint(6, 0);
        arrowPolygon.addPoint(3, -3);
        arrowPolygon.addPoint(3, -1);
        arrowPolygon.addPoint(-6, -1);

        Point2D.Double midPoint = midpoint(fromPt, toPt);

        double rotate = Math.atan2(toPt.y - fromPt.y, toPt.x - fromPt.x);

        AffineTransform transform = new AffineTransform();
        transform.translate(midPoint.x, midPoint.y);
        double ptDistance = fromPt.distance(toPt);
        double scale = ptDistance / 12.0; // 12 because it's the length of the arrow
polygon.
        transform.scale(scale, scale);
        transform.rotate(rotate);

        return transform.createTransformedShape(arrowPolygon);
    }

    // Επιστρέφει το μέσο της απόστασης δύο σημείων.
    private Point2D.Double midpoint(Point2D.Double p1, Point2D.Double p2) {
        return new Point2D.Double(((p1.x + p2.x) / 2.0),((p1.y + p2.y) / 2.0));
}

    // Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Η εν λόγω μέθοδος
    // ζωγραφίζει μία ακμή στο γραφικό περιβάλλον.
@Override
    public void draw(Graphics g) {

// Σχεδιασμός της γραμμής
Graphics2Dg2d = (Graphics2D) g;
```

```
if (isSelected)
            g2d.setColor(Color.RED);

        Rectangle2D startBounds = startsFrom.getBounds();
        Rectangle2D endBounds = endsTo.getBounds();

        double lineStartX = startBounds.getCenterX();
        double lineStartY = startBounds.getCenterY();
        double lineEndX = endBounds.getCenterX();
        double lineEndY = endBounds.getCenterY();

        Shape startShape;
        Shape endShape;

        if (startsFrom instanceof Place) {
            startShape = ((Place) startsFrom).getPlaceShape();
            endShape = ((Transition) endsTo).getTransitionShape();
        } else {
            startShape = ((Transition) startsFrom).getTransitionShape();
            endShape = ((Place) endsTo).getPlaceShape();
        }

        double x1, y1, x2, y2;
        x1 = startBounds.getCenterX();
        y1 = startBounds.getCenterY();
        x2 = endBounds.getCenterX();
        y2 = endBounds.getCenterY();

        double step = sign(x1,x2) * 0.1;

        boolean foundStart = false;
        boolean foundEnd = false;
        boolean foundArrowStart = false;

        double bx = 0, by = 0;

        for (double i = x1, j = y1; !foundStart || !foundEnd; i += step) {
             if (!foundStart && !startShape.contains(i,j)) {
                 lineStartX = i;
                 lineStartY = j;
                 foundStart = true;
             }
             if (foundStart && endShape.contains(i,j)) {
                 lineEndX = i;
                 lineEndY = j;
                 foundEnd = true;
             } else {
                 double distance = Point2D.distance(i,j,lineEndX,lineEndY);
                 if (!foundArrowStart && distance < 35) {
                     bx = i;
                     by = j;
```

```
                    foundArrowStart = true;
                }
            }
            j = ((y1 - y2)/(x1-x2))*i + (x1*y2)/(x1-x2) - (x2*y1)/(x1-x2);
        }

        line = new Line2D.Double(lineStartX,lineStartY,lineEndX,lineEndY);
        g2d.draw(line);

        Point2D.Double p1 = new Point2D.Double(bx, by);
        Point2D.Double p2 = new Point2D.Double(lineEndX,lineEndY);

// Σχεδιασμός του βέλους
arrow = createArrowShape(p1,p2);

g2d.fill(arrow);

        Font weightFont = new Font("Verdana",Font.BOLD,12);
        g2d.setFont(weightFont);
        g2d.drawString(Integer.toString(weight),
                    (float) line.getBounds2D().getCenterX()-7,
                    (float) line.getBounds2D().getCenterY()-10);

        if (isSelected)
            g2d.setColor(Color.BLACK);

}

// Επανορισμός της μεθόδου της υπερκλάσης PetriComponent.
    // Η εν λόγω μέθοδος ελεγχει αν μία ακμή περιέχει ένα συγκεκριμένο σημείο,
    // το οποίο δίνεται ως όρισμα.
@Override
    public boolean contains(Point p) {
        return line.getBounds().contains(p) || arrow.getBounds().contains(p);
}

    // Επανορισμός της μεθόδου της υπερκλάσης PetriComponent. Η εν λόγω μέθοδος
    // επιστρέφει ένα τετράγωνο το οποίο περιλαμβάνει πλήρως την
    // ακμή μέσα στα όριά του.
@Override
    public Rectangle2D getBounds() {
        return line.getBounds();
}

    // Επανορισμός της μεθόδου της υπερκλάσης PetriComponent.
    // Στην περίπτωση της ακμής αυτή δεν μπορεί να μετακινηθεί,
    // καθώς εξαρτάται από την θέση των στοιχείων που ενώνει.
@Override
    public void updatePosition(int x, int y, Graphics g) {
    }
```

```
}
```

Κλάσηpetrinet.PetriNet

```
// ΗκλάσηPetriNetαναπαριστάέναFPN.

package petrinet;

import java.awt.Graphics;
import java.awt.Point;
import java.util.ArrayList;
import java.util.Random;

public class PetriNet {

// Λίστα με τις θέσεις του δικτύου
privateArrayList<Place>places;
    // Λίστα με τις μεταβάσεις του δικτύου
private ArrayList<Transition> transitions;
    // Λίστα με τις ακμές του δικτύου
    privateArrayList<Arc>arcs;
    // Flag το οποίο αποθηκεύει την πληροφορία για το αν κάποιο στοιχείο
    // του PetriNet έχει επιλεγεί στο γραφικό περιβάλλον
private boolean selectedComponentOn;

    // Constructor
    public PetriNet() {
        places = new ArrayList<>();
        transitions = new ArrayList<>();
        arcs = new ArrayList<>();
    }

    // Getters/Setters
    public ArrayList<Place> getPlaces() {
        return places;
    }

    public void setPlaces(ArrayList<Place> places) {
        this.places = places;
    }

    public ArrayList<Transition> getTransitions() {
        return transitions;
    }

    public ArrayList<Arc> getArcs() {
        return arcs;
    }

    public void setTransitions(ArrayList<Transition> transitions) {
        this.transitions = transitions;
```

```
    }

    public void setSelectedComponentOn() {
        selectedComponentOn = true;
    }

    public boolean hasSelectedComponent() {
        return selectedComponentOn;
    }

    // Μέθοδοι χειρισμού των ArrayLists

    public void addPlace(Place p) {
        places.add(p);
        p.setPetriNet(this);
    }

    public void removePlace(Place p) {
        places.remove(p);
        p.setPetriNet(null);
        // Αφαίρεσε επίσης τυχόν συνδεδεμένες ακμές
        ArrayList<Arc> arcsToRemove = new ArrayList<>();
        for (Arc a : arcs) {
             if ((a.getStartsFrom() instanceof Place &&
                  a.getStartsFrom() == p) ||
                 (a.getEndsTo() instanceof Place &&
                  a.getEndsTo() == p)) {
                 arcsToRemove.add(a);
             }
        }
        for (Arc a : arcsToRemove)
             this.removeArc(a);
        for (Transition t : transitions) {
             t.removeIncomingPlace(p);
             t.removeOutgoingPlace(p);
        }
    }

    public void addTransition(Transition t) {
        transitions.add(t);
        t.setPetriNet(this);
    }

    public void removeTransition(Transition t) {
        transitions.remove(t);
        t.setPetriNet(null);
        // Αφαίρεσε επίσης τυχόν συνδεδεμένες ακμές
        ArrayList<Arc> arcsToRemove = new ArrayList<>();
        for (Arc a : arcs) {
             if ((a.getStartsFrom() instanceof Transition &&
                  a.getStartsFrom() == t) ||
```

```
                (a.getEndsTo() instanceof Transition &&
                 a.getEndsTo() == t)) {
                arcsToRemove.add(a);
            }
        }
        for (Arc a : arcsToRemove)
            this.removeArc(a);
        for (Place p : places) {
            p.removeIncomingTransition(t);
            p.removeOutgoingTransition(t);
        }
    }

    public void addArc(Arc a) {
        arcs.add(a);
    }

    public void removeArc(Arc a) {
        arcs.remove(a);
        if (a.getStartsFrom() instanceof Place) {
            ((Place)         a.getStartsFrom()).getOutgoingTransitions().remove((Transition)
a.getEndsTo());
            ((Transition)                  a.getEndsTo()).getIncomingPlaces().remove((Place)
a.getStartsFrom());
        } else if (a.getStartsFrom() instanceof Transition) {
            ((Transition)              a.getStartsFrom()).getOutgoingPlaces().remove((Place)
a.getEndsTo());
            ((Place)             a.getEndsTo()).getIncomingTransitions().remove((Transition)
a.getStartsFrom());
}
    }

    // Ζωγραφίζει το FPN στο γραφικό περιβάλλον, καλώντας τις επιμέρους
    // μεθόδους draw των στοιχείων που το απαρτίζουν.
public void draw(Graphics g) {
// Σχεδιασμός όλων των θέσεων
for (Placep : places)
p.draw(g);
        // Σχεδιασμός όλων των μεταβάσεων
for (Transition t : transitions)
            t.draw(g);
// Σχεδιασμός όλων των ακμών
for (Arca : arcs)
a.draw(g);
    }

    // Επιστρέφει το στοιχείο του FPN στο οποίο περιέχεται το σημείο το
    // οποίο η μέθοδος δέχεται σαν όρισμα.
public PetriComponent getSelected(Point point) {
        for (Place p : places) {
            if (p.contains(point)) {
                selectedComponentOn = true;
```

```
                return p;
            }
        }
        for (Transition t : transitions) {
            if (t.contains(point)) {
                selectedComponentOn = true;
                return t;
            }
        }
        for (Arc a : arcs) {
            if (a.contains(point)) {
                selectedComponentOn = true;
                returna;
            }
        }

returnnnull;
    }

    // Επιστρέφει το στοιχείο του FPN το οποίο είναι επιλεγμένο. Αν κανένα
    // στοιχείο δεν είναι επιλεγμένο, τότε επιστρέφει null.
publicPetriComponentgetSelected() {

for (Place p : places) {
            if (p.isSelected())
                return p;
        }
        for (Transition t : transitions) {
            if (t.isSelected())
                return t;
        }
        for (Arc a : arcs) {
            if (a.isSelected())
                returna;
        }

returnnull;

    }

    // Διαγράφει ένα στοιχείο από το FPN.
public void removeComponent(PetriComponent c) {
        if (c instanceof Place)
            this.removePlace((Place) c);
        else if (c instanceof Transition)
            this.removeTransition((Transition) c);
        else
            this.removeArc((Arc) c);
}

    // Απο-επιλέγει όλα τα επιλεγμένα στοιχεία του FPN.
```

```
public void deselectAll() {
        for (Place p : places)
             p.setIsSelected(false);
        for (Transition t : transitions)
             t.setIsSelected(false);
        for (Arc a : arcs)
             a.setIsSelected(false);
        selectedComponentOn = false;
}

    // Ελέγχει αν ένα σύνολο μεταβάσεων αποτελεί υποσύνολο ενός άλλου.
    // Η μέθοδος αυτή χρησιμοποιείται από την μέθοδο stepSimulation().
public boolean subGroup(ArrayList<Transition> group1, ArrayList<Transition> group2) {
        for (Transition t : group1) {
             if (!group2.contains(t))
                 returnfalse;
        }
returntrue;
    }

    // Η μέθοδος αυτή βρίσκει όλες τις ικανοποιήσιμες μεταβάσεις του FPN
    // σε μία δεδομένη χρονική στιγμή και τις πυροδοτεί, σύμφωνα πάντα
    // με τους κανόνες πυροδότησης. Η μέθοδος επιστρέφει τον αριθμό
    // των μεταβάσεων που εν τέλει πυροδοτήθηκαν.
publicintstepSimulation() {

        // Στο εν λόγω ArrayList αποθηκεύονται οι ικανοποιήσιμες μεταβάσεις
ArrayList<Transition> satisfiable = new ArrayList<>();

// Βήμα 1: βρες αρχικά τις ικανοποιήσιμες μεταβάσεις
for (Transition t : transitions) {
             boolean flag = true;
             // Εισερχόμενες ακμές
             for (Place p : t.getIncomingPlaces()) {
// Βρες αρχικά την συνδεδεμένη ακμή
for (Arca : arcs) {
if (a.getStartsFrom() instanceof Place &&
                           a.getStartsFrom() == p &&
                           a.getEndsTo() instanceof Transition &&
                           a.getEndsTo() == t) {
                           // Έλεγξε τα τεκμήρια της θέσης σε σχέση με το
                           // βάρος της ακμής καθώς και τον βαθμό αληθείας της
                           // θέσης σε σχέση με το κατώφλι της μετάβασης
if (p.getTokensNo() < a.getWeight() ||
                               p.getTruthFactor() < t.getThreshold())
                               flag = false;
// Σε κάθε περίπτωση τερμάτισε την επανάληψη
break;
                      }
                  }
             }
```

```
            // Εξερχόμενες ακμές
for (Placep : t.getOutgoingPlaces()) {
                  // Βρες αρχικά την συνδεδεμένη ακμή
for (Arca : arcs) {
if (a.getStartsFrom() instanceof Transition &&
                           a.getStartsFrom() == t &&
                           a.getEndsTo() instanceof Place &&
                           a.getEndsTo() == p) {
                           // Έλεγξε τα τεκμήρια της θέσης σε σχέση με το
// βάρος της ακμής
                           if (p.getTokensNo() + a.getWeight() > p.getMaxCapacity())
                               flag = false;
                           // Σε κάθε περίπτωση τερμάτισε την επανάληψη
break;
                      }
                  }
            }
            // Αν η μετάβαση πληρεί τα κριτήρια τότε κάνε εισαγωγή αυτής
// στο ArrayList satisfiable
            if (flag)
                satisfiable.add(t);
        }

        // Βήμα 2: βρες ποιες από τις ικανοποιήσιμες μεταβάσεις είναι ισοπίθανες
ArrayList<ArrayList<Transition>> equalProbableGroups = new ArrayList<>();

        for (Place p : places) {
            ArrayList<Transition> transitionGroup1 = new ArrayList<>();
            for (Transition t : p.getOutgoingTransitions()) {
                 if (satisfiable.contains(t))
                     transitionGroup1.add(t);
            }
            if (!transitionGroup1.isEmpty())
                equalProbableGroups.add(transitionGroup1);
            ArrayList<Transition> transitionGroup2 = new ArrayList<>();
            for (Transition t : p.getIncomingTransitions()) {
                 if (satisfiable.contains(t))
                     transitionGroup2.add(t);
            }
            if (!transitionGroup2.isEmpty())
                equalProbableGroups.add(transitionGroup2);
}

        // Ειδική μέριμνα για τις μεταβάσεις που δεν έχουν εισόδους
// ή εξόδους άρα είναι ικανοποιήσιμες
        for (Transition t : transitions) {
            if (t.getIncomingPlaces().isEmpty() && t.getOutgoingPlaces().isEmpty()) {
                ArrayList<Transition> transitionGroup = new ArrayList<>();
                transitionGroup.add(t);
                equalProbableGroups.add(transitionGroup);
            }
```

```
        }

        boolean removed = true;
        while (removed) {
            removed = false;
            for (int i = 0; i < equalProbableGroups.size(); i++) {
                 for (int j = 0; j < equalProbableGroups.size(); j++) {
                     if (i == j)
                         continue;
                     if
(this.subGroup(equalProbableGroups.get(i),equalProbableGroups.get(j))) {
                         equalProbableGroups.remove(i);
                         removed = true;
                         break;
                     }                                  else                                  if
(this.subGroup(equalProbableGroups.get(j),equalProbableGroups.get(i))) {
                         equalProbableGroups.remove(j);
                         removed = true;
                         break;
                     }
                 }
if (removed)
break;
            }
        }

        // Επειδή η πυροδότηση των ισοπίθανων μεταβάσεων θα πρέπει να είναι
        // τυχαία, αναδιάταξε με τυχαίο τρόπο τα στοιχεία του
Random r = new Random();
        ArrayList<ArrayList<Transition>> tmp = new ArrayList<>();
        while (!equalProbableGroups.isEmpty()) {
            int i = r.nextInt(equalProbableGroups.size());
            ArrayList<Transition> group = equalProbableGroups.get(i);
            tmp.add(group);
            equalProbableGroups.remove(group);
}
equalProbableGroups = tmp;

        // Πυροδότηση των μεταβάσεων
        // Μετρητής ο οποίος μετρά πόσες μεταβάσεις πυροδοτήθηκαν
intfireCounter = 0;
        // Διατρέχουμε κάθε group ισοπίθανων μεταβάσεων
this.deselectAll();
        for (int i = 0; i < equalProbableGroups.size(); i++) {
             ArrayList<Transition> group = equalProbableGroups.get(i);
// Αν το group είναι κενό, αγνόησέ το
if (group.isEmpty())
continue;
             // Διάλεξε τυχαία μία μετάβαση από το group
int x = r.nextInt(group.size());
             Transition selected = group.get(x);
```

```
            // Πυροδότησε την μετάβαση
            selected.setIsSelected(true);
            selected.fire();
// Αύξησε τον μετρητή κατά 1
fireCounter++;
// Ενημέρωση των υπόλοιπων groups
            for (int j = 0; j < equalProbableGroups.size(); j++) {
                 if (i == j)
                     continue;
                 ArrayList<Transition> nextGroup = equalProbableGroups.get(j);
                 if (nextGroup.contains(selected))
                      nextGroup.clear();
                 else {
                     for (Transition t : group) {
                          if (t == selected)
                              continue;
                          if (nextGroup.contains(t))
                              nextGroup.remove(t);
                     }
                 }
            }
        }

        // Η μέθοδος επιστρέφει τον αριθμό των μεταβάσεων που πυροδοτήθηκαν
returnfireCounter;
    }

    // Θέτει σε όλες τις θέσεις την αρχική τους χωρητικότητα.
public void reset() {

        for (Place p : places)
             p.setTokensNo(p.getInitialCapacity());

    }

    // Επιστρέφει το στοιχείο του FPN συναρτήσει της ταυτότητάς του,
    // την οποία δέχεται ως όρισμα.
public PetriComponent getComponentById(int id) {

        for (Place p : places) {
             if (p.getId() == id)
                 return p;
        }

        for (Transition t : transitions) {
            if (t.getId() == id)
                return t;
        }

        for (Arc a : arcs) {
             if (a.getId() == id)
```

```
                returna;
        }

returnnull;

    }

    // Βρίσκει και επιστρέφει την μέγιστη ταυτότητα που έχει δοθεί σε
    // κάποιο στοιχείο του FPN.
publicintfindMaxId() {

int max = 0;

        for (Place p : places) {
             if (p.getId() > max)
                 max = p.getId();
        }

        for (Transition t : transitions) {
            if (t.getId() > max)
                max = t.getId();
        }

        for (Arc a : arcs) {
             if (a.getId() > max)
                 max = a.getId();
        }

        return max;

    }

}
```

Κλάσηpetrinet.PetriNetFileHandler

```
// ΗκλάσηPetriNetFileHandlerχρησιμοποιείταιτηναποθήκευσηκαιτην
// ανάκτηση FPN σε αντίστοιχα αρχεία (με κατάληξη .fpn).

package petrinet;

import gui.PetriPanel;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.util.Locale;
import java.util.Scanner;
import javax.swing.JOptionPane;

public class PetriNetFileHandler {
```

```
    // Αποθηκεύει ένα FPN στο αρχείο με ονομασία filename.
    public void savePetriNet(PetriNet petriNet, String filename) {

        try {
            // Άνοιγμα του αρχείου εξόδου
            File outFile = new File(filename);
            BufferedWriter output = new BufferedWriter(new FileWriter(outFile));
            // Εγγραφή των θέσεων
            int placesNum = petriNet.getPlaces().size();
            output.write(placesNum + "\n");
            for (int i = 0; i < placesNum; i++) {
                 int placeId = petriNet.getPlaces().get(i).getId();
                 String placeName = petriNet.getPlaces().get(i).getName();
                 int                    x                    =                    (int)
petriNet.getPlaces().get(i).getPlaceShape().getBounds().getX();
                 int                    y                    =                    (int)
petriNet.getPlaces().get(i).getPlaceShape().getBounds().getY();
                 int maxCapacity = petriNet.getPlaces().get(i).getMaxCapacity();
                 int initialCapacity = petriNet.getPlaces().get(i).getInitialCapacity();
                 int tokensNo = petriNet.getPlaces().get(i).getTokensNo();
                 double truthFactor = petriNet.getPlaces().get(i).getTruthFactor();
                 output.write(placeId + " " + placeName + " " + x + " " + y + " " +
                              maxCapacity + " "  + initialCapacity + " " +
                              tokensNo + " " + truthFactor + "\n");
            }
            // Εγγραφή των μεταβάσεων
            int transitionsNum = petriNet.getTransitions().size();
            output.write(transitionsNum + "\n");
            for (int i = 0; i < transitionsNum; i++) {
                 int transitionId = petriNet.getTransitions().get(i).getId();
                 String transitionName = petriNet.getTransitions().get(i).getName();
                 int                    x                    =                    (int)
petriNet.getTransitions().get(i).getTransitionShape().getBounds().getX();
                 int                    y                    =                    (int)
petriNet.getTransitions().get(i).getTransitionShape().getBounds().getY();
                 double threshold = petriNet.getTransitions().get(i).getThreshold();
                 output.write(transitionId + " " + transitionName + " " +
                              x + " " + y + " " + threshold + "\n");
            }
            // Εγγραφή των ακμών
            int arcsNum = petriNet.getArcs().size();
            output.write(arcsNum + "\n");
            for (int i = 0; i < arcsNum; i++) {
                 int arcId = petriNet.getArcs().get(i).getId();
                 PetriComponent startsFrom = petriNet.getArcs().get(i).getStartsFrom();
                 PetriComponent endsTo = petriNet.getArcs().get(i).getEndsTo();
                 int startsFromId = startsFrom.getId();
                 int endsToId = endsTo.getId();
                 output.write(arcId + " " + startsFromId + " " + endsToId + "\n");
}
            // Κλείσιμο του αρχείου εξόδου
output.close();
```

```
}
        catch(Exception ex) {
            JOptionPane.showMessageDialog(null,
"Πρόβλημα κατά την εγγραφή του αρχείου:\n" +
ex.getMessage(),
                                    "Σφάλμα",
                                    JOptionPane.WARNING_MESSAGE);
}

    }

    // Ανακτά ένα FPN από το αρχείο με ονομασία filename.
public void readPetriNet(PetriPanel petriPanel, String filename) {

        try {
            // Άνοιγμα του αρχείου εισόδου
            Scanner input = new Scanner(new File(filename));
            input.useLocale(Locale.US);
            // Δημιουργία ενός νέου δικτύου Petri
            PetriNet petriNet = new PetriNet();
            PetriComponent.drawOn = false;
// Θέτουμε το νέο δίκτυο Petri στο PetriPanel
petriPanel.setPetriNet(petriNet);
            // Διάβασμα των θέσεων
            int placesNum = Integer.parseInt(input.nextLine());
            for (int i = 0; i < placesNum; i++) {
                 String placeString = input.nextLine();
                 String[] placeDataStr = placeString.split(" ");
                 int placeId = Integer.parseInt(placeDataStr[0]);
                 String placeName = placeDataStr[1];
                 int x = Integer.parseInt(placeDataStr[2]);
                 int y = Integer.parseInt(placeDataStr[3]);
                 int maxCapacity = Integer.parseInt(placeDataStr[4]);
                 int initialCapacity = Integer.parseInt(placeDataStr[5]);
                 int tokensNo = Integer.parseInt(placeDataStr[6]);
                 double truthFactor = Double.parseDouble(placeDataStr[7]);
                 Place p = new Place(x, y, petriPanel.getGraphics());
                 p.setId(placeId);
                 p.setName(placeName);
                 p.setMaxCapacity(maxCapacity);
                 p.setInitialCapacity(initialCapacity);
                 p.setTokensNo(tokensNo);
                 p.setTruthFactor(truthFactor);
                 petriNet.addPlace(p);
            }
            // Διάβασμα των μεταβάσεων
int transitionsNum = Integer.parseInt(input.nextLine());
            for (int i = 0; i < transitionsNum; i++) {
                 String transitionString = input.nextLine();
                 String[] transitionDataStr = transitionString.split(" ");
                 int transitionId = Integer.parseInt(transitionDataStr[0]);
```

```
                String transitionName = transitionDataStr[1];
                int x = Integer.parseInt(transitionDataStr[2]);
                int y = Integer.parseInt(transitionDataStr[3]);
                double threshold = Double.parseDouble(transitionDataStr[4]);
                Transition t = new Transition(x, y, petriPanel.getGraphics());
                t.setId(transitionId);
                t.setName(transitionName);
                t.setThreshold(threshold);
                petriNet.addTransition(t);
            }
            // Διάβασμα των ακμών
            int arcsNum = Integer.parseInt(input.nextLine());
            for (int i = 0; i < arcsNum; i++) {
                String arcString = input.nextLine();
                String[] arcDataStr = arcString.split(" ");
                int arcId = Integer.parseInt(arcDataStr[0]);
                PetriComponent                          startsFrom                          =
petriNet.getComponentById(Integer.parseInt(arcDataStr[1]));
                PetriComponent                           endsTo                             =
petriNet.getComponentById(Integer.parseInt(arcDataStr[2]));
                Arc a = new Arc(startsFrom, endsTo, petriPanel.getGraphics());
                if (startsFrom instanceof Place) {
                    ((Place) startsFrom).addOutgoingTransition((Transition) endsTo);
                    ((Transition) endsTo).addIncomingPlaces((Place) startsFrom);
                } else {
                    ((Transition) startsFrom).addOutgoingPlace((Place) endsTo);
                    ((Place) endsTo).addIncomingTransition((Transition) startsFrom);
                }
                a.setId(arcId);
                petriNet.addArc(a);
}
            // Θέτουμε εκ νέου το τρέχον μέγιστο id
PetriComponent.nextId = petriNet.findMaxId() + 1;
            PetriComponent.drawOn = true;
        }
        catch(Exception ex) {
            JOptionPane.showMessageDialog(null,
"Πρόβλημα κατά το διάβασμα του αρχείου:\n" +
ex.getMessage(),
                                "Σφάλμα",
                                JOptionPane.WARNING_MESSAGE);
}

    }

}
```

Κλάση gui.PetriPanel

```
// Η κλάση PetriPanel αποτελεί τον καμβά στον οποίον ο χρήστης σχεδιάζει το FPN.
```

```
package gui;

import petrinet.PetriNet;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import javax.swing.JPanel;

public class PetriPanel extends JPanel {

// Το FPN το οποίο σχεδιάζεται
privatePetriNetpetriNet;
    // Αναφορά στο JFrame το οποίο περιέχει το PetriPanel
privateMainWindowparentWindow;
    // Flag το οποίο δείχνει αν η προσομοίωση είναι ενεργή
privatebooleansimulationOn;
    // Flag το οποίο δείχνει αν πραγματοποιήθηκαν αλλαγές στο FPN
privatebooleanchanged;
    // Flag το οποίο δείχνει αν το FPN έχει αποθηκευθεί
private boolean saved;

    // Constructor
    public PetriPanel(MainWindow parentWindow) {
        this.parentWindow = parentWindow;
        this.setBackground(Color.WHITE);
        petriNet = new PetriNet();
        changed = false;
        saved = false;
    }

    // Getters/Setters
    public MainWindow getParentWindow() {
        return parentWindow;
    }

    public PetriNet getPetriNet() {
        return petriNet;
    }

    public void setPetriNet(PetriNet petriNet) {
        this.petriNet = petriNet;
    }

    public boolean isSimulationOn() {
        return simulationOn;
    }

    public void setSimulationOn(boolean simulationOn) {
        this.simulationOn = simulationOn;
    }
```

```
    public boolean hasChanged() {
        return changed;
    }

    public void setChanged(boolean changed) {
        this.changed = changed;
    }

    public boolean isSaved() {
        return saved;
    }

    public void setSaved(boolean saved) {
        this.saved = saved;
    }

    // Ζωγραφίζει το περιεχόμενο του PetriPanel, καλώντας την μέθοδο
    // paintBackGround αλλά και την μέθοδο draw του PetriNet.
@Override
    public void paintComponent(Graphics g) {

        super.paintComponent(g);
        this.paintBackGround(g);
        petriNet.draw(g);

    }

    // Ζωγραφίζει το background του PetriPanel.
    public void paintBackGround(Graphics g) {

        Graphics2D g2d = (Graphics2D) g;

        int panelWidth = this.getWidth();
        int panelHeight = this.getHeight();

        g2d.setColor(Color.BLACK);

        if (!simulationOn) {
            int pointY = 2;
            while (pointY < panelHeight) {
                 int pointX = 2;
                 while (pointX < panelWidth) {
                      g2d.drawLine(pointX, pointY, pointX, pointY);
                      pointX += 10;
                 }
                 pointY += 10;
            }
        }

    }
```

```
    // Καλεί την μέθοδο stepSimulation() του PetriNet.
    public int stepSimulation() {
        return petriNet.stepSimulation();
}

    // Καλεί την μέθοδο reset() του PetriNet.
public void resetPetriNet() {
        petriNet.reset();
    }

}
```

Κλάσηgui.PetriPanelKeyListener

```
// ΗκλάσηPetriPanelKeyListenerδιαχειρίζεταιταγεγονότακατάταοποία
// ο χρήστης πληκτρολογεί κάτι μέσα στο PetriPanel.

package gui;

import petrinet.PetriComponent;
import petrinet.PetriNet;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

public class PetriPanelKeyListener implements KeyListener {

    // Αναφορά προς το PetriPanel
    private PetriPanel petriPanel;
    // Αναφορά προς το FPN
    private PetriNet petriNet;

    // Constructor
    public PetriPanelKeyListener(PetriPanel petriPanel) {
        this.petriPanel = petriPanel;
        this.petriNet = petriPanel.getPetriNet();
    }

    // Getters/Setters
    public void setPetriNet(PetriNet petriNet) {
        this.petriNet = petriNet;
    }

    // Εκτελείται όταν ο χρήστης κρατά πατημένο ένα πλήκτρο.
    // Η μέθοδος αυτή δεν έχει καμία πρακτική σημασία στο συγκεκριμένο λογισμικό.
@Override
    public void keyTyped(KeyEvent e) {
}

    // Εκτελείται όταν ο χρήστης παρά ένα πλήκτρο (πίεση και ελευθέρωση).
    // Η μέθοδος ελέγχει αν το πλήκτρο ήταν το Delete. Αν ναι και ο χρήστης
    // έχει ήδη επιλέξει ένα στοιχείο το στοιχείο αυτό διαγράφεται.
```

```
@Override
    public void keyPressed(KeyEvent e) {
        if (petriPanel.isSimulationOn())
            return;
        if (e.getKeyCode() == KeyEvent.VK_DELETE) {
            PetriComponent c = petriNet.getSelected();
            if (c != null) {
                petriNet.removeComponent(c);
                petriPanel.repaint();
}
        }
    }

    // Εκτελείται όταν ο χρήστης αφήνει ένα πλήκτρο το οποίο είχε πατήσει.
    // Η μέθοδος αυτή δεν έχει καμία πρακτική σημασία στο συγκεκριμένο λογισμικό.
@Override
    public void keyReleased(KeyEvent e) {
    }

}
```

Κλάσηgui.PetriPanelMouseListener

```
// Η κλάση PetriPanelMouseListener διαχειρίζεται τα γεγονότα κατά τα οποία ο
// χρήστης επιλέγει κάτι με το ποντίκι μέσα στο PetriPanel.

package gui;

import petrinet.Arc;
import petrinet.PetriComponent;
import petrinet.PetriNet;
import petrinet.Place;
import petrinet.Transition;
import java.awt.Point;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import javax.swing.JOptionPane;
import javax.swing.SwingUtilities;

public class PetriPanelMouseListener implements MouseListener {

// Flag το οποίο αποθηκεύει την πληροφορία για το
    // αν μία θέση έχει επιλεγεί
privatebooleanplaceCreationOn;
    // Flag το οποίο αποθηκεύει την πληροφορία για το
// αν μία μετάβαση έχει επιλεγεί
    private boolean transitionCreationOn;
// Flag το οποίο αποθηκεύει την πληροφορία για το
    // αν μία ακμή έχει επιλεγεί
privatebooleanarcCreationOn;
    // Αναφορά προς το PetriPanel
```

```
private PetriPanel petriPanel;
    // Αναφορά προς το FPN
    privatePetriNetpetriNet;
    // Σε περίπτωση που ο χρήστης αρχίζει να σύρει το ποντίκι, τότε θα πρέπει
    // να γνωρίζουμε το σημείο από όπου το γεγονός αυτό αρχίζει.
privatePointstartingPoint;
    // Σε περίπτωση που ο χρήστης αρχίζει να σύρει το ποντίκι, τότε θα πρέπει
    // να γνωρίζουμε το σημείο από όπου το γεγονός αυτό τερματίζει.
private Point endingPoint;

    // Constructor
    public PetriPanelMouseListener(PetriPanel panel) {
        this.petriPanel = panel;
        petriNet = petriPanel.getPetriNet();
    }

    // Getters/Setters
    public void setPetriNet(PetriNet petriNet) {
        this.petriNet = petriNet;
    }

    // Εκτελείται όταν ο χρήστης κάνει κλικ στον καμβά (πίεση και αποπίεση).
@Override
    public void mouseClicked(MouseEvent e) {
        if (petriPanel.isSimulationOn())
            return;
        petriPanel.setChanged(true);
        petriPanel.getParentWindow().updateTitle();
        if (e.getClickCount() == 2) {
            PetriComponent comp = petriNet.getSelected(e.getPoint());
            if (comp != null) {
                if (comp instanceof Place) {
                    PlacePropertiesWindow placePropertiesWindow =
                            new                          PlacePropertiesWindow((Place)comp,
petriPanel.getParentWindow());
                    petriPanel.getParentWindow().setEnabled(false);
                    placePropertiesWindow.setVisible(true);
                } else if (comp instanceof Transition) {
                    TransitionPropertiesWindow transitionPropertiesWindow =
                            new                TransitionPropertiesWindow((Transition)comp,
petriPanel.getParentWindow());
                    petriPanel.getParentWindow().setVisible(true);
                    transitionPropertiesWindow.setVisible(true);
                } else {
                    ArcPropertiesWindow arcPropertiesWindow =
                            new                              ArcPropertiesWindow((Arc)comp,
petriPanel.getParentWindow());
                    petriPanel.getParentWindow().setVisible(true);
                    arcPropertiesWindow.setVisible(true);
}
            }
```

```
        }
    }

    // Ζωγραφίζει εξ'αρχής το PetriNet, καθαρίζοντας τις όποιες επιλογές
    // στοιχείων σε αυτό.
publicvoidclear() {
petriNet.deselectAll();
petriPanel.repaint();
    }

    // Εκτελείται όταν ο χρήστης κάνει κλικ στον καμβά και κρατά το
// πλήκτρο του ποντικού πατημένο.
    @Override
    public void mousePressed(MouseEvent e) {
        if (petriPanel.isSimulationOn())
            return;
        petriPanel.setChanged(true);
        petriPanel.getParentWindow().updateTitle();
        PetriPanelMouseMotionListener mouseMotionListener =
                        (PetriPanelMouseMotionListener)
petriPanel.getMouseMotionListeners()[0];
        // Σε περίπτωση επιλογής του αριστερού κουμπιού του ποντικιού
        if (SwingUtilities.isLeftMouseButton(e)) {
            // Σχεδιασμός μιας θέσης
            if (placeCreationOn) {
                Place place = new Place(e.getX(),e.getY(),petriPanel.getGraphics());
                petriNet.addPlace(place);
            // Σχεδιασμός μιας μετάβασης
} else if (transitionCreationOn) {
                Transition                transition                =                new
Transition(e.getX(),e.getY(),petriPanel.getGraphics());
                petriNet.addTransition(transition);
            } else if (arcCreationOn) {
                // Πάρε το αρχικό σημείο
                startingPoint = new Point(e.getX(),e.getY());
                mouseMotionListener.setStartingPoint(startingPoint);
                mouseMotionListener.setDrawLineOn(true);
            } else {
                this.clear();
                PetriComponent comp = petriNet.getSelected(e.getPoint());
                if (comp != null) {
                    petriNet.deselectAll();
                    comp.setIsSelected(true);
                    petriNet.setSelectedComponentOn();
                    comp.draw(petriPanel.getGraphics());
                    mouseMotionListener.setStartingPoint(e.getPoint());
}
            }
        }
    }
```

```
    // Εκτελείται όταν ο χρήστης έχει κάνει ήδη κρατήσει πατημένο το
    // πλήκτρο του ποντικιού και στην συνέχεια το απελευθερώνει.
@Override
    public void mouseReleased(MouseEvent e) {
        if (petriPanel.isSimulationOn())
            return;
        PetriPanelMouseMotionListener mouseMotionListener =
                        (PetriPanelMouseMotionListener)
petriPanel.getMouseMotionListeners()[0];
        if (arcCreationOn && mouseMotionListener.isDrawLineOn()) {
            mouseMotionListener.setDrawLineOn(false);
            // Έλεγχος εγκυρότητας της ακμής
// Παρε το τελικό σημείο (έχουμε ήδη το αρχικό)
endingPoint = e.getPoint();
            // Έλεγξε αν υπάρχει κάποιο στοιχείο στο δίκτυο Petri (θέση ή
            // μετάβαση) από το οποίο το αρχικό σημείο εμπεριέχεται
PetriComponent startsFrom = null;
            for (Place p : petriNet.getPlaces()) {
                 if (p.contains(startingPoint))
                     startsFrom = p;
            }
            for (Transition t : petriNet.getTransitions()) {
                 if (t.contains(startingPoint))
                     startsFrom = t;
            }
            if (startsFrom == null) {
                JOptionPane.showMessageDialog(null,
"Η ακμή δεν αρχίζει από κάποια θέση ή μετάβαση.",
"Σφάλμα",
                                    JOptionPane.WARNING_MESSAGE);
                return;
}
            // Έλεγξε αν υπάρχει κάποιο στοιχείο στο δίκτυο Petri (θέση ή
            // μετάβαση) στο οποίο το τελικό σημείο εμπεριέχεται
PetriComponent endsTo = null;
            for (Place p : petriNet.getPlaces()) {
                 if (p.contains(endingPoint))
                     endsTo = p;
            }
            for (Transition t : petriNet.getTransitions()) {
                 if (t.contains(endingPoint))
                     endsTo = t;
            }
            if (endsTo == null) {
                JOptionPane.showMessageDialog(null,
"Η ακμή δεν τερματίζει σε κάποια θέση ή μετάβαση.",
"Σφάλμα",
                                    JOptionPane.WARNING_MESSAGE);
                return;
}
            // Έλεγξε αν τα δύο στοιχεία του δικτύου Petri είναι διαφορετικά
```

```
// μεταξύ τους
            if ((startsFrom instanceof Place && endsTo instanceof Place) ||
                (startsFrom instanceof Transition && endsTo instanceof Transition)) {
                 JOptionPane.showMessageDialog(null,
                                "Τα στοιχεία που η ακμή ενώνει θα πρέπει να είναι
διαφορετικού τύπου.",
"Σφάλμα",
                                JOptionPane.WARNING_MESSAGE);
                return;
            }
            // Δημιουργία της νέας κατευθυνόμενης ακμής
            Arc arc = new Arc(startsFrom,endsTo,petriPanel.getGraphics());
            petriNet.addArc(arc);
            // Σύνδεση των απαραίτητων στοιχείων στο δίκτυο Petri
if (startsFrom instanceof Place) {
                ((Place) startsFrom).addOutgoingTransition((Transition) endsTo);
                ((Transition) endsTo).addIncomingPlaces((Place) startsFrom);
            } else {
                ((Transition) startsFrom).addOutgoingPlace((Place) endsTo);
                ((Place) endsTo).addIncomingTransition((Transition) startsFrom);
}
        }
    }

    // Εκτελείται όταν ο δείκτης του ποντικιού βρίσκεται μέσα στα όρια
    // του καμβά. Η μέθοδος αυτή δεν έχει καμία πρακτική χρησιμότητα
// στο συγκεκριμένο λογισμικό.
    @Override
    public void mouseEntered(MouseEvent e) {
}

    // Εκτελείται όταν ο δείκτης του ποντικιού βγαίνει έξω από τα όρια
    // του καμβά. Η μέθοδος αυτή δεν έχει καμία πρακτική χρησιμότητα
    // στο συγκεκριμένο λογισμικό.
    @Override
publicvoidmouseExited(MouseEvente) {
    }

// Getters/Setters
    public boolean isPlaceCreationOn() {
        return placeCreationOn;
    }

    public void setPlaceCreationOn(boolean placeCreationOn) {
        this.placeCreationOn = placeCreationOn;
        if (placeCreationOn) {
            transitionCreationOn = false;
            arcCreationOn = false;
        }
    }
```

```
    public boolean isTransitionCreationOn() {
        return transitionCreationOn;
    }

    public void setTransitionCreationOn(boolean transitionCreationOn) {
        this.transitionCreationOn = transitionCreationOn;
        if (transitionCreationOn) {
            placeCreationOn = false;
            arcCreationOn = false;
        }
    }

    public boolean isArcCreationOn() {
        return arcCreationOn;
    }

    public void setArcCreationOn(boolean arcCreationOn) {
        this.arcCreationOn = arcCreationOn;
        if (arcCreationOn) {
            placeCreationOn = false;
            transitionCreationOn = false;
        }
    }

}
```

Κλάση gui.PetriPanelMouseMotionListener

```
// ΗκλάσηPetriPanelMouseMotionListenerδιαχειρίζεταιταγεγονότακατάτα
// οποία ο χρήστης σύρει τον δείκτη του ποντικιού μέσα στο PetriPanel.

package gui;

import petrinet.PetriComponent;
import petrinet.PetriNet;
import petrinet.Place;
import petrinet.Transition;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionListener;

public class PetriPanelMouseMotionListener implements MouseMotionListener {

// Flag το οποίο αποθηκεύει την πληροφορία για το αν ο χρήστης σύρει
    // το ποντίκι για να ζωγραφίσει μία ακμή ή για να μεταφέρει ένα στοιχείο
    // του FPN από ένα σημείο σε ένα άλλο.
privatebooleandrawLineOn;
    // Σε περίπτωση που ο χρήστης αρχίζει να σύρει το ποντίκι, τότε θα πρέπει
    // να γνωρίζουμε το σημείο από όπου το γεγονός αυτό αρχίζει.
private Point startingPoint;
```

```
    // Αναφορά προς το PetriPanel
    private PetriPanel petriPanel;
    // Αναφορά προς το FPN
    private PetriNet petriNet;

    // Constructor
    public PetriPanelMouseMotionListener(PetriPanel petriPanel) {
        this.petriPanel = petriPanel;
        petriNet = petriPanel.getPetriNet();
    }

    // Getters/Setters
    public boolean isDrawLineOn() {
        return drawLineOn;
    }

    public void setDrawLineOn(boolean drawLineOn) {
        this.drawLineOn = drawLineOn;
    }

    public Point getStartingPoint() {
        return startingPoint;
    }

    public void setStartingPoint(Point startingPoint) {
        this.startingPoint = startingPoint;
    }

    public void setPetriNet(PetriNet petriNet) {
        this.petriNet = petriNet;
    }

    // Εκτελείται όταν ο χρήστης σύρει τον δείκτη του ποντικιού ενώ
    // πρώτα έχει πατήσει και έχει κρατήσει το αριστερό πλήκτρο.
@Override
    public void mouseDragged(MouseEvent e) {
        if (petriPanel.isSimulationOn())
            return;
        if (drawLineOn) {
            petriPanel.repaint();
            Graphics2D g2d = (Graphics2D) petriPanel.getGraphics();

g2d.drawLine((int)startingPoint.getX(),(int)startingPoint.getY(),e.getX(),e.getY());
        } else if (petriNet.hasSelectedComponent()) {
            PetriComponent comp = petriNet.getSelected(startingPoint);
            if (comp instanceof Place || comp instanceof Transition) {
                comp.updatePosition(e.getX(), e.getY(), petriPanel.getGraphics());
                startingPoint = e.getPoint();
                petriPanel.repaint();
}
        }
```

```
    }

    // Εκτελείται όταν ο χρήστης απλά μετακινεί το ποντίκι μέσα στον καμβά.
    // Η μέθοδος δεν έχει καμία πρακτική σημασία στο εν λόγω λογισμικό.
@Override
    public void mouseMoved(MouseEvent e) {
    }

}
```

Κλάσηgui.PlacePropertiesWindow

```
// ΗκλάσηPlacePropertiesWindowαντιστοιχείστοπαράθυροτου
// γραφικού περιβάλλοντος στο οποίο ο χρήστης μπορεί να θέσει τις παραμέτρους
// μίας θέσης.

package gui;

import petrinet.Place;
import javax.swing.JOptionPane;

public class PlacePropertiesWindow extends javax.swing.JFrame {

// Η θέση για την οποία θέτουμε τις παραμέτρους
privatePlaceplace;
    // Αναφορά στο γονικό παράθυρο του γραφικού περιβάλλοντος
private MainWindow parent;

    // Constructor
    public PlacePropertiesWindow(Place place, MainWindow parent) {
        this.place = place;
        this.parent = parent;
        initComponents();
this.setInitialData();
        // Ρυθμίζουμε το παράθυρο να εμφανίζεται στο κέντρο της οθόνης
setLocationRelativeTo(null);
    }

    // Αρχικοποιεί τις πληροφορίες που παρουσιάζονται στο παράθυρο
    // ανάλογα με τα δεδομένα της θέσης.
public void setInitialData() {
        placeNameTF.setText(place.getName());
        maxCapacitySpinner.setValue(place.getMaxCapacity());
        initialCapacitySpinner.setValue(place.getInitialCapacity());
        truthFactorSpinner.setValue(place.getTruthFactor());
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
```

```
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jSpinner1 = new javax.swing.JSpinner();
        jLabel1 = new javax.swing.JLabel();
        placeNameTF = new javax.swing.JTextField();
        cancelButton = new javax.swing.JButton();
        saveButton = new javax.swing.JButton();
        jLabel3 = new javax.swing.JLabel();
        truthFactorSpinner = new javax.swing.JSpinner();
        jLabel4 = new javax.swing.JLabel();
        maxCapacitySpinner = new javax.swing.JSpinner();
        jLabel5 = new javax.swing.JLabel();
        initialCapacitySpinner = new javax.swing.JSpinner();

        setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
        setTitle("Place Properties");
        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                formWindowClosing(evt);
            }
        });

        jLabel1.setText("Name:");

        placeNameTF.setText("jTextField1");

        cancelButton.setText("Cancel");
        cancelButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelButtonActionPerformed(evt);
            }
        });

        saveButton.setText("Save");
        saveButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                saveButtonActionPerformed(evt);
            }
        });

        jLabel3.setText("Truth Factor:");

        truthFactorSpinner.setModel(new  javax.swing.SpinnerNumberModel(0.0d,  0.0d,  1.0d,
0.10000000149011612d));

        jLabel4.setText("Max Capacity:");
```

```
        maxCapacitySpinner.setModel(new javax.swing.SpinnerNumberModel(1, 1, 999, 1));

        jLabel5.setText("Initial Capacity:");

        initialCapacitySpinner.setModel(new javax.swing.SpinnerNumberModel(0, 0, 999, 1));

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 88, Short.MAX_VALUE)
                        .addComponent(placeNameTF,   javax.swing.GroupLayout.PREFERRED_SIZE,
136, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(81, 81, 81))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel4)
                        .addContainerGap(267, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(maxCapacitySpinner,
javax.swing.GroupLayout.PREFERRED_SIZE, 51, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jLabel3)
                                .addGap(55, 55, 55)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                    .addComponent(truthFactorSpinner,
javax.swing.GroupLayout.DEFAULT_SIZE, 51, Short.MAX_VALUE)
                                    .addComponent(initialCapacitySpinner))))
                        .addGap(0, 0, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                                .addGap(0, 0, Short.MAX_VALUE)
                                .addComponent(cancelButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(saveButton))
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jLabel5)
                                .addGap(0, 0, Short.MAX_VALUE)))
                        .addContainerGap())))
        );
```

```
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(placeNameTF,       javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(maxCapacitySpinner,
javax.swing.GroupLayout.PREFERRED_SIZE,               javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel5)
                    .addComponent(initialCapacitySpinner,
javax.swing.GroupLayout.PREFERRED_SIZE,               javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel3)
                    .addComponent(truthFactorSpinner,
javax.swing.GroupLayout.PREFERRED_SIZE,               javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(cancelButton)
                    .addComponent(saveButton))
                .addContainerGap())
);

pack();
    }// </editor-fold>

    // Εκτελείται όταν ο χρήστης πατά το κουμπί cancel.
private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt) {

        parent.setEnabled(true);
        parent.refresh();
        this.dispose();

    }

    // Εκτελείται όταν ο χρήστης πατά το κουμπί save.
private void saveButtonActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        if (placeNameTF.getText().length() == 0) {
            JOptionPane.showMessageDialog(this,
"Το όνομα της θέσης δεν μπορεί να είναι κενό!",
"Σφάλμα",
                                JOptionPane.WARNING_MESSAGE);
            return;
        }
        if (placeNameTF.getText().length() > 20) {
            JOptionPane.showMessageDialog(this,
"Το όνομα της θέσης είναι πολύ μεγάλο!",
"Σφάλμα",
                                JOptionPane.WARNING_MESSAGE);
            return;
        }
        if ((Integer) maxCapacitySpinner.getValue() < 0 ||
            (Integer) maxCapacitySpinner.getValue() > 999) {
            JOptionPane.showMessageDialog(this,
                                "Η μέγιστη χωρητικότητα πρέπει να κυμαίνεται στο διάστημα
[0..999]",
"Σφάλμα",
                                JOptionPane.WARNING_MESSAGE);
            return;
        }
        if        ((Integer)        maxCapacitySpinner.getValue()        <        (Integer)
initialCapacitySpinner.getValue()) {
            JOptionPane.showMessageDialog(this,
                                "Η μέγιστη χωρητικότητα πρέπει να είναι μεγαλύτερη ή ίση
της αρχικής χωρητικότητας",
"Σφάλμα",
                                JOptionPane.WARNING_MESSAGE);
            return;
        }
        if ((Double) truthFactorSpinner.getValue() < 0.0 ||
            (Double) truthFactorSpinner.getValue() > 1.0) {
            JOptionPane.showMessageDialog(this,
                                "Ο βαθμός βεβαιότητας πρέπει να κυμαίνεται στο διάστημα
[0..1]",
"Σφάλμα",
                                JOptionPane.WARNING_MESSAGE);
            return;
        }

        place.setName(placeNameTF.getText());
        place.setMaxCapacity((Integer) maxCapacitySpinner.getValue());
        place.setInitialCapacity((Integer) initialCapacitySpinner.getValue());
        place.setTokensNo((Integer) initialCapacitySpinner.getValue());
        Double value = (Double) truthFactorSpinner.getValue();
        value = (double)Math.round(value * 1000) / 1000;
        place.setTruthFactor(value);

        parent.setEnabled(true);
        parent.refresh();
```

```
this.dispose();

    }

    // Εκτελείται όταν ο χρήστης κλείνει το παράθυρο.
private void formWindowClosing(java.awt.event.WindowEvent evt) {

        parent.setEnabled(true);
        parent.refresh();
        this.dispose();

    }


    // Variables declaration - do not modify
    private javax.swing.JButton cancelButton;
    private javax.swing.JSpinner initialCapacitySpinner;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JSpinner jSpinner1;
    private javax.swing.JSpinner maxCapacitySpinner;
    private javax.swing.JTextField placeNameTF;
    private javax.swing.JButton saveButton;
    private javax.swing.JSpinner truthFactorSpinner;
    // End of variables declaration
}
```

Κλάσηgui.TransitionPropertiesWindow

```
// Η κλάση TransitionPropertiesWindow αντιστοιχεί στο παράθυρο του γραφικού
// περιβάλλοντος στο οποίο ο χρήστης μπορεί να θέσει τις παραμέτρους μίας
// μετάβασης.

package gui;

import petrinet.Transition;
import javax.swing.JOptionPane;

public class TransitionPropertiesWindow extends javax.swing.JFrame {

// Η μετάβαση για την οποία θέτουμε τις παραμέτρους
privateTransitiontransition;
    // Αναφορά στο γονικό παράθυρο του γραφικού περιβάλλοντος
private MainWindow parent;

    // Constructor
    public TransitionPropertiesWindow(Transition transition, MainWindow parent) {
        this.transition = transition;
        this.parent = parent;
```

```
        initComponents();
this.setInitialData();
        // Ρυθμίζουμε το παράθυρο να εμφανίζεται στο κέντρο της οθόνης
setLocationRelativeTo(null);
    }

    // Αρχικοποιεί τις πληροφορίες που παρουσιάζονται στο παράθυρο
    // ανάλογα με τα δεδομένα της μετάβασης.
public void setInitialData() {
        transitionNameTF.setText(transition.getName());
        thresholSpinner.setValue(transition.getThreshold());
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        transitionNameTF = new javax.swing.JTextField();
        thresholSpinner = new javax.swing.JSpinner();
        saveButton = new javax.swing.JButton();
        cancelButton = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
        setTitle("Transition Properties");
        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                formWindowClosing(evt);
            }
        });

        jLabel1.setText("Name:");

        jLabel2.setText("Threshold:");

        transitionNameTF.setText("jTextField1");

        thresholSpinner.setModel(new   javax.swing.SpinnerNumberModel(0.0d,   0.0d,   1.0d,
0.1d));

        saveButton.setText("Save");
        saveButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                saveButtonActionPerformed(evt);
            }
        });

        cancelButton.setText("Cancel");
        cancelButton.addActionListener(new java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel1)
                    .addComponent(jLabel2))
                .addGap(30, 30, 30)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(transitionNameTF,  javax.swing.GroupLayout.PREFERRED_SIZE,
126, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(thresholSpinner,   javax.swing.GroupLayout.PREFERRED_SIZE,
55, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(124, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(cancelButton)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(saveButton)
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(transitionNameTF,  javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(thresholSpinner,   javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,     23,
Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(saveButton)
                    .addComponent(cancelButton))
                .addContainerGap())
```

```
);

pack();
    }// </editor-fold>

    // Εκτελείται όταν ο χρήστης πατά το κουμπί cancel.
private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt) {

        parent.setEnabled(true);
        parent.refresh();
        this.dispose();

    }

    // Εκτελείται όταν ο χρήστης πατά το κουμπί save.
private void saveButtonActionPerformed(java.awt.event.ActionEvent evt) {

        if (transitionNameTF.getText().length() == 0) {
            JOptionPane.showMessageDialog(this,
"Το όνομα της μετάβασης δεν μπορεί να είναι κενό!",
"Σφάλμα",
                                JOptionPane.WARNING_MESSAGE);
            return;
        }
        if (transitionNameTF.getText().length() > 20) {
            JOptionPane.showMessageDialog(this,
"Το όνομα της μετάβασης είναι πολύ μεγάλο!",
"Σφάλμα",
                                JOptionPane.WARNING_MESSAGE);
            return;
        }
        if ((Double) thresholSpinner.getValue() < 0.0 ||
            (Double) thresholSpinner.getValue() > 1.0) {
            JOptionPane.showMessageDialog(this,
                                "Ο βαθμός βεβαιότητας πρέπει να κυμαίνεται στο διάστημα
[0..1]",
"Σφάλμα",
                                JOptionPane.WARNING_MESSAGE);
            return;
        }

        transition.setName(transitionNameTF.getText());
        Double value = (Double) thresholSpinner.getValue();
        value = (double)Math.round(value * 1000) / 1000;
        transition.setThreshold(value);

        parent.setEnabled(true);
        parent.refresh();
this.dispose();

    }
```

```
    // Εκτελείται όταν ο χρήστης κλείνει το παράθυρο.
private void formWindowClosing(java.awt.event.WindowEvent evt) {

        parent.setEnabled(true);
        parent.refresh();
        this.dispose();

    }


    // Variables declaration - do not modify
    private javax.swing.JButton cancelButton;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JButton saveButton;
    private javax.swing.JSpinner thresholSpinner;
    private javax.swing.JTextField transitionNameTF;
    // End of variables declaration
}
```

Κλάσηgui.ArcPropertiesWindow

```
// Η κλάση ArcPropertiesWindow αντιστοιχεί στο παράθυρο του
// γραφικού περιβάλλοντος στο οποίο ο χρήστης μπορεί να θέσει τις
// παραμέτρους μίας ακμής.

packagegui;

import petrinet.Arc;

public class ArcPropertiesWindow extends javax.swing.JFrame {

// Η ακμή για την οποία θέτουμε τις παραμέτρους
privateArcarc;
    // Αναφορά στο γονικό παράθυρο του γραφικού περιβάλλοντος
private MainWindow parent;

    // Constructor
    public ArcPropertiesWindow(Arc arc, MainWindow parent) {
        this.arc = arc;
        this.parent = parent;
        initComponents();
        weightSpinner.setValue(arc.getWeight());
// Ρυθμίζουμε το παράθυρο να εμφανίζεται στο κέντρο της οθόνης
setLocationRelativeTo(null);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
```

```
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        weightSpinner = new javax.swing.JSpinner();
        saveButton = new javax.swing.JButton();
        cancelButton = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
        setTitle("Arc Properties");
        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                formWindowClosing(evt);
            }
        });

        jLabel1.setText("Weight:");

        weightSpinner.setModel(new javax.swing.SpinnerNumberModel(1, 1, 999, 1));

        saveButton.setText("Save");
        saveButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                saveButtonActionPerformed(evt);
            }
        });

        cancelButton.setText("Cancel");
        cancelButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(weightSpinner,   javax.swing.GroupLayout.PREFERRED_SIZE,   48,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
```

```
                .addContainerGap(224, Short.MAX_VALUE)
                .addComponent(cancelButton)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(saveButton)
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(weightSpinner,     javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,     23,
Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(saveButton)
                    .addComponent(cancelButton))
                .addContainerGap())
);

pack();
    }// </editor-fold>

    // Εκτελείται όταν ο χρήστης πατά το κουμπί cancel.
private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt) {

        parent.setEnabled(true);
        parent.refresh();
        this.dispose();

    }

    // Εκτελείται όταν ο χρήστης πατά το κουμπί save.
private void saveButtonActionPerformed(java.awt.event.ActionEvent evt) {

        arc.setWeight((Integer) weightSpinner.getValue());

        parent.setEnabled(true);
        parent.refresh();
        this.dispose();

    }

    private void formWindowClosing(java.awt.event.WindowEvent evt) {

        parent.setEnabled(true);
        parent.refresh();
        this.dispose();
```

```
    }

    // Variables declaration - do not modify
    private javax.swing.JButton cancelButton;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JButton saveButton;
    private javax.swing.JSpinner weightSpinner;
    // End of variables declaration
}
```

Κλάσηgui.FPNFileFilter

```
// Η κλάση FPNFileFilter φιλτράρει τα αρχεία τα οποία έχουν κατάληξη .fpn,
// σε έναν επιλογέα αρχείων.

packagegui;

import java.io.File;
import javax.swing.filechooser.FileFilter;

public class FPNFileFilter extends FileFilter {

// Ελέγχει αν ένα αρχείο, το οποίο δέχεται σαν όρισμα, είναι ένα
    // αρχείο .fpn και επιστρέφει true ή false ανάλογα.
@Override
    public boolean accept(File f) {
        if (f == null) {
            return false;
        }
        if (f.isDirectory()) {
            return true;
        }
        if (f.getName().toLowerCase().endsWith(".fpn")) {
            return true;
        } else {
            return false;
}
    }

    // Επιστρέφει την περιγραφή (κείμενο) ενός αρχείου .fpn.
@Override
    public String getDescription() {
        return "Fuzzy Petri Net files (*.fpn)";
    }

}
```

Κλάσηgui.FPNFileChooser

```
package gui;
```

```
import java.io.File;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

public class FPNFileChooser extends JFileChooser {

    // Constructor
    public FPNFileChooser() {
        FPNFileFilter xmlFiler = new FPNFileFilter();
        setFileFilter(xmlFiler);
}

    // Στην περίπτωση που το αρχείο ήδη υπάρχει, τότε θα πρέπει να εμφανίζεται
    // διάλογος επιβεβαίωσης για την αντικατάστασή του.
@Override
    public void approveSelection() {
        if (getDialogType() == SAVE_DIALOG) {
            File selectedFile = getSelectedFile();
            if ((selectedFile != null) && selectedFile.exists()) {
                int ret = JOptionPane.showConfirmDialog(this,
"Το αρχείο " + selectedFile.getName()
                        + " ήδη υπάρχει. Θέλετε να το αντικαταστήσετε;",
                        "Αντικατάσταση αρχείου", JOptionPane.YES_NO_OPTION,
JOptionPane.WARNING_MESSAGE);
                if (ret != JOptionPane.YES_OPTION) {
                    return;
                }
            }
        }
        super.approveSelection();
}

}
```

Κλάση gui.MainWindow

```
// Η κλάση MainWindow αντιστοιχεί στο κύριο παράθυρο του γραφικού περιβάλλοντος,
// το οποίο περιλαμβάνει τον καμβά σχεδίασης καθώς και όλες τις επιλογές
// σχεδίασης και διαχείρισης των αρχείων.

package gui;

import petrinet.PetriComponent;
import petrinet.PetriNet;
import petrinet.PetriNetFileHandler;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyListener;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
```

```
import java.io.File;
import javax.swing.BorderFactory;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.Timer;
import javax.swing.border.Border;

public class MainWindow extends javax.swing.JFrame {

// Flagto οποίο αποθηκεύει την πληροφορία για το αν ο χρήστης
// σχεδιάζει μία θέση
    private boolean placeCreationOn;
// Flagto οποίο αποθηκεύει την πληροφορία για το αν ο χρήστης
// σχεδιάζει μία μετάβαση
    private boolean transitionCreationOn;
// Flagto οποίο αποθηκεύει την πληροφορία για το αν ο χρήστης
// σχεδιάζει μία ακμή
    private boolean arcCreationOn;
// Flagto οποίο αποθηκεύει την πληροφορία για το αν η προσομοίωση
// είναι ενεργή
    private boolean simulationOn;
// Timer ο οποίος προσθέτει καθυστέρηση ανάμεσα στα βήματα
// της προσομοίωσης
    private Timer timer;
// Σταθερά η οποία αποθηκεύει το πρόθεμα (prefix) κάθε νέου αρχείου
private final String DEFAULT_FILE_NAME = "Untitled";
    // Το όνομα του αρχείου .fpn
    private String filename;
// Αναφορά προς το αρχείο .fpn
privateFilefpnFile;
    // Αύξων αριθμός για τον αριθμό του κάθε νέου αρχείου που δημιουργείται
privateintfileCounter;
    // Ο border σχεδίασης ενός κουμπιού στην περίπτωση που αυτό είναι
// μη επιλεγμένο
    private Border defaultBorder;

    // Constructor
    public MainWindow() {
        initComponents();
        timer = new Timer(1000, new ActionListener() {
              @Override
              public void actionPerformed(ActionEvent e) {
                  int fireCounter = ((PetriPanel) petriPanel).stepSimulation();
                  if (fireCounter == 0) {
                      timer.stop();
                  }
                  ((PetriPanel) petriPanel).repaint();
}
        });
statusLabel.setText("CursorON");
        // Ρυθμίζουμε το παράθυρο να εμφανίζεται στο κέντρο της οθόνης
```

```
setLocationRelativeTo(null);
        fileCounter = 1;
        filename = DEFAULT_FILE_NAME + fileCounter++ + ".fpn";
        this.setTitle(filename);
        defaultBorder = cursorButton.getBorder();
        cursorButton.setBorder(BorderFactory.createLoweredBevelBorder());
    }

    // Ανανεώνει τον τίτλο του παραθύρου.
    public void updateTitle() {
        if (!((PetriPanel) petriPanel).hasChanged())
            this.setTitle(filename);
        else
            this.setTitle(filename + " *");
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jToolBar1 = new javax.swing.JToolBar();
        cursorButton = new javax.swing.JButton();
        placeButton = new javax.swing.JButton();
        transitionButton = new javax.swing.JButton();
        arcButton = new javax.swing.JButton();
        bottomPanel = new javax.swing.JPanel();
        statusLabel = new javax.swing.JLabel();
        petriScrollPane = new javax.swing.JScrollPane();
        petriPanel = new PetriPanel(this);
        jToolBar2 = new javax.swing.JToolBar();
        simulationButton = new javax.swing.JButton();
        stepButton = new javax.swing.JButton();
        playButton = new javax.swing.JButton();
        pauseButton = new javax.swing.JButton();
        resetButton = new javax.swing.JButton();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();
        newFileMenuItem = new javax.swing.JMenuItem();
        openFileMenuItem = new javax.swing.JMenuItem();
        saveMenuItem = new javax.swing.JMenuItem();
        saveAsMenuItem = new javax.swing.JMenuItem();
        exitMenuItem = new javax.swing.JMenuItem();

        setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
```

```
            formWindowClosing(evt);
        }
    });

    jToolBar1.setFloatable(false);
    jToolBar1.setRollover(true);

    cursorButton.setText("Cursor");
    cursorButton.setFocusable(false);
    cursorButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    cursorButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
    cursorButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            cursorButtonActionPerformed(evt);
        }
    });
    jToolBar1.add(cursorButton);

    placeButton.setText("Place");
    placeButton.setFocusable(false);
    placeButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    placeButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
    placeButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            placeButtonActionPerformed(evt);
        }
    });
    jToolBar1.add(placeButton);

    transitionButton.setText("Transition");
    transitionButton.setFocusable(false);
    transitionButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    transitionButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
    transitionButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            transitionButtonActionPerformed(evt);
        }
    });
    jToolBar1.add(transitionButton);

    arcButton.setText("Arc");
    arcButton.setFocusable(false);
    arcButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    arcButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
    arcButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            arcButtonActionPerformed(evt);
        }
    });
    jToolBar1.add(arcButton);
```

```
        statusLabel.setText("jLabel1");

        javax.swing.GroupLayout bottomPanelLayout = new
javax.swing.GroupLayout(bottomPanel);
        bottomPanel.setLayout(bottomPanelLayout);
        bottomPanelLayout.setHorizontalGroup(

bottomPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(bottomPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(statusLabel)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        bottomPanelLayout.setVerticalGroup(

bottomPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(bottomPanelLayout.createSequentialGroup()
                .addComponent(statusLabel)
                .addGap(0, 11, Short.MAX_VALUE))
        );

        petriScrollPane.setDoubleBuffered(true);
        petriScrollPane.setMinimumSize(new java.awt.Dimension(0, 0));

        gui.PetriPanelMouseListener mouseListener = new
gui.PetriPanelMouseListener((PetriPanel) petriPanel);
        petriPanel.addMouseListener(mouseListener);
        gui.PetriPanelMouseMotionListener mouseMotionListener = new
gui.PetriPanelMouseMotionListener((PetriPanel) petriPanel);
        petriPanel.addMouseMotionListener(mouseMotionListener);
        gui.PetriPanelKeyListener keyListener = new gui.PetriPanelKeyListener((PetriPanel)
petriPanel);
        petriPanel.addKeyListener(keyListener);
        petriPanel.setBackground(new java.awt.Color(255, 255, 255));
        petriPanel.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        petriPanel.setAutoscrolls(true);
        petriPanel.setPreferredSize(new java.awt.Dimension(900, 900));

        javax.swing.GroupLayout petriPanelLayout = new javax.swing.GroupLayout(petriPanel);
        petriPanel.setLayout(petriPanelLayout);
        petriPanelLayout.setHorizontalGroup(
            petriPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 0, Short.MAX_VALUE)
        );
        petriPanelLayout.setVerticalGroup(
            petriPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 898, Short.MAX_VALUE)
        );

        petriScrollPane.setViewportView(petriPanel);
        petriPanel.setFocusable(true);
```

```
        jToolBar2.setFloatable(false);
        jToolBar2.setRollover(true);

        simulationButton.setText("Simulation");
        simulationButton.setFocusable(false);
        simulationButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        simulationButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        simulationButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                simulationButtonActionPerformed(evt);
            }
        });
        jToolBar2.add(simulationButton);

        stepButton.setText("Step");
        stepButton.setFocusable(false);
        stepButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        stepButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        stepButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                stepButtonActionPerformed(evt);
            }
        });
        jToolBar2.add(stepButton);

        playButton.setText("Play");
        playButton.setFocusable(false);
        playButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        playButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        playButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                playButtonActionPerformed(evt);
            }
        });
        jToolBar2.add(playButton);

        pauseButton.setText("Pause");
        pauseButton.setFocusable(false);
        pauseButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        pauseButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        pauseButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                pauseButtonActionPerformed(evt);
            }
        });
        jToolBar2.add(pauseButton);

        resetButton.setText("Reset");
        resetButton.setFocusable(false);
        resetButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
```

```
        resetButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        resetButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                resetButtonActionPerformed(evt);
            }
        });
        jToolBar2.add(resetButton);

        jMenu1.setText("File");


newFileMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.V
K_N, java.awt.event.InputEvent.CTRL_MASK));
        newFileMenuItem.setText("New File...");
        newFileMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                newFileMenuItemActionPerformed(evt);
            }
        });
        jMenu1.add(newFileMenuItem);


openFileMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.
VK_O, java.awt.event.InputEvent.CTRL_MASK));
        openFileMenuItem.setText("Open File...");
        openFileMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                openFileMenuItemActionPerformed(evt);
            }
        });
        jMenu1.add(openFileMenuItem);


saveMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_S
, java.awt.event.InputEvent.CTRL_MASK));
        saveMenuItem.setText("Save");
        saveMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                saveMenuItemActionPerformed(evt);
            }
        });
        jMenu1.add(saveMenuItem);

        saveAsMenuItem.setText("Save As...");
        saveAsMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                saveAsMenuItemActionPerformed(evt);
            }
        });
        jMenu1.add(saveAsMenuItem);

        exitMenuItem.setText("Exit");
```

```
        exitMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                exitMenuItemActionPerformed(evt);
            }
        });
        jMenu1.add(exitMenuItem);

        jMenuBar1.add(jMenu1);

        setJMenuBar(jMenuBar1);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(bottomPanel,                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(petriScrollPane,    javax.swing.GroupLayout.DEFAULT_SIZE,    745,
Short.MAX_VALUE)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jToolBar1,    javax.swing.GroupLayout.PREFERRED_SIZE,    156,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jToolBar2,               javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jToolBar1,   javax.swing.GroupLayout.PREFERRED_SIZE,   25,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jToolBar2,   javax.swing.GroupLayout.PREFERRED_SIZE,   25,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(petriScrollPane,   javax.swing.GroupLayout.DEFAULT_SIZE,   327,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(bottomPanel,           javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        );

        petriScrollPane.getVerticalScrollBar().setUnitIncrement(16);
        petriScrollPane.getHorizontalScrollBar().setUnitIncrement(16);

        pack();
    }// </editor-fold>

    // Απενεργοποιεί τις επιλογές όλων των κουμπιών.
public void unpressAll() {

        cursorButton.setBorder(defaultBorder);
```

```
        placeButton.setBorder(defaultBorder);
        transitionButton.setBorder(defaultBorder);
        arcButton.setBorder(defaultBorder);
        simulationButton.setBorder(defaultBorder);
        stepButton.setBorder(defaultBorder);
        playButton.setBorder(defaultBorder);
        pauseButton.setBorder(defaultBorder);

    }

    // Εκτελείται όταν ο χρήστης πατά το πλήκτρο δημιουργίας μίας θέσης.
private void placeButtonActionPerformed(java.awt.event.ActionEvent evt) {

        if (simulationOn)
            return;

        this.unpressAll();
        placeButton.setBorder(BorderFactory.createLoweredBevelBorder());

        PetriPanelMouseListener mouseListener = (PetriPanelMouseListener)
petriPanel.getMouseListeners()[0];
        mouseListener.clear();

        if (!placeCreationOn) {
            placeCreationOn = true;
            transitionCreationOn = false;
            arcCreationOn = false;
            mouseListener.setPlaceCreationOn(true);
        } else {
            placeCreationOn = false;
            mouseListener.setPlaceCreationOn(false);
        }

        statusLabel.setText("Place Creation ON");

}

    // Εκτελείται όταν ο χρήστης πατά το πλήκτρο δημιουργίας μίας μετάβασης.
private void transitionButtonActionPerformed(java.awt.event.ActionEvent evt) {

        if (simulationOn)
            return;

        this.unpressAll();
        transitionButton.setBorder(BorderFactory.createLoweredBevelBorder());

        PetriPanelMouseListener mouseListener = (PetriPanelMouseListener)
petriPanel.getMouseListeners()[0];
        mouseListener.clear();

        if (!transitionCreationOn) {
```

```
            transitionCreationOn = true;
            placeCreationOn = false;
            arcCreationOn = false;
            mouseListener.setTransitionCreationOn(true);
        } else {
            transitionCreationOn = false;
            mouseListener.setTransitionCreationOn(false);
        }

        statusLabel.setText("Transition Creation ON");

}

    // Εκτελείται όταν ο χρήστης πατά το πλήκτρο δημιουργίας μίας ακμής.
private void arcButtonActionPerformed(java.awt.event.ActionEvent evt) {

        if (simulationOn)
            return;

        this.unpressAll();
        arcButton.setBorder(BorderFactory.createLoweredBevelBorder());

        PetriPanelMouseListener mouseListener = (PetriPanelMouseListener)
petriPanel.getMouseListeners()[0];
        mouseListener.clear();

        if (!arcCreationOn) {
            arcCreationOn = true;
            placeCreationOn = false;
            transitionCreationOn = false;
            mouseListener.setArcCreationOn(true);
        } else {
            arcCreationOn = false;
            mouseListener.setArcCreationOn(false);
        }

        statusLabel.setText("Arc Creation ON");

}

    // Εκτελείται όταν ο χρήστης πατά το πλήκτρο επιλογής ενός στοιχείου.
private void cursorButtonActionPerformed(java.awt.event.ActionEvent evt) {

        if (simulationOn)
            return;

        this.unpressAll();
        cursorButton.setBorder(BorderFactory.createLoweredBevelBorder());

        PetriPanelMouseListener mouseListener = (PetriPanelMouseListener)
petriPanel.getMouseListeners()[0];
```

```
        mouseListener.setPlaceCreationOn(false);
        mouseListener.setTransitionCreationOn(false);
        mouseListener.setArcCreationOn(false);

        arcCreationOn = false;
        placeCreationOn = false;
        transitionCreationOn = false;

        statusLabel.setText("Selection ON");

}

    // Εκτελείται όταν ο χρήστης πατά το πλήκτρο της προσομοίωσης.
private void simulationButtonActionPerformed(java.awt.event.ActionEvent evt) {

        simulationOn = !simulationOn;
        ((PetriPanel) petriPanel).setSimulationOn(simulationOn);
        if (timer.isRunning())
            timer.stop();
        PetriPanelMouseListener       mouseListener       =       (PetriPanelMouseListener)
petriPanel.getMouseListeners()[0];
        mouseListener.clear();

        if (simulationOn) {
            this.unpressAll();
            simulationButton.setBorder(BorderFactory.createLoweredBevelBorder());
            statusLabel.setText("Simulation ON");
        } else {
            this.cursorButtonActionPerformed(null);
        }

    }

    // Εκτελείται όταν ο χρήστης πατά το πλήκτρο εκτέλεσης ενός
    // βήματος της προσομοίωσης.
    private void stepButtonActionPerformed(java.awt.event.ActionEvent evt) {

        if (!simulationOn)
            return;

        if (timer.isRunning())
            return;

        stepButton.setBorder(BorderFactory.createLoweredBevelBorder());
        pauseButton.setBorder(defaultBorder);

        ((PetriPanel) petriPanel).stepSimulation();

        this.refresh();

        stepButton.setBorder(defaultBorder);
```

```
}

    // Εκτελείται όταν ο χρήστης πατά το πλήκτρο εκτέλεσης όλης
// της προσομοίωσης συνολικά.
    private void playButtonActionPerformed(java.awt.event.ActionEvent evt) {

        if (!simulationOn)
            return;

        playButton.setBorder(BorderFactory.createLoweredBevelBorder());
        pauseButton.setBorder(defaultBorder);

        timer.start();

}

    // Εκτελείται όταν ο χρήστης πατά το πλήκτρο reset, έτσι ώστε το
    // FPN να επανέλθει στην αρχική του κατάσταση.
private void resetButtonActionPerformed(java.awt.event.ActionEvent evt) {

        if (timer.isRunning())
            return;

        ((PetriPanel) petriPanel).resetPetriNet();

        this.refresh();

    }

    // Εκτελείται όταν ο χρήστης πατά το πλήκτρο pause.
private void pauseButtonActionPerformed(java.awt.event.ActionEvent evt) {

        if (!simulationOn)
            return;

        if (!timer.isRunning())
            return;

        playButton.setBorder(defaultBorder);
        pauseButton.setBorder(BorderFactory.createLoweredBevelBorder());

        timer.stop();

        ((PetriPanel) petriPanel).repaint();
}

    // Εκτελείται όταν ο χρήστης επιλέγει την δημιουργία ενός νέου αρχείου.
private void newFileMenuItemActionPerformed(java.awt.event.ActionEvent evt) {

// Αν το τρέχον δίκτυο Petri έχει γίνει edit
```

```
if (((PetriPanel) petriPanel).hasChanged()) {
            // Επιβεβαίωση αποθήκευσης
            int choice = JOptionPane.showConfirmDialog(this,
"Το αρχείο " + filename + " έχει τροποποιηθεί\n"
                    + "Αποθήκευση του αρχείου;",
                    "Επιβεβαίωση αποθήκευσης",
JOptionPane.YES_NO_OPTION);

            if (choice == 0) {
                if (((PetriPanel) petriPanel).isSaved())
                    this.saveMenuItemActionPerformed(null);
                else
                    this.saveAsMenuItemActionPerformed(null);
            }

        }

        // Δημιουργία ενός νέου δικτύου
        filename = DEFAULT_FILE_NAME + fileCounter++ + ".fpn";
        this.setTitle(filename + " *");

        PetriNet petriNet = new PetriNet();
        PetriComponent.nextId = petriNet.findMaxId() + 1;
        PetriComponent.drawOn = true;
        ((PetriPanel) petriPanel).setPetriNet(petriNet);
        MouseListener mouseListener = petriPanel.getMouseListeners()[0];
        ((PetriPanelMouseListener)                  mouseListener).setPetriNet(((PetriPanel)
petriPanel).getPetriNet());
        MouseMotionListener mouseMotionListener = petriPanel.getMouseMotionListeners()[0];
        ((PetriPanelMouseMotionListener)      mouseMotionListener).setPetriNet(((PetriPanel)
petriPanel).getPetriNet());
        KeyListener keyListener = petriPanel.getKeyListeners()[0];
        ((PetriPanelKeyListener)                      keyListener).setPetriNet(((PetriPanel)
petriPanel).getPetriNet());
        this.refresh();
this.updateTitle();

    }

    // Εκτελείται όταν ο χρήστης επιλέγει την αποθήκευση του τρέχοντος αρχείου.
private void saveMenuItemActionPerformed(java.awt.event.ActionEvent evt) {

        try {
            if (!((PetriPanel) petriPanel).isSaved()) {
                this.saveAsMenuItemActionPerformed(null);
                return;
            }
            // Δημιουργία ενός νέου PetriNetFileHandler
            PetriNetFileHandler fileHandler = new PetriNetFileHandler();
            fileHandler.savePetriNet(((PetriPanel)               petriPanel).getPetriNet(),
fpnFile.getAbsolutePath());
            ((PetriPanel) petriPanel).setSaved(true);
```

```
            ((PetriPanel) petriPanel).setChanged(false);
            this.readFileName();
            this.updateTitle();
        } catch(Exception ex) {
            JOptionPane.showMessageDialog(this, ex.getMessage(), "Σφάλμα",
                                          JOptionPane.WARNING_MESSAGE);
        }

    }

    // Εκτελείται όταν ο χρήστης επιλέγει το άνοιγμα ενός αρχείου.
private void openFileMenuItemActionPerformed(java.awt.event.ActionEvent evt) {

        try {

            // Αν το τρέχον δίκτυο Petri έχει γίνει edit
if (((PetriPanel) petriPanel).hasChanged()) {
                // Επιβεβαίωση αποθήκευσης
                int choice = JOptionPane.showConfirmDialog(this,
"Το αρχείο " + filename + " έχει τροποποιηθεί\n"
                                + "Αποθήκευση του αρχείου;",
                        "Επιβεβαίωση αποθήκευσης",
JOptionPane.YES_NO_OPTION);

                if (choice == 0) {
                    if (((PetriPanel) petriPanel).isSaved())
                        this.saveMenuItemActionPerformed(null);
                    else
                        this.saveAsMenuItemActionPerformed(null);
                }

            }

            // Δημιούργησε έναν επιλογέα αρχείων για αρχεία FPN
            FPNFileChooser fchooser = new FPNFileChooser();

            // Επιλογή ενός αρχείου .fpn
            int ret = fchooser.showOpenDialog(this);
            if (ret == JFileChooser.APPROVE_OPTION) {
                // Πάρε το επιλεγμένο αρχείο
                File inputFPNFile = fchooser.getSelectedFile();
                fpnFile = inputFPNFile;
                // Ανάκτηση του δικτύου Petri
                PetriNetFileHandler fileHandler = new PetriNetFileHandler();
                fileHandler.readPetriNet((PetriPanel)                          petriPanel,
inputFPNFile.getAbsolutePath());
                MouseListener mouseListener = petriPanel.getMouseListeners()[0];
                ((PetriPanelMouseListener)          mouseListener).setPetriNet(((PetriPanel)
petriPanel).getPetriNet());
                MouseMotionListener                  mouseMotionListener                  =
petriPanel.getMouseMotionListeners()[0];
```

```
                ((PetriPanelMouseMotionListener)
mouseMotionListener).setPetriNet(((PetriPanel) petriPanel).getPetriNet());
                KeyListener keyListener = petriPanel.getKeyListeners()[0];
                ((PetriPanelKeyListener)             keyListener).setPetriNet(((PetriPanel)
petriPanel).getPetriNet());
                this.refresh();
                ((PetriPanel) petriPanel).setSaved(true);
                ((PetriPanel) petriPanel).setChanged(false);
                this.readFileName();
                this.updateTitle();
            }
        } catch(Exception ex) {
            JOptionPane.showMessageDialog(this, ex.getMessage(), "Σφάλμα",
                                          JOptionPane.WARNING_MESSAGE);
        }

    }

    // Εκτελείται όταν ο χρήστης επιλέγει την αποθήκευση του τρέχοντος αρχείου
// ως...
    private void saveAsMenuItemActionPerformed(java.awt.event.ActionEvent evt) {

        try {
            // Δημιούργησε έναν επιλογέα αρχείων για αρχεία FPN
FPNFileChooser fchooser = new FPNFileChooser();
            File saveFile;
            if (((PetriPanel) petriPanel).isSaved())
                saveFile = new File(fpnFile.getAbsolutePath());
            else
                saveFile = new File(filename);
            fchooser.setSelectedFile(saveFile);
            // Επιλογή ενός αρχείου .fpn
            int ret = fchooser.showSaveDialog(this);
            if (ret == FPNFileChooser.APPROVE_OPTION) {
                // Πάρε το επιλεγμένο αρχείο
                File outputFPNFile = fchooser.getSelectedFile();
                fpnFile = outputFPNFile;
                // Το αρχείο θα πρέπει να έχει την κατάληξη .fpn
if (!outputFPNFile.getName().toLowerCase().endsWith(".fpn")) {
                    outputFPNFile = new File(outputFPNFile + ".fpn");
                }
                // Δημιουργία ενός νέου PetriNetFileHandler
                PetriNetFileHandler fileHandler = new PetriNetFileHandler();
                fileHandler.savePetriNet(((PetriPanel)           petriPanel).getPetriNet(),
outputFPNFile.getAbsolutePath());
                ((PetriPanel) petriPanel).setSaved(true);
                ((PetriPanel) petriPanel).setChanged(false);
                this.readFileName();
                this.updateTitle();
            }
        } catch(Exception ex) {
```

```
            JOptionPane.showMessageDialog(this, ex.getMessage(), "Σφάλμα",
                                          JOptionPane.WARNING_MESSAGE);
        }

    }

    // Εκτελείται όταν ο χρήστης κλείσει το παράθυρο.
private void formWindowClosing(java.awt.event.WindowEvent evt) {

        if (((PetriPanel) petriPanel).hasChanged()) {
            // Επιβεβαίωση αποθήκευσης
            int choice = JOptionPane.showConfirmDialog(this,
"Το αρχείο " + filename + " έχει τροποποιηθεί\n"
                    + "Αποθήκευση του αρχείου;",
                    "Επιβεβαίωση αποθήκευσης",
JOptionPane.YES_NO_OPTION);

            if (choice == 0) {
                if (((PetriPanel) petriPanel).isSaved()) {
                    this.saveMenuItemActionPerformed(null);
                } else {
                    this.saveAsMenuItemActionPerformed(null);
                }
            }
        }

        this.dispose();
        System.exit(0);

    }

    // Εκτελείται όταν ο χρήστης επιλέγει την επιλογή εξόδου από το menu.
private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {

        this.formWindowClosing(null);

    }

    // Ανανεώνει τον καμβά.
    public void refresh() {
        petriPanel.repaint();
}

    // Εξάγει το όνομα του αρχείου από το απόλυτο μονοπάτι αυτού.
private void readFileName() {

        String[] parts = fpnFile.getAbsolutePath().split("\\\\");
        filename = parts[parts.length-1];

    }
```

```
    // Variables declaration - do not modify
    private javax.swing.JButton arcButton;
    private javax.swing.JPanel bottomPanel;
    private javax.swing.JButton cursorButton;
    private javax.swing.JMenuItem exitMenuItem;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JToolBar jToolBar1;
    private javax.swing.JToolBar jToolBar2;
    private javax.swing.JMenuItem newFileMenuItem;
    private javax.swing.JMenuItem openFileMenuItem;
    private javax.swing.JButton pauseButton;
    private javax.swing.JPanel petriPanel;
    private javax.swing.JScrollPane petriScrollPane;
    private javax.swing.JButton placeButton;
    private javax.swing.JButton playButton;
    private javax.swing.JButton resetButton;
    private javax.swing.JMenuItem saveAsMenuItem;
    private javax.swing.JMenuItem saveMenuItem;
    private javax.swing.JButton simulationButton;
    private javax.swing.JLabel statusLabel;
    private javax.swing.JButton stepButton;
    private javax.swing.JButton transitionButton;
    // End of variables declaration
}
```