ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Σύνθεση Υπηρεσιών Με Χρήση Petri Nets

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΚΩΣΤΕΛΛΟΥ ΜΑΡΙΟΥ-ΧΡΗΣΤΟΥ

Επιβλέπων : Γρηγόριος Μέντζας
Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Σύνθεση Υπηρεσιών Με Χρήση Petri Nets

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΚΩΣΤΕΛΛΟΥ ΜΑΡΙΟΥ-ΧΡΗΣΤΟΥ

Επιβλέπων : Γρηγόριος Μέντζας
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 24η Φεβρουαρίου 2009.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γρηγόριος Μέντζας Καθηγητής Ε.Μ.Π.	Δημήτριος Ασκούνης Καθηγητής Ε.Μ.Π.	Ιωάννης Ψαρράς Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2009

(Υπογραφή)

.....

ΚΩΣΤΕΛΛΟΣ ΜΑΡΙΟΣ-ΧΡΗΣΤΟΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τα τελευταία χρόνια η τεχνολογία των web services έχει γίνει ο ακρογωνιαίος λίθος της δημιουργίας πληροφοριακών συστημάτων. Ένα απλό Web service εκτελεί μία συγκεκριμένη εργασία ενώ μια σύνθετη βασίζεται σε άλλες Web services. Η συμπεριφορά μίας Web service είναι βασικά ένα μερικά διατεταγμένο σύνολο λειτουργιών. Γι' αυτό, είναι εύκολο να μοντελοποιηθούν με την χρήση των Petri Nets. Τα Petri Nets (PN) είναι ένα εποπτικός τρόπος αναπαράστασης που εξυπηρετεί την τυποποιημένη περιγραφή της ροής των διαδικασιών σε ένα πολύπλοκο σύστημα. Εμβαθύνουμε στην ανάλυση καταστάσεων (state space analysis) που είναι ουσιαστικά το εργαλείο με το οποίο κρίνουμε για την ικανότητα σύνθεσης δύο ή περισσοτέρων υπηρεσιών και αναλύουμε το αλγεβρικό μοντέλο αναπαράστασης των web services. Τέλος παρατίθεται ένα παράδειγμα σύνθεσης τεσσάρων υπηρεσιών σε μια σύνθετη βασιζόμενοι στην επιμέρους μοντελοποίηση των υπηρεσιών αυτών.

Abstract

Last years the web services technology has become the cornerstone of the information system development. A simple web service runs a certain task, while a composite web service based on other web services. The behavior of a web service is a kind of ordered set of functions. Therefore, is easy to modeled with the use of Petri Nets. Petri Nets (PN) is a surveillant way to represent, which serves the formal description of the flow of the procedures in a complicated system. Then, we take a look at the state space analysis, which is the tool that helps us to decide if we can combine two or more web services and analyze the algebraic model of the representation of web services. Finally, we give an example of the combination of four different web services to one composite web service, based on the model of each simple web service.

Πίνακας Περιεχομένων

Πρόλογος

Τα τελευταία χρόνια η τεχνολογία των web services έχει γίνει ο ακρογωνιαίος λίθος της δημιουργίας πληροφοριακών συστημάτων. Στα πλαίσια αυτά υπάρχει η όλο και αυξανόμενη τάση της μεταφοράς της επιχειρησιακής λογικής των εφαρμογών σε αυτοτελή web services. Προκειμένου το B2B ηλεκτρονικό εμπόριο να μπορέσει να προηχθεί, υπάρχει η ανάγκη να συνθέτονται και να εξελίσσονται ημιαυτόματα Web services σε ένα λογικό χρονικό πλαίσιο. Αυτό έχει γεννήσει την ιδέα των 'σύνθετων' web services. Ένα απλό Web service εκτελεί μία συγκεκριμένη εργασία ενώ μια σύνθετη βασίζεται σε άλλες Web services.

Αυτό σημαίνει ότι οι υπάρχουσες υπηρεσίες είναι ικανές να συνεργαστούν παρόλο που η συνεργασία δεν είχε σχεδιαστεί εκ των προτέρων. Η σύνθεση υπηρεσιών μπορεί να είναι στατική (οι συνιστώσες της υπηρεσίας αλληλεπιδρούν η μία με την άλλη με ένα προκαθορισμένο τρόπο) ή δυναμική (ανακαλύπτουν η μία την άλλη και διαπραγματεύονται on the fly).

Όσο αφορά την σύνθεση των υπηρεσιών, έχουν προταθεί στην βιβλιογραφία πολλές τεχνικές για την επίτευξη της. Οι υπάρχουσες ad-hoc λύσεις έχουν εμποδίσει σε μεγάλο βαθμό την ταχύτερη αξιοποίηση των υφιστάμενων υπηρεσιών.

Οι σύγχρονες τεχνολογίες βασισμένες σε πρότυπα όπως το UDDI, το WSDL, και το SOAP δεν είναι ικανές να μοντελοποιήσουν όλες τις παραμέτρους μια υπηρεσίας που είναι απαραίτητες για την σύνθεση της σε μια πιο σύνθετη υπηρεσία. Οι απαραίτητες παράμετροι σχετίζονται με την συμπεριφορά μιας υπηρεσίας (behavioral aspects). Ως εκ τούτου, τα πρότυπα αυτά, παρέχουν περιορισμένη υποστήριξη στην σύνθεση υπηρεσιών.

Η συμπεριφορά μίας Web service είναι βασικά ένα μερικά διατεταγμένο σύνολο λειτουργιών. Γι' αυτό, είναι εύκολο να το μοντελοποιήσει κάποιος με την χρήση των Petri Nets. Τα Petri Nets (PN) είναι ένα εποπτικός τρόπος αναπαράστασης που εξυπηρετεί την τυποποιημένη περιγραφή της ροής των διαδικασιών σε ένα πολύπλοκο σύστημα.

Στα πλαίσια της παρούσας πτυχιακής εργασίας ασχολούμαστε με την ημιαυτόματη σύνθεση των υπηρεσιών (web services) χρησιμοποιώντας το μαθηματικό υπόβαθρο των Petri-Nets και ειδικότερα το State space analysis μιας υποκατηγορίας των Coloured-Petri Nets. Η γενικότερη ιδέα είναι ότι με δεδομένη μια μοντελοποίηση σε PetriNets κάποιων υπηρεσιών (ενός συστήματος υπηρεσιών) μπορούμε από την ανάλυση όλων των πιθανών καταστάσεων στις οποίες αυτά περιέρχονται να κρίνουμε για το κατά πόσο οι υπηρεσίες αυτές μπορούν να

συνεργαστούν προκειμένου να συστήσουν μια καινούρια ‘σύνθετη’ υπηρεσία.

Η οργάνωση της εργασίας έχει ως ακολούθως: στο πρώτο κεφάλαιο κάνουμε μια εισαγωγή στις έννοιες των Petri Nets. Στο δεύτερο κεφάλαιο αναλύουμε πιο διεξοδικά τα Coloured Petri Nets που αποτελούν τη βάση του μαθηματικού μας μοντέλου. Στο τρίτο κεφάλαιο εμβαθύνουμε στην ανάλυση καταστάσεων (state space analysis) που είναι ουσιαστικά το εργαλείο με το οποίο κρίνουμε για την ικανότητα σύνθεση δύο ή περισσοτέρων υπηρεσιών. Στο τέταρτο κεφάλαιο αναλύεται το αλγεβρικό μοντέλο αναπαράστασης των web services. Τέλος στο τελευταίο κεφάλαιο παρατίθεται ένα παράδειγμα σύνθεσης τεσσάρων υπηρεσιών σε μια σύνθετη βασιζόμενοι στην επιμέρους μοντελοποίηση των υπηρεσιών αυτών.

I

Κεφάλαιο

Εισαγωγή στα Petri Nets

1 Εισαγωγή στα Petri Nets

1.1 Εισαγωγή

Τα Petri Nets (PN) είναι ένα εποπτικός τρόπος αναπαράστασης που εξυπηρετεί την τυποποιημένη περιγραφή της ροής των διαδικασιών σε ένα πολύπλοκο σύστημα. Συγκρινόμενα με άλλες πιο δημοφιλείς τεχνικές εποπτικής αναπαράστασης συστημάτων (όπως block διαγράμματα ή λογικά δέντρα), τα PN είναι ιδιαίτερα κατάλληλα για να αναπαριστούν με φυσικό τρόπο λογικές αντιδράσεις μεταξύ μερών ή διαδικασιών ενός συστήματος. Συνηθισμένες καταστάσεις που μπορούν να μοντελοποιηθούν με την χρήση των PN είναι ο συγχρονισμός (concurrency) και η σύγκρουση (conflict).

Η θεωρία των PN κατάγεται στην διδακτορική διατριβή του C.A.Petri το 1962[1]. Από τότε, η τυποποιημένη γλώσσα των PN έχει αναπτυχθεί και χρησιμοποιηθεί σε πολλές θεωρητικές και εφαρμόσιμες περιοχές.

Τα κλασσικά PN δεν φέρουν καμία έννοια του χρόνου. Προκειμένου όμως να χρησιμοποιήσουμε την τυποποίηση των PN για την ποσοτική ανάλυση της απόδοσης και της ρεαλιστικότητας των συστημάτων ενάντια στο χρόνο, εισήχθηκε η έννοια των Timed PN (TPN). Οι χρονικές μεταβλητές που σχετίζονται με τα PN μπορούν νε είναι είτε ντετερμινιστικές (deterministic) μεταβλητές (οδηγώντας στην κατηγορία των μοντέλων που ονομάζονται ντετερμινιστικά PN), είτε τυχαίες (random) μεταβλητές (οδηγώντας στην κατηγορία των μοντέλων που ονομάζονται Stochastic PN-SPN).

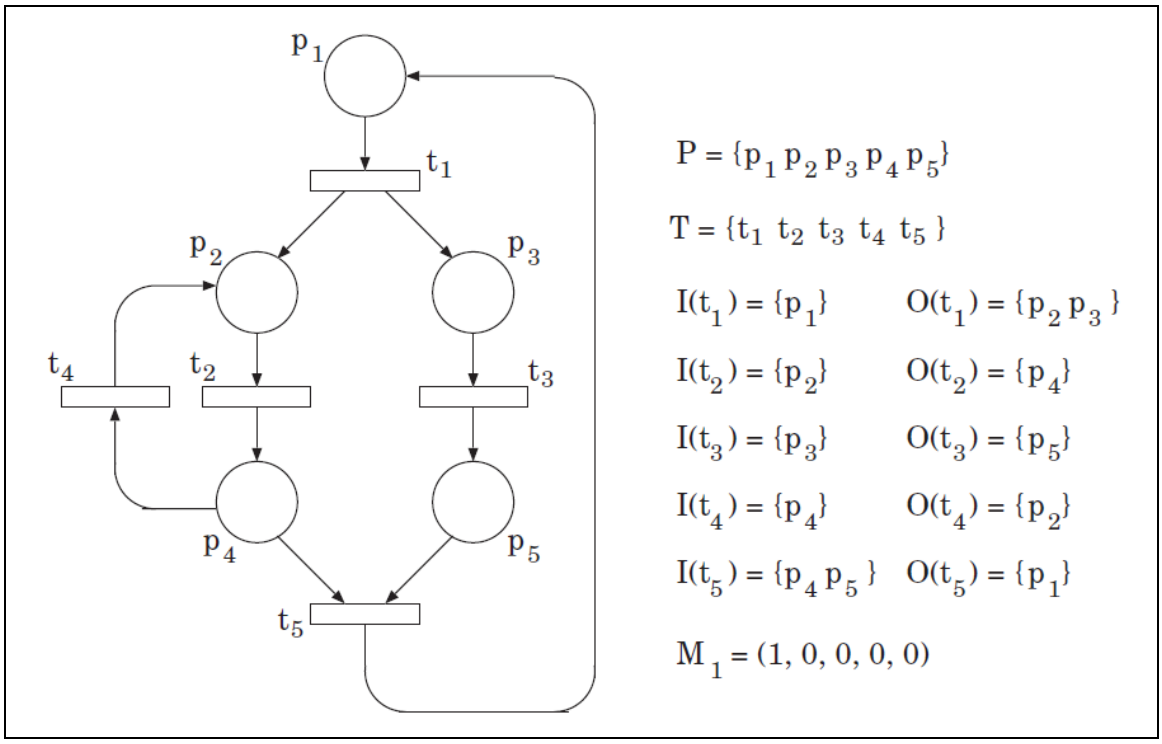
1.2 Τα βασικά στοιχεία ενός Petri net

Ένα σημαδεμένο (marked) PN είναι ένα πενταπλό σύνολο (quintuple), όπου :

- $P = \{p_1, p_2, \ldots, p_{np}\}$ είναι το σύνολο των n_p places (σχεδιασμένοι ως κύκλοι στην γραφική αναπαράσταση),
- $T = \{ t_1, t_2, \ldots, t_{nt}\}$ είναι το σύνολο των n_t transitions (σχεδιασμένοι ως γραμμές),
- I είναι η σχέση εισόδου του transition και αναπαριστάται με ένα βέλος κατευθυνόμενο από τον place στον transition,
- O είναι η σχέση εξόδου του transition και αναπαριστάται με ένα βέλος κατευθυνόμενο από τον transition στον place,
- $M = \{m_1, m_2, \ldots, m_{np}\}$ είναι η σήμανση (marking). Η γενική είσοδος m_i είναι ο αριθμός των token (σχεδιασμένα ως μαύρες κουκίδες) στον place p_i στη σήμανση M.

Κωστέλλος Μαριος-Χρηστος
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Η γραφική δομή ενός PN είναι ένας διμερής προσανατολισμένος γράφος: οι κόμβοι ανήκουν σε δύο διαφορετικές κλάσεις (places και transitions) και οι ακμές (βέλη) επιτρέπεται να ενώνουν μόνο κόμβους διαφορετικής κλάσης (πολλαπλά βέλη είναι δυνατό να υπάρχουν στον ορισμό των I και O σχέσεων). Το σχήμα 1.1 είναι ένα PN[2].



Σχήμα 1.1. Ένας γράφος PN με συναρτήσεις εισόδου και εξόδου

Η δυναμική των PN λαμβάνεται μετακινώντας τα token στους places με τις έννοιες των ακόλουθων κανόνων εκτέλεσης :

- Ένας transition είναι ενεργός στην σήμανση *M* αν όλες του οι place εισόδου έχει τουλάχιστο ένα token,
- ένας ενεργός transition πυροδοτείται μετακινώντας ένα token ανά βέλος από κάθε place εισόδου και προσθέτοντας ένα token ανά βέλος για κάθε place εξόδου.

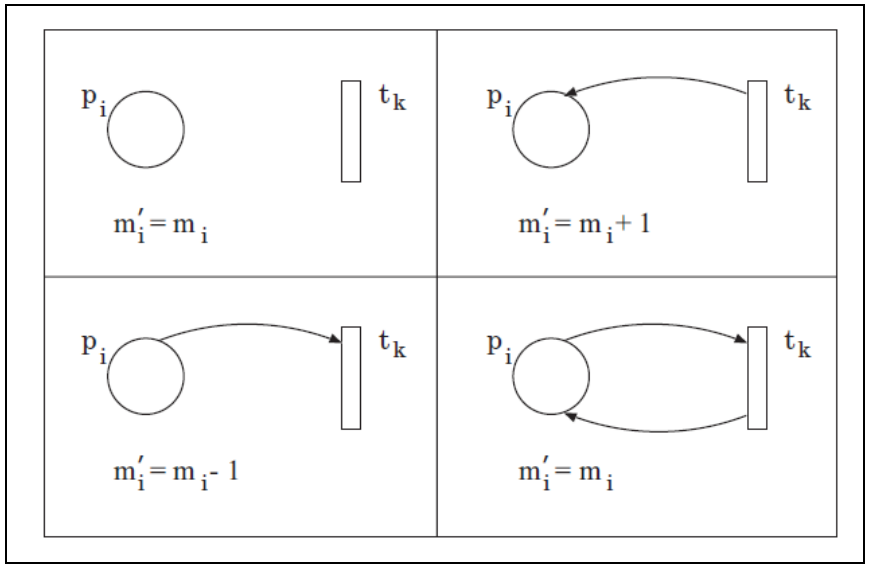
Δοθείσας μίας αρχικής σήμανσης M_1 , το σύνολο προσεγγισιμότητας $R(M_1)$ είναι το σύνολο όλων των σημάνσεων που μπορούμε να λάβουμε με επαναληπτική εφαρμογή των παραπάνω κανόνων. Περισσότερο τυπικά μπορούμε να λέμε ότι ο t_k είναι ενεργός στη σήμανση M αν :

Για κάθε p_i Ε $I(t_k)$, $m_i \geq 1$

Η σήμανση Μ', που λαμβάνεται από την Μ με πυροδότηση του t_k, λέγεται ότι είναι απευθείας προσεγγίσιμη από την Μ, και η λειτουργία πυροδότησης δηλώνεται από το σύμβολο (M - t_k -> M'). Το token που υπάρχει στην Μ' αναπαριστάται με εικόνες στο σχήμα 1.2, και δίνεται από την ακόλουθη σχέση :

$$M'(p_i) = \begin{cases} M(p_i) + 1 & \text{if} \quad p_i \in O(t_k)\,, \quad p_i \notin I(t_k) \\ M(p_i) - 1 & \text{if} \quad p_i \notin O(t_k)\,, \quad p_i \in I(t_k) \\ M(p_i) & \text{otherwise} \end{cases}$$

Ας εξετάσουμε την γένεση του συνόλου προσεγγισιμότητας του PN του σχήματος 1.1 δοθείσας της αρχικής σήμανσης M_1 = (1,0,0,0,0). Στην M_1 ο μόνος ενεργός transition είναι ο t_1, η πυροδότηση του t_1 μετακινεί το token από τον p_1 και το τοποθετεί ένα token στον p_2 και p_3 παράγοντας την νέα σήμανση M_2= (0,1,1,0,0).



Σχήμα 1.2. Τροποποίηση του αριθμού των token στον place p_i μετά την πυροδότηση του transition t_k

Στην M_2 οι transitions t_2 και t_3 είναι και οι δύο ενεργοί και μπορούν να πυροδοτηθούν ταυτόχρονα. Η πυροδότηση του t_3 οδηγεί στην M_3= (0,1,0,0,1) και μετά η πυροδότηση του t_2 οδηγεί στην M_4= (0,0,0,1,1). Στην M_4 οι transitions t_4 και t_5 είναι και οι δύο ενεργοί, αλλά η πυροδότηση του ενός απενεργοποιεί τον άλλο, οι δύο transitions είναι σε σύγκρουση. Η πυροδότηση του t_4 στην M_4 παράγει την σήμανση M_3 , ενώ η πυροδότηση του t_4 στην M_4 παράγει την αρχική σήμανση M_1 . Σημειώνουμε ότι μία

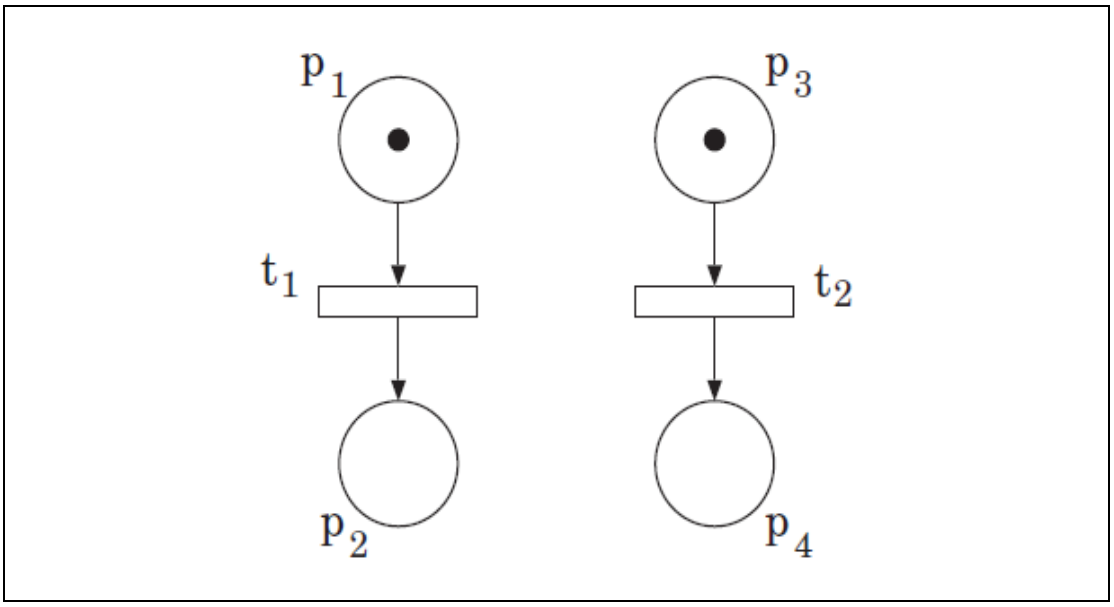
διαφορετική σειρά πυροδότησης μπορεί να ενεργοποιηθεί από την σήμανση M_2 , αφήνοντας τον t_2 να πυροδοτήσει πρώτος και να λάβουμε την σήμανση M_5= (0,0,1,1,0). Με αυτό, όλες οι πιθανές σειρές πυροδότησης έχουν εξεταστεί, και το σύνολο προσεγγισιμότητας $R(M_1)$ του δικτύου του σχήματος 1.1 καταλήγει να έχει 5 στοιχεία M_1 ,M_2 ,M_3 ,M_4 ,και M_5 .

1.3 Τα Petri net και η μοντελοποίηση συστημάτων

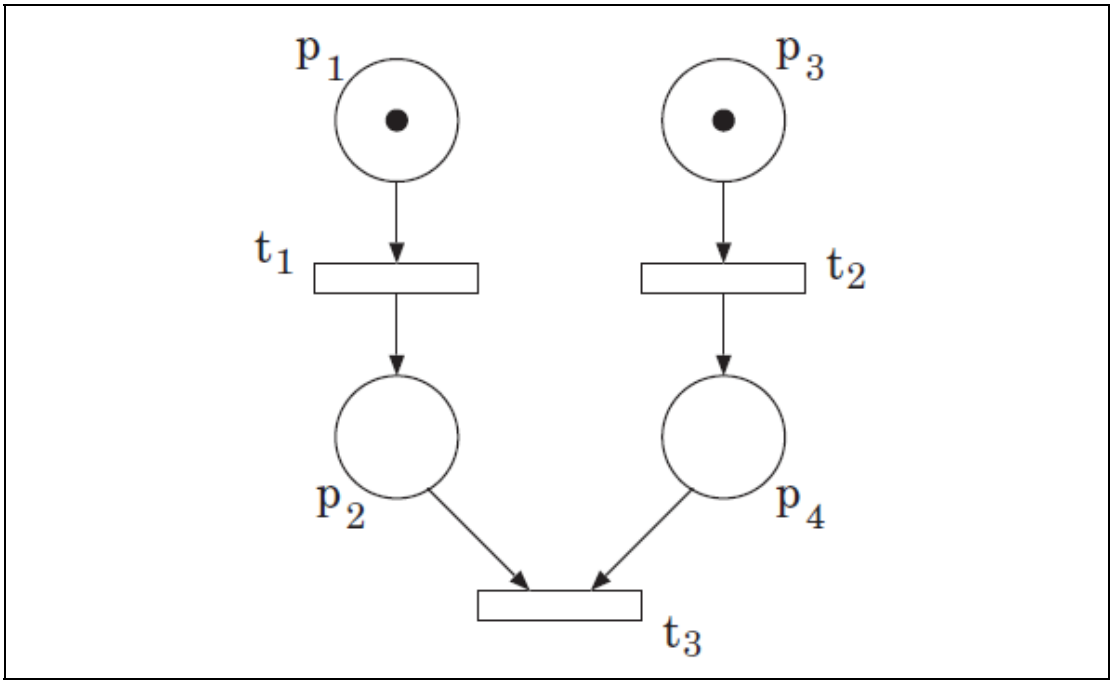
Τα PN που χρησιμοποιούνται για την μοντελοποίηση πραγματικών συστημάτων συχνά αναφέρονται ως *Condition/Events* δίκτυα. Οι places προσδιορίζουν τις συνθήκες (conditions) από τα μέρη του συστήματος (απασχολημένα, αδρανή, σε αναμονή, λανθασμένα), και οι transitions περιγράφουν το πέρασμα από μία συνθήκη σε μία άλλη (τέλος εργασίας, αποτυχία, επισκευή...). Ένα γεγονός λαμβάνει δράση (πυροδότηση transition) όταν όλες οι συνθήκες ικανοποιούνται και δίνει άδεια στο γεγονός. Το συμβάν ενός γεγονότος τροποποιεί εντελώς ή σε κομμάτια την κατάσταση των συνθηκών (σήμανση). Ο αριθμός των token σε ένα place μπορεί να χρησιμοποιηθεί για την αναγνώριση του αριθμού των πόρων που υπάρχουν στην συνθήκη που δηλώθηκε από τον place αυτό.

1.3.1 Concurrency (Παραλληλισμός)

Στο PN του σχήματος 1.3 οι transitions t_1 και t_2 είναι ενεργοί ταυτόχρονα, δηλαδή η πυροδότηση του ενός δεν τροποποιεί την κατάσταση του άλλου. Οι δραστηριότητες που μοντελοποιούνται από τους δύο transitions τρέχουν ταυτόχρονα. Σε αξιόπιστη μοντελοποίηση, το PN του σχήματος 1.3 μπορεί να αναπαριστά δύο μέρη C_1 και C_2 σε παράλληλη εκτέλεση, σε αυτή την περίπτωση, οι places p_1 και p_3 αντιπροσωπεύουν την συνθήκη που εργάζεται, οι p_2 και p_4 την λανθασμένη συνθήκη και οι t_1 και t_2 γεγονός αποτυχίας των C_1 και C_2 αντίστοιχα.



Σχήμα 1.3. Το PN μου μοντελοποιεί δύο παράλληλες διαδικασίες



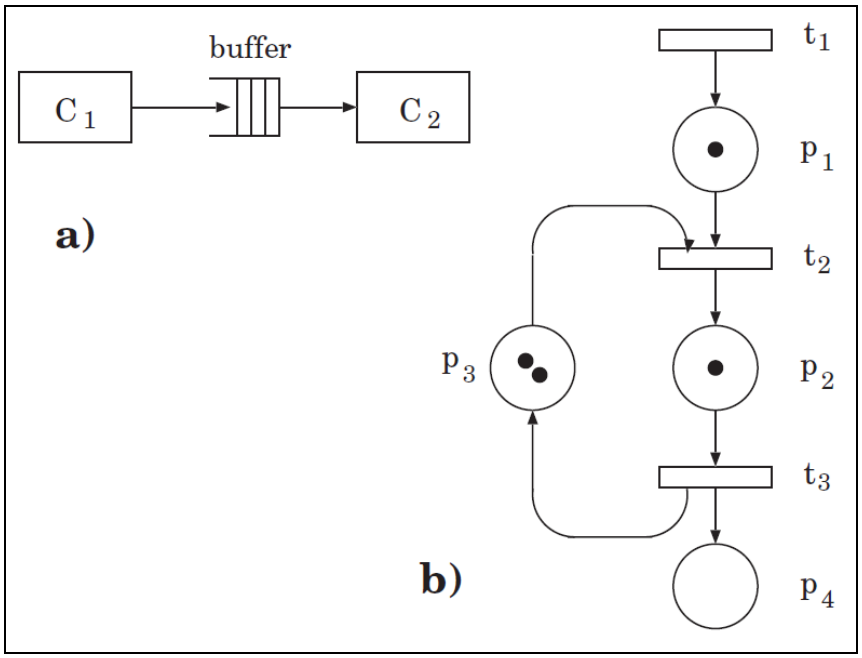
Σχήμα 1.4. Το PN που μοντελοποιεί δύο παράλληλες διαδικασίες με συγχρονισμό

1.3.2 Synchronization (Συγχρονισμός)

Στο σχήμα 1.3 οι διαδικασίες μοντελοποιούνται με τους t_1 και t_2 να τρέχουν ταυτόχρονα, ωστόσο, αν αντιπροσώπευαν ρουτίνες ενός παράλληλου προγράμματος, και οι δύο θα έπρεπε να τερματίσουν πριν η εκτέλεση του προγράμματος να μπορεί να προχωρήσει. Ο συγχρονισμένη δραστηριότητα μοντελοποιείται στο σχήμα 1.4 με την έννοια του transition t_3 του οποίου η πυροδότηση απαιτεί ένα token και στον p_2 και p_4.

1.3.3 Limited resources (περιορισμένοι πόροι)

Ένας συνηθισμένος παράγοντας που επηρεάζει την απόδοση κατανεμημένων συστημάτων (συστήματα πολλών επεξεργαστών, ευέλικτα κατασκευαστικά συστήματα και άλλα) είναι ο περιορισμένος αριθμός διαθέσιμων πόρων.



Σχήμα 1.5. Block διάγραμμα και το PN για ένα buffer με πεπερασμένο μέγεθος

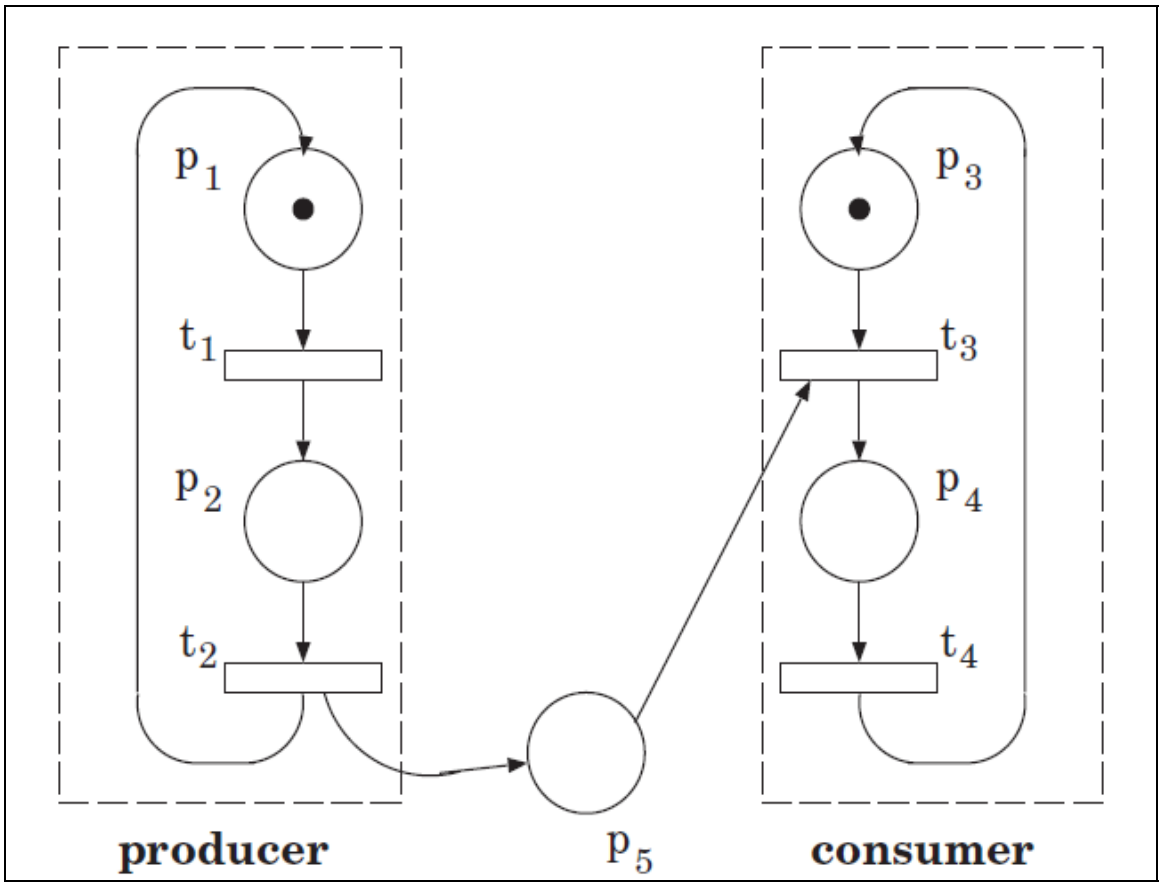
Η εξάντληση των πόρων εμποδίζει τις διαδικασίες να συνεχίσουν την διεργασία και μπλοκάρουν το σύστημα. Η μοντελοποίηση συστημάτων που έχουν μπλοκάρει είναι δύσκολο αντικείμενο σχεδόν σε όλες τις τεχνικές μοντελοποίησης[3,4]. Η αναπαράσταση PN ενός buffer με περιορισμένο μέγεθος φαίνεται στο σχήμα 1.5b (το σχήμα 1.5a δείχνει την αναπαράσταση σε block διάγραμμα).Ο place p_3 μοντελοποιεί τον αριθμό

των ελεύθερων θέσεων στο buffer ενώ ο p_2 τον αριθμό των συμπληρωμένων θέσεων, σημειώνουμε ότι το άθροισμα των token στους p_2 και p_3 είναι σταθερός και μοντελοποιεί το συνολικό αριθμό των διαθέσιμων θέσεων στο buffer (τρείς θέσεις στο σχήμα). Ο transition t_2 μοντελοποιεί το γέμισμα μίας θέσης του buffer και πυροδοτείται αν υπάρχει μία θέση κενή (τουλάχιστο ένα token στον p_3) και μία εργασία είναι διαθέσιμη να αποθηκευτεί (τουλάχιστο ένα token στον p_1). Ο transition t_3 είναι ενεργός όταν τουλάχιστο μία θέση στο buffer είναι γεμάτη, και η πυροδότηση του μετακινεί ένα token από τον p_2 στον p_3 .

1.3.4 Sequentiality (το πρόβλημα παραγωγού/καταναλωτή)

Ένας παραγωγός παράγει αντικείμενα τα οποία τοποθετούνται σε ένα buffer από το οποίο μπορούν να μετακινηθούν και να καταναλωθούν από ένα καταναλωτή. Η διαδικασία κατανάλωσης πρέπει να είναι σε σειρά με σεβασμό στην διαδικασία παραγωγής. Η λύση με χρήση PN σε αυτό το πρόβλημα αναφέρεται στο σχήμα 1.6. Ένα token στον p_1 σημαίνει ότι ο παραγωγός είναι έτοιμος να παράγει. Με την πυροδότηση των t_1 και t_2 ένα αντικείμενο παράγεται (ένα token τοποθετείται στο buffer p_5) και ο παραγωγός είναι έτοιμος πάλι. Αν ο καταναλωτής είναι έτοιμος να καταναλώσει (token στον p_3) και υπάρχει ένα αντικείμενο στο buffer, ο transition t_3 μπορεί να πυροδοτήσει μετακινώντας ένα token από τον p_5 .

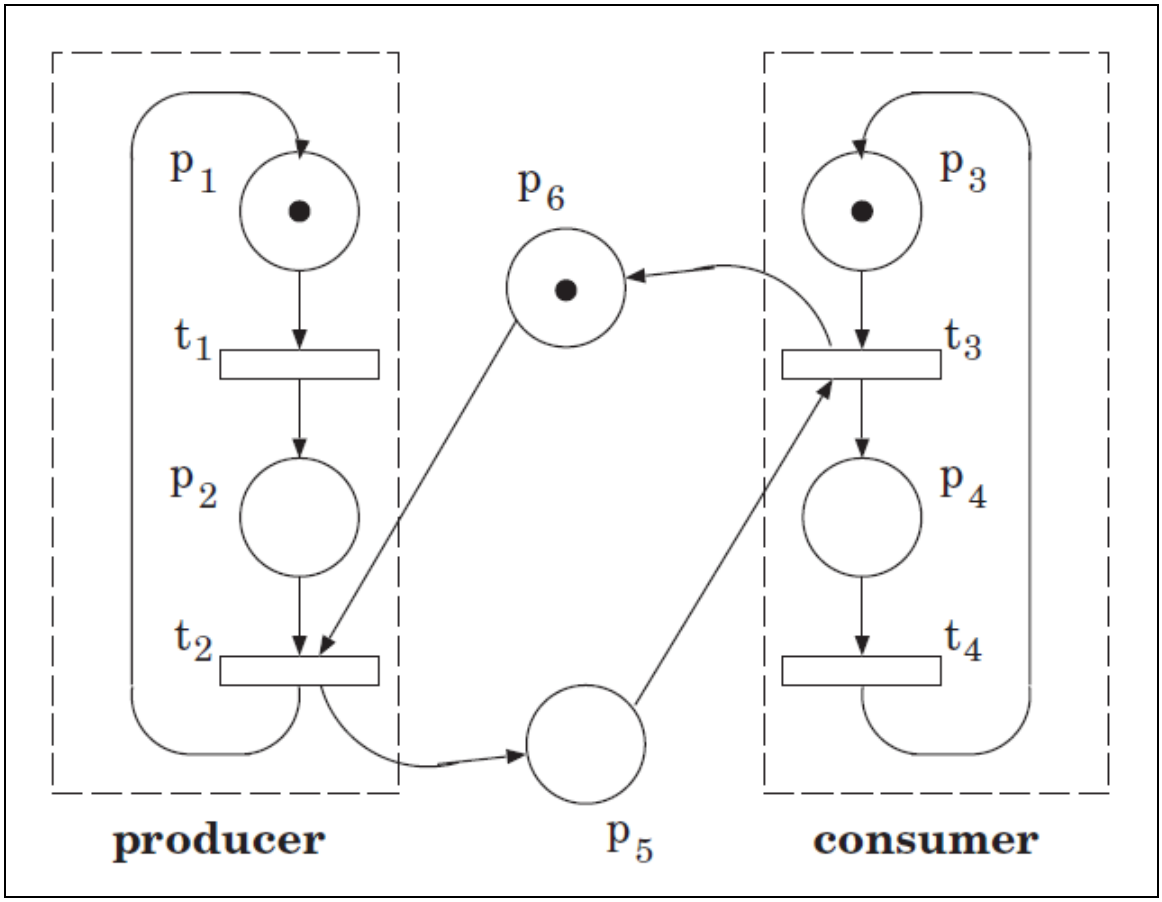
Στο PN του σχήματος 1.6 η παραγωγή και η συσσώρευση αντικειμένων στο buffer είναι χωρίς όριο. Μία πιο πραγματική κατάσταση μπορεί να ληφθεί αν χρησιμοποιήσουμε ένα buffer με περιορισμένη χωρητικότητα (όπως στην προηγούμενη παράγραφο). Το σχετικό PN παρατίθεται στο σχήμα 1.7. Ο place p_6 μοντελοποιεί τις ελεύθερες θέσεις του buffer και ο place p_5 μοντελοποιεί τις συμπληρωμένες θέσεις του buffer, ο αριθμός των token στον p_5 και τον p_6 είναι σταθερός και αντιπροσωπεύει τις συνολικές θέσεις που είναι διαθέσιμες από το buffer. Αν ένα από token τοποθετηθεί στον p_6 στην αρχική σήμανση, μοντελοποιούμε την κατάσταση όπου ο παραγωγός δεν μπορεί να παράγει πλέον μέχρι ο καταναλωτής να καταναλώσει το αντικείμενο στο buffer (μία αυστηρώς διαδοχική διάταξη των διαδικασιών).



Σχήμα 1.6. Το πρόβλημα παραγωγού/καταναλωτή με buffer χωρίς όριο

1.3.5 Mutual exclusion (conflict)

Δύο διαδικασίες C_1 και C_2 επιτρέπετε να δουλεύουν παράλληλα, αλλά συνδέονται σε μία κοινή διαδικασία C_s η οποία δεν μπορεί να είναι προσβάσιμη από τις C_1 και C_2 ταυτόχρονα (block διάγραμμα του σχήματος 1.8a). Το σχετικό PN είναι στο σχήμα 1.8b. Οι places p_1 και p_5 αντιπροσωπεύουν τις C_1 και C_2 που δουλεύουν ανεξάρτητα, οι p_2 και p_6 αντιπροσωπεύουν τις C_1 και C_2 που ζητούν πρόσβαση στην C_s , οι p_3 και p_7 αντιπροσωπεύουν την C_s απασχολημένη με τις C_1 και C_2 αντίστοιχα. Ο place p_4 προσδιορίζει ποιά διαδικασία μπορεί πραγματικά να έχει πρόσβαση στην C_s , και εμποδίζει τους places p_3 και p_7 να είναι σημαδεμένοι την ίδια στιγμή, στην πραγματικότητα όταν ο p_2 και ο p_6 είναι και οι δύο σημαδεμένοι έχουμε σύγκρουση (conflict). Η πυροδότηση του ενός από αυτούς απενεργοποιεί τον άλλο. Η πυροδότηση των t_3 και t_6 μοντελοποιεί την απελευθέρωση της κοινής διαδικασίας (το token πίσω στον p_4) και την επιστροφή σε λειτουργική συνθήκη.



Σχήμα 1.7. Το πρόβλημα παραγωγού/καταναλωτή με πεπερασμένο buffer

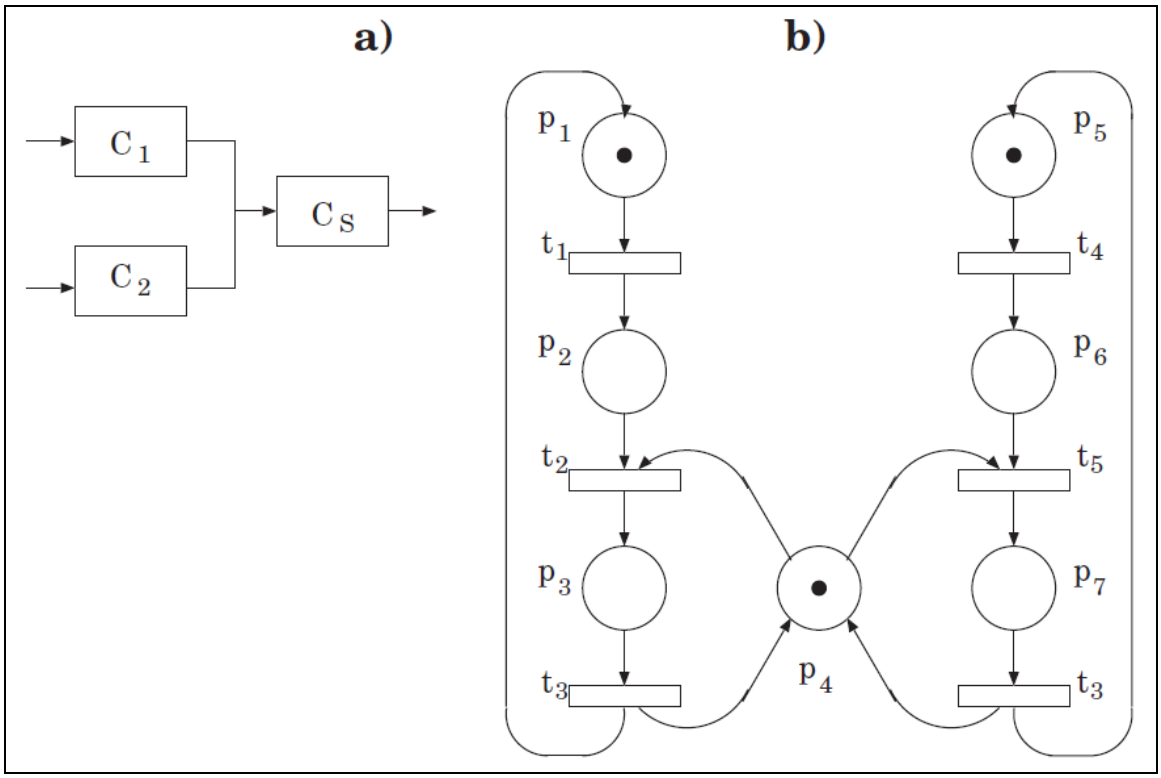
1.4 Ιδιότητες των Petri nets

Θα απαριθμήσουμε διαφορετικές ιδιότητες οι οποίες μας επιτρέπουν να ταξινομήσουμε τα βασικά στοιχεία των PN.

1.4.1 Liveness

Ένας transition είναι δυνητικά πυροδοτήσιμος (potentially firable) στην σήμανση M αν υπάρχει μία σειρά από πυροδοτήσεις transitions οι οποίες οδηγούν σε μία σήμανση στην οποία ο transition είναι ενεργός. Ένας transition είναι ζωντανός (live) αν είναι δυνητικά πυροδοτήσιμος σε κάθε σήμανση του $R(M_1)$. Ένας transition ονομάζεται νεκρός (dead) στην

σήμανση M αν δεν είναι δυνητικά πυροδοτήσιμος, αν το PN μπαίνει στην σήμανση M ο νεκρός transition δεν μπορεί να πυροδοτήσει πλέον.



Σχήμα 1.8. Το πρόβλημα της αμοιβαίας απόκλισης: δύο παράλληλες διαδικασίες με κοινό πόρο.

1.4.2 Safeness (Ασφάλεια)

Ένας place είναι safe αν ο μετρητής των token δεν ξεπερνά το 1 σε κάθε σήμανση του $R\{M_1\}$. Ένα PN είναι safe αν κάθε place είναι safe. Τα PNs των σχημάτων 1, 3, και 8b είναι safe.

1.4.3 Boundedness (Οριοθέτηση)

Μία απλή γενίκευση της έννοιας safeness είναι η ιδέα της έννοιας boundedness. Ένας place είναι οριοθετημένος (bounded) με όριο k, αν ο μετρητής token δεν ξεπερνά το k σε καμία σήμανση του $R\{M_1\}$. Ένα PN είναι k-οριοθετημένο (k-bounded) αν κάθε place είναι k-οριοθετημένος. Το PN του σχήματος 1.7 είναι k-οριοθετημένο όπου k είναι ο αριθμός των θέσεων του buffer. Σε αντίθεση, το PN του σχήματος 1.6 είναι μη οριοθετημένο.

1.4.4 Conservation

Ένα PN είναι αυστηρά conservative αν ο συνολικός αριθμός των token είναι σταθερός σε κάθε σήμανση του $R\{M_1\}$. Το PN του σχήματος 1.7 είναι k-οριοθετημένο και αυστηρά conservative, ενώ το PN του σχήματος 1.8b είναι safe αλλά όχι αυστηρά conservative. Ένα υποσύνολο των place από μία place-invariant[5] (αναλλοίωτη-place) αν είναι αυστηρά conservative. Στο δίκτυο του σχήματος 1.8b τα υποσύνολα $\{p_1, p_2, p_3\}$, $\{p_5, p_6, p_7\}$ και $\{p_3, p_4, p_7\}$ είναι place-invariants.

1.5 Τεχνικές ανάλυσης

Η επιτυχία κάθε μοντέλου εξαρτάται από δύο παράγοντες: την δύναμη μοντελοποίησης και την δύναμη απόφασης. Η δύναμη μοντελοποίησης αναφέρεται στην ικανότητα να μπορεί να αναπαραστήσει σωστά το σύστημα που είναι προς μοντελοποίηση, η δύναμη απόφασης αναφέρεται στην ικανότητα να αναλύει το μοντέλο και να προσδιορίζει ιδιότητες του μοντελοποιημένου συστήματος. Η δύναμη μοντελοποίησης των PN έχει εξεταστεί στις προηγούμενες ενότητες, και σε αυτή την ενότητα θα δώσουμε βάρος στις τεχνικές ανάλυσης των PN.

1.5.1 Το δέντρο προσεγγισμότητας και ο γράφος προσεγγισημότητας

Το σύνολο προσεγγισημότητας $R\{M_1\}$ ενός PN γεννάται με τις έννοιες του δέντρου προσεγγισημότητας. Η αρχική σήμανση M_1 είναι η ρίζα του δέντρου προσεγγισημότητας. Αρχίζοντας από την ρίζα ψάχνουμε για όλους τους ενεργούς transitions, η πυροδότηση ενός ενεργού transition δημιουργεί μία καινούργια σήμανση η οποία αναπαριστάται ως ένα καινούργιο φύλλο στο δέντρο, από το οποίο η διαδικασία επαναλαμβάνεται.

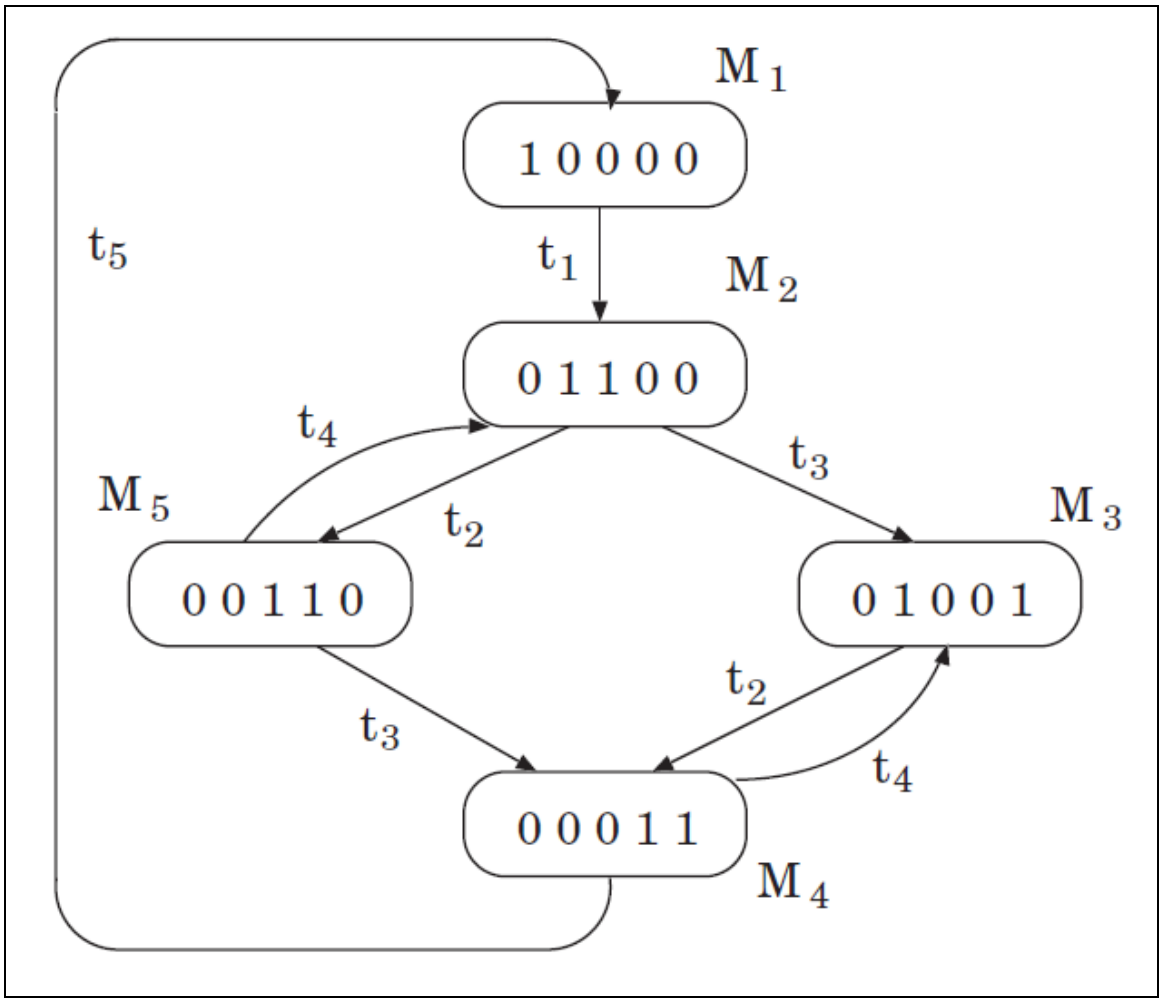
Με κανονικό εντοπισμό των αρχικών κόμβων του δέντρου, η γένεση του δέντρου προσεγγισημότητας εμπλέκει πεπερασμένο αριθμό βημάτων[6], ακόμη και αν το PN είναι μη οριοθετημένο. Ας εισάγουμε τρία είδη αρχικών κόμβων:

- τερματικοί κόμβοι (terminal nodes[dead]): κόμβοι στους οποίους κανένας transition δεν είναι ενεργός,
- διπλοί κόμβοι (duplicate nodes): κόμβοι οι οποίοι έχουν ήδη γεννηθεί στο δέντρο,

- ατέρμονα αναπαραγόμενοι κόμβοι (infinitely reproducible nodes): μία σήμανση M^n είναι ένας ατέρμονα αναπαραγόμενος κόμβος αν $M'' \geq M'$ ($m_i'' \geq m_i'$, I = 1, 2, . . . , n_p) για κάποια M' που έχουν ήδη γεννηθεί στο δέντρο. Λόγω της στατικής σχέσης, η σειρά των transition από την οποία η M'' έχει ξεκινήσει να γεννάται από την M' είναι σίγουρα πυροδοτήσιμη στην M''. Γι' αυτό, η σειρά M' -> M'' μπορεί να παραχθεί απείρως συχνά, ώστε ο μετρητής token στους place για τους οποίους $m_i'' \geq m_i'$ να μπορεί να αυξηθεί επ' αόριστο. Αναπαριστούμε αυτό τον αυθαίρετα μεγάλο αριθμό των token που είναι αποτέλεσμα των ατέρμονα αναπαραγόμενων κόμβων με τον ορισμό ενός συμβόλου ω με τις ακόλουθες ιδιότητες:

$$\omega + a = \omega$$
$$\omega - a = \omega$$
$$a < \omega$$

για κάθε δυνατή θετική σταθερά a.

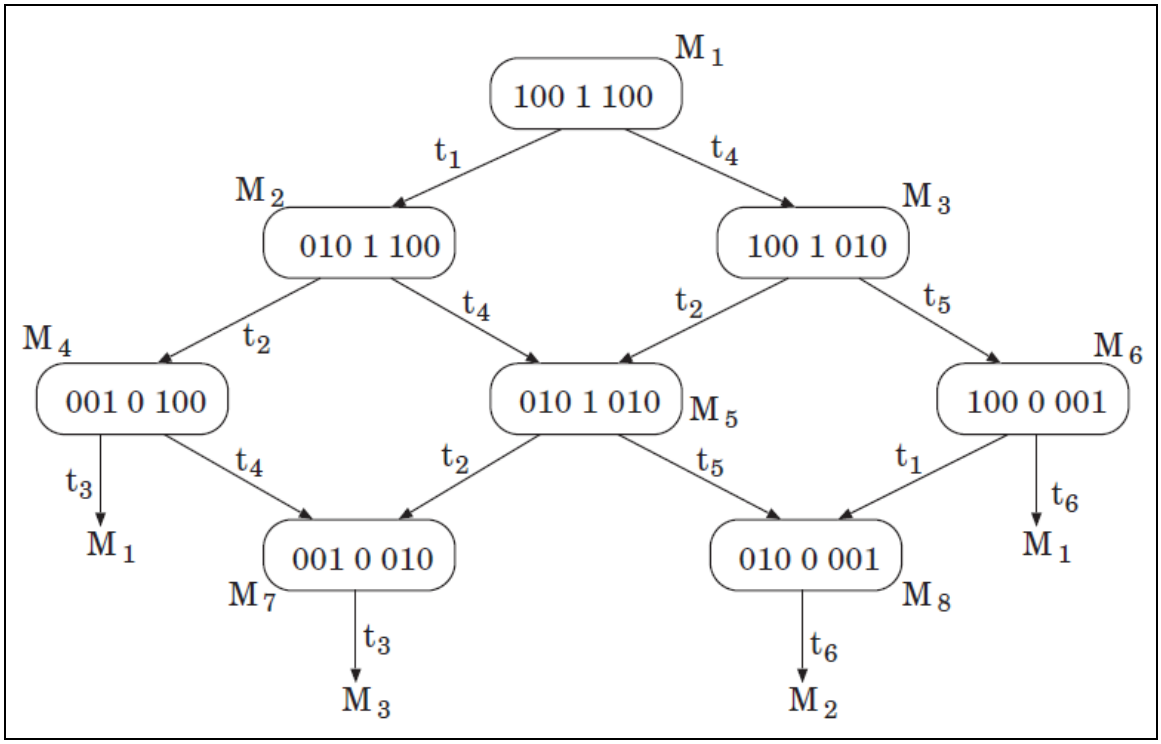


Σχήμα 1.9. Ο γράφος προσεγγισημότητας G_R (M_1) για το δίκτυο του σχήματος 1.1

Επιτρέποντας στο ω να είναι ένα νόμιμο σύμβολο στον προσδιορισμό του δέντρου προσεγγισημότητας, μπορούμε να δείξουμε ότι η γένεση του δέντρου προσεγγισημότητας εμπλέκει πάντα ένα πεπερασμένο αλγόριθμο[6]. Αν η γένεση του δέντρου προσεγγισημότητας τερματίζει χωρίς να φτάνει σε ατέρμονα αναπαραγόμενους κόμβους, το PN είναι οριοθετημένο. Σε αυτή την περίπτωση το σύνολο προσεγγισημότητας είναι πεπερασμένο και μπορεί να αναπαρασταθεί ως ένας προσανατολισμένος γράφος με ετικέτες του οποίου κορυφές είναι τα στοιχεία του $R(M_1)$ και για κάθε δυνατή πυροδότηση transition $M_i - t_k \rightarrow M_j$ υπάρχει ένα βέλος (i, j) με ετικέτα k. Ο γράφος προσεγγισημότητας συνδεδεμένος με ένα σύνολο προσεγγισημότητας $R(M_1)$ θα δηλώνεται με G_R (M_1).

Κωστέλλος Μαριος-Χρηστος
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Το σχήμα 1.9 δείχνει τον γράφο προσεγγισημότητας για το PN του σχήματος 1.1 με αρχική σήμανση $M_1 = (1, 0, 0, 0, 0)$, όπως αναφέρθηκε σε προηγούμενη ενότητα. Το σχήμα 1.10 δείχνει τον γράφο προσεγγισημότητας για το πρόβλημα της αμοιβαίας απόκλισης του σχήματος 1.8 με αρχική σήμανση $M_1 = (1, 0, 0, 1, 1, 0, 0)$.

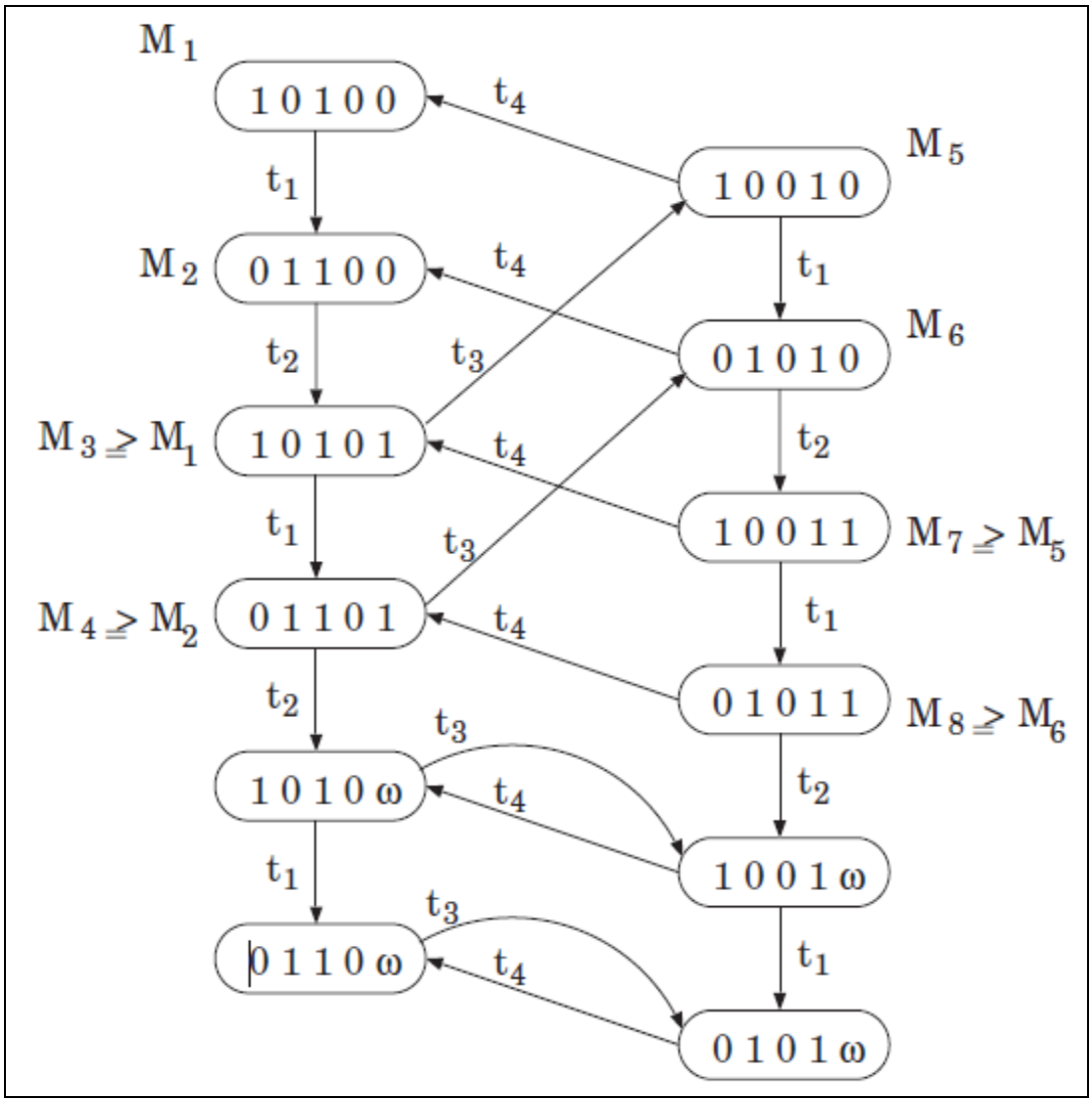


Σχήμα 1.10. Ο γράφος προσεγγισημότητας $G_R(M_1)$ για το δίκτυο του σχήματος 1.8

Στο σχήμα 1.11 αναφέρουμε το δέντρο προσεγγισημότητας του PN του σχήματος 1.6, δεδομένου ότι το δίκτυο είναι μη οριοθετημένο, με στόχο να κρατήσουμε τον αλγόριθμο γένεσης πεπερασμένο, το σύμβολο ω έχει εισαχθεί.

Αν το PN έχει ένα πεπερασμένο $R(M_1)$ όλες οι ιδιότητες του δικτύου (safeness, liveness, κλπ) μπορούν να αναλυθούν με επιθεώρηση του γράφου προσεγγισημότητας. Αν το δίκτυο είναι μη οριοθετημένο η αναπαράσταση του πεπερασμένου δέντρου προσεγγισημότητας, με την έννοια του συμβόλου ω, μπορεί να είναι μία ατελής περιγραφή του δικτύου (είναι δυνατό να βρεθούν PNs με διαφορετικές ιδιότητες και συμπεριφορές που να μην μπορούν να διακριθούν μέσα από το δέντρο

προσεγγισημότητας, λόγω των ημιτελών πληροφοριών που κουβαλάει το ω)[6].



Σχήμα 1.11. Γένεση του δέντρου προσεγγισημότητας για το PN του σχήματος 1.6 με buffer χωρίς οριοθέτηση

1.5.2 Μήτρες Ανάλυσης

Οι συναρτήσεις εισόδου και εξόδου ενός PN μπορούν αντίστοιχα να οριστούν χρησιμοποιώντας ένα συμβολισμό μήτρας. Έστω το D^- δηλώνει την μήτρα εισόδου. Ο D^- είναι μία (n_t x n_p) μήτρα, της οποίας το γενικό στοιχείο d^-_{ij} είναι ίσο με τον αριθμό των βελών τον place p_j με τον transition t_i. Όμοια ορίζουμε την μήτρα εξόδου D^+ ως μία (n_t x n_p) μήτρα, της οποίας

το γενικό στοιχείο d^+_{ij} είναι ίσο με τον αριθμό των βελών που ενώνουν τον transition t_i με τον place p_j. Η μήτρα συχνότητας εμφάνισης (incidence matrix) D ορίζεται από την ακόλουθη σχέση:

$$D = D^+ - D^-$$

Οι μήτρες D^-,D^+ και D για το σχήμα 1.8b αναφέρονται παρακάτω:

$$D^- = \begin{array}{c|ccccccc} & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 \\ \hline t_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ t_2 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ t_3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ t_4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ t_5 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ t_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

$$D^+ = \begin{array}{c|ccccccc} & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 \\ \hline t_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ t_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ t_3 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ t_4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ t_5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ t_6 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}$$

$$D = \begin{array}{c|ccccccc} & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 \\ \hline t_1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ t_2 & 0 & -1 & 1 & -1 & 0 & 0 & 0 \\ t_3 & 1 & 0 & -1 & 1 & 0 & 0 & 0 \\ t_4 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ t_5 & 0 & 0 & 0 & -1 & 0 & -1 & 1 \\ t_6 & 0 & 0 & 0 & 1 & 1 & 0 & -1 \end{array}$$

Ας εισάγουμε τώρα το διάνυσμα e_j το οποίο είναι ένα n_t-διάστασης σειράς διάνυσμα με όλες τις εισόδους ίσες με 0 εκτός την είσοδο j που είναι ίση με 1. Με αυτό το συμβολισμό οι κανόνες εκτέλεσης ενός PN γίνονται:

- ένας transition t_j είναι ενεργός σε μία σήμανση M αν και μόνο αν $M \geq e_j D^-$,
- η πυροδότηση του t_j στην M παράγει μία σήμανση M' που δίνεται από:

$$M' = M - e_j D^- + e_j D^+ = M + e_j D$$

Από τους παραπάνω ορισμούς ακολουθεί ότι, δοθέντος ενός PN με αρχική σήμανση M_1 και σειρά πυροδότησης $t_i \rightarrow t_j \rightarrow t_k \rightarrow t_j \rightarrow t_i$, η σήμανση που θα λάβουμε στο τέλος της σειράς δίνεται από την ακόλουθη εξίσωση με μήτρες:

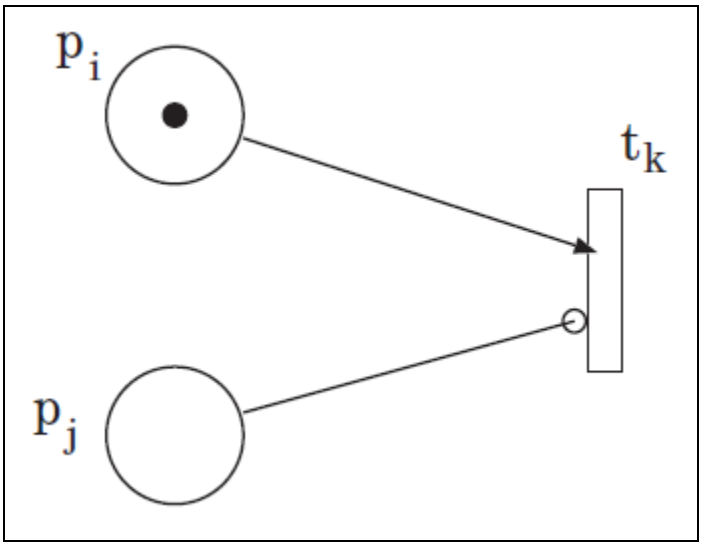
$$M_{fin} = M_1 + (e_i + e_j + e_k + e_j + e_i)D$$

1.6 Επεκτάσεις (extensions)

Στην χρήση των PN για την μοντελοποίηση πραγματικών συστημάτων διάφοροι συντάκτες έχουν βρει βολικό να παρουσιάζουν ειδικές κατασκευές είτε για να φτιάχνουν την αναπαράσταση μοντέλου πιο μικρή σε μία δοθείσα εφαρμογή ή για να επεκτείνουν την δύναμη μοντελοποίησης της τυποποίησης των PN. Όλες οι επεκτάσεις που σημειώνονται είναι ισοδύναμες από την άποψη της δύναμης μοντελοποίησης, γι' αυτό η χρήση τους εξαρτάται από την ευκολία που προσφέρουν ή πόσο βολεύουν στην εφαρμογή[7].

1.6.1 Βέλη αναστολέων

Ένα βέλος αναστολέα από τον place p_j στον transition t_k τροποποιεί τους κανόνες πυροδότησης με την έννοια ότι ο transition μπορεί να πυροδοτήσει μόνο αν ο place p_j δεν περιέχει tokens. Η συνάρτηση αναστολής αναπαρίσταται συνήθως με βέλη με κύκλο στην μύτη τους, όπως στο σχήμα 1.12 όπου ο transition t_k μπορεί να πυροδοτήσει αν και μόνο αν ο p_j περιέχει τουλάχιστο ένα token, αλλά κανένα token δεν παρευρίσκεται στον p_j.



Σχήμα 1.12. Βέλος αναστολέα

Στο πρόβλημα του αμοιβαίου αποκλεισμού του σχήματος 1.8, η βασική PN γλώσσα δεν παρέχει καμία έννοια για να καθιερώσει κανόνες προτεραιότητας στην περίπτωση των δύο πόρων C_1 και C_2 ζητούν ταυτόχρονα πρόσβαση στον κοινό πόρο C_s (οι place p_2 και p_5 μαρκάρονται ταυτόχρονα). Με την παρεμβολή ενός βέλους αναστολέα από τον place p_2 στον transition t_5 (σχήμα 1.13), μοντελοποιούμε την κατάσταση στην οποία, καθώς μία σύγκρουση ανακύπτει μεταξύ των C_1 και C_2 , ο C_1 έχει πάντα προτεραιότητα, και σταματά (αναστέλλει) τον C_2 μέχρι ο κοινός πόρος να ελευθερωθεί.

Με σεβασμό στον γράφο προσεγγισημότητας $G_R(M_1)$ του αυθεντικού PN του σχήματος 1.8 (δείχνεται στο σχήμα 1.10), ο γράφος προσεγγισημότητας του τροποποιημένου PN του σχήματος 1.13 είναι τέτοιος ώστε από την σήμανση M_5 μόνο ο transition t_2 μπορεί να πυροδοτήσει ενώ ο t_5 είναι σε αναστολή.

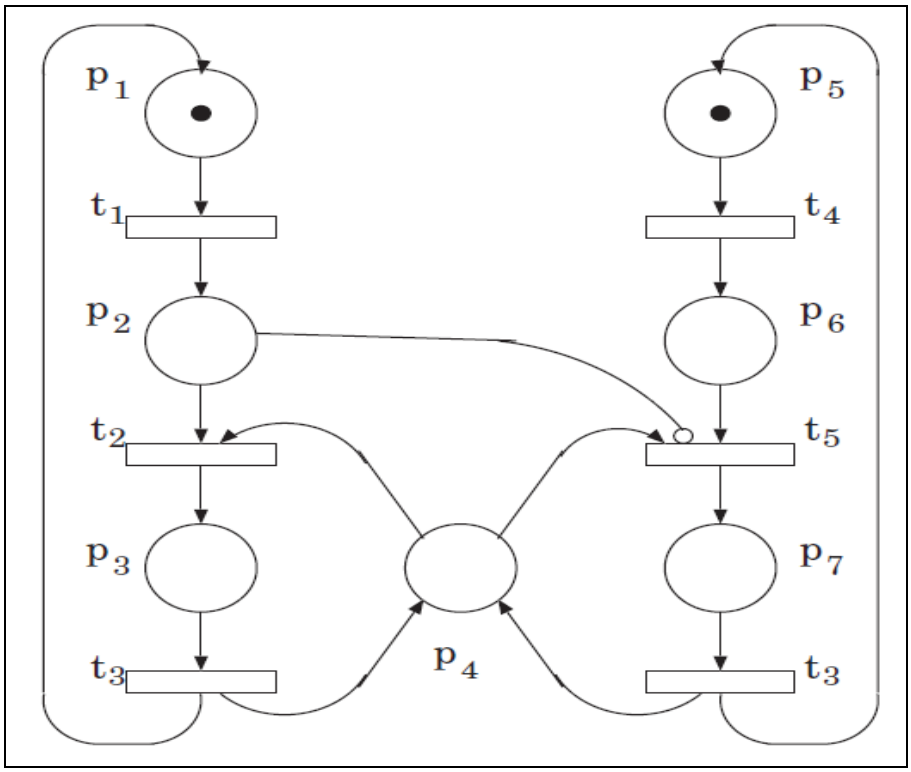
1.6.2 Επίπεδα προτεραιότητας

Ένας εναλλακτικός, αλλά ισοδύναμος τρόπος να μοντελοποιήσουμε τα ίδια χαρακτηριστικά με αυτά με την εισαγωγή των βελών αναστολέων, λαμβάνεται επισυνάπτοντας σε κάθε transition του PN ένα επίπεδο προτεραιότητας. Οι βασικοί κανόνες εκτέλεσης τροποποιούνται με την έννοια ότι, σε όλους τους ενεργούς transition σε μία δοθείσα σήμανση, μόνο αυτοί που είναι συνδεδεμένοι με το υψηλότερο επίπεδο προτεραιότητας επιτρέπεται να πυροδοτήσουν. Στο σχήμα 1.13 ακριβώς η ίδια πολιτική προτεραιότητας μπορεί να μοντελοποιηθεί με την επισύναψη

στον transition t_2 ένα επίπεδο προτεραιότητας υψηλότερο από αυτό που έχει επισυναφθεί στον transition t_5. στην σήμανση M_5 (βλέπε σχήμα 1.10) στην οποία και οι δύο transitions είναι ενεργοί, μόνο ο t_2 μπορεί να πυροδοτήσει.

1.6.3 Συναρτήσεις συνθηκών (conditioning functions)

Πιο περίπλοκες λογικές αλληλεπιδράσεις μεταξύ βασικών στοιχείων ενός PN μπορούν να εξεταστούν εισάγοντας λογικές συναρτήσεις συνθηκών[8]. Δοθείσας μία σήμανσης M, ένας PN transition είναι ενεργός αν, ανάμεσα στις κανονικές απαιτήσεις ενεργοποίησης (περιλαμβανομένων βελών αναστολέων και προτεραιότητες),



Σχήμα 1.13. Το πρόβλημα αμοιβαίου αποκλεισμού του σχήματος 1.8, με επισυνημμένη προτεραιότητα

η συνάρτηση συνθήκης να είναι ‘‘αλήθεια’’ (true). Οι συναρτήσεις συνθηκών μπορεί να είναι πολύ αποτελεσματικές στο να μειώσουν την γραφική πολυπλοκότητα ενός PN, ακόμη και αν δεν επεκτείνουν την δύναμη μοντελοποίησης με σεβασμό στα βέλη αναστολέων ή τα επίπεδα προτεραιότητας.

1.7 Petri nets υψηλού επιπέδου (high level Petri nets)

Στα μοντέλα PN που είδαμε ως τώρα, τα ξεχωριστά tokens είναι δυσδιάκριτα. Η σημασιολογία του μοντέλου δεν μας επιτρέπει να ακολουθήσουμε την συμπεριφορά των ξεχωριστών token μέσα από το δίκτυο. Για να ξεπεράσουμε αυτό τον περιορισμό μία νέα κλάση μοντέλων προτάθηκε και συζητήθηκε. Τα κοινά χαρακτηριστικά αυτών των μοντέλων, που συνήθως αναφέρονται ως High level PN, ότι η θέση ενός απλού token μπορεί να εντοπιστεί μέσα στο PN. Δύο ονομασμένες τεχνικές προταθήκαν αρχικά: η τεχνική του χρωματισμού των tokens (τα coloured PN εισήχθησαν από τον Jensen) και η τεχνική που επισυνάπτει σε κάθε token ένα χαρακτηριστικό (predicate /transition δίκτυα εισήχθησαν από τους Genrich και Lautenbach). Τα coloured Petri nets θα αναλυθούν εκτενέστερα στο επόμενο κεφάλαιο.

1.8 Χρονικά Petri nets

Μία σειρά εκτέλεσης *E* σε ένα σημαδεμένο PN, είναι σειρά από λογικές σημάνσεις που λαμβάνεται από την πυροδότηση μίας σειράς ενεργών transitions:

$$E = \{(M_1, t_1); (M_2, t_2); \ldots; (M_j, t_j); \ldots;\}$$

Μία σειρά εκτέλεσης *E* μπορούμε να την δούμε ως ένα συνδεδεμένο μονοπάτι στον γράφο προσεγγισημότητας $G_R(M_1)$ του δικτύου.

Μία χρονική σειρά εκτέλεσης T_E ενός σημαδεμένου PN με αρχική σήμανση M_1, είναι μία σειρά εκτέλεσης *E* επαυξημένη με μία μη-φθίνουσα σειρά από πραγματικές τιμές που αντιπροσωπεύουν τις χρονικές στιγμές πυροδότησης του κάθε transition, όπως οι συνεχόμενοι transitions $(t_j; t_{j+1})$ στην *E* αντιστοιχούν σε διατεταγμένες χρονικές στιγμές $\tau_j \leq \tau_{j+1}$ στην T_E. Αυτό τυποποιημένο:

$$T_E = \{(M_1, t_1, \tau_1); (M_2, t_2, \tau_2); \ldots; (M_{,}, t_j, \tau_j); \ldots\}$$

Το χρονικό διάστημα $\tau_j - \tau_{j+1}$ μεταξύ συνεχόμενων χρονικών στιγμών αντιπροσωπεύει την περίοδο κατά την οποία το PN παραμένει στη σήμανση M_j. Στην συνέχεια πάντα υποθέτουμε ως αρχική χρονική στιγμή $\tau_1=0$.

Ορισμός – Ένα χρονικό PN (TPN) είναι ένα σημαδεμένο PN στο οποίο παρέχεται ένα σύνολο από προσδιορισμούς και ορίζονται ένα

σύνολο κανόνων τέτοια ώστε για κάθε νόμιμη σειρά εκτέλεσης *E* μία χρονική σειρά εκτέλεσης T_E να μπορεί σίγουρα να συσχετιστεί.

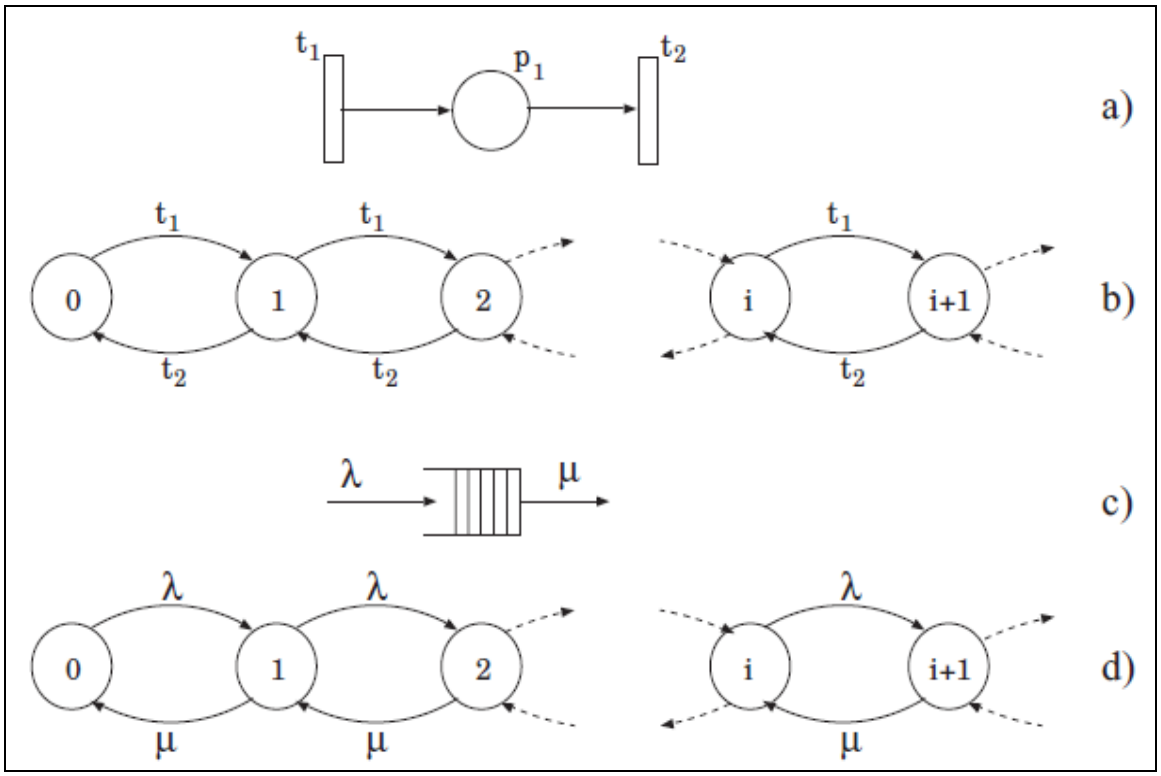
Μία ποικιλία χρονικών μηχανισμών έχει προταθεί στη βιβλιογραφία. Τα διακριτικά χαρακτηριστικά των χρονικών μηχανισμών είναι πότε η διάρκεια των γεγονότων μοντελοποιείται από ντετερμινιστικές μεταβλητές ή από τυχαίες μεταβλητές, και πότε ο χρόνος σχετίζεται στο PN με τους places, τους transitions ή τα tokens.

Η εφαρμογή των ντετερμινιστικών –PN μοντέλων είναι διαθέσιμη σε διάφορες περιοχές, όπως: πρωτόκολλα επικοινωνίας, αξιολόγηση επίδοσης, κατασκευή. Ωστόσο, στην περιοχή της αξιοπιστίας η στοχαστική μοντελοποίηση είναι πιο κατάλληλη, και γι' αυτό θα επικεντρωθούμε σε TPN στα οποία ο μηχανισμός του χρόνου είναι στοχαστικός, και θα τα αναφέρουμε ως Stochastic PN (SPN).

Τα SPN αρχικά προτάθηκαν σε δύο διδακτορικές θέσεις. Σε αυτά τα μοντέλα, μεταφράστηκαν τα PN ως δίκτυα συνθήκης/γεγονός, ο χρόνος φυσικά σχετίστηκε με τις δραστηριότητες οι οποίες προκαλούν τις αλλαγές καταστάσεων, ως εκ τούτου με την καθυστέρηση που πραγματοποιήθηκε πριν από την πυροδότηση των transitions. Παρά τις όποιες δυνατότητες έχουν εξερευνηθεί, η επιλογή συσχέτισης του χρόνου με PN transitions είναι η πιο κοινή στην βιβλιογραφία.

Όταν τυχαίες μεταβλητές σχετίζονται με τους PN transitions είναι εκθετικά κατανεμημένες, η δυναμική συμπεριφορά των PN μπορεί να παρατηρηθεί σε χρονικά-συνεχόμενα ομοιογενή Markov αλυσίδες με ισομορφική state space σε σχέση με τον γράφο προσεγγισημότητας του PN.

Επεκτάσεις για να καλύψουν την περίπτωση της γενικής κατανομής του χρόνου πυροδότησης των transitions έχουν αναλυθεί σε πολλές δημοσιεύσεις. Αφήνοντας την αδύναμης μνήμης ιδιότητα της εκθετικής κατανομής, με στόχο να σχετίσουμε πλήρως σε κάθε σειρά εκτέλεσης *E* μία χρονική σειρά εκτέλεσης T_E στα SPN η πολιτική εκτέλεσης χρειάζεται να εισαχθεί. Η πολιτική εκτέλεσης αποτελείται από δύο μέρη: στον τρόπο με τον οποίο επιλέγεται ένας transition να πυροδοτήσει ανάμεσα σε τόσους ενεργούς σε μία δοθείσα σήμανση, και στον τρόπο με τον οποίο καλύπτεται ο χρόνος που δαπανήθηκε μετά την πυροδότηση ενός transition. Ωστόσο, λόγω της πολυπλοκότητας της σημασιολογίας των SPN μοντέλων, και των σχετιζόμενων στοχαστικών διαδικασιών.



Σχήμα 1.14. η ουρά M/M/1: a) Η αναπαράσταση με PN, b) Ο γράφος προσεγγισημότητας, c) Η αναπαράσταση σε block διάγραμμα, d) Ο αντίστοιχος Markov γράφος transition

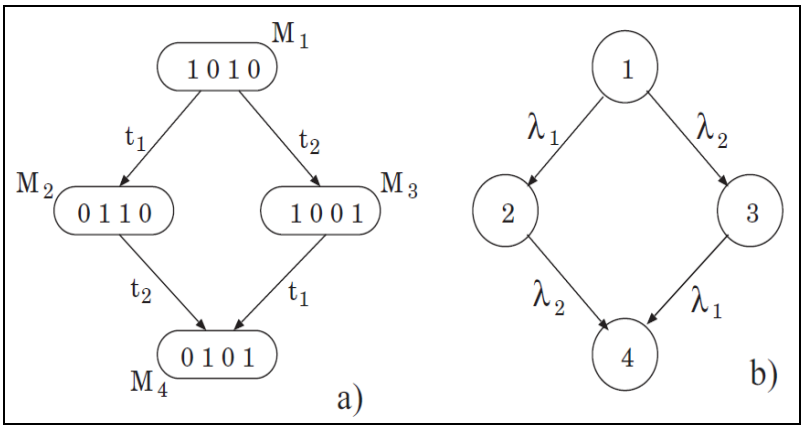
1.9 Ομοιογενές Markov SPN (HMSPN)

Ας υποθέσουμε ότι η δραστηριότητα μοντελοποιείται από ένα PN transition που παίρνει εκθετικά κατανεμημένες τυχαίες τιμές χρόνου για να ολοκληρωθεί από όταν ξεκίνησε. Αυτό σημαίνει ότι μία εκθετικά κατανεμημένη τυχαία μεταβλητή θ_j με παράμετρο $\lambda_j(M)$ συνδέεται σε κάθε PN transition t_j. Η πυροδότηση ενός ενεργού transition t_j στην σήμανση M γίνεται τυχαίο γεγονός το οποίο δρα με ένα χρονικά ανεξάρτητο ρυθμό πυροδότησης λ_j (M). Γι' αυτό, γνωρίζοντας ποιοί transitions είναι ενεργοί σε μία δοθείσα σήμανση και τους συνδεδεμένους ρυθμούς πυροδότησης, μπορούμε μονοσήμαντα να δημιουργήσουμε την στοχαστική χρονική σειρά T_E από κάθε σειρά εκτέλεσης *E*. Με άλλα λόγια, ο γράφος προσεγγισημότητας $G_R(M_1)$ ενός σημαδεμένου PN μπορεί μονοσήμαντα να εντοπιστεί σε μία διακριτής κατάστασης συνεχόμενου χρόνου ομοιογενή Markov αλυσίδα, αφήνοντας κάθε σήμανση του $G_R(M_1)$ να αντιστοιχεί σε μία κατάσταση της αλυσίδας Markov, και αντικαθιστώντας την ετικέτα του

PN transition σε κάθε άκρη του $G_R(M_1)$ με τον ρυθμό πυροδότησης του αντίστοιχου transition.

Παράδειγμα 1- Η ουρά M/M/1. Παρατηρώντας το PN του σχήματος 1.14a όπου φαίνεται ότι ένας transition με κανέναν place εισόδου είναι πάντα ενεργός. Ο αντίστοιχος γράφος προσεγγισημότητας είναι στο σχήμα 1.14b. η ετικέτα μέσα σε κάθε κατάσταση είναι η σήμανση, π.χ., ο αριθμός των token στον place p_1. Η πυροδότηση του t_1 αυξάνει τον μετρητή των token κατά 1, ενώ η πυροδότηση του t_2 μειώνει τον μετρητή των token κατά 1. Σχετίζοντας στον transition t_1 το ρυθμό αφίξεων λ και στον t_2 τον ρυθμό εξυπηρέτησης μ, το PN του σχήματος 1.14a μοντελοποιεί το σύστημα αναμονής M/M/1. η συνηθισμένη αναπαράσταση σε block διάγραμμα δίνεται στο σχήμα 1.14c και ο αντίστοιχος γράφος μετασχηματισμών Markov δίνεται στο σχήμα 1.14d.

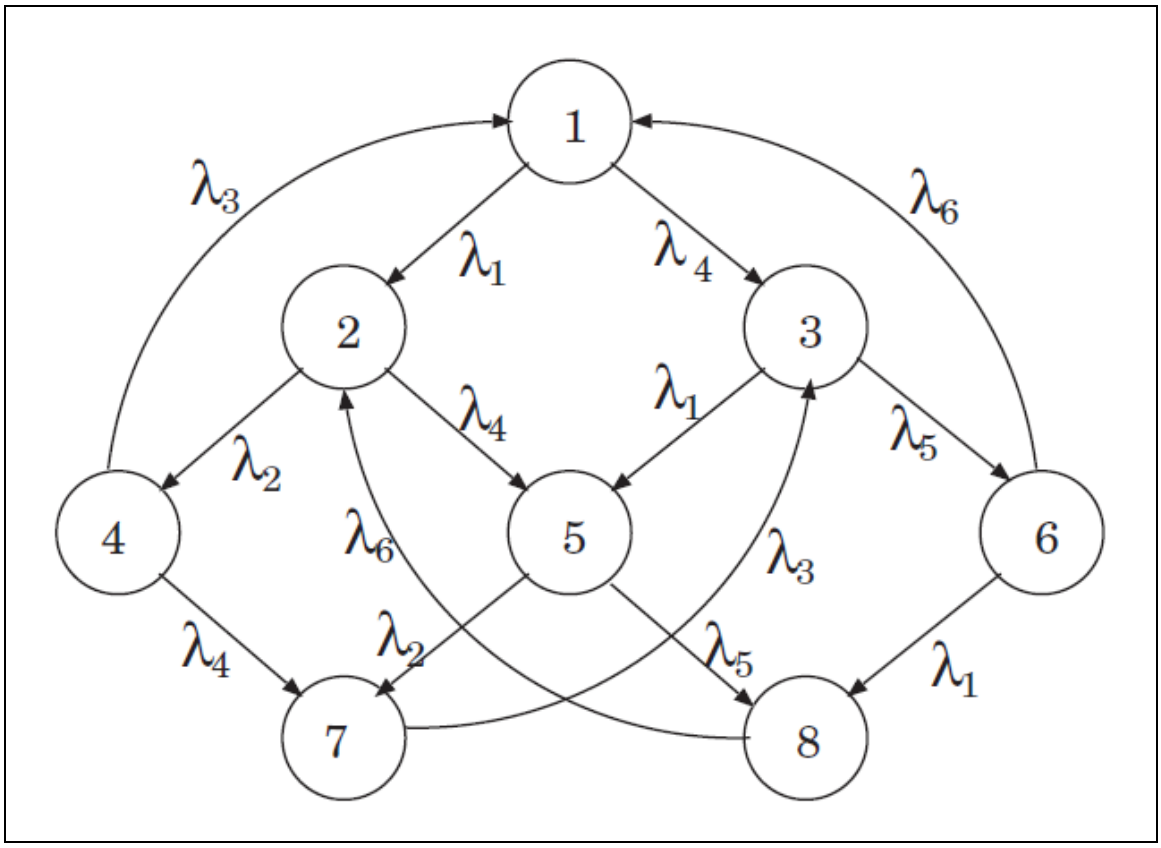
Παράδειγμα 2- Έστω ότι το PN του σχήματος 1.3 δηλώνει την διαδικασία αποτυχίας δύο μερών σε παράλληλη απόλυση, t_1 είναι το γεγονός αποτυχίας του μέρους 1 στο οποίο επισυνάπτεται ρυθμός αποτυχίας λ_1. Όμοια επισυνάπτουμε στον t_2 το ρυθμό αποτυχίας λ_2 του μέρους 2. Το σχήμα 1.15a δείχνει το γράφο προσεγγισημότητας του δικτύου και το σχήμα 1.15b την σχετική αλυσίδα Markov που αντιπροσωπεύει την δυναμική συμπεριφορά του δικτύου στο χρόνο.



Σχήμα 1.15. Ο γράφος προσεγγισημότητας a) και η αντίστοιχη αλυσίδα Markov b) του SPN του σχήματος 1.3

Η πιθανότητα το αρχικό PN να βρεθεί στη σήμανση M_4 σε χρόνο t όπου και τα δύο μέρη αποτυγχάνουν μπορεί να υπολογιστεί ως η πιθανότητα να βρίσκεται το σύστημα στην κατάσταση 4 σε χρόνο t στην αντίστοιχη αλυσίδα Markov.

Παράδειγμα 3- Ο γράφος προσεγγισημότητας του PN του σχήματος 1.8 φαίνεται στο σχήμα 1.10. Αν σε όλους τους PN transitions έχουν επισυναφθεί χρονικά ανεξάρτητοι ρυθμοί πυροδότησης, ο γράφος προσεγγισημότητας του σχήματος 1.10 εντοπίζεται στην αλυσίδα Markov του σχήματος 1.16.



Σχήμα 1.16. Η αλυσίδα Markov αντίστοιχη στον γράφο προσεγγισημότητας του σχήματος 1.10

1.10 Τυποποιημένος ορισμός του μοντέλου

Το HMSPN είναι ένα εξαπλό σύνολο:

$$HMSPN = (P, T, I, O, M, L)$$

όπου P, T, I, O, M έχουν την ίδια έννοια με αυτή που συναντήσαμε στην αρχή, και το $L = \{ \lambda_1(M), \lambda_2(M), \ldots, \lambda_{nt}(M)\}$ είναι ένα σύνολο n_t μη αρνητικών

πραγματικών αριθμών που αντιπροσωπεύουν τους ρυθμούς πυροδότησης των εκθετικών τυχαίων μεταβλητών που σχετίζονται με κάθε PN-transition.

Η γνώση του γράφου προσεγγισημότητας μας επιτρέπει να δημιουργήσουμε αυτόματα τον πίνακα ρυθμού των transitions (**Λ**) της σχετικής ομοιογενούς αλυσίδας Markov. Ο **Λ** είναι ένας NxN πίνακας, όπου N ο αριθμός των μελών του συνόλου προσεγγισημότητας $R(M_1)$.

Ας ορίσουμε τώρα το Q(t) ένα N-διάστατο διάνυσμα πιθανότητας κατάστασης, του οποίου η γενική είσοδος $q_i(t)$ είναι η πιθανότητα να είμαστε στην κατάσταση i (i = 1, 2, . . ., N) σε χρόνο t στην σχετική αλυσίδα Markov. Το Q(t) είναι η λύση της βασικής γραμμικής διαφορικής εξίσωσης Markov:

$$\frac{d\underline{Q}(t)}{dt} = \Lambda\,\underline{Q}(t)$$

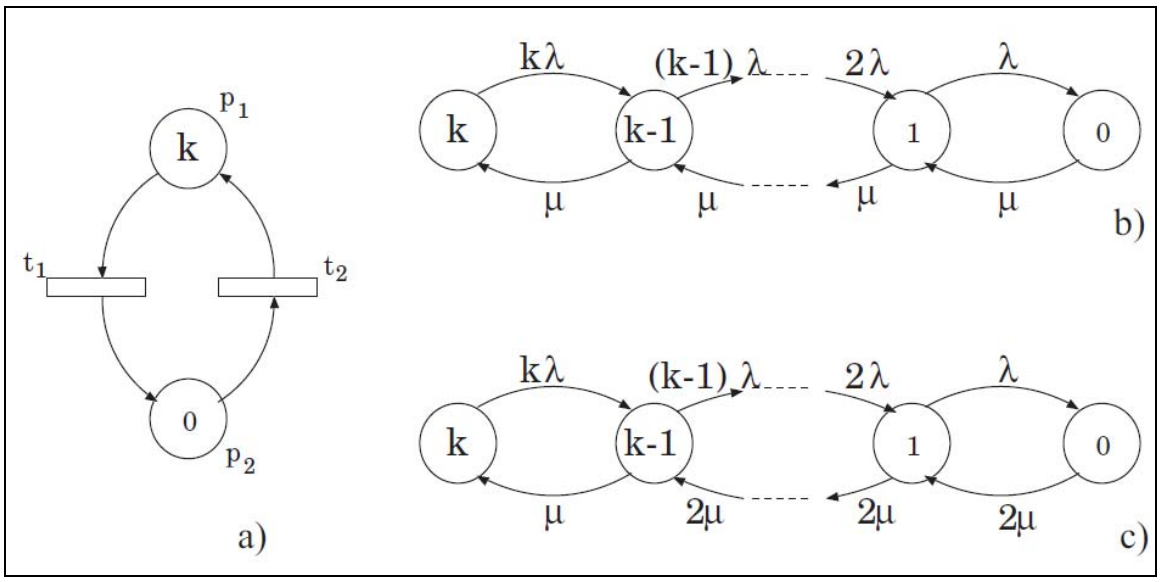
με αρχική συνθήκη $Q(0) = [1, 0, 0, \ldots, 0]^T$. Αν σταθερό διάνυσμα πιθανότητας κατάστασης $Q(\infty)$ της αλυσίδας Markov υπάρχει, μπορεί να υπολογιστεί από την εξίσωση:

$$\Lambda\,\underline{Q}(\infty) = \underline{0} \qquad \text{with} \qquad \sum_{i=1}^{N} q_i(\infty) = 1$$

1.11 Ρυθμοί πυροδότησης ανεξάρτητοι σήμανσης

Στον τυποποιημένο ορισμό του μοντέλου HMSPN οι ρυθμοί πυροδότησης που σχετίζονται με κάθε transition εξετάζονται ως ανεξάρτητοι σήμανσης. Αυτή η δυνατότητα αυξάνει την ελαστικότητα του μοντέλου και συχνά χρησιμοποιείτε για να κάνει τα μοντέλα πιο πυκνά στην περίπτωση παρουσίας πολλαπλών ταυτόσημων πόρων.

Παράδειγμα 4- Το PN του σχήματος 1.17α έχει την ακόλουθη φυσική σημασία: ο place p_1 αντιπροσωπεύει την λειτουργία, ο place p_2 την μη λειτουργία, ο transition t_1 την αποτυχία και ο transition t_2 την επισκευή. Υποθέτουμε ότι έχουμε *K* ταυτόσημα μέρη με παράλληλη απόλυση το καθένα με ρυθμό αποτυχίας *λ*. Μπορούμε να μοντελοποιήσουμε την λειτουργία του συστήματος με το PN του σχήματος 1.17α με αρχική σήμανση $M_1=(K, 0)$ και να σχετίσουμε στον transition t_1 τον ανεξάρτητο σήμανσης ρυθμό πυροδότησης $\lambda_{t1}(M_X)= m_{1x}\lambda$, όπου m_{1x} είναι ο αριθμός των token στον place p_1 στη σήμανση M_x.



Σχήμα 1.17. a) το PN που μοντελοποιεί *K* ταυτόσημα παράλληλα μέρη, b) Η σχετική αλυσίδα Markov με ένα repairman, c) Η σχετική αλυσίδα Markov με δύο repairmen

1.12 Άμεσοι και χρονισμένοι PN transitions

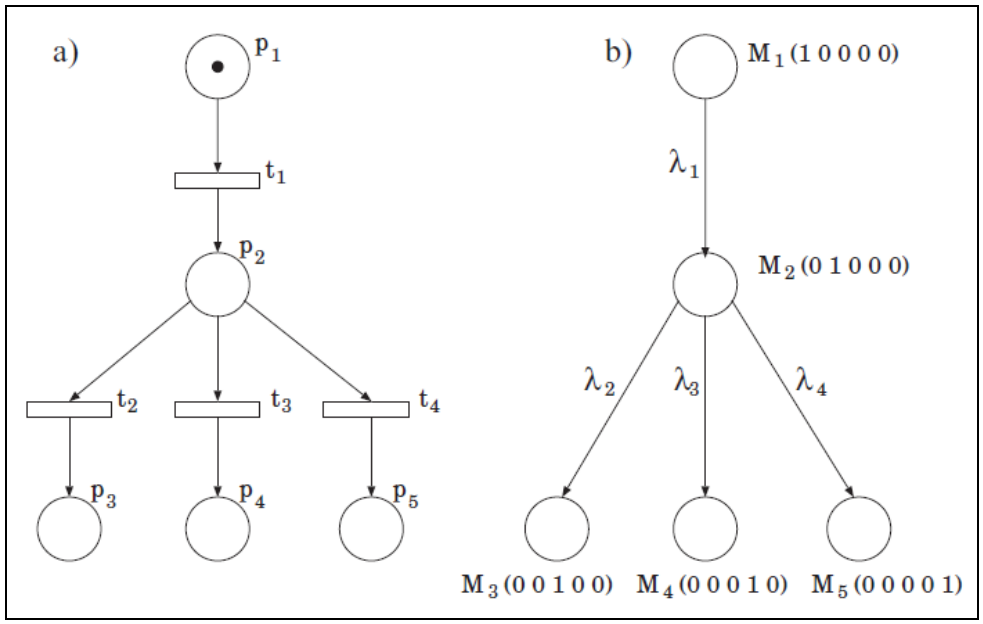
Πολλοί ερευνητές έχουν διαπιστώσει ότι η χρήση των SPN για την μοντελοποίηση πραγματικών συστημάτων εμπλέκει την παρουσία πολύ μικρών ή γρήγορων transitions, των οποίων η διάρκεια είναι σύντομη, ή ακόμη αμελητέα, με σεβασμό στο χρονικό μέρος του προβλήματος. Διαφορετικές τεχνικές έχουν προταθεί για να ξεπεραστεί αυτό το πρόβλημα.

Η αρχική υπόθεση στο μοντέλο GSPN[9] είναι το ότι είναι επιθυμητό να συσχετιστεί ένας τυχαίος χρόνος μόνο σε αυτούς τους transitions οι οποίοι πιστεύεται ότι έχουν την μεγαλύτερη επίδραση στην λειτουργία του συστήματος. Οι transitions χωρίζονται σε δύο διαφορετικές κλάσεις: τους άμεσους transitions και τους χρονισμένους transitions. Οι άμεσοι transitions πυροδοτούν σε χρόνο μηδέν εφόσον είναι ενεργοί και έχουν υψηλότερη προτεραιότητα από τους χρονισμένους transitions. Οι χρονισμένοι transitions πυροδοτούν μετά από ένα εκθετικά κατανεμημένο χρόνο πυροδότησης. Στην γραφική αναπαράσταση του GSPN, οι άμεσοι transitions σχεδιάζονται ως λεπτές ράβδοι ενώ οι χρονισμένοι transitions σχεδιάζονται ως παχύς ράβδοι.

Οι σημάνσεις (καταστάσεις) που ενεργοποιούν τους άμεσους transitions περνούν σε χρόνο μηδέν και ονομάζονται vanishing states. Οι

σημάνσεις που ενεργοποιούν μόνο χρονισμένους transitions ονομάζονται tangible. Εφόσον η διαδικασία δεν ξοδεύει χρόνο στις vanishing states, δεν συμβάλουν στην χρονική συμπεριφορά του συστήματος τόσο ώστε να μπορεί μια διαδικασία να τους εξαλείψει από την τελική αλυσίδα Markov. Με τον διαχωρισμό των PN σε χρονισμένα και άμεσα, εισάγουμε μεγαλύτερη ελαστικότητα στο επίπεδο μοντελοποίησης χωρίς να αυξάνουμε τις διαστάσεις του τελικού state space.

Δοθείσας μίας σήμανσης M ε $G_R(M_1)$ ενός GSPN, τρείς διαφορετικές καταστάσεις μπορούν να παρουσιαστούν:



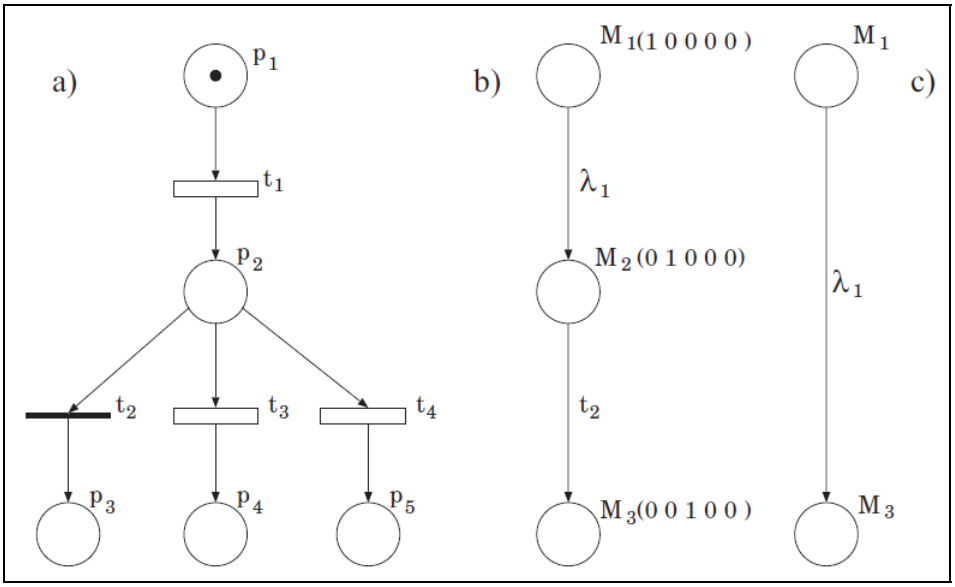
Σχήμα 1.19. a) Ένα SPN με χρονισμένους transitions μόνο, b) Η αντίστοιχη αλυσίδα Markov

Κατάσταση 1 (σχήμα 1.19)

Μόνο ενεργά transitions είναι ενεργά (σχήμα 1.19a) έτσι ώστε να γεννιούνται μόνο ενσωματωμένες σημάνσεις (σχήμα 1.19b). Το μοντέλο, σε αυτή την περίπτωση, συμπίπτει με το HMSPN που περιγράφηκε σε προηγούμενη ενότητα.

Κατάσταση 2 (σχήμα 1.20)

Χρονισμένοι transitions είναι ενεργοί ταυτόχρονα σε ένα άμεσο transition (σχήμα 1.20a). Μόνο ο άμεσος transition επιτρέπεται να πυροδοτήσει, δημιουργώντας την σχετική αλυσίδα Markov του σχήματος 1.20b. Ωστόσο, η σήμανση M_2 είναι αμελητέα και μπορεί να παραληφθεί από την αλυσίδα για να μειώσουμε την αλυσίδα Markov του σχήματος 1.20b, στην οποία όλες οι καταστάσεις είναι ενσωματωμένες.

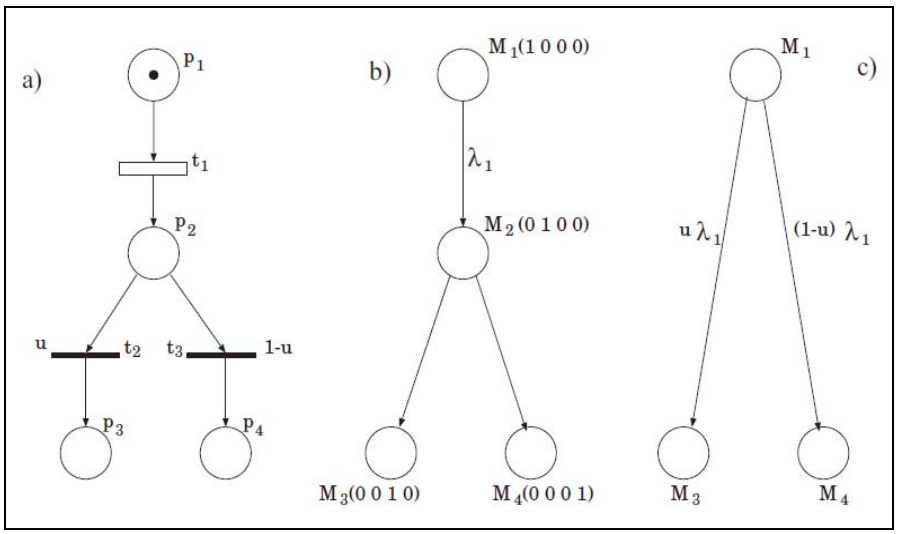


Σχήμα 1.20. a) Ένα SPN με ένα άμεσο transition, b) Ο γράφος προσεγγισημότητας, c) Η μειωμένη αλυσίδα Markov ορισμένη μόνο με αμελητέες σημάνσεις

Κατάσταση 3 (σχήμα 1.21)

Διάφοροι άμεσοι transitions είναι ενεργοί σε μία σήμανση. Σε αυτή την περίπτωση με στόχο να προσδιορίσουμε ποιός είναι ο transition που πυροδοτεί πρώτος, μία συνάρτηση πυκνότητας πιθανότητας χρειάζεται να προσδιοριστεί: στην γλώσσα του GSPN αυτή η κατασκευή ονομάζεται τυχαίος διακόπτης (random switch), και η συνάρτηση πυκνότητας πιθανότητας ονομάζεται διακόπτης διανομής (switching distribution). Στο σχήμα 1.21a ο άμεσος transition t_2 πυροδοτεί με πιθανότητα *u* και ο άμεσος transition t_3 με την συμπληρωματική πιθανότητα 1-*u*. Η ισοδύναμη αλυσίδα Markov φαίνεται στο σχήμα 1.21b. Η κατάσταση M_2 είναι αμελητέα και μπορεί να παραληφθεί ενσωματώνοντας τον switching distribution στον ρυθμό που οδηγεί στην κατάσταση M_2. Η παράληψη της αμελητέας κατάστασης οδηγεί στην αλυσίδα Markov του σχήματος 1.21c, το οποίο περιέχει μόνο ενσωματωμένες καταστάσεις.

Ένας αυτόματος αλγόριθμος[9] μπορεί να εφαρμοστεί ο οποίος να αναγνωρίζει τις τρείς καταστάσεις που παρουσιάστηκαν πριν και προοδευτικά να παραλείπει αμελητέες καταστάσεις μέχρι μία ομοιογενή αλυσίδα Markov, ορισμένη μόνο με ενσωματωμένες καταστάσεις, να ληφθεί. Με αυτό τον τρόπο η διαδικασία μείωσης γίνεται εντελώς διαφανή στον αναλυτή.



Σχήμα 1.21. a) Ο τυχαίος διακόπτης, b) ο γράφος προσεγγισημότητας, c) Η μειωμένη αλυσίδα Markov ορισμένη μόνο με ενσωματωμένες καταστάσεις

1.13 Υπολογισμός των μέτρων της αξιοπιστίας και της απόδοσης

Ένα πολύ σημαντικό σημείο της εξαρτημένης από το χρόνο αναπαράστασης της συμπεριφοράς του συστήματος μέσα από τα SPN, είναι ότι επιτρέπει στο χρήστη να προσδιορίσει με απλό και φυσικό τρόπο ένα μεγάλο αριθμό από μέτρα που σχετίζονται με την αξιοπιστία και την απόδοση του συστήματος. Για να μπορέσουμε να εκμεταλλευτούμε αυτή την ιδιομορφία, η γλώσσα εισόδου θα πρέπει να είναι έτσι δομημένη ώστε να παρέχει ένα φιλικό περιβάλλον για τα μέτρα εξόδου.

Η στοχαστική συμπεριφορά ενός SPN προσδιορίζεται με τον υπολογισμό των πιθανοτήτων εμφάνισης πάνω στις καταστάσεις του συνόλου προσεγγισημότητας $R(M_1)$. Γι' αυτό, τα μέτρα εξόδου προσδιορίζονται από το επίπεδο του δικτύου, και ο αριθμητικός υπολογισμός γίνεται αυτόματα από την επίλυση της σχετικής εξίσωσης:

$$\frac{d\underline{Q}(t)}{dt} = \Lambda\,\underline{Q}(t)$$

και παρατηρώντας τις καταστάσεις στο $R(M_1)$.

Δεδομένου ότι μερικά από τα μέτρα εξόδου εξαρτώνται από το ολοκλήρωμα των πιθανοτήτων παρά από τις πιθανότητες αυτές καθαυτές,

είναι απαραίτητο να παρέχουμε το πακέτο με τον κατάλληλο υπολογισμό για το ολοκλήρωμα των πιθανοτήτων κατάστασης. Παρακάτω εννοείται ότι ο χρόνος t εκτείνεται από το 0 ως το ∞.

1.13.1 Η πιθανότητα μίας δοθείσας συνθήκης σε ένα SPN

Με τις έννοιες των λογικών ή αλγεβρικών συναρτήσεων του αριθμού των tokens στους place ενός PN, μπορούμε να προσδιορίσουμε μία συνθήκη εξόδου (π.χ., κανένα token στον place αποτυχίας). Αναγνωρίζουμε στο $R(M_1)$ το υποσύνολο των places S για το οποίο η συνθήκη εξόδου είναι αληθής. Το μέτρο εξόδου:

Qs(t) = prob {συνθήκη που είναι αληθής σε χρόνο t}

δίνεται από:

$$Q_S(t) = \sum_{s \in S} q_s(t)$$

όπου $q_s(t)$ είναι η πιθανότητα να βρίσκεται στην κατάσταση s σε χρόνο t. Για παράδειγμα, αν το S είναι το σύνολο των λειτουργικών καταστάσεων, Qs(t) είναι ο συνηθισμένος ορισμός της αξιοπιστίας.

1.13.2 Ο χρόνος που δαπανάται σε μία σήμανση

Έστω S το υποσύνολο των σημάνσεων στο οποίο μία συγκεκριμένη συνθήκη πληρείται. Ο προσδοκώμενος χρόνος $\psi_s(t)$ που ξοδεύεται στις σημάνσεις s ε S μέσα στο διάστημα 0 – t δίνεται από:

$$\psi_S(t) = \sum_{s \in S} \int_0^t q_s(z)\, dz$$

Επιπλέον, είναι γνωστό από την θεωρία των αλυσίδων Markov ότι καθώς ο t πλησιάζει το άπειρο το ποσοστό του χρόνου που δαπανάται στις καταστάσεις s ε S ισούται με την ασυμπτωτική πιθανότητα:

$$Q_S(\infty) = \sum_{s \in S} q_s(\infty)$$

Αν το S είναι το σύνολο των καταστάσεων που δουλεύουν, τότε $\psi_s(t)$ είναι το αναμενόμενο διάστημα διαθεσιμότητας.

1.13.3 Μέσος χρόνος περάσματος

Δοθέντος ότι το Qs(t), όπως υπολογίστηκε πριν, είναι η πιθανότητα να έχουμε μπει στο σύνολο S πριν από τον χρόνο t για πρώτη φορά, ο μέσος χρόνος περάσματος Φ_s ,έχει την συνηθισμένη έκφραση:

$$\phi_S = \int_0^\infty [1 - Q_S(z)]\, dz$$

1.13.4 Διανομή των token σε ένα place

Έστω p_i ο γενικός place ενός PN. Η αθροιστική συνάρτηση κατανομής (Cdf) του αριθμού των tokens στον p_i στον χρόνο t είναι μία βηματική συνάρτηση της οποίας το πλάτος του *k-th* βήματος λαμβάνεται αθροίζοντας τις πιθανότητες όλων των σημάνσεων στην $R(M_1)$ που περιέχουν *k* tokens (*k*=0, 1, 2, . . .) στον p_i τον χρόνο t. Η αναμενόμενη τιμή του αριθμού των tokens στον place p_i στον χρόνο t είναι:

$$E\,[m_i(t)\,] = \sum_{k=0}^{\infty} k\, f_i(k,t)$$

Ως παράδειγμα, αν οι place p_i αντιπροσωπεύουν πανομοιότυπες μονάδας που περιμένουν πρόσβαση σε ένα κοινό πόρο οι παραπάνω ποσότητες είναι η Cdf και η αναμενόμενη τιμή αριθμού μονάδων σε αναμονή σε σύγκριση με τον χρόνο. Στην ανάλυση αξιοπιστίας μία πολύ ενδιαφέρουσα περίπτωση παρατηρείται όταν ο place p_i αντιπροσωπεύει αποτυχόντα μέρη. Οι παραπάνω ποσότητες παρέχουν την Cdf και την αναμενόμενη τιμή του αριθμού των token στα αποτυχόντα μέρη στον χρόνο t.

1.13.5 Αναμενόμενος αριθμός πυροδοτήσεων ενός PN-transition

Δοθέντος ενός διαστήματος (0, t) αυτή η ποσότητα δείχνει πόσες φορές, κατά μέσο όρο, ένα γεγονός που μοντελοποιείται από ένα PN transition έχει δράσει μέσα σε αυτό το διάστημα. Έστω t_k είναι ο γενικός PN

transition, και έστω S είναι το υποσύνολο του $R(M_1)$ το οποίο περικλείει όλες τις σημάνσεις που ενεργοποιούν τον t_k. Ο αναμενόμενος αριθμός πυροδοτήσεων του t_k στο διάστημα (0, t) δίνεται από:

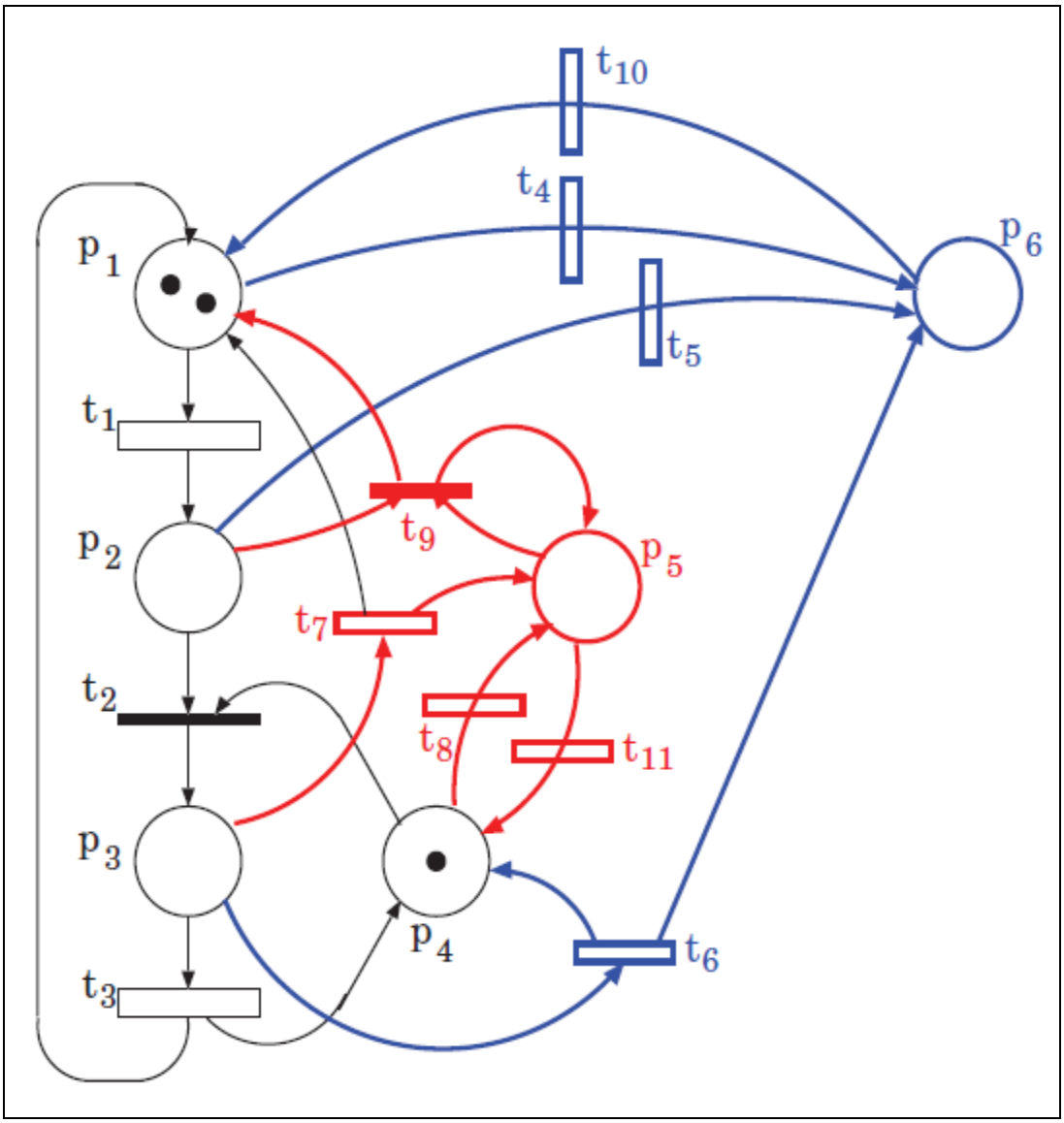
$$\eta_k(t) = \sum_{s \in S} \int_0^t q_s(z)\, \lambda_k(s)\, dz$$

όπου $\lambda_k(s)$ είναι ο ρυθμός πυροδότησης του t_k στην σήμανση s.

Σε σταθερή κατάσταση, ο αναμενόμενος αριθμός πυροδοτήσεων στη μονάδα του χρόνου γίνεται:

$$\nu_k = \sum_{s \in S} q_s(\infty)\, \lambda_k(s)$$

όπου $q_s(\infty)$ είναι η πιθανότητα σταθερής κατάστασης της κατάστασης s.



Σχήμα 1.23. Το σύστημα του σχήματος 1.18 με αποτυχίες και επιδιορθώσεις

Σαν παράδειγμα, αν ο transition t_k δείχνει αποτυχία (επιδιόρθωση) ενός μέρους, το $\eta_k(t)$ μας δίνει τον μέσο αριθμό αποτυχιών (επιδιορθώσεων) αυτού του μέρους στο διάστημα (0, t).

1.14 Μοντελοποίηση επίδοσης/αξιοπιστίας μέσα από τα SPN

Η ανάλυση αξιοπιστίας προσανατολισμένη στην επίδοση είναι το αντικείμενο μίας εκτενούς βιβλιογραφίας τα τελευταία χρόνια. Θα δείξουμε,

με χρήση πλήρως ανεπτυγμένων παραδειγμάτων, ότι η γλώσσα SPN, που περιγράφηκε προηγουμένως, είναι πολύ κατάλληλη για να μοντελοποιήσουμε τέτοιου είδους προβλήματα.

Πίνακας 1

Σημασία των places και transitions στο SPN του σχήματος 1.23

P_1	μονάδα που λειτουργεί ανεξάρτητα	
P_2	μονάδα που περιμένει πρόσβαση στη C_s	
P_3	μονάδα που λειτουργεί με την C_s	
P_4	η C_s ελεύθερη	
P_5	αποτυχία της C_s	
P_6	αποτυχία της μονάδας	
		ρυθμός πυροδότησης
T_1	η μονάδα που απαιτεί πρόσβαση στην C_s	1 m_1
T_2	η μονάδα έχει πρόσβαση στην C_s	10^4
T_3	η μονάδα απελευθερώνει την C_s	5
T_4	αποτυχία της μονάδας σε τοπικό επίπεδο	10^{-4} m_1
T_5	αποτυχία της μονάδας ενώ αναμένει	10^{-4} m_2
T_6	αποτυχία της μονάδας ενώ δουλεύει με την C_s	10^{-4}
T_7	αποτυχία της C_s ενώ δουλεύει	10^{-4}
T_8	αποτυχία της C_s ενώ είναι ελεύθερη	10^{-4}
T_9	επιστροφή σε τοπικό επίπεδο όταν αποτυγχάνει η C_s	10^{-4}
T_{10}	επιδιόρθωση της μονάδας	10^{-2}
T_{11}	επιδιόρθωση της C_s	10^{-2}

1.14.1 Παράλληλες μονάδες με μοιραζόμενους πόρους

Αυτή η κατάσταση έχει απεικονισθεί στο σχήμα 1.8, και παρουσιάζεται πολύ συχνά στα κατανεμημένα συστήματα. Με αναφορά στο σχήμα 1.8 ένα σύστημα πολλαπλών επεξεργαστών οι C_1 και C_2 είναι ανεξάρτητοι επεξεργαστές που εργάζονται τοπικά στις δικές τους ιδιωτικές μνήμες και C_s είναι μία μοιραζόμενη γενική μνήμη η οποία περιέχει κοινά δεδομένα για τους δύο επεξεργαστές. Σε ένα κατασκευαζόμενο σύστημα οι C_1 και C_2 είναι δύο σταθμοί εργασίας που συνδέονται στο ίδιο σύστημα μεταφοράς ή στην ίδια συσκευή load/unload C_s.

Υποθέτοντας ότι οι C_1 και C_2 είναι πανομοιότυπες μονάδες, το SPN που μοντελοποιεί την ελεύθερη σφάλματος λειτουργία του συστήματος δείχνεται στο σχήμα 1.18. Λαμβάνοντας υπόψη τις αποτυχίες και τις επιδιορθώσεις κάθε μονάδας η λειτουργία του συστήματος μοντελοποιείται από το SPN του σχήματος 1.23.

Ο πίνακας 1 αναφέρει την σημασία των places και των transitions του σχήματος 1.23, και τις αριθμητικές τιμές του ρυθμού πυροδότησης που σχετίζεται με κάθε transition. Με την αρχική σήμανση M_1 που φαίνεται στο σχήμα 1.10, το σύνολο προσεγγισημότητας $R(M_1)$ αποτελείται από 15 καταστάσεις των οποίων η κατανομή των token αναφέρεται στον πίνακα 2. Με επιθεώρηση των πινάκων 1 και 2 τα ακόλουθα υποσύνολα καταστάσεων αναγνωρίζονται:

- Καταστάσεις 1, 2, 5, 6, 11: λειτουργία του συστήματος ελεύθερη λαθών
- Καταστάσεις 3, 7, 13: κανονική λειτουργία της μονάδας όταν η άλλη είναι σε μία συνθήκη αποτυχίας
- Καταστάσεις 4, 8, 12: δύο μονάδες λειτουργούν και ο μοιραζόμενος πόρος έχει αποτύχει
- Καταστάσεις 10, 14: μία μονάδα λειτουργεί ενώ η άλλη και ο μοιραζόμενος πόρος έχουν αποτύχει
- Κατάσταση 9: οι δύο μονάδες απέτυχαν
- Κατάσταση 15: οι δύο μονάδες και ο μοιραζόμενος πόρος απέτυχαν

Πίνακας 2

Σύνολο προσεγγισημότητας και κατανομή των token του SPN του σχήματος 1.23

State	Marking					
	m_1	m_2	m_3	m_4	m_5	m_6
1	2	0	0	1	0	0
2	1	1	0	1	0	0
3	1	0	0	1	0	1
4	2	0	0	0	1	0
5	0	2	0	1	0	0
6	1	0	1	0	0	0
7	0	1	0	1	0	1
8	1	1	0	0	1	0
9	0	0	0	1	0	2
10	1	0	0	0	1	1
11	0	1	1	0	0	0
12	0	2	0	0	1	0
13	0	0	1	0	0	1
14	0	1	0	0	1	1
15	0	0	0	0	1	2

Ο πίνακας 3 είναι η κατά γράμμα περιγραφή του γράφου προσεγγισημότητας $G_R(M_1)$ του SPN, για κάθε κατάσταση του $R(M_1)$ στην πρώτη στήλη, οι ενεργοί transitions και οι άμεσα προσεγγίσιμες καταστάσεις (στις παρενθέσεις) αναφέρονται. Αντικαθιστώντας τις αριθμητικές τιμές των ρυθμών πυροδότησης που αναφέρονται στον πίνακα 1 στις ετικέτες των transitions του πίνακα 3, ο πίνακας ρυθμού των transition **Λ** της σχετικής αλυσίδας Markov μπορεί αυτόματα να δημιουργηθεί.

Θεωρώντας τον χρόνο πρόσβασης στην C_s ως αμελητέο με σεβασμό στη χρονική σταθερά του συστήματος, οι t_2 και t_9 μπορούν να θεωρηθούν άμεσοι transitions. Με αυτή την υπόθεση φαίνεται στον πίνακα 3 ότι οι καταστάσεις {2, 5, 7, 8, 12, 14} γίνονται άμεσες εφόσον σε αυτές τις καταστάσεις ένας από τους άμεσους transitions είναι ενεργός. Μειώνοντας το state space, η τελική αλυσίδα Markov προσδιορίζεται με ένα state space που περιέχει 9 ενσωματωμένες καταστάσεις.

Ένα ενδιαφέρον μέτρο επίδοσης/αξιοπιστίας για αυτό το σύστημα είναι ο αριθμός των μονάδων που εκτελούν χρήσιμη εργασία σε χρόνο t,

όπου χρήσιμη εργασία εννοούμε την εργασία που θα γινόταν από κάθε μονάδα όταν θα λειτουργούσε ανεξάρτητα. Αυτό το μέτρο λαμβάνει υπόψη την μείωση στην απόδοση του συστήματος λόγω των διαφορετικών επιδράσεων: τις καθυστερήσεις συμφόρησης λόγω του διαμοιρασμού του κοινού πόρου, την μεταφορά δεδομένων ή κομμάτια από κάθε μονάδα στο C_s και τους κύκλους αποτυχίας ή επιδιόρθωσης. Χρησιμοποιώντας τους ορισμούς της προηγούμενης παραγράφου και βλέποντας τον πίνακα 1, φαίνεται ότι αυτό το μέτρο συμπίπτει με τον αναμενόμενο αριθμό tokens στον place p_1.

Πίνακας 3

Κατά γράμμα περιγραφή του γράφου προσεγγισημότητας $G_R(M_1)$

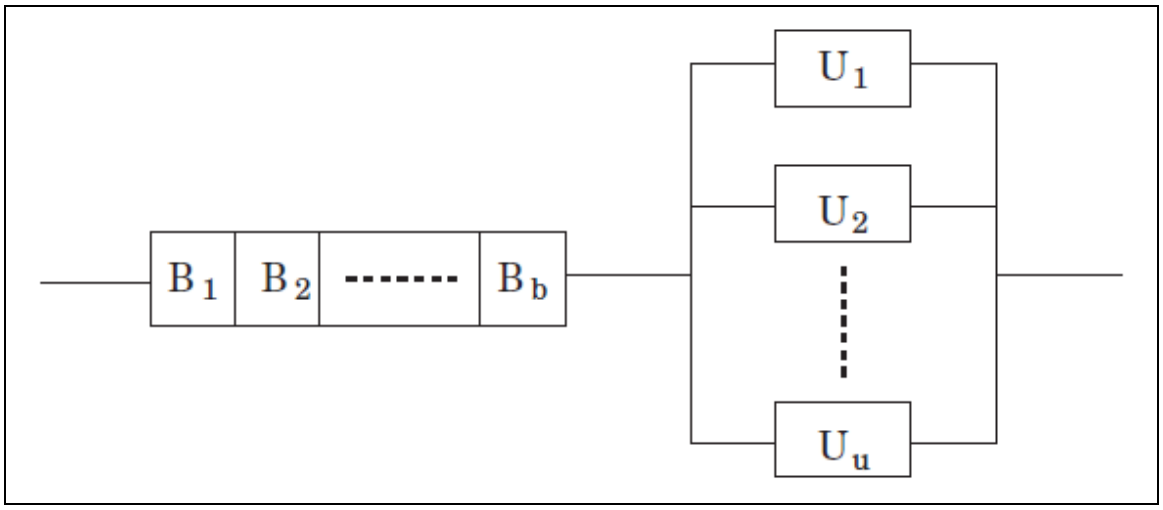
State	Enabled transition and immediately reachable state											
1	1	(2)	4	(3)	8	(4)						
2	1	(5)	2	(6)	4	(7)	5	(3)	8	(8)		
3	1	(7)	4	(9)	8	(10)	10	(1)				
4	1	(8)	4	(10)	11	(1)						
5	2	(11)	5	(7)	8	(12)						
6	1	(11)	3	(1)	4	(13)	6	(3)	7	(4)		
7	2	(13)	5	(9)	8	(14)	10	(2)				
8	1	(12)	4	(14)	5	(10)	9	(4)	11	(2)		
9	8	(15)	10	(3)								
10	1	(14)	4	(15)	10	(4)	11	(3)				
11	3	(2)	5	(13)	6	(7)	7	(8)				
12	5	(14)	9	(8)	11	(5)						
13	3	(3)	6	(9)	7	(10)	10	(6)				
14	5	(15)	9	(10)	10	(8)	11	(7)				
15	10	(10)	11	(9)								

1.14.2 Παράλληλα συστήματα με πεπερασμένο buffer εισόδου

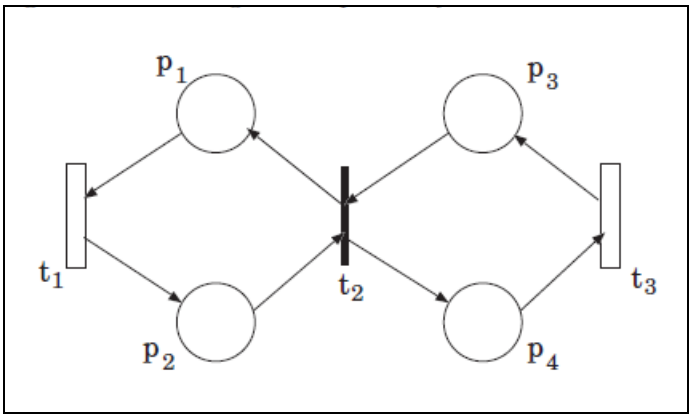
Το block διάγραμμα του συστήματος δείχνεται στο σχήμα 1.24. Αποτελείται από *υ* ταυτόσημες μονάδες $\mathbf{U}_1$, $\mathbf{U}_2$, ...,$\mathbf{U}_U$ και από ένα buffer εισόδου με b θέσεις $\mathbf{B}_1$,$\mathbf{B}_2$, ..., $\mathbf{B}_b$.

Το μοντέλο GSPN της ελεύθερης σφαλμάτων λειτουργίας του συστήματος δείχνεται στο σχήμα 1.25, το άθροισμα των tokens στους p_1 και p_2 είναι ίσο με b (αριθμός των θέσεων του buffer), από την άλλη το άθροισμα των tokens των p_3 και p_4 είναι ίσο με u (αριθμός των παράλληλων μονάδων).

Ο ρυθμός πυροδότησης που σχετίζεται με τον t_1 είναι ο ρυθμός άφιξης εργασιών λ, ενώ ο ρυθμός πυροδότησης που σχετίζεται με τον t_3 είναι ο ρυθμός εξυπηρέτησης ανάλογος με τον με τον αριθμό των ενεργών μονάδων m_4μ, όπου μ ο ρυθμός εξυπηρέτησης μίας απλής μονάδος και m_4 ο αριθμός των tokens στον p_4. Ο t_2 είναι ένας άμεσος transition.



Σχήμα 1.24. Block διάγραμμα ενός παράλληλου συστήματος με πεπερασμένο buffer

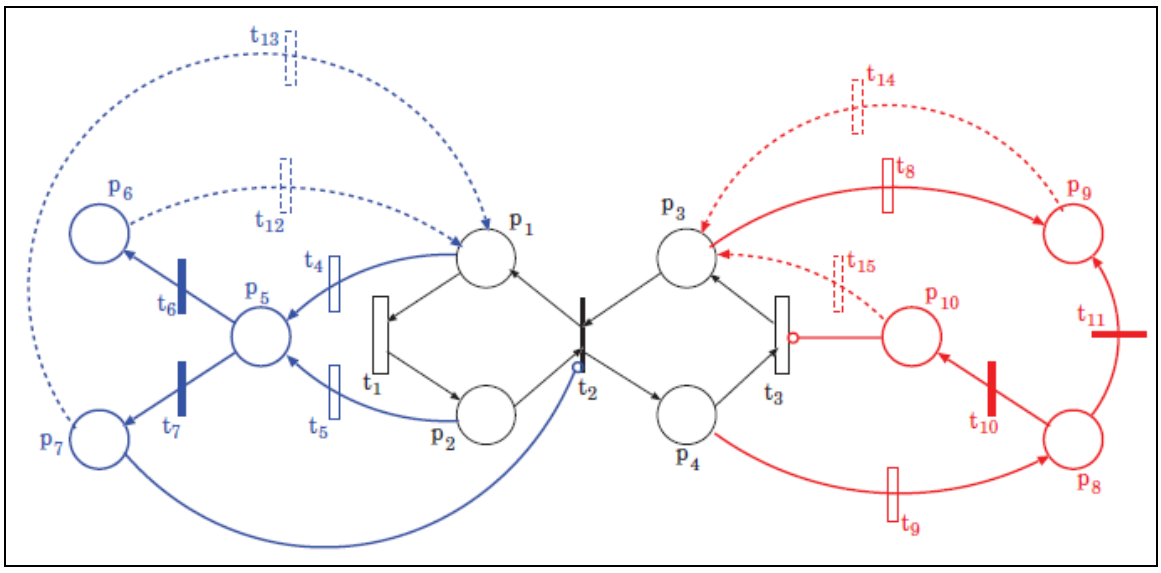


Σχήμα 1.25. Ελεύθερο σφαλμάτων SPN μοντέλο του συστήματος του σχήματος 1.24

Όταν αποτυχίες και επιδιορθώσεις συντελούνται, το GSPN μοντέλο γίνεται όπως στο σχήμα 1.26. Οι έντονες γραμμές αντιπροσωπεύουν την ελεύθερη σφαλμάτων λειτουργία, οι light γραμμές τις αποτυχίες και οι διακεκομμένες γραμμές τις επιδιορθώσεις. Εστιάζοντας στους transitions που αποτυγχάνουν, με αναφορά στο σχήμα 1.26 οι ακόλουθες υποθέσεις γίνονται:

- Τα στάδια των buffer αποτυγχάνουν μία φορά, είτε όταν είναι ελεύθερα (t_4), είτε όταν είναι απασχολημένα (t_5), με πιθανούς διαφορετικούς ρυθμούς αποτυχίας. Οι t_6 και t_7 σχηματίζουν ένα τυχαίο διακόπτη που μοντελοποιεί το γεγονός ότι με πιθανότητα u_B μία αποτυχία του σταδίου του buffer ανακτάται (το buffer συνεχίζει να είναι λειτουργικό με δυνατότητα αποθήκευσης μειωμένη κατά ένα στάδιο), και με πιθανότητα 1- u_B ότι η αποτυχία δεν ανακτάται και το buffer «κλειδώνει» (βέλος αναστολέα από τον p_7 στον t_2).

- Οι μονάδες $\mathbf{U}_i$ (i= 1, 2, . . . , υ) αποτυγχάνουν είτε όταν είναι αδρανείς (t_8) ή όταν είναι ενεργοί (t_9), με πιθανότατα διαφορετικούς ρυθμούς αποτυχίας. Η αποτυχία μίας αδρανούς μονάδας ανακτάται με πιθανότητα ένα (1), ενώ η αποτυχία μίας ενεργούς μονάδας ανακτάται με πιθανότητα u_U (τυχαίος διακόπτης t_{10} t_{11}). Μία εργασία χάνεται όταν μία ενεργή μονάδα αποτυγχάνει.



Σχήμα 1.26. Το SPN μοντέλο του συστήματος του σχήματος 1.24 με αποτυχίες και επιδιορθώσεις

Με μία μικρή τροποποίηση του GSPN του σχήματος 1.26, διαφορετικές εναλλακτικές σχεδιασμού ή στρατηγικές ανάκτησης μπορούν να υιοθετηθούν. Όταν μία επιδιόρθωση γίνεται, οι t_{12} και t_{13} αναφέρεται στην επιδιόρθωση του σταδίου των buffer, και οι t_{14} και t_{15} στην επιδιόρθωση του επεξεργαστή, το συγκεκριμένο μοντέλο μας επιτρέπει να έχουμε διαφορετικούς ρυθμούς επιδιόρθωσης για ανακτώμενες και μη ανακτώμενες αποτυχίες.

Η αρχική σήμανση M_1 αποτελείται από b tokens στον place p_1 και u tokens στον place p_3. Ως μέτρα που χαρακτηρίζουν την επίδοση και την αξιοπιστία του συστήματος, ορίζουμε τα ακόλουθα :

- *Μέσο κλάσμα των ληφθέντων εργασιών που επεξεργάστηκαν σε 0-t*

Ο μέσος αριθμός των εργασιών που επεξεργάστηκαν σε 0-t δίνεται από τον μέσο αριθμό των πυροδοτήσεων του t_3. Ο μέσος αριθμός των ληφθέντων εργασιών σε 0-t είναι απλά λ*t, αν υποθέσουμε διαδικασία αφίξεων Poisson με ρυθμό λ. Για αυτό ο δείκτης επίδοσης/αξιοπιστίας Y(t), αντιπροσωπεύει το μέσο κλάσμα των ληφθέντων εργασιών που επεξεργάστηκαν σε 0-t που υπολογίζεται σε:

$$Y(t) = \frac{\eta_3(t)}{\lambda t}$$

- *Μέσος αριθμός αποτυχιών (επιδιορθώσεων) σε 0-t*

Αυτή η ποσότητα από $[\eta_4(t) + \eta_5(t)]$ για την αποτυχία του σταδίου του buffer ($[\eta_{12}(t) + \eta_{13}(t)]$ για επιδιόρθωση), και με $[\eta_8(t) + \eta_9(t)]$ για αποτυχία μονάδας ($[\eta_{14}(t) + \eta_{15}(t)]$ για επιδιόρθωση μονάδας).

- *Cdf και μέσος αριθμός ενεργών, αδρανών, αποτυχόντων, μονάδων ή σταδίων buffer*

Αυτές οι ποσότητες λαμβάνονται εφαρμόζοντας την διαδικασία που ακολουθήθηκε στην παράγραφο για την κατανομή των token σε ένα place,για τον place p_4 σε ενεργές μονάδες, για τον place p_5 για αδρανείς μονάδες, και για τους places $[p_9+p_{10}]$ για μονάδες με αποτυχία. Ομοίως, ο place p_1 δείχνει τα στάδια ελεύθερου buffer, ο p_2 τα στάδια γεμάτου buffer, και $[p_6+p_7]$ τα στάδια αποτυχίας του buffer.

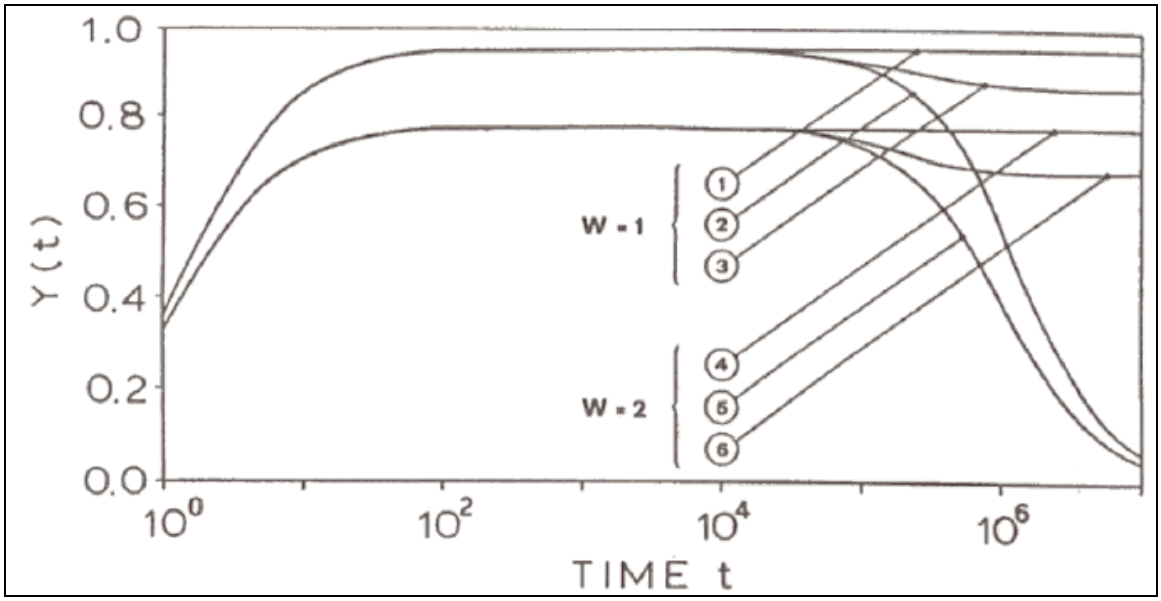
Πίνακας 4

Η σημασία των places και transitions του SPN του σχήματος 1.26

P_1	ελεύθερο στάδιο buffer	
P_2	απασχολημένο στάδιο buffer	
P_3	αδρανής μονάδα	
P_4	ενεργή μονάδα	
P_5	αποτυχία σταδίου buffer	
P_6	ανάκτηση αποτυχίας σταδίου buffer	
P_7	μη ανάκτηση αποτυχίας σταδίου buffer	
P_8	αποτυχία ενεργούς μονάδος	
P_9	ανάκτηση αποτυχίας μονάδας	
p_{10}	μη ανάκτηση αποτυχίας μονάδας	
		ρυθμός πυροδότησης
T_1	το στάδιο buffer γίνεται απασχολημένο	λ
T_2	μεταφορά από buffer σε μονάδα	άμεσα
T_3	η μονάδα τελειώνει μία εργασία	$m_4\mu$
T_4	αποτυχία σταδίου ελεύθερου buffer	$m_1\gamma_4$
T_5	αποτυχία σταδίου απασχολημένου buffer	$m_2\gamma_5$
T_6	αποτυχία σταδίου buffer ανακτάται	υ_B
T_7	αποτυχία σταδίου buffer δεν ανακτάται	$(1-\upsilon_B)$
T_8	αποτυχία αδρανούς μονάδος	$m_3\gamma_8$
T_9	αποτυχία ενεργής μονάδος	$m_4\gamma_9$
t_{10}	αποτυχία μονάδος δεν ανακτάται	$(1-\upsilon_U)$
t_{11}	αποτυχία μονάδος ανακτάται	υ_U
t_{12}	επιδιόρθωση ανακτημένου σταδίου buffer	ρ_{12}
t_{13}	επιδιόρθωση μη ανακτημένου σταδίου buffer	ρ_{13}
t_{14}	επιδιόρθωση ανακτημένης μονάδος	ρ_{14}
t_{15}	επιδιόρθωση μη ανακτημένης μονάδος	ρ_{15}

Ένα αριθμητικό παράδειγμα έχει εκτελεστεί με υ=2 και b=2. Το σύνολο προσεγγισημότητας, σε αυτή την περίπτωση, περιλαμβάνει 88 ενσωματωμένες καταστάσεις και 84 αμελητέες καταστάσεις. Με αναφορά στον πίνακα 4, δώσαμε στις παραμέτρους τις παρακάτω αριθμητικές τιμές (με $w = \lambda/\mu$ η ένταση φορτίου του συστήματος):

$$
\begin{aligned}
\mu &= 1 \\
\lambda &= w\,\mu \\
\gamma_4 &= \gamma_5 = \gamma_8 = \gamma_9 = \gamma = 1.0\,10^{-6} \\
\rho_{12} &= \rho_{13} = \rho_{14} = \rho_{15} = 10\,\gamma \\
v_B &= v_U = 0.9
\end{aligned}
$$



Σχήμα 1.27. Μέσο κλάσμα των ληφθέντων εργασιών που επεξεργάστηκαν σε
0-t σε σχέση με το χρόνο

Δύο περιπτώσεις εξετάζονται με διαφορετικές εντάσεις φορτίου: w=1 και w=2. Η τελευταία περίπτωση αντιπροσωπεύει την ιδανική ένταση φορτίου εφόσον ο ρυθμός άφιξης είναι δύο φορές μεγαλύτερος από τον ρυθμό εξυπηρέτησης, αλλά υπάρχουν δύο παράλληλες μονάδες εξυπηρέτησης. Τα αριθμητικά αποτελέσματα έχουν παρθεί μα χρήση του προγράμματος ESP και χρήση μίας τεχνικής αποσύνθεσης λόγω της σημαντικής απόκλισης στις τιμές του ρυθμού πυροδότησης. Το σχήμα 1.27 δείχνει Y(t) ως συνάρτηση του t για τις δύο επιλεγμένες τιμές του w, και σε τρείς διαφορετικές συνθήκες, ονομαστικά : ελεύθερη σφαλμάτων λειτουργία (καμπύλες 1 και 4), με αποτυχίες (καμπύλες 2 και 5), με αποτυχίες και επιδιορθώσεις (καμπύλες 3 και 6). Το σχήμα 1.27 δείχνει πως η επίδοση του συστήματος (ρυθμαπόδοση) υποβαθμίζεται όταν υπάρχουν αποτυχίες και αποτυχίες/επιδιορθώσεις, και μπορεί να είναι μία αξιόλογη υποστήριξη στο επίπεδο σχεδιασμού.

II

Κεφάλαιο

Εισαγωγή στα Coloured Petri *N*ets

2 Εισαγωγή στα Coloured Petri Nets

2.1 Εισαγωγή

Ένας αυξανόμενος αριθμός των system development projects ασχολούνται με κατανεμημένα και σύγχρονα συστήματα. Υπάρχουν πολλά παραδείγματα, τα οποία κυμαίνονται από μεγάλης κλίμακας συστήματα, στους τομείς των τηλεπικοινωνιών και των εφαρμογών που βασίζονται στην WWW τεχνολογία, σε μεσαίες ή μικρές κλίμακας συστήματα, στην περιοχή των ενσωματωμένων συστημάτων.

Η ανάπτυξη των σύγχρονων και κατανεμημένων συστημάτων είναι πολύπλοκη. Ένας σημαντικός λόγος είναι ότι η εκτέλεση των τέτοιων συστημάτων μπορεί να προχωρήσει πολλούς διαφορετικούς τρόπους, π.χ., ανάλογα με το αν τα μηνύματα χάνονται, η ταχύτητα των διεργασιών που εμπλέκονται, καθώς και η χρονική στιγμή κατά την οποία οι είσοδοι λαμβάνονται από το περιβάλλον. Ως αποτέλεσμα, τα κατανεμημένα και σύγχρονα συστήματα είναι από τη φύση τους περίπλοκα και δύσκολα στον σχεδιασμό και τη δοκιμή.

Τα Coloured Petri-Net[10] (CP-nets ή CPNs) παρέχουν ένα πλαίσιο για την κατασκευή και την ανάλυση των σύγχρονων και κατανεμημένων συστημάτων. Ένα CPN μοντέλο του συστήματος περιγράφει τις καταστάσεις που το σύστημα μπορεί να είναι και τις μεταβάσεις μεταξύ των καταστάσεων αυτών. CP-Nets έχουν εφαρμοστεί σε ένα ευρύ φάσμα τομέων εφαρμογής, και πολλά έργα που έχουν πραγματοποιηθεί στη βιομηχανία[10] και καταγράφονται στη βιβλιογραφία, π.χ., στους τομείς των πρωτοκόλλων επικοινωνίας, τα λειτουργικά συστήματα, το σχεδιασμό hardware, τα ενσωματωμένα συστήματα, software system designs, καθώς και business process re-engineering.

Η ανάπτυξη της CP-nets ωθήθηκε από το επιθυμία να αναπτυχθεί μια δυνατή βιομηχανική modeling language ,ταυτόχρονα όμως καλά βασισμένη και αρκετά ευέλικτη ώστε να χρησιμοποιείτε στην πράξη για συστήματα στο μέγεθος και την πολυπλοκότητα των συνηθισμένων βιομηχανικών έργων . Για να επιτευχθεί αυτό, έχουμε συνδυάσει τη δύναμη των Petri-net , με τη δύναμη των γλωσσών προγραμματισμού. Τα Petri-nets παρέχουν το πλεονέκτημα της σύγχρονης περιγραφής σύγχρονων συστημάτων ,ενώ οι γλώσσες προγραμματισμού παρέχουν το πλεονέκτημα του ορισμού των τύπων δεδομένων (colour sets) και τον χειρισμό των τιμών των δεδομένων.

Τα CPN μοντέλα μπορεί να είναι δομημένα σε μια σειρά συναφών ενοτήτων. Αυτό είναι ιδιαίτερα σημαντικό όταν πρόκειται για την διαχείριση μεγάλων συστημάτων με χρήση CPN . Η έννοια της ενότητας στα CPN-nets

βασίζεται σε ένα μηχανισμό με ιεραρχική δόμηση, ο οποίος υποστηρίζει την bottom-up, καθώς και topdown τρόπο εργασίας. Νέες ενότητες που μπορούν να δημιουργηθούν από υφιστάμενες ενότητες, και οι ενότητες μπορούν να επαναχρησιμοποιηθούν σε διάφορα τμήματα του CPN model. Μέσω του δομημένου μηχανισμού είναι δυνατόν να μπορέσουμε να αντλήσουμε διαφορετικά επίπεδα του πρότυπου συστήματος με το ίδιο CPN model. Ένα CPN- μοντέλο το οποίο αντιπροσωπεύει ένα υψηλό επίπεδο λήψης είναι συνήθως φτιαγμένο κατά τα πρώτα στάδια του σχεδιασμού ή ανάλυσης. Αυτό το μοντέλο είναι που σταδιακά επαναπροσδιορίζεται με μια πιο λεπτομερή και ακριβή περιγραφή του συστήματος που είναι υπό εξέταση.

Τα CPN μοντέλα είναι εκτελέσιμα. Αυτό σημαίνει ότι είναι δυνατόν να διερευνηθεί η συμπεριφορά του συστήματος με το να κάνεις προσομοιώσεις του CPN model. Πολύ συχνά, οι προσομοιώσεις γίνονται για να επικυρωθεί το σχέδιο του συστήματος. Ωστόσο, οι προσομοιώσεις μπορούν εξίσου καλά να χρησιμεύσουν ως βάση για τη διερεύνηση της απόδοσης του συστήματος.

Visualisation είναι μια τεχνική που είναι στενά συνδεδεμένη με την προσομοίωση των CPN models. Παρατήρηση του κάθε βήματος σε μια προσομοίωση είναι συχνά πολύ λεπτομερές για ένα επίπεδο παρατήρησης της συμπεριφοράς του συστήματος. Παραδίδει στον παρατηρητή μία τεράστια ποσότητα λεπτομέρειας ,ιδιαίτερα για πολύ μεγάλα CPN- μοντέλα. Μέσω της υψηλού επιπέδου οπτικής ανατροφοδότησης από τις προσομοιώσεις, πληροφορίες σχετικά με την εκτέλεση του συστήματος μπορούν να αποκτηθούν σε ένα πιο κατάλληλο επίπεδο λεπτομέρειας. Μια άλλη σημαντική εφαρμογή της οπτικοποίησης των CPN είναι η πιθανότητα να παρουσιάζονται σχεδιαστικές ιδέες και αποτελέσματα χρησιμοποιώντας εφαρμοσμένους τομείς εννοιών. Αυτό είναι ιδιαίτερα σημαντικό σε συζητήσεις με τους ανθρώπους και τους συναδέλφους που δεν έχουν εξοικειωθεί με τα CPN-nets.

Ο χρόνος παίζει σημαντικό ρόλο σε ένα ευρύ φάσμα των κατανεμημένων και σύγχρονων συστημάτων. Η σωστή λειτουργία πολλών συστημάτων εξαρτάται αποφασιστικά από τη στιγμή που ελήφθησαν ορισμένες αποφάσεις, και διαφορετικές αποφάσεις σχεδιασμού μπορεί να έχουν σημαντική επίπτωση στην απόδοση ενός συστήματος. Τα CP-nets περιλαμβάνουν μια χρονική έννοια, η οποία καθιστά δυνατό να καταλάβουμε το χρόνο που χρειάστηκαν διαφορετικές δραστηριότητες στο σύστημα. Τα Timed CPN models και η προσομοίωση τους μπορεί να χρησιμοποιηθεί για να αναλυθούν οι επιδόσεις ενός συστήματος, π.χ., να ερευνήσει την ποιότητα της υπηρεσίας (π.χ. καθυστέρηση), ή η ποσότητα των υπηρεσιών (π.χ., throughput) που παρέχονται από το σύστημα. Η έννοια του χρόνου στα CP-nets του είναι κατά κύριο λόγο κατάλληλη για τη διερεύνηση ενός συστήματος μέσο προσομοιώσεων. Αυτό έρχεται σε

αντίθεση αναλυτικών προσεγγίσεις για την ανάλυση των επιδόσεων , και γλωσσών μοντελοποίησης που αποσκοπούν στο model checking , των χρονικών και υβριδικών συστημάτων.

Η state space μέθοδος στα CP-nets καθιστά δυνατή την επικύρωση και επαλήθευση της ορθότητας των συστημάτων. Η state space μέθοδος βασίζεται στον υπολογισμό όλων των πιθανών καταστάσεων και τις πιθανές αλλαγές κατάστασης του συστήματος, και χρησιμοποιεί μια σαφή απαρίθμηση καταστάσεων. Με μια κατασκευασμένη state space, μπορούν να επαληθευθούν οι ιδιότητες του συστήματος. Παραδείγματα τέτοιων ιδιοτήτων είναι η απουσία του αδιεξόδου στο σύστημα, η δυνατότητα πάντα να μπορούμε να οδηγηθούμε σε μία συγκεκριμένη κατάσταση, καθώς και η εγγυημένη περάτωση μίας συγκεκριμένης υπηρεσίας. Η state space μέθοδος των CP-nets μπορεί επίσης να εφαρμοστεί σε timed CP-nets. Ως εκ τούτου, είναι εφικτό να διερευνηθεί η λειτουργική ορθότητα συστημάτων που μοντελοποιήθηκαν με χρήση του πρότυπου των timed CP-nets.

Τα CP-nets έχουν αναπτυχθεί τα τελευταία 20 χρόνια. Ο κύριος φορέας ανάπτυξης είναι η CPN-group στο πανεπιστήμιο του Aarhus, στη Δανία, με επικεφαλή τον Kurt Jensen. Έχουν αναπτύξει το βασικό μοντέλο, συμπεριλαμβανομένης της χρήσης τύπων δεδομένων και ιεραρχικές δομές, έχουν ορίσει τις βασικές έννοιες, όπως δυναμικές ιδιότητες, και έχουν αναπτύξει τη θεωρία πίσω από πολλές από τις υπάρχουσες μεθόδους ανάλυσης. Μαζί με τη Meta Software Corporation, έχουν διαδραματίσει καθοριστικό ρόλο στην ανάπτυξη υψηλής ποιότητας εργαλεία υποστήριξης της χρήσης των CP-nets.

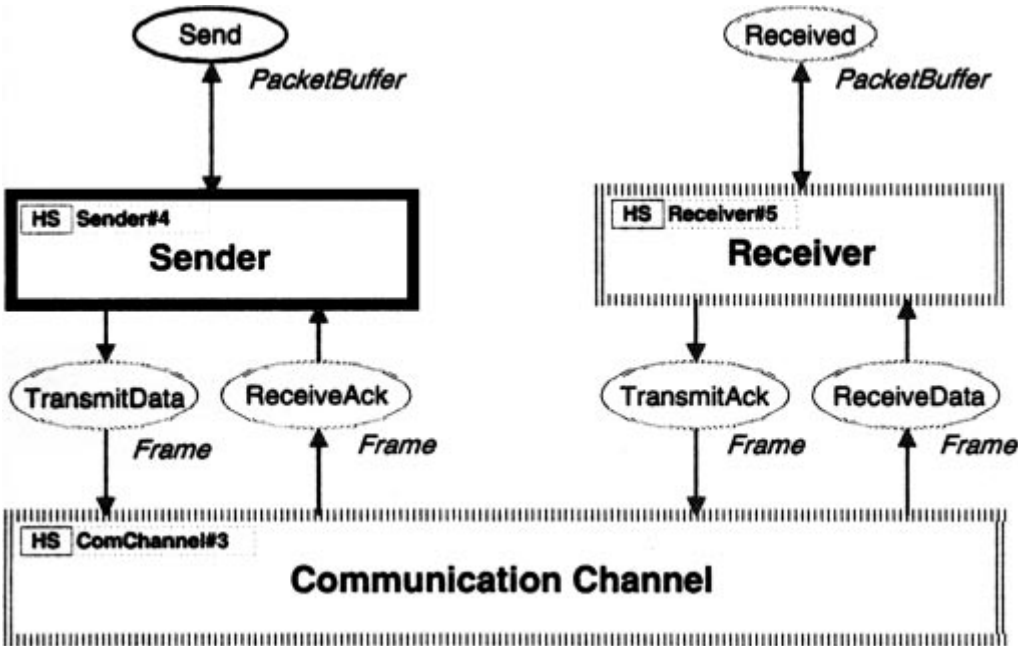
2.2 Coloured Petri nets

Αυτή η ενότητα δίνει μια άτυπη εισαγωγή στο συντακτικό και τη σημασιολογία των CP-nets. Ένα μικρό πρωτόκολλο επικοινωνίας χρησιμοποιείται σε ολόκληρη την ενότητα, προκειμένου να τεκμηριωθούν οι βασικές έννοιες της γλώσσας CPN[11]. Αργότερα, το πρωτόκολλο επικοινωνίας θα χρησιμοποιηθεί επίσης για να εισαγάγει την έννοια του χρόνου και την state space μέθοδο των CP-nets.

2.2.1 Παράδειγμα 1: Ένα πρωτόκολλο επικοινωνίας

Θεωρούμε ένα stop-and-wait πρωτόκολλο από το datalink έλεγχο στρώματος της OSI αρχιτεκτονικής δικτύου. Το stop-and-wait πρωτόκολλο[12] είναι αρκετά απλό, και με κανένα τρόπο δεν αποτελεί ένα εξελιγμένο πρωτόκολλο. Ωστόσο, το πρωτόκολλο είναι αρκετά ενδιαφέρον και αξίζει λεπτομερέστερη έρευνα, και είναι και αρκετά περίπλοκο για την εισαγωγή στις βασικές έννοιες της κατασκευής CP-nets.

Στο σχήμα 2.1 παρουσιάζεται μια επισκόπηση του πρωτοκόλλου που εξετάζουμε. Το σύστημα αποτελείται από ένα Sender (αριστερά) που διαβιβάζει πακέτα δεδομένων σε ένα Receiver (δεξιά). Τα πακέτα δεδομένων που πρέπει να διαβιβαστούν βρίσκονται σε ένα send buffer στη πλευρά του αποστολέα. Η επικοινωνία με το δέκτη λαμβάνει χώρα σε μια αμφίδρομη σύνδεση διαύλου επικοινωνίας (κάτω).



σχήμα 2.1. Stop-and-wait πρωτόκολλο επικοινωνίας

Τα πακέτα δεδομένων πρέπει να παραδίδονται μία φορά και με τη σωστή σειρά στο Received buffer στη πλευρά του δέκτη. Η επιδίωξη αυτής της υπηρεσίας είναι περίπλοκη διότι ο αποστολέας και παραλήπτης επικοινωνούν μέσο αναξιόπιστου διαύλου επικοινωνίας, δηλαδή, τα πακέτα μπορούν να χαθούν κατά τη διάρκεια της μετάδοσης και τα πακέτα μπορεί να προσπεράσει το ένα το άλλο. Ένας τρόπος για την επίτευξη του επιθυμητού αποτελέσματος είναι να χρησιμοποιήσουμε την ονομαζόμενη stop-and-wait στρατηγική αναμετάδοσης. Η ιδέα είναι ότι ο αποστολέας στέλνει το ίδιο πακέτο δεδομένων μέχρι να λάβει μία αποδεκτή απόδειξη παραλαβής(acknowledgement),και όταν τη λάβει μπορεί να μεταδώσει το επόμενο πακέτο. Για λόγους απλότητας, αυτό το stop-and-wait πρωτόκολλο χρησιμοποιεί απεριόριστο αριθμό αναμεταδόσεων.

Ο αποστολέας στέλνει ένα πακέτο δεδομένων στο κανάλι επικοινωνίας, κατασκευάζοντας ένα πλαίσιο δεδομένων(data frame) και τοποθετώντας το στο Transmit Data buffer. Ένα πλαίσιο δεδομένων είναι ένα ζευγάρι που αποτελείται από έναν αριθμό σειράς και ένα πακέτο

δεδομένων. Το κανάλι στη συνέχεια θα επιχειρήσει να μεταδώσει το πλαίσιο δεδομένων, και, εφόσον αποβεί επιτυχής, το πλαίσιο δεδομένων θα πρέπει να παραδοθεί στο Receive Data buffer, όπου μπορεί να επεξεργαστεί από τον δέκτη. Ο δέκτης παραδίδει το πακέτο δεδομένων στα ανώτερα στρώματα του πρωτοκόλλου με τη χρήση των πακέτων δεδομένων που έλαβε το Received buffer. Το πρωτόκολλο χρησιμοποιεί αλληλουχία αριθμών, ώστε να είναι ικανό να ταιριάξει αναγνωρίσεις και πακέτα δεδομένων, δηλαδή, να είναι σε θέση να συναγάγει το ποια αναγνώριση αντιστοιχεί σε ποιο πακέτο δεδομένων, και να είναι σε θέση να καταλάβει αν ένα συγκεκριμένο πακέτο δεδομένων έχει ήδη ληφθεί.

Ο δέκτης στέλνει μια απόδειξη ότι έλαβε ένα πλαίσιο δεδομένων κατασκευάζοντας ένα πλαίσιο απόδειξης και το τοποθετεί στο Transmit Ack buffer.Ένα πλαίσιο απόδειξης αποτελείται από έναν αριθμό που υποδηλώνει τον αριθμό σειράς του επόμενου πακέτου δεδομένων που περιμένει ο δέκτης. Παρόμοια με τα πλαίσια δεδομένων, το κανάλι στη συνέχεια θα επιχειρήσει να μεταδώσει, και, εφόσον αποβεί επιτυχής η μετάδοση, έχει σειρά το Receive Ack buffer στην πλευρά του αποστολέα, όπου μπορούν να επεξεργαστούν από τον αποστολέα.

Αργότερα θα αναφερθούμε στην επιγραφές που τοποθετούνται δίπλα στις ελλείψεις και στην πάνω αριστερή γωνία των κουτιών.

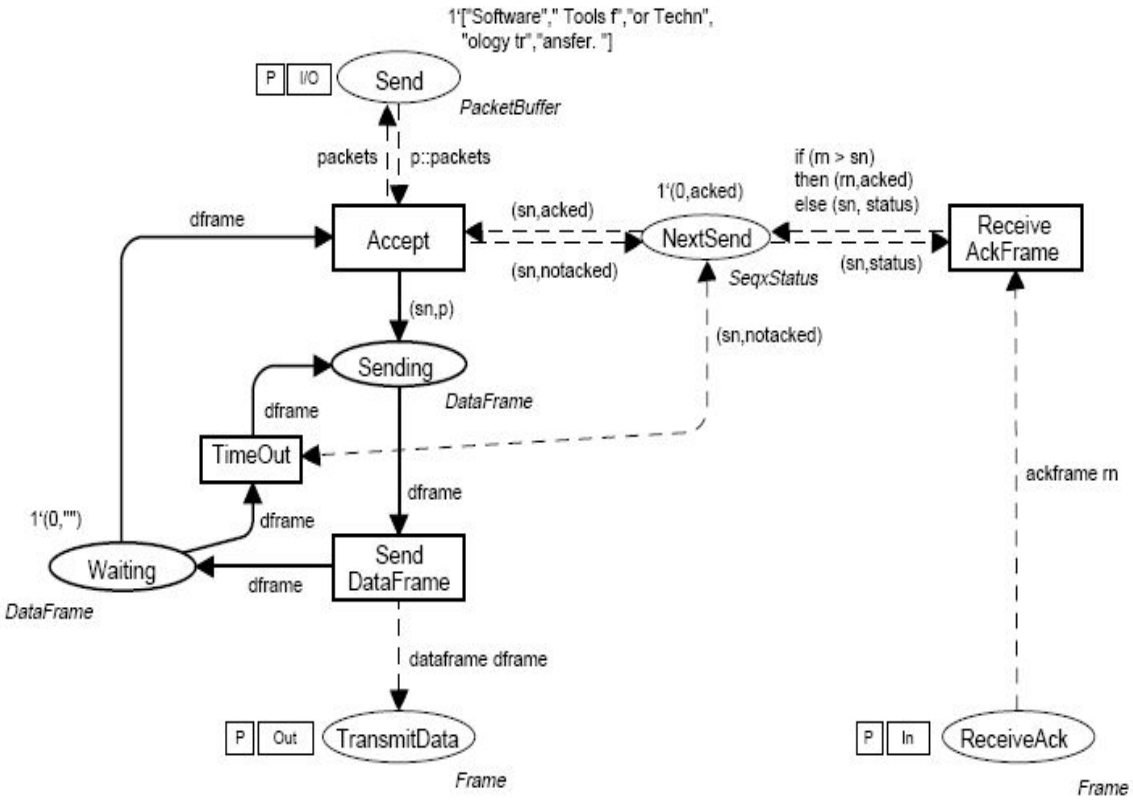
2.2.2 Μοντελοποίηση των καταστάσεων

Τα Petri-net είναι, σε αντίθεση με τις περισσότερες γλώσσες περιγραφής, είναι ταυτόχρονα προσανατολισμένα και στην δράση (action) και την κατάσταση(state), παρέχοντας σαφή περιγραφή των καταστάσεων και των δράσεων του συστήματος. Αυτό σημαίνει ότι, σε μια δεδομένη στιγμή, ο modeler μπορούν ελεύθερα να καθορίσει εάν θα επικεντρωθεί στην κατάσταση ή στη δράση.

Ένα CP-net πάντα αναπτύσσετε ως ένα γραφικό σχέδιο. Το σχήμα 2.2 δείχνει το CP-net που υλοποιεί το μοντέλο του αποστολέα στο stop-and-wait πρωτόκολλο. Αποτελείται από δύο μέρη. Το τμήμα στην αριστερή μοντελοποιεί την αποστολή των πλαισίων δεδομένων, και το μέρος δεξιά που μοντελοποιεί τη βεβαίωση παραλαβής πλαισίων. Θα εξηγήσουμε το σχήμα 2.2 με μεγάλη λεπτομέρεια στις ακόλουθες ενότητες οι οποίες εισάγουν τις βασικές δομές των CP-Nets.

Places: Οι καταστάσεις ενός CP-net εκπροσωπούνται μέσο places (οι οποίοι έχουν συνταχθεί ως ελλείψεις ή κύκλοι). Ο αποστολέας έχει μοντελοποιηθεί έχοντας έξι διαφορετικούς places. Κατά σύμβαση, τα ονόματα των places γράφονται μέσα στις ελλείψεις. Τα ονόματα δεν έχουν καμία επίσημη έννοια, αλλά έχουν τεράστια πρακτική σημασία για την αναγνωσιμότητα ενός CP-net (όπως η χρήση των μνημονικών ονομάτων στον παραδοσιακό προγραμματισμό). Παρόμοια παρατήρηση ισχύει και

για την εμφάνιση γραφικών των places, δηλαδή, το πάχος, το μέγεθος, το χρώμα, γραμματοσειράς, θέση, κλπ.



Σχήμα 2.2. Αποστολέας του stop-and-wait πρωτοκόλλου

Είναι σημαντικό να έχουμε στο μυαλό μας ότι οι places σε αντίθεση, για παράδειγμα, με συστήματα transitions(μεταβάσεων), δεν συνηθίζεται να περιγράφουν ευθέως τις δυνατές καταστάσεις των CP-net. Η κατάσταση ενός CP-net είναι μια λεγόμενη σήμανση(marking) των places του CP-net.Η έννοια του marking εξηγείται μετά την επόμενη παράγραφο.

Ο place SEND (επάνω αριστερά) μοντελοποιεί την απομόνωση των πακέτων δεδομένων, που περιέχουν τα δεδομένα τα οποία ο αποστολέας έχει να διαβιβάσει στον δέκτη. Οι places Transmit Data και Receive Ack (στο βάθος) μοντελοποιούν τον απομονωτή πλαισίων ανάμεσα στον αποστολέα και του διαύλου επικοινωνίας. Ο place Next Send (μεσαία) μοντελοποιεί την εσωτερική κατάσταση του αποστολέα- θα μπορεί να παρακολουθεί τον αύξοντα αριθμό του πακέτου δεδομένων που πρέπει να σταλεί ως επόμενο, και θα αναφέρει αν έχει ληφθεί βεβαίωση για το πακέτο δεδομένων που μεταδίδεται εκείνη την ώρα. Οι places Sending και Waiting μοντελοποιούν τις δύο πιθανές θέσεις της ροής ελέγχου του αποστολέα-

είτε ο αποστολέας είναι έτοιμος να ξεκινήσει την αποστολή(Sending) ενός πλαισίου δεδομένων, ή ο αποστολέας περιμένει(Waiting) αφού έστειλε ένα πλαίσιο δεδομένων.

Types: Κάθε place έχει ένα συναφή type (το χρώμα που έχει) που καθορίζει το είδος των δεδομένων το οποίο ο place μπορεί να περιέχει. Κατά σύμβαση, το type του place είναι γραμμένο με πλάγια γράμματα, κάτω αριστερά ή δεξιά του τόπου. Οι types είναι παρόμοιοι με τα είδη σε μια γλώσσα προγραμματισμού. Οι types ενός CPnet μπορεί να είναι αυθαίρετα περίπλοκοι, π.χ., μία εγγραφή όταν ένα πεδίο είναι πραγματικό (real), ένα άλλο κείμενο (string), και ένα τρίτο λίστα (list) ακεραίων. Το σχήμα 2.3 δείχνει τον ορισμό των types που χρησιμοποιούνται στο stop-and-wait πρωτόκολλο.

Χρησιμοποιούμε οκτώ διαφορετικούς types(color set) στο μοντέλο του αποστολέα. Ο Place Send έχει τον type "PacketBuffer". Αυτός ο type έχει οριστεί ως μία λίστα από "Packets" αντιπροσωπεύοντας το πιθανό περιεχόμενο του απομονωτή πακέτων που μοντελοποιεί ο place Send. Ο type "Packet" έχει οριστεί ως κείμενο(string),δηλώνοντας το σύνολο από strings.

```
(* —— data packets —— *)
color Packet = string ;
color PacketBuffer = list Packet ;

(* —— status and sequence numbers —— *)
color Seq = int ;
color Status = with acked | notacked ;

color SeqxStatus = product Seq * Status ;

(* —— data and acknowledgement frames —— *)
color DataFrame = product Seq * Packet ;
color AckFrame = Seq ;

color Frame = union
      dataframe : DataFrame + ackframe : AckFrame ;
```

Σχήμα 2.3. Ορισμοί type (colour set)

Το type του place Next Send είναι SeqxStatus. Αυτό το type είναι ορισμένος ως το προϊόν (ή ζευγάρι) των type Seq και Status. Ο type Seq είναι ορισμένος ως int (ακέραιοι), καθώς και Status είναι ορισμένος ως απαρίθμηση τύπου που περιέχει δύο πιθανές τιμές: acked και notacked. Ως εκ τούτου, το type SeqxStatus περιέχει όλα τα ζεύγη κατά τα οποία η πρώτη συνιστώσα είναι ένας ακέραιος (δηλωτικός αριθμό σειράς) και η δεύτερη είναι είτε acked ή notacked (αναφέροντας κατά πόσον η βεβαίωση έχει παραληφθεί για τα πακέτα δεδομένων που στέλνονται εκείνη τη στιγμή).

Οι places έχουν Sending και Waiting έχουν τον type DataFrame, ο οποίος είναι ορισμένος ως προϊόν της Sequence αριθμών και Packets. Οι places TransmitData και ReceiveAck έχουν τον type Frame, ο οποίος είναι ορισμένος ως η ένωση του type DataFrame και του type AckFrame. Ο type AckFrame είναι απλά ένας Sequence number(αριθμό σειράς).

Markings: Μια κατάσταση ενός CP-net ονομάζεται marking. Συνίσταται από ένα αριθμό από «μάρκες»(tokens) που έχουν τοποθετηθεί σε επιμέρους places. Κάθε token είναι εφοδιασμένο με μία τιμή (colour), η οποία ανήκει στο type του place στον οποίο το token υπάρχει. Τα tokens που υπάρχουν σε ένα place,αποτελούν και το marking του place αυτού. Για ιστορικούς λόγους, μερικές φορές αναφερόμαστε στην αξία του token ως token colour, με τον ίδιο τρόπο όπως αναφερόμαστε στους τύπους δεδομένων ως colour sets. Αυτή είναι μία μεταφορική εικόνα όπου ενδιαφερόμαστε τα token ενός CP-net πρέπει να διακρίνονται και άρα να είναι "coloured"(χρωματισμένα),σε αντίθεση με τα χαμηλού επίπεδο Petri-net που έχουν μαύρα δυσδιάκριτα token.

Η σήμανση(marking) ενός place είναι, σε γενικές γραμμές, ένα multi-set από αξίες τον token. Ένα multi-set είναι παρόμοιο με ένα σύνολο, εκτός του ότι μπορεί να υπάρχουν αρκετές εμφανίσεις του ίδιου στοιχείου. Αυτό σημαίνει ότι ένα place μπορεί να έχει πολλά token που με την ίδια τιμή. Ως παράδειγμα, ένα πιθανό marking του place TransmitData είναι τα ακόλουθα:

2 `dataframe (0," CP-nets ") + 4` dataframe (1, "CPN")

Η σήμανση περιλαμβάνει δύο token με αξία dataframe (0, "CP-nets") και τέσσερα token με αξία dataframe (1, "CPN"). Κατά σύμβαση, τα multi-sets είναι γραμμένα ως άθροισμα (+) που χρησιμοποιούν το σύμβολο prime (`) για να υποδηλώσει τον αριθμό των εμφανίσεων ενός στοιχείου.

Initial Marking: Ένα CP-net έχει διακριτή σήμανση - Initial Marking, η οποία χρησιμοποιείται για να περιγράψει την αρχική κατάσταση του

συστήματος. Η αρχική σήμανση ενός place είναι, κατά συνθήκη, γραμμένο επάνω αριστερά ή δεξιά του place. Ο place Send έχει αρχική σήμανση αποτελούμενη από ένα μόνο token με τη συμβολική αξία ["Software"," Toolsf","or Techn","ology Tr","ansfer. "], δηλαδή, μια λίστα με πέντε πακέτα. Ο place NextSend αρχικά περιέχει ένα ενιαίο token με την τιμή (0, acked), που δηλώνει ότι στο πακέτο που θα αποστέλλεται πρώτο θα ανατεθεί ο αύξων αριθμός 0, και ότι η βεβαίωση έχει παραληφθεί για το προηγούμενο πακέτο δεδομένων (αφού αρχικά δεν υπάρχει προηγούμενο πακέτο). Αρχικά ο αποστολέας είναι σε κατάσταση αναμονής, όπως φαίνεται από την αρχική σήμανση του place Waiting. Αρχικά, οι υπόλοιποι τρεις places δεν περιλαμβάνουν tokens. Οι προδιαγραφές της αρχικής σήμανσης (κατά συνθήκη) παραλείπονται για αυτούς τους τρείς places.

2.2.3 Μοντελοποίηση των δράσεων

Transitions: Οι δράσεις ενός CP-net, εκπροσωπούνται από την έννοια του transition (τα οποία έχουν συνταχθεί ως ορθογώνια). Όπως με τους places, γράφουμε το όνομα των transitions στο εσωτερικό των ορθογωνίων. Ο αποστολέας αποτελείται από τέσσερις μεταβάσεις. Η μετάβαση Accept μοντελοποιεί την πράξη της λήψης όταν το επόμενο πακέτο δεδομένων είναι αποδεκτό για μετάδοση. Η μετάβαση SendDataFrame μοντελοποιεί την αποστολή ενός πλαισίου δεδομένων, ReceiveAckFrame μοντελοποιεί την παραλαβή του πλαισίου βεβαίωσης.TimeOut είναι το μοντέλο που χρησιμοποιείται για την εμφάνιση των λήξεων χρόνου, έτσι ώστε τα πλαίσια δεδομένων να μπορούν να αναμεταδοθούν.

Arcs and arc expressions: Μεταβάσεις και θέσεις συνδέονται με τόξα. Οι δράσεις ενός CP-net αποτελούνται από ενεργοποιήσεις στις μεταβάσεις. Η ενεργοποίηση μίας μετάβασης αφαιρεί token από places που συνδέονται στα εισερχόμενα τόξα της μετάβασης, και προσθέτει token σε places που συνδέονται σε εξερχόμενα τόξα , έτσι αλλάζει και το marking(κατάσταση) του CP-net. Αυτό αναφέρεται επίσης ως token game. Ως παράδειγμα, η μετάβαση Accept έχει τρία εισερχόμενα τόξα και τρία απερχόμενα τόξα. Ως εκ τούτου, ένα συμβάν αυτής της μετάβασης θα πάρει tokens από τους places Waiting, Send και NextSend,και θα προσθέσει tokens στους places Sending,Send, και NextSend.

Ο ακριβής αριθμός των token που προστίθεται και αφαιρείται από ένα συμβάν μίας μετάβασης, και οι τιμές των δεδομένων τους ορίζονται από τα arc expressions, τα οποία είναι τοποθετημένα δίπλα στα τόξα. Το πώς καθορίζουν τις τιμές των token που αφαιρούνται και προστίθενται θα εξηγηθεί στην επόμενη υποενότητα. Ένα διπλό τόξο, όπως το διακεκομμένο τόξο μεταξύ των TimeOut και NextSend, είναι συντομογραφία για δύο

αντίρροπα τόξα με ταυτόσημη arc expression. Όπως θα δούμε στην επόμενη ενότητα, μια ενέργεια ενός CP-net συνίσταται, γενικά, όταν μία ή περισσότερες μεταβάσεις συμβαίνουν ταυτόχρονα.

2.2.4 Δυναμική συμπεριφορά

Τώρα θα περιγράψουμε τη δυναμική συμπεριφορά των CP-Nets. Δηλαδή, οι συνθήκες υπό τις οποίες οι μεταβάσεις μπορεί να ενεργοποιηθούν, και την επίδραση μίας ενεργοποίησης μίας μετάβασης στο marking ενός CP-net.

Variables and bindings: Για να μιλήσουμε για το συμβάν μίας μετάβασης, πρέπει να εκχωρήσουμε τιμές δεδομένων για τις (ελεύθερες) μεταβλητές που περιλαμβάνονται στην έκφραση των τόξων που περιβάλλουν τη μετάβαση αυτή. Διαφορετικά, είναι αδύνατο να αξιολογηθούν οι arc expressions. Η μετάβαση Accept έχει τέσσερις μεταβλητές: p of type Packet, packets of typePacketBu_er, sn of type Seq, και dframe of type DataFrame.

Η δήλωση μεταβλητών, προσδιορίζει τις μεταβλητές και τους τύπους τους, παρουσιάζονται στο Σχήμα 2.4. Στις μεταβλητές μπορεί να αποδοθούν τιμές δεδομένων που ανήκουν στο type της μεταβλητής.

```
(* --- data packets --- *)
var p : Packet ;
var packets : PacketBuffer ;

(* --- status and sequence numbers --- *)
var sn,rn : Seq ;
var status : Status ;

(* --- data frames --- *)
var dframe : DataFrame ;
```

Σχήμα 2.4. Δήλωση μεταβλητών για το stop-and-wait πρωτόκολλο

Ας υποθέσουμε τώρα ότι έχουμε αποδώσει τιμές δεδομένων στις μεταβλητές της μετάβασης Accept από τη δημιουργία της κλειστής λίστας στο σχήμα 2.5, όπου → θα πρέπει να διαβαστεί "δεσμεύεται".

p	←	"Software"
packets	←	[" Tools f","or Techn","ology Tr","ansfer. "]
sn	←	0
dframe	←	(0,"")

Σχήμα 2.5. Μία δέσμευση για την μετάβαση Accept

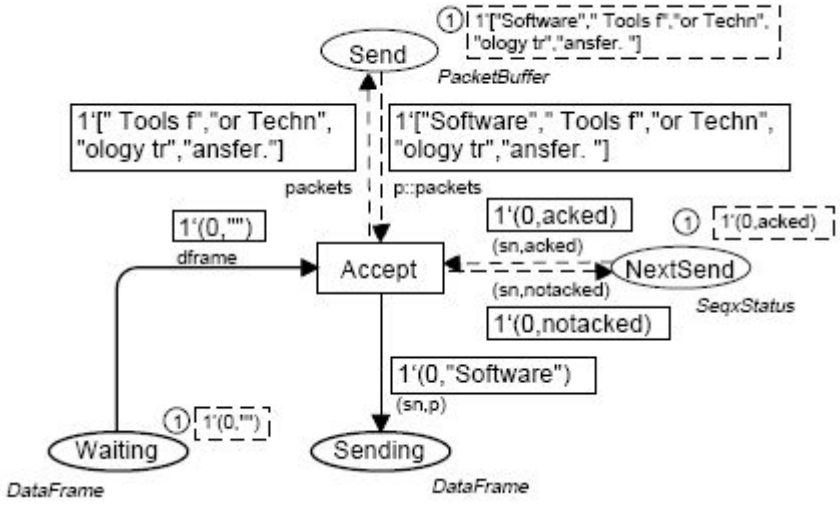
Το σχήμα 2.6 δείχνει, για κάθε τόξο γύρω από το transition Accept, τα multi-set των token που προκύπτουν από την αξιολόγηση των αντίστοιχων arc expressions από την δεσμευτική λίστα στο σχήμα 2.5. Σε αυτή την περίπτωση όλα τα multi-sets περιέχουν ένα μόνο token. Για όλες τις arc expressions, το αποτέλεσμα είναι άμεσο. Τη μόνη εξαίρεση αποτελεί η έκφραση p::packets για το εισερχόμενο τόξο από το place Send. Το σύμβολο :: είναι ο βασικός τελεστής για λίστα. Αυτό σημαίνει ότι το αποτέλεσμα της αντιστοίχισης της έκφρασης p::packets στο binding του σχήματος 2.5 είναι η λίστα που λαμβάνεται εισάγοντας την τιμή "Software", δεσμεύεται στο p, που είναι μπροστά στην δεσμευμένη λίστα από το packets.

Το σχήμα 2.6 δείχνει επίσης την αρχική σήμανση των places που περιβάλλουν τη μετάβαση Accept. Η σήμανση του κάθε place αναγράφεται δίπλα στον τόπο. Ο αριθμός των token σχετικά με τον place παρουσιάζεται στο μικρό κύκλο, ενώ οι λεπτομερείς τιμές των token αναφέρονται στα διακεκομμένα τετράγωνο δίπλα στον μικρό κύκλο.

Ένα δεσμευτικό της μετάβασης γράφεται επίσης με τη μορφή: [υ1 = d1, υ2 = d2,…υn = dn] όπου vi for i 1.. n είναι μια μεταβλητή, και di είναι η τιμή που αποδίδεται στην ui. Ως παράδειγμα, το binding στο σχήμα 2.5 μπορεί να γραφτεί ως:

{p = "Software", sn = 0, dframe = (0, ""),
packets = [" Tools f","or Techn","ology Tr","ansfer. "]}

Επιπρόσθετα στις arc expressions ,είναι δυνατό να και μια Boolean έκφραση με μεταβλητές, σε κάθε μετάβαση. Μια Boolean έκφρασης (με μεταβλητές) για κάθε μετάβαση. Η Boolean έκφραση ονομάζεται φρουρός(guard). Εξ ορισμού θα κάνουμε δεκτά τα bindings για τα οποία η Boolean έκφραση θα έχει την τιμή ‘‘true’’. Ωστόσο, καμία από αυτές τις μεταβάσεις του αποστολέα δεν χρησιμοποιεί φρουρό.



Σχήμα 2.6. Αντιστοίχιση των arc expressions για την μετάβαση Accept

Enabling: Για να μπορεί να ενεργοποιηθεί ένας transition σε ένα marking, δηλαδή, να είναι έτοιμος να συμβεί, θα πρέπει να μπορεί να δεσμεύει (εκχωρήσει) τιμές δεδομένων στις μεταβλητές που εμφανίζονται στα arc expressions γύρω από το transition και στον φρουρό(guard) του transition τέτοια ώστε: 1) κάθε μία από τις arc expressions να αντιστοιχείτε σε tokens που υπάρχουν στον εν λόγω place που αποτελεί την είσοδο του transition και 2) οι μεταβλητές του φρουρού (εάν υπάρχει) να ικανοποιούνται.

Το σχήμα 2.6 δείχνει ότι το συμβάν της Accept (με τη δέσμευση του σχήματος 2.5) αφαιρεί ένα token με τιμή (0,'') από το place Waiting,ένα token με τιμή ["Software"," Tools f", "or Techn","ology Tr","ansfer. "] από το place Send ,και ένα token με τιμή (0,acked) από το place NextSend.

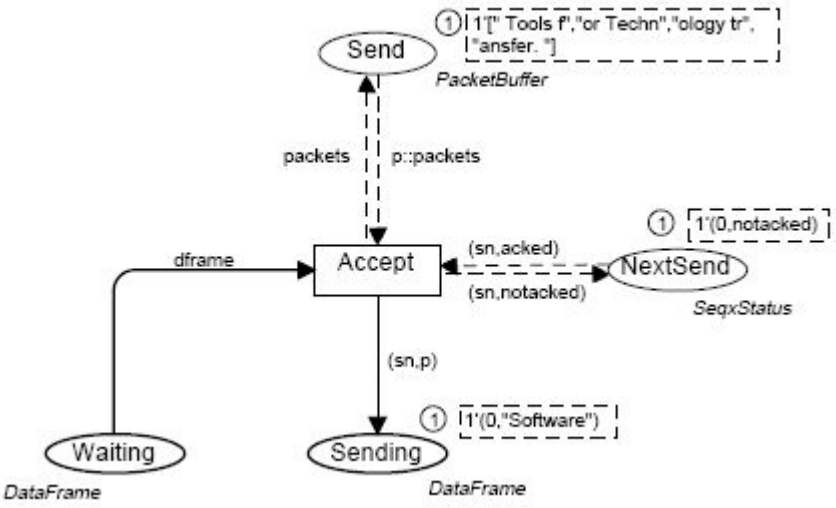
Η δέσμευση της Accept είναι ενεργοποιημένη στην αρχική σήμανση (initial marking),εάν τα token στα οποία η arc expression αντιστοιχεί είναι παρόντα στον εν λόγω place που αποτελεί είσοδο του transition.

Προφανώς, υπάρχουν πολλές άλλες δεσμεύσεις που μπορούμε να δοκιμάσουμε για τον transition Accept στην αρχική σήμανση. Ωστόσο, καμία από αυτές δεν είναι ενεργοποιημένη στην αρχική σήμανση(initial marking). Αυτό μπορεί να εξηγηθεί ως εξής. Ο Send έχει μόνο ένα token. Γι 'αυτό θα πρέπει να δεσμεύσουμε το p στην κεφαλή της λίστας και το packets στην ουρά. Ο NextSend να έχει επίσης ένα token με τιμή (0, acked). Ως εκ τούτου, θα πρέπει να δεσμεύσουμε το sn στο 0. Ο place Waiting περιέχει επίσης ένα token με τιμή (0 ,""), άρα έχουμε την ανάγκη να δεσμεύσουμε το dframe σε αυτήν την τιμή. Άρα, οι δεσμεύσεις που

ενεργοποιούν το transition είναι μοναδικά ορισμένες από τα tokens που υπάρχουν στους places που είναι είσοδοι του transition.

Οι δυνατές δεσμεύσεις των τιμών δεδομένων στις μεταβλητές του transition αντιστοιχούν στους δυνατούς τρόπους που ένας transition μπορεί να δράσει. Ωστόσο, όπως προαναφέραμε, μόνο ένα υποσύνολο τους, γενικά, μπορεί να ενεργοποιηθεί στη δοσμένη σήμανση(marking).

Occurrence: Η ενεργοποίηση ενός transition είναι η ενεργοποίηση μίας δέσμευσης, η οποία αφαιρεί tokens από τον place που λειτουργεί ως είσοδος και προσθέτει tokens στους places που λειτουργούν ως έξοδος στον transition.Οι τιμές των token που αφαιρούνται από το place εισόδου είναι συγκεκριμένες(determined) ικανοποιώντας τις arc expressions του τόξου εισόδου. Ομοίως, οι τιμές των tokens που προστίθενται σε έναν place εξόδου καθορίζονται κατά αντιστοιχία της arc expression του τόξου εξόδου(βλέπε σχήμα 2.6). Το σχήμα 2.7 δείχνει τη σήμανση των γύρω places του transition Accept ,λόγω της ενεργοποίησης του από την αρχική σήμανση(initial marking) με την δέσμευση του σχήματος 2.5.



Σχήμα 2.7. Η σήμανση μετά την ενεργοποίηση της μετάβασης Accept

Ως εκ τούτου, η ενεργοποίηση της μετάβασης Accept έχει ως αποτέλεσμα ότι το πακέτο δεδομένων στην κορυφή της λίστας, που υπήρχαν στον place Send, έχει μετακινηθεί. Η κατάσταση του αποστολέα, που μοντελοποιείτε από τον place NextSend, έχει ενημερωθεί έτσι ώστε τα token, που υπάρχουν στο NextSend, δηλώνουν ότι η βεβαίωση δεν έχει ληφθεί ακόμα (notacked) για το πακέτο δεδομένων, το οποίο τώρα

μεταδίδεται. Ο έλεγχος της ροής του αποστολέα έχει αλλάξει από την κατάσταση αναμονής στην κατάσταση αποστολής, μετακινώντας το token, το οποίο αναφερόταν στο πλαίσιο δεδομένων που μεταδιδόταν προηγουμένως από τον place Waiting, και τοποθετώντας το στον place Sending αναφερόμενο πλέον στο νέο πλαίσιο δεδομένων, που τώρα μεταδίδεται.

Τώρα θα κάνουμε μια σύντομη αναφορά των υπολοίπων στοιχείων του σχήματος 2.2. Η μετάβαση TimeOut είναι ενεργοποιημένη όταν το πλαίσιο δεδομένων που στέλνεται επί του παρόντος βρίσκεται στο Waiting, και δεν έχει ληφθεί ακόμη βεβαίωση (notacked) για το υπό αποστολή πακέτο. Μία ενεργοποίηση του TimeOut αλλάζει τον έλεγχο του αποστολέα από την κατάσταση αναμονής, στην κατάσταση αποστολής, μετακινώντας το πακέτο δεδομένων από τον Waiting και προσθέτοντας το στο place Sending ώστε να μπορεί να αναμεταδοθεί. Με μία πρώτη ματιά, φαίνεται περίεργο το ότι δεν αναφέρουμε τις συνθήκες κάτω από τις οποίες το πακέτο αναμεταδίδεται. Ωστόσο, αυτό δεν είναι απαραίτητο. Τα περισσότερα CP-nets χρησιμοποιούνται για να ερευνήσουν την λογική και λειτουργική ορθότητα του σχεδίου ενός συστήματος. Για αυτό το λόγο είναι προτιμότερο να περιγράφεις τις αναμεταδόσεις που ίσως εμφανιστούν, παράδειγμα, επειδή το κανάλι επικοινωνίας είναι αργό. Ωστόσο μπορεί να μην είναι απαραίτητο, ή και εργονομικό, αν αναλογιστούμε πόσο συχνά συμβαίνει κάτι τέτοιο- πρωτόκολλο πρέπει να είναι ικανό να ανταπεξέλθει σε όλα τα είδη καναλιών επικοινωνίας, και αυτά που είναι τόσο καλά που δεν απαιτείται ποτέ καμία αναμετάδοση, αλλά και αυτά στα οποία οι αναμεταδόσεις είναι συχνές. Αργότερα θα δούμε ότι τα CP-nets μπορούν να επεκταθούν, χρησιμοποιώντας την έννοια του χρόνου που τους επιτρέπει να περιγράψουν την διάρκεια της κάθε δράσης και κάθε κατάστασης ξεχωριστά. Αυτό θα μας επιτρέψει να ερευνήσουμε την απόδοση του μοντελοποιημένου συστήματος, για παράδειγμα, πόσο γρήγορα και αποτελεσματικά λειτουργεί. Τότε θα μία περισσότερο ακριβή περιγραφή των αναμεταδόσεων (π.χ. ότι πραγματοποιούνται όταν δεν έχει ληφθεί βεβαίωση μέσα σε χρόνο 200 ms).

Η παραλαβή των πλαισίων βεβαίωσης μοντελοποιείται από το ReceiveAckFrame. Η ενεργοποίηση αυτού του transition μετακινεί ένα πλαίσιο βεβαίωσης από τον place ReceiveAck και συγκρίνει τον αριθμό σειράς rn του πλαισίου βεβαίωσης με τον αριθμό σειράς sn του πλαισίου δεδομένων που αποστέλλεται εκείνη την ώρα . Αν ο αριθμός σειράς στην βεβαίωση είναι μεγαλύτερος από τον αριθμό σειράς του πλαισίου δεδομένων που αποστέλλεται εκείνη την ώρα (θυμόμαστε ότι ο δέκτης στέλνει τον αριθμό σειράς του πλαισίου δεδομένων που περιμένει ως επόμενο), τότε ο αριθμός σειράς του επόμενου πακέτου δεδομένων ανανεώνεται, και η κατάσταση της βεβαίωσης αλλάζει στο να δείξει ότι μία βεβαίωση έχει ληφθεί για το πλαίσιο δεδομένων που μόλις μεταδόθηκε.

Αυτό επιτυγχάνεται με την if-then-else δομή που χρησιμοποιείτε στο τόξο από το ReceiveAckFrame στο NextSend, το οποίο τοποθετεί ένα token στο NextSend με μία τιμή σύμφωνη στην παραπάνω περιγραφή.

Occurrence sequences and steps: Η εκτέλεση ενός CP-net περιγράφεται από την έννοια μίας σειράς συμβάντων (occurrence sequence). Συγκεκριμενοποιεί τις σημάνσεις οι οποίες επιτυγχάνονται και τα βήματα τα οποία συνέβησαν. Προηγουμένως, είδαμε μία σειρά συμβάντων μήκους ''ένα''. Αποτελούταν από ένα μόνο βήμα, την ενεργοποίηση του Accept με την δέσμευση του σχήματος 2.5 στην αρχική σήμανση (initial marking), και οδηγηθήκαμε στην σήμανση του σχήματος 2.7.

Σε αυτή τη σήμανση, η μετάβαση SendDataFrame ενεργοποιείται σε μία δέσμευση στην οποία στη μεταβλητή dframe έχει εκχωρηθεί η τιμή (0,''Software''). Ως εκ τούτου, σειρά συμβάντων μπορεί να συνεχιστεί με ένα βήμα σχετικό με την ενεργοποίηση της SendDataFrame, με αυτή τη δέσμευση. Αυτό οδηγεί σε μία νέα σήμανση στην οποία η σήμανση του Send και του ReceiveAck παραμένουν αναλλοίωτες, ένα token με τιμή dataframe(0,''Software'') είναι στο TransmitData (σύμφωνα με το πλαίσιο δεδομένων που τοποθετήθηκε στο TransmitData buffer για μετάδοση), και ένα token με τιμή (0,''Software'') στο Waiting.

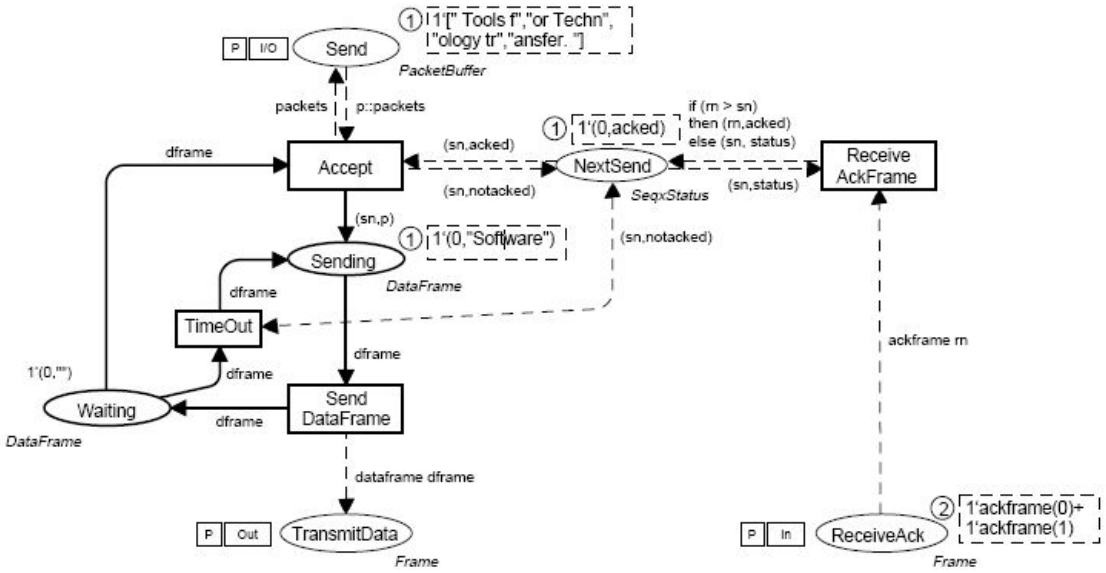
Γενικά, ένα βήμα (step) μπορεί να αποτελείται από πολλά ''δεσμευτικά στοιχεία'' (binding element) ενεργοποιημένα ταυτόχρονα. Ένα ''δεσμευτικό στοιχείο'' είναι ένα ζευγάρι αποτελούμενο από μία μετάβαση και μία δέσμευση των μεταβλητών της. Ως παράδειγμα, μπορούμε να δούμε την σήμανση του αποστολέα που φαίνεται στο σχήμα 2.8. Αυτή η σήμανση είναι ταυτόσημη με την σήμανση που επετεύχθη μετά την δράση της Accept στην αρχική σήμανση, εκτός από το ότι δύο token υπάρχουν στον place ReceiveAck, ένα με τιμή ackframe(0), και ένα με τιμή ackframe(1). Επιπλέον, το token που είναι τώρα στον place NextSend τώρα δείχνει ότι μία βεβαίωση έχει ληφθεί. Σε αυτή τη σήμανση, τα τρία ''δεσμευτικά στοιχεία'' που παρατίθενται παρακάτω είναι ενεργά.

1. (SendDataFrame, [dframe = (0, "Software")])
2. (ReceiveAckFrame, [rn = 0, sn = 0, status = acked])
3. (ReceiveAckFrame, [rn = 1, sn = 0, status = acked])

Τα ''δεσμευτικά στοιχεία'' 1 και 2είναι ταυτόχρονα ενεργά, εφόσον κάθε ένα μπορεί να έχει τα token που χρειάζεται (π.χ., αυτά που ορίζονται από τις arc expressions των εισερχόμενων τόξων)- χωρίς να μοιράζονται τα token με το άλλο ''δεσμευτικό στοιχείο''. Το ίδιο ισχύει και για τα ''δεσμευτικά στοιχεία'' 1 και 3. Ωστόσο, τα ''δεσμευτικά στοιχεία'' 2 και 3 δεν είναι ταυτόχρονα ενεργά, εφόσον δεν μπορούν να πάρουν και τα δύο το μόνο

(0, acked) token του NextSend. Αυτά τα δύο ‘‘δεσμευτικά στοιχεία’’ είναι σε σύγκρουση (conflict).

Σε γενικές γραμμές, είναι δυνατό μία μετάβαση να είναι ταυτόχρονα ενεργή με τον εαυτό της (χρησιμοποιώντας δύο διαφορετικές δεσμεύσεις ή μία δύο φορές). Ως εκ τούτου, ένα βήμα, από μία σήμανση στην επόμενη, μπορεί να εμπλέκει ένα multi-set από ‘‘δεσμευτικά στοιχεία’’. Η σήμανση, που είναι αποτέλεσμα της πραγματοποίησης ενός βήματος που αποτελείται από αρκετά ‘‘δεσμευτικά στοιχεία’’, είναι η ίδια με την σήμανση που επετεύχθη από την δράση των ξεχωριστών ‘‘δεσμευτικών στοιχείων’’ σε κάποια τυχαία σειρά. Αυτή η σήμανση είναι καλά ορισμένη εφόσον το αποτέλεσμα στη σήμανση από την επίδραση ενός multi-set ταυτόχρονα ενεργοποιημένων ‘‘δεσμευτικών στοιχείων’’, είναι ανεξάρτητη από την σειρά που τα ξεχωριστά στοιχεία δρουν.



Σχήμα 2.8. Η σήμανση του αποστολέα

Mia finite occurrence sequence είναι μια ακολουθία συμβάντων που αποτελείται από πεπερασμένες σημάνσεις και βήματα. Μία infinite occurrence sequence είναι μία ακολουθία συμβάντων που αποτελείται από ένα ‘‘άπειρο’’ αριθμό σημάνσεων και βημάτων. Η δεύτερη αναφέρεται σε ένα σύστημα το οποίο η εκτέλεση του δεν τερματίζεται ποτέ.

2.3 Ιεραρχικά CP-nets

Η βασική ιδέα πίσω από τα ιεραρχικά CP-nets είναι να δοθεί στον σχεδιαστή να κατασκευάζει μεγάλα μοντέλα χρησιμοποιώντας κάποιο αριθμό άλλων μικρότερων όπως αυτό στο σχήμα 2.2. Αυτά τα μικρά CP-nets ονομάζονται pages. Αυτές οι pages συσχετίζονται μεταξύ τους με ένα καλά ορισμένο τρόπο όπως αναφέρθηκε προηγουμένως. Αυτό είναι παρόμοιο με την κατάσταση κατά την οποία ο προγραμματιστής κατασκευάζει ένα μεγάλο πρόγραμμα χρησιμοποιώντας ένα σετ από ενότητες. Πολλά CP-nets αποτελούνται από περισσότερες από εκατό pages με συνολικά εκατοντάδες places και transitions. Χωρίς την δυνατότητα κατασκευής ιεραρχικών CP-nets, ένα τέτοιο μοντέλο θα έπρεπε να σχεδιαστεί ως ενιαίο, και θα γινόταν εντελώς ακατανόητο.

Στην προηγούμενη ενότητα περιγράψαμε ένα CPN μοντέλο για τον αποστολέα. Σε αυτή την ενότητα περιγράφουμε μοντέλα για τον δέκτη και το κανάλι επικοινωνίας, και δείχνουμε πώς αυτά τα τρία υπομοντέλα μπορούν να τοποθετηθούν μαζί και να σχηματίσουν ολόκληρο το stop-and-wait πρωτόκολλο.

Σε ένα ιεραρχικό CP-net, είναι δυνατό να συσχετίσεις μία μετάβαση (και τα περιβάλλοντα σε αυτή τόξα και places) σε ένα ξεχωριστό CP-net, παρέχοντας μία πιο ακριβή και λεπτομερή περιγραφή της δραστηριότητας που παρουσιάζεται από την μετάβαση αυτή. Η ιδέα είναι ανάλογη στην ιεραρχική δομή που βρίσκεται σε πολλές γραφικές γλώσσες περιγραφής (π.χ., ροή δεδομένων και SADT διαγράμματα). Είναι επίσης, με κάποιες επιφυλάξεις, ανάλογη στην ιδέα των ενοτήτων που βρίσκουμε σε πολλές μοντέρνες γλώσσες προγραμματισμού. Σε ένα επίπεδο, θέλουμε να δώσουμε μία απλή περιγραφή της μοντελοποιημένης δραστηριότητας χωρίς να επικεντρωθούμε στις εσωτερικές λειτουργίες και πως αυτές διαπεραιώνονται. Σε ένα άλλο επίπεδο, θέλουμε να προσέξουμε πιο λεπτομερώς την συμπεριφορά του συστήματος. Επιπλέον, θέλουμε να μπορούμε να αναβαθμίσουμε τις πιο λεπτομερείς προδιαγραφές με την πιο σχολαστική περιγραφή – και αυτή η αναβάθμιση πρέπει να γίνει με τέτοιο τρόπο ώστε να έχει νόημα να μας πει σχετικά με την συμπεριφορά του σύνθετου συστήματος.

Substitution transitions and subpages. Αρχίζουμε να δουλεύουμε με ένα από πάνω προς τα κάτω τρόπο. Το πιο περιληπτικό Cp-net που περιγράφει το stop-and-wait πρωτόκολλο είναι αυτό που δείχνεται στο σχήμα 2.1. Σε αυτό έχουμε τρείς μεταβάσεις:Sender, Receiver και Communication Channel. Κάθε μία από αυτές τις μεταβάσεις σημειώνεται από μία HS-tag (στην πάνω αριστερή γωνία) υποδεικνύοντας ότι πρόκειται για ένα substitution transition (HS = hierarchy + Substitution).Τα διακεκομμένα κουτιά που βρίσκονται δίπλα στις HS-tags ονομάζονται hierarchy inscriptions και ορίζουν τις λεπτομέρειες της υποκατάστασης.

Κάθε hierarchy inscription ορίζει την subpage, π.χ., το CP-net που περιέχει την λεπτομερή περιγραφή της δραστηριότητας παρουσιάζεται από την εξεταζόμενη υποκατάσταση μετάβασης (substitution transition).Η υποσελίδα (subpage) που αναφέρεται στη υποκατάσταση μετάβασης Sender ονομάζεται Sender και είναι το CP-net που δείχνεται στο σχήμα 2.2. Το ίδιο ισχύει και για τον Receiver και για το Communication Channel. Θα επιστρέψουμε αργότερα στις τελευταίες δύο pages αργότερα σε αυτή την υποενότητα, όπου θα εξηγήσουμε πως μοντελοποιούνται ο Receiver και το Communication Channel.

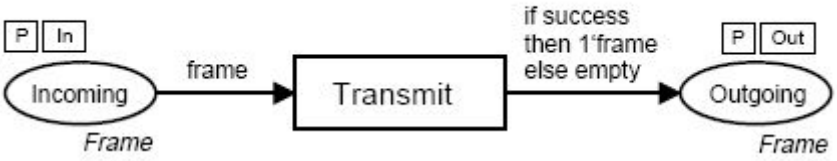
Port and socket places. Κάθε υποσελίδα έχει ένα αριθμό places οι οποίοι σημαδεύονται από μία In-tag, out-tag ή I/O-tag. Αυτοί οι places ονομάζονται ports και αποτελούν τη διεπαφή μέσω της οποίας η υποσελίδα επικοινωνεί με τα περιβάλλοντα στοιχεία. Μέσω των ports εισόδου (In-tags) η υποσελίδα λαμβάνει tokens από τα περιβάλλοντα στοιχεία. Ανάλογα με αυτό, η υποσελίδα εκχωρεί token στα περιβάλλοντα στοιχεία με χρήση των ports εξόδου (Out-tags). Ένας place με I/O-tag είναι και port εισόδου και port εξόδου ταυτόχρονα. Η σελίδα Sender έχει μία port εισόδου ReceiveAck, και μία port εξόδου TransmitData, και μία port I/O Send.

Η υποκατάσταση μετάβασης Sender στο σχήμα 2.1 έχει ένα place εισόδου ReceiveAck, ένα place εξόδου TransmitData και ένα εισόδου/εξόδου place Send. Αυτοί οι places ονομάζονται socket places. Για να κάνουμε ξεκάθαρη την σχέση μεταξύ μίας υποκατάστασης μετάβαση και της υποσελίδας της, πρέπει να περιγράψουμε πως οι port places της υποσελίδας σχετίζονται με τα socket places της υποκατάστασης μετάβαση. Αυτό επιτυγχάνεται παρέχοντας μία port assignment. Για την υποσελίδα Sender, σχετίζουμε την port εισόδου ReceiveAck του σχήματος 2.2 με το socket εισόδου ReceiveAck του σχήματος 2.1. Ανάλογα, η port εξόδου TransmitData του σχήματος 2.2 σχετίζεται με το socket εξόδου TransmitData του σχήματος 2.1, και η I/O port Send στο I/O socket Send.

Όταν ένα port place εκχωρείτε σε ένα socket place, οι δύο places γίνονται ταυτόσημοι. Ο port place και ο socket place αποτελούν δύο διαφορετικές αναπαραστάσεις του ίδιου εννοιολογικά place. Πιο συγκεκριμένα, αυτό σημαίνει ότι οι port και οι socket places πάντα έχουν ταυτόσημη σήμανση. Όταν ένα socket εισόδου λαμβάνει ένα token από τα περιβάλλοντα στοιχεία της υποκατάστασης μετάβαση, αυτό το token γίνεται διαθέσιμο και στο port εισόδου της υποσελίδας. Ανάλογα, η υποσελίδα μπορεί να παράγει token σε ένα port εξόδου. Αυτά τα token είναι επίσης διαθέσιμα και στο αναφερόμενο socket εξόδου και ως εκ τούτου μπορούν να χρησιμοποιηθούν από τα περιβάλλοντα στοιχεία της υποκατάστασης μετάβαση.

The communication channel. Ας επικεντρωθούμε τώρα στο κανάλι επικοινωνίας. Το κανάλι επικοινωνίας απαιτείται να είναι ένα αναξιόπιστο διπλής κατεύθυνσης κανάλι επικοινωνίας. Τώρα θα εργαστούμε με ένα από κάτω πάνω τρόπο, μοντελοποιώντας ένα αναξιόπιστο μίας κατεύθυνσης κανάλι και μετά θα συνθέσουμε δύο τέτοια μίας κατεύθυνσης κανάλια, ώστε να σχηματίσουμε ένα διπλής κατεύθυνσης κανάλι.

Το CP-net που μοντελοποιεί το μίας κατεύθυνσης κανάλι φαίνεται στο σχήμα 2.9. Αποτελείται από δύο places Incoming και Outgoing, και μία απλή μετάβαση Transmit. Το port εισόδου του place Incoming μοντελοποιεί την εισερχόμενη κίνηση η οποία πρόκειται να μεταδοθεί από το κανάλι. Το port εξόδου του place Outgoing μοντελοποιεί την εξερχόμενη κίνηση η οποία έχει μεταδοθεί επιτυχώς από το κανάλι. Και οι δύο places έχουν τον τύπο Frame εφόσον το κανάλι θα μεταδίδει πλαίσια (frames). Μία δράση της μετάβασης Transmit μετακινεί ένα πλαίσιο από τον place Incoming. Εξαρτάται από την δέσμευση της Boolean μεταβλητής success (η οποία μπορεί να είναι "αλήθεια" ή "ψέμα"), το πλαίσιο είτε θα τοποθετηθεί στον place Outgoing, όταν πρόκειται για μία επιτυχή μετάδοση, είτε το πλαίσιο δεν τοποθετείται στον place Outgoing, όταν πρόκειται για ένα χαμένο πλαίσιο. Η σταθερά empty δηλώνει το κενό multi-set από token.

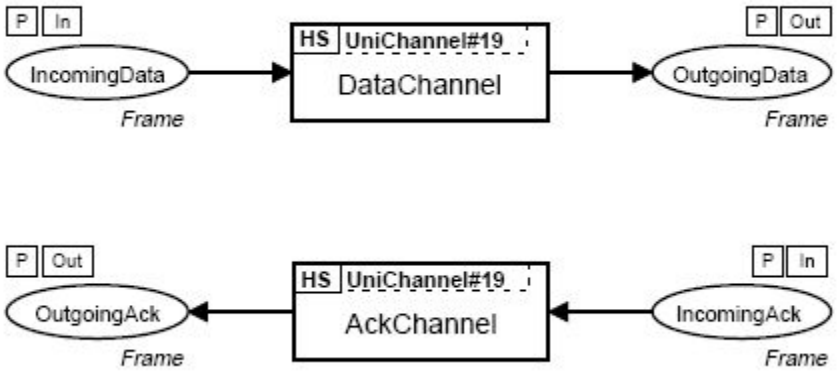


Σχήμα 2.9. Κανάλι μίας κατεύθυνσης

Τα CP-nets μοντελοποιούν το αναξιόπιστο διπλής κατεύθυνσης κανάλι επικοινωνίας που φαίνεται στο σχήμα 2.10. Το CP-net έχει δύο υποκαταστάσεις μετάβασης: DataChannel και AckChannel.

Το DataChannel μοντελοποιεί την κίνηση στην κατεύθυνση αποστολέα προς παραλήπτη. AckChannel μοντελοποιεί την κίνηση στην κατεύθυνση παραλήπτη προς αποστολέα. Και οι δύο υποκαταστάσεις μετάβασης έχουν το CP-net που μοντελοποιεί το κανάλι μίας κατεύθυνσης, ως υποσελίδα. Αυτό σημαίνει ότι έχουμε ξαναχρησιμοποιήσει το Cp-net του μίας κατεύθυνσης κανάλι. Κατά την διάρκεια της εκτέλεσης του CP-net, θα έχουμε δύο ξεχωριστές εκφάνσεις της σελίδας που μοντελοποιεί το κανάλι μίας κατεύθυνσης: μία σχετικά με το κανάλι δεδομένων, και άλλη μία

για το κανάλι βεβαιώσεων. Κάθε μία από αυτές τις εκφάνσεις θα έχει την δική της σήμανση που είναι εντελώς ανεξάρτητη με την σήμανση των άλλων εκφάνσεων της σελίδας (με παρόμοιο τρόπο για να επεξεργαστούμε κλήσεις που έχουν ιδιωτικά αντίγραφα τοπικών μεταβλητών). Η εκχώρηση των ports για τα socket places του σχήματος 2.10 είναι όπως αναμένουμε.



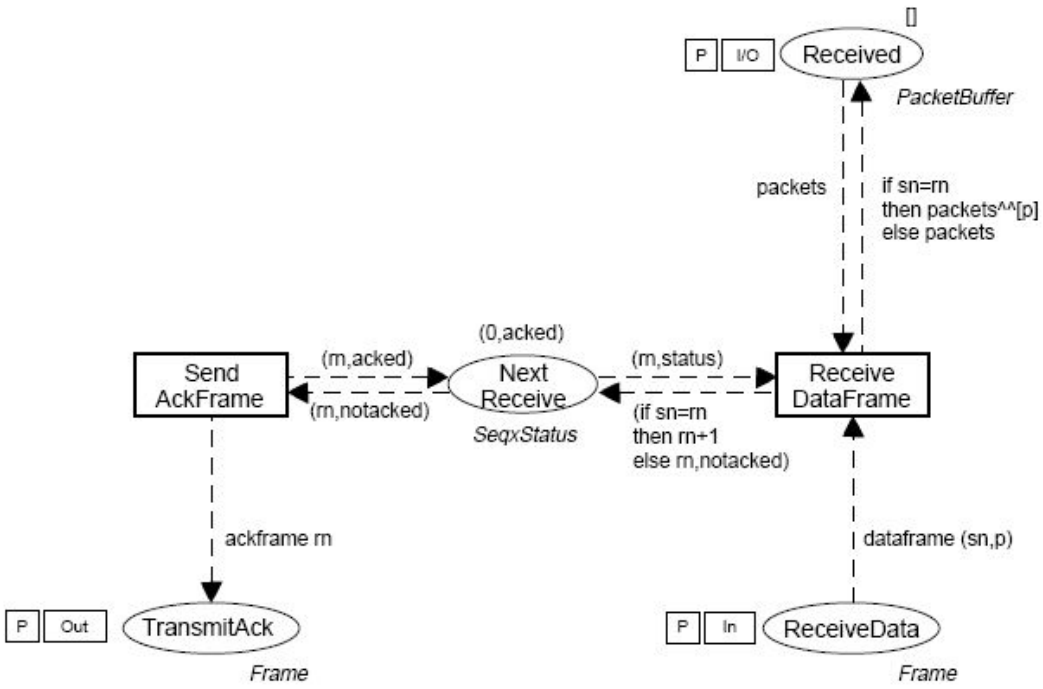
Σχήμα 2.10. Κανάλι δύο κατευθύνσεων

Τα δύο socket εισόδου places των IncomingData και IncomingAck έχουν εκχωρηθεί στο port του place του σχήματος 2.9. Τα δύο socket places εξόδου OutgoingData και OutgoingAck έχουν εκχωρηθεί στο port place Outgoing του σχήματος 2.9.

Το CP-net που μοντελοποιεί το διπλής κατεύθυνσης κανάλι είναι η υποσελίδα της υποκατάστασης μετάβασης Communication Channel στο σχήμα 2.1. Αξίζει να παρατηρήσουμε ότι οι places IncomingData, OutgoingData του σχήματος 2.10 είναι sockets με σεβασμό στη υποκατάσταση μετάβασης DataChannel αφού αποτελούν port places με σεβασμό στην υποκατάσταση μετάβασης CommunicationChannel.

The receiver. Το σχήμα 2.11 απεικονίζει το CPN μοντέλο του παραλήπτη. Η δεξιά μεριά του μοντελοποιεί την παραλαβή πλαισίων δεδομένων ενώ η αριστερή μοντελοποιεί την αποστολή βεβαιώσεων. Ο place NextReceive (στη μέση) είναι παρόμοιος με τον place NextSend στην σελίδα του αποστολέα. Χρησιμοποιείται για να παρακολουθείται η κατάσταση του παραλήπτη. Προσδιορίζει τον αριθμό σειράς του πλαισίου δεδομένων που αναμένεται ως επόμενο, και δείχνει αν έχει σταλεί βεβαίωση

για το τελευταίο πλαίσιο δεδομένων που έχει ληφθεί. Αρχικά, ο παραλήπτης αναμένει το πλαίσιο δεδομένων με τον αριθμό σειράς 0, και το προηγούμενο πλαίσιο δεδομένων έχει βεβαιωθεί (εφόσον, αρχικά, δεν υπάρχει τέτοιο πλαίσιο δεδομένων). Αυτό προσδιορίζεται από την αρχική σήμανση (0, acked) του place NextReceive. Ο place Received μοντελοποιεί το buffer πακέτων δεδομένων στην πλευρά του παραλήπτη. Αρχικά, το buffer είναι άδειο και δείχνεται από την αρχική σήμανση [], που δηλώνει κενή λίστα. Οι port places TransmitAck και ReceiveData είναι παρόμοιοι με τους places με όμοια ονόματα στην πλευρά του αποστολέα, εδώ μοντελοποιούν τα buffers πλαισίων μεταξύ του παραλήπτη και του καναλιού επικοινωνίας.



Σχήμα 2.11. Ο παραλήπτης του stop-and-wait πρωτοκόλλου

Η μετάβαση ReceiveDataFrame μοντελοποιεί την παραλαβή πλαισίων δεδομένων. Όταν παραλαμβάνονται πλαίσια δεδομένων, απομακρύνονται από τον place ReceiveData και ο αριθμός σειράς sn συγκρίνεται με τον αριθμό σειράς rn του πλαισίου δεδομένων που ο παραλήπτης αναμένει ως επόμενο. Αν ο αριθμοί σειράς ταιριάζουν (rn=sn), τότε το πακέτο p προσαρτάται στην λίστα που υπάρχει στον place

Received και ο αριθμός σειράς του πλαισίου δεδομένων που αναμένεται μετά αυξάνεται κατά ένα. Σε κάθε περίπτωση, η κατάσταση του παραλήπτη ανανεώνεται στην κατάσταση notacked.

Η μετάβαση SendAckFrame μοντελοποιεί την αποστολή πλαισίων βεβαιώσεων. Όταν η εσωτερική κατάσταση του παραλήπτη είναι notacked , τότε ένα πλαίσιο βεβαίωσης με αριθμό σειράς rn τοποθετείται στον place TransmitAck, και η εσωτερική κατάσταση του παραλήπτη ανανεώνεται σε acked, δείχνοντας ότι μία βεβαίωση έχει σταλεί.

Fusion places. Τα ιεραρχικά CP-nets επιπρόσθετα προσφέρουν την δυνατότητα μίας έννοιας γνωστής ως fusion places. Αυτό επιτρέπει στον σχεδιαστή να προσδιορίσει ότι ένα σύνολο places μπορούν να θεωρηθούν ταυτόσημοι, π.χ., όλοι αποτελούν ένα απλό εννοιολογικά place, ακόμη και αν αυτοί έχουν σχεδιαστεί ως ένας αριθμός ξεχωριστών places. Όταν ένα token προστίθεται/απομακρύνεται σε ένα place, ένα ταυτόσημο token θα προστεθεί/αφαιρεθεί σε όλους τους places του συνόλου fusion. Από αυτή την περιγραφή, είναι εύκολο να δούμε ότι η σχέση μεταξύ μελών ενός συνόλου fusion (μέσα σε κάποια όρια) είναι όμοια με την σχέση δύο places οι οποίοι έχουν εκχωρηθεί ο ένας στον άλλο με εκχώρηση port.

Όταν όλα τα μέλη ενός fusion συνόλου ανήκουν σε μία απλή σελίδα και αυτή η σελίδα έχει μόνο μία έκφανση, place fusion δεν είναι τίποτα περισσότερο από ένα σχέδιο που βοηθά τον χρήστη να αποφύγει πολλά διασταυρωμένα τόξα. Ωστόσο, τα πράγματα γίνονται πολύ πιο ενδιαφέροντα όταν τα μέλη ενός fusion συνόλου ανήκουν σε διαφορετικές υποσελίδες ή σε σελίδα που έχει πολλές εκφάνσεις . Σε αυτή την περίπτωση, τα σύνολα fusion επιτρέπουν στο χρήστη να προσδιορίζει μία συμπεριφορά, η οποία θα ήταν πολύ δύσκολο να προσδιοριστεί χωρίς σύνολα fusion.

Υπάρχουν τρία διαφορετικά είδη από σύνολα fusion: τα global fusion sets μπορούν να έχουν μέλη από πολλές διαφορετικές σελίδες, ενώ τα page fusion sets και instance fusion sets έχουν μέλη μόνο από μια απλή σελίδα. Η διαφορά μεταξύ των δύο τελευταίων είναι η ακόλουθη. Μία page fusion ενοποιεί όλες τις εκφάνσεις των place που ανήκουν σε αυτή (ανεξάρτητα από την έκφανση της σελίδας στην οποία αυτοί εμφανίζονται), και αυτό σημαίνει ότι το σύνολο fusion έχει μόνο ένα "resulting place" το οποίο "μοιράζονται" όλες οι εκφάνσεις της υπό εξέταση σελίδας. Σε αντίθεση, ένα instance fusion set μόνο προσδιορίζει τις εκφάνσεις των places που ανήκουν στην ίδια έκφανση της σελίδας, και αυτό σημαίνει ότι το σύνολο fusion έχει ένα "resulting place" για κάθε έκφανση σελίδας. Η σημασιολογία ενός global fusion set είναι ανάλογη με αυτή ενός page fusion set – υπό την έννοια ότι υπάρχει μόνο ένας "resulting place" (ο οποίος είναι κοινός για όλες τις εκφάνσεις των

σελίδων που παίρνουν μέρος). Για να λάβουμε τα κέρδη του δομικού σχεδιασμού και ανάλυσης, τα global fusions sets πρέπει να χρησιμοποιούνται με φειδώ.

Στο παράδειγμα με το πρωτόκολλο, έχουμε μόνο τρία επίπεδα στην ιεραρχία των σελίδων. Ωστόσο, στην πράξη, συχνά υπάρχουν ως και δέκα επίπεδα ιεραρχίας. Όπως είδαμε με την σελίδα που μοντελοποιεί το διπλής κατεύθυνσης κανάλι, μία υποσελίδα μπορεί να περιέχει υποκαταστάσεις μετάβασης και αυτές να έχουν τις δικές τους υποσελίδες. Πολύ συχνά, μία σελίδα έχει και μεταβάσεις και υποκαταστάσεις μεταβάσεων, π.χ., μερικές δραστηριότητες περιγράφονται με πλήρη λεπτομέρεια, ενώ άλλες δραστηριότητες περιγράφονται με ένα πιο επιφανειακό τρόπο – αφήνοντας την λεπτομερή περιγραφή σε μία υποσελίδα.

Μπορεί να δειχθεί ότι κάθε ιεραρχικό CP-net έχει ένα ισοδύναμο μη-ιεραρχικό CP-net. Για να λάβουμε το μη-ιεραρχικό net, πρέπει απλά να αντικαταστήσουμε κάθε υποκατάσταση μετάβασης (και τα τόξα που έχει γύρω της) με ένα αντίγραφο της υποσελίδας της, "κολλώντας" κάθε port place σε κάθε socket place στο οποίο έχει εκχωρηθεί.

Πρέπει να σημειωθεί ότι οι υποκαταστάσεις μεταβάσεων ποτέ δεν ενεργοποιούνται και ποτέ δεν δρουν. Οι υποκαταστάσεις μεταβάσεων λειτουργούν ως ένας macro μηχανισμός. Αυτές επιτρέπουν στις υποσελίδες να εισέρχονται εννοιολογικά στην θέση των υποκαταστάσεων μετάβασης. Στο σχήμα 2.1, δεν έχουμε δώσει καμία arc expression για τα τόξα που περιβάλλουν τις υποκαταστάσεις μεταβάσεων. Αυτό είναι περιττό εφόσον οι υποκαταστάσεις μεταβάσεων ποτέ δεν ενεργοποιούνται και ποτέ δεν δρουν. Ωστόσο, μπορεί να φανεί πολύ χρήσιμο δίνοντας μία πρώτη εντύπωση για την λειτουργία της υποσελίδας στον αναγνώστη του μοντέλου.

Είναι σημαντικό να σημειώσουμε ότι οι ιεραρχικές κατασκευές δεν εγγυώνται μία τυπική σχέση στα όρια της ισοδυναμίας ανάμεσα στις arc expressions των τόξων που περιβάλλουν την υποκατάσταση μετάβασης, και της συμπεριφοράς της συγκεκριμένης υποσελίδας. Αυτό σημαίνει ότι η ιεραρχική έννοια των CP-nets προσφέρει δυνατότητα σύντηξης[13] στο συντακτικό επίπεδο παρά στο σχεδιαστικό.

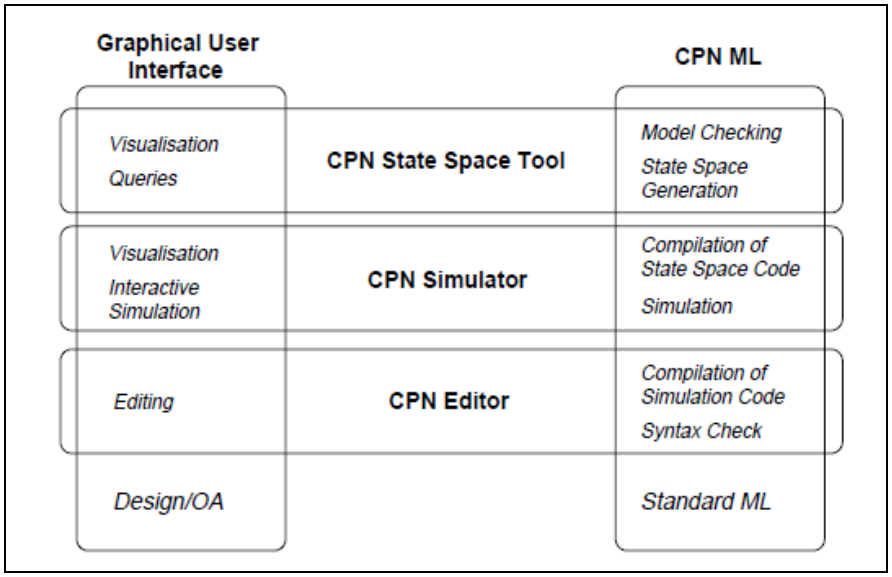
2.4 Επισκόπηση του CPN tools

Σε αυτό το κομμάτι παρουσιάζουμε μία επισκόπηση του CPN tools περιγράφοντας τα κύρια χαρακτηριστικά του και τον γενικό τρόπο εφαρμογής του εργαλείου.

Η γενική αρχιτεκτονική που διέπει το CPN tools παρουσιάζεται στο σχήμα 2.12. Το CPN tools αποτελείται από δύο κύρια μέρη: το *graphical user interface (GUI) και την CPN ML. Μαζί, αυτά τα δύο μέρη δημιουργούν*

τα τρία άκρως συνδεδεμένα εργαλεία που αποτελούν το CPN tools: τον CPN editor, τον CPN simulator, και το CPN state space tool.

Το GUI είναι η γραφική απεικόνιση που αλληλεπιδρά με τον χρήστη. Έχει χτιστεί πάνω στο γενικό γραφικό πακέτο Design/OA. Η CPN ML υλοποιεί μέρη της CPN ML γλώσσας, η οποία είναι η γλώσσα προγραμματισμού που χρησιμοποιείται για δηλώσεις μεταβλητών, δηλώσεις των τύπων, και για περιγραφές του δικτύου (π.χ., εκφράσεις των τόξων, φρουρούς, κλπ) στα CP nets. Τα σχήματα 2.3 και 2.4 της προηγούμενης παραγράφου είναι παραδείγματα δηλώσεων τύπων και μεταβλητών γραμμένα σε CPN ML. Πρόσθετα στην CPN ML γλώσσα, το CPN ML component υλοποιεί τον μηχανισμό προσομοίωσης και τον κεντρικό αλγόριθμο για την δημιουργία και την χρήση των state spaces. Αυτό σημαίνει ότι το CPN ML component είναι υπεύθυνο για τον υπολογισμό τον ενεργών δεσμευτικών στοιχείων στις σημάνσεις που ανέκυψαν στο CPN μοντέλο.



Σχήμα 2.12. Αρχιτεκτονική επισκόπηση του CPN tools

Η CPN ML χτίστηκε πάνω στην Standard ML (SML) γλώσσα[14]. Η CPN ML γλώσσα είναι η SML γλώσσα εκτεταμένη με κάποια συντακτική ευκολία ώστε να απλοποιήσει την δήλωση των τύπων, των μεταβλητών, κλπ. Το γεγονός ότι η γλώσσα περιγραφής βασίζεται στην SML έχει διάφορα πλεονεκτήματα. Πρώτον, η εκφραστικότητα της SML έχει κληρονομηθεί από το CPN tools. Δεύτερον, η SML είναι strongly typed, επιτρέποντας πολλά λάθη στην μοντελοποίηση να γίνονται γνωστά νωρίς. Ένα τρίτο πλεονέκτημα είναι ότι ο πολυμορφισμός, η υπερφόρτωση, και ο

ορισμός των infix τελεστών στην SML επιτρέπει της περιγραφές του δικτύου να γράφονται φυσική, μαθηματική σύνταξη. Τέλος, η SML είναι καλά τεκμηριωμένη, δοκιμασμένη και διατηρημένη. Η επιλογή της SML αποδείχθηκε ότι ήταν μία από τις πιο επιτυχημένες σχεδιαστικές αποφάσεις για τον CPN tools. Αποδείχθηκε πλεονέκτημα στο να χτιστεί το CPN tools πάνω στο Design/OA και την SML, τα οποία είναι και τα δύο διαθέσιμα σε διαφορετικές πλατφόρμες.

Για τις περιγραφές ενός CP net (π.χ., εκφράσεις τόξων, φρουρούς, κλπ) μόνο το καθαρά συναρτησιακό κομμάτι της SML γλώσσας χρησιμοποιείται. Αυτό είναι συνεπές με την λειτουργική σημασιολογία των CP nets, όπως περιγράφονται σε προηγούμενη παράγραφο. Δεν θα κάνει αίσθηση αν, για παράδειγμα, η αξιολόγηση ενός φρουρού μίας μετάβασης μπορεί να έχει επίδραση στην σήμανση μερικών places.

Το συνηθισμένο στυλ δουλειάς, όταν εργαζόμαστε με αυτό, είναι να αρχίζουμε να χρησιμοποιούμε τον editor για την κατασκευή ενός πρώτου CPN μοντέλου του συστήματος που εξετάζουμε. Το ιεραρχικό CP net της προηγούμενης παραγράφου (σχήμα 2.1) μοντελοποιεί το stop-and-wait πρωτόκολλο, είναι ένα παράδειγμα ενός CPN διαγράμματος κατασκευασμένο χρησιμοποιώντας τον CPN editor. Τα σχήματα της προηγούμενης παραγράφου είναι αποτυπώσεις οθόνης από τον editor. Ο χρήστης δουλεύει απευθείας στην γραφική απεικόνιση ενός CPN μοντέλου. Για να είναι σε θέση να μπορεί να προσομοιώσει το CPN μοντέλο, θα πρέπει να αποτελεί ένα συντακτικά και στους types σωστό CPN net. Όταν τα αναφερθέντα συντακτικά και types λάθη (αν υπάρχουν) έχουν διορθωθεί, ο χρήστης μπορεί να μεταβεί στο CPN simulator. Η μετάβαση στον CPN simulator ενεργοποιεί τον κώδικα που είναι απαραίτητος για την προσομοίωση του CPN μοντέλου. Ο συντακτικός έλεγχος όπως και το compilation διαχειρίζονται από CPN ML κομμάτι του editor.

Μόλις ο κώδικας για την προσομοίωση έχει μεταφραστεί, ο χρήστης είναι έτοιμος να εξερευνήσει την συμπεριφορά του συστήματος με τις έννοιες της προσομοίωσης. Αυτές οι πρώτες προσομοιώσεις έχουν τα χαρακτηριστικά ενός απλού βήματος κατά το οποίο το token game παρατηρείται με μεγάλη λεπτομέρεια, και ο χρήστης επιλέγει τα επόμενα δεσμευτικά στοιχεία που θα είναι ενεργά μετά. Κατά την διάρκεια τέτοιων προσομοιώσεων, οι σημάνσεις των places δείχνονται απευθείας στο CPN διάγραμμα όμοια με το σχήμα 2.8. Οι προσομοιώσεις συνήθως αποκαλύπτουν κενά και/ή λάθη στο CPN μοντέλο τα οποία πρέπει να λυθούν. Ως εκ τούτου, η πρώτη φάση φυσιολογικά αποτελείται από ένα αριθμό επαναλήψεων εναλλάσσοντας πίσω μπρος στον editor και στον simulator, τελειοποιώντας και βελτιώνοντας βαθμιαία το CPN μοντέλο. Η προσομοίωση/εκτέλεση ενός CPN μοντέλου οδηγείτε από τον μηχανισμό προσομοίωσης της CPN ML. Χρησιμοποιεί το μέρος προσομοίωσης του GUI για την οπτικοποίηση του token game.

Συνήθως, η επόμενη φάση είναι να κάνουμε κάποιες πιο μεγάλες προσομοιώσεις για να παραχθεί μία πιο επεξεργασμένη επικύρωση του σχεδίου και/ή του CPN μοντέλου. Σε τέτοιες μεγάλες προσομοιώσεις, η παρακολούθηση του token game συνήθως της συμπεριφοράς του συστήματος. Παραδείγματα αυτού περιλαμβάνουν επιχειρηματικούς πίνακες, πίνακες συχνότητας μηνυμάτων (MSC), και διάφορα είδη γραφικών, ειδικευμένα για το domain της εφαρμογής.

Σε περίπτωση που ο λόγος δημιουργίας του CPN μοντέλου είναι να κάνουμε ανάλυση επίδοσης, ο CPN προσομοιωτής είναι εντεταλμένος να συλλέγει δεδομένα κατά τη διάρκεια μεγάλων προσομοιώσεων. Τα συλλεγμένα δεδομένα μπορεί να χρησιμοποιηθούν αργότερα σε μία post-processing φάση κατά την οποία λαμβάνονται τα σχήματα κλειδιά για την επίδοση του συστήματος.

Μία πιθανή επόμενη φάση είναι να εφαρμόσουμε το state space εργαλείο για να επαληθεύσουμε και να επικυρώσουμε τη συναρτησιακή ορθότητα του συστήματος. Για να μπορούμε να εφαρμόσουμε το CPN state space εργαλείο, ο χρήστης μεταπηδά από τον CPN προσομοιωτή στο CPN state space εργαλείο. Η εναλλαγή είναι παρόμοια με αυτή από τον editor στον προσομοιωτή, και αποτελείται από μία μετάφραση των απαραίτητων εσωτερικών δομών δεδομένων για να δουλέψει το state space εργαλείο. Αυτή η μετάφραση διαχειρίζεται από το κομμάτι προσομοίωσης της CPN ML. Η πρώτη φάση εφαρμογής του state space εργαλείου τυπικά αποτελείται από το να κάνει το CPN μοντέλο ανιχνεύσιμο για την state space ανάλυση. Το επόμενο βήμα τότε είναι να ενεργοποιηθεί το state space (ή κομμάτι του). Η ενεργοποίηση διαχειρίζεται από το CPN ML κομμάτι του state space εργαλείου. Ο χρήστης τώρα μπορεί να κάνει ερωτήσεις σχετικά με την συμπεριφορά του συστήματος, χρησιμοποιώντας την διαθέσιμη γλώσσα για της ερωτήσεις. Οι ερωτήσεις μπορούν να γραφούν σαν τύποι σε μία state και action προσανατολισμένη παραλλαγή της προσωρινής λογικής CTL ή με τη χρήση ενός αριθμού διαθέσιμων συναρτήσεων έρευνας. Οι απαντήσεις σε αυτές τις ερωτήσεις υπολογίζονται από τον model checker στο CPN ML κομμάτι του state space εργαλείου. Χρησιμοποιεί το GUI για να οπτικοποιήσει πληροφορίες σχετικά με το state space και για να απεικονίσει τα αποτελέσματα των ερωτήσεων. Η μέθοδος state space παρουσιάζεται αναλυτικότερα σε επόμενες σελίδες.

Το GUI και η CPN ML τρέχουν ως δύο ξεχωριστές διεργασίες που είναι σε επικοινωνία. Η επικοινωνία ανάμεσα στις δύο διεργασίες χρειάζεται για να μπορεί να μεταμορφωθεί η γραφική αναπαράσταση από το GUI σε περιεκτικές/εσωτερικές αναπαραστάσεις που χρησιμοποιούνται στην CPM ML και ανάποδα. Έχοντας το GUI και την CPN ML ως δύο ξεχωριστές διεργασίες έχει πολλά πλεονεκτήματα. Το κύριο πλεονέκτημα προέρχεται από το γεγονός ότι ηCPN ML χειρίζεται τον συντακτικό έλεγχο, την

ενεργοποίηση κώδικα, την προσομοίωση, και την state space ενεργοποίηση τα οποία είναι τα μέρη που απαιτούν τον όγκο των υπολογιστικών πόρων σε έννοιες μνήμης και ταχύτητας. Το GUI έχει κάπως μέτριες απαιτήσεις με σεβασμό στους υπολογιστικούς πόρους. Εφόσον τα δύο μέρη είναι ξεχωριστά το GUI μπορεί να τρέχει σε ένα μικρό σταθμό εργασίας, και η CPN ML μπορεί να τρέχει στο παρασκήνιο ενός πιο ισχυρού σταθμού εργασίας. Στα περισσότερα βιομηχανικά έργα στα οποία έχει εφαρμοστεί το CPN tools, ένας σταθμός εργασίας εξοπλισμένος με 64 MB εσωτερικής μνήμης και με ταχύτητα ισοδύναμη με ένα Sun Ultra Sparc είναι αρκετό για τρέχει το CPN ML κομμάτι. Ωστόσο, το state space εργαλείο συχνά απαιτεί επιπρόσθετους υπολογιστικούς πόρους με την έννοια της μνήμης.

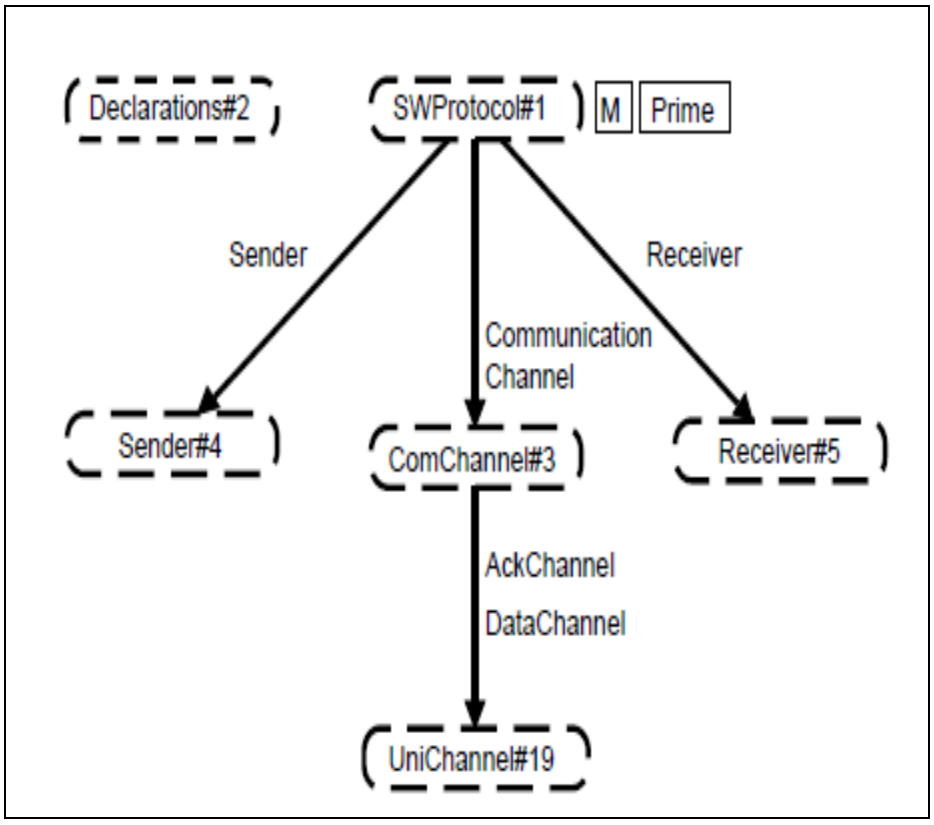
2.5 Κατασκευή των CPN μοντέλων

Ο CPN editor υποστηρίζει την κατασκευή, την διόρθωση, και τον συντακτικό έλεγχο CPN διαγραμμάτων. Σε αυτή την παράγραφο περιγράφουμε την βασική λειτουργία του editor.

2.5.1 Δουλεύοντας με ιεραρχικά CPN μοντέλα

Σε μία τυπική βιομηχανική εφαρμογή, ένα CPN διάγραμμα αποτελείται από 10 έως 100 σελίδες με ποικίλη πολυπλοκότητα. Ο σχεδιαστής πρέπει να βρει ένα κατάλληλο τρόπο να διαιρέσει το μοντέλο σε σελίδες. Επιπλέον, ο σχεδιαστής πρέπει να βρει και μία κατάλληλη ισορροπία ανάμεσα σε δηλώσεις(declarations), επιγραφές του δικτύου(net inscriptions, και τη δομή του δικτύου(net structure) [π.χ., places, transitions, και arcs].

Η σελίδα ιεραρχίας. Ο editor παρέχει μία γενική άποψη των σελίδων ενός CPN μοντέλου και της εσωτερικής διασύνδεσης αυτών δημιουργώντας και διατηρώντας μία σελίδα ονομαζόμενη και σελίδα ιεραρχίας, η οποία είναι παρόμοια με πολλούς browsers που βρίσκουμε σε πολλά συμβατικά προγραμματιστικά περιβάλλοντα. Η σελίδα ιεραρχίας για το stop-and-wait πρωτόκολλο παρουσιάζεται στο σχήμα 2.13.



Σχήμα 2.13. Σελίδα ιεραρχίας για το πρωτόκολλο stop-and-wait

Η σελίδα ιεραρχίας έχει ένα κόμβο σελίδας για κάθε σελίδα. Ένα βέλος ανάμεσα σε δύο κόμβους σελίδων υποδηλώνει ότι η κατάληξη του βέλους είναι υποσελίδα της αρχής, π.χ. η σελίδα πηγή περιέχει μία υποκατάσταση μετάβασης που χρησιμοποιεί την κατάληξη του βέλους ως υποσελίδα. Κάθε κόμβος σελίδας έχει επιγραφή το όνομα της σελίδας και τον αριθμό της. Αναλογικά, κάθε βέλος έχει ένα κείμενο που προσδιορίζει το όνομα της υποκατάστασης μετάβασης. Σαν παράδειγμα, κόμβος σελίδας SWProtocol αντιπροσωπεύει την σελίδα που φαίνεται στο σχήμα 2.1. Ο κόμβος σελίδας έχει τρία εξερχόμενα βέλη κατά αντιστοιχία στις τρείς υποκαταστάσεις μεταβάσεων Sender, Receiver, και Communication Channel. Αυτά τα τρία βέλη οδηγούν στους κόμβους σελίδας που αντιπροσωπεύουν τις σελίδες Sender (σχήμα 2.2), Receiver (σχήμα 2.11), και ComChannel (σχήμα 2.10), αντίστοιχα. Ο κόμβος σελίδας αντιπροσωπεύει την σελίδα ComChannel έχει ένα εξερχόμενο βέλος που οδηγεί στον κόμβο σελίδας που αντιπροσωπεύει την σελίδα UniChannel (σχήμα 2.9). Αυτό το βέλος αντιστοιχεί στις δύο υποκαταστάσεις μεταβάσεων DataChannel και AckChannel.

Όταν ο χρήστης κάνει διπλό κλικ σε ένα κόμβο σελίδας, η συγκεκριμένη σελίδα ανοίγει και φαίνεται, και ο χρήστης μπορεί να αρχίσει να επεμβαίνει στην σελίδα. Ένα διπλό κλικ σε μία υποκατάσταση μετάβασης όμοια ανοίγει και τοποθετεί την συγκεκριμένη υποσελίδα μπροστά.

Η σελίδα SWprotocol έχει μία μικρή Prime ετικέτα δίπλα της. Αυτή δείχνει ότι η SWProtocol είναι πρωτεύουσα σελίδα, π.χ., μία σελίδα στο πιο περιεκτικό επίπεδο. Ένα CPN μοντέλο έχει μία έκφανση σελίδας για κάθε πρωτεύουσα σελίδα. Για κάθε υποκατάσταση μετάβασης σε μία πρωτεύουσα σελίδα έχουμε μία έκφανση σελίδας της εν λόγω υποσελίδας. Αν αυτές οι εκφάνσεις σελίδας έχουν υποκαταστάσεις μετάβασης, παίρνουμε εκφάνσεις σελίδας και για αυτές, και ου το καθεξής, μέχρι να φτάσουμε στο τέλος της σελίδας ιεραρχίας (η οποία απαιτείται να είναι μη κυκλική). Το CPN μοντέλο του stop-and-wait πρωτοκόλλου έχει έξι εκφάνσεις σελίδας. Όλες οι σελίδες έχουν μία έκφανση, εκτός από το UniChannel, το οποίο έχει δύο εκφάνσεις, μία για κάθε υποκατάσταση μετάβασης στη σελίδα ComChannel.

Το CPN μοντέλο έχει επίσης μία σελίδα που ονομάζεται Declarations(δηλώσεις). Αυτή η σελίδα περιέχει τον global declaration node (κόμβο καθολικών δηλώσεων). Ο κόμβος καθολικών δηλώσεων περιέχει όλους τις δηλώσεις των types, σταθερών, συναρτήσεων, και μεταβλητών, που χρησιμοποιούνται στην περιγραφή του δικτύου ενός CPN μοντέλου. Για το stop-and-wait πρωτόκολλο, ο κόμβος καθολικών δηλώσεων περιέχει τις δηλώσεις των τύπων και των μεταβλητών που φαίνονται στο σχήμα 2.3 και σχήμα 2.4.

Μετακίνηση σε υποσελίδα. Ο editor κάνει εύκολο να προσθέσεις νέες υποσελίδες, ή να αναδιατάξεις την σελίδα ιεραρχίας σε άλλη μορφή. Όταν μία σελίδα έχει πάρα πολλούς places και μεταβάσεις, μπορούμε να μετακινήσουμε μερικούς από αυτούς σε μία καινούργια υποσελίδα. Αυτό γίνεται με μία απλή λειτουργία του editor. Ο χρήστης επιλέγει τους κόμβους που θέλει να μετακινηθούν και καλεί την Move to Subpage εντολή. Τότε ο editor :

- Ελέγχει την νομιμότητα της επιλογής (πρέπει να σχηματίζεται ένα υποδίκτυο οριοθετημένο από μεταβάσεις)
- Δημιουργεί την νέα σελίδα
- Μετακινεί το υποδίκτυο στη νέα σελίδα
- Δημιουργεί τους port places αντιγράφοντας αυτούς τους places οι οποίοι είναι παρακείμενοι του επιλεγμένου υποδικτύου
- Υπολογίζει τους τύπους των port (In, Out, ή I/O)
- Κατασκευάζει τα απαραίτητα τόξα ανάμεσα στα port των κόμβων και το επιλεγμένο υποδίκτυο
- Παρακινεί τον χρήστη να δημιουργήσει μία νέα μετάβαση η οποία θα είναι η υποκατάσταση μετάβασης για την νέα υποσελίδα
- Σχεδιάζει τα τόξα που περιβάλλουν την νέα μετάβαση
- Δημιουργεί μία περιγραφή ιεραρχίας για την νέα μετάβαση
- Ανανεώνει την σελίδα ιεραρχίας

Όπως μπορούμε να δούμε, πολλά από διάφορες πολύπλοκες διαδικασίες εμπλέκονται στην εντολή Move to Subpage. Ωστόσο, σχεδόν όλες διενεργούνται αυτόματα από τον editor. Ο χρήστης μόνο επιλέγει το υποδίκτυο, καλεί την εντολή, και δημιουργείται η νέα υποκατάσταση μετάβασης. Η υπόλοιπη δουλειά γίνεται από τον editor. Αυτό, φυσικά, είναι δυνατό μόνο επειδή ο editor αναγνωρίζει ένα CPN διάγραμμα ως ένα ιεραρχικό CP-net, και όχι μόνο ως ένα μαθηματικό γράφο ή ένα σύνολο από ασύνδετα αντικείμενα. Χωρίς αυτή την ιδιότητα, ο χρήστης θα έπρεπε να κάνει όλη την δουλειά με την χρήση των συνηθισμένων λειτουργιών (αντιγραφή, μετακίνηση, και δημιουργία των απαραιτήτων αντικειμένων). Αυτό είναι δυνατό – αλλά είναι πολύ χρονοβόρο και πολύ εύκολο να γίνουν λάθη.

Δημιουργώντας μία υποκατάσταση μετάβασης. Υπάρχει επίσης μία εντολή του editor η οποία μετατρέπει μία υπάρχουσα μετάβαση σε μία υποκατάσταση μετάβασης – σχετίζοντας τη με μία υπάρχουσα σελίδα. Πάλι, η περισσότερη δουλειά γίνεται από τον editor. Ο χρήστης επιλέγει την μετάβαση και καλεί την εντολή. Έπειτα ο editor :

- Κάνει την σελίδα ιεραρχίας ενεργή
- Παρακινεί τον χρήστη να επιλέξει την επιθυμητή υποσελίδα; Όταν το ποντίκι μετακινείται πάνω από τον κόμβο σελίδας αναβοσβήνει, εκτός αν είναι παράνομο (επειδή η επιλογή μπορεί να κάνει την σελίδα ιεραρχίας κυκλική)
- Περιμένει μέχρι ένας κόμβος σελίδας από αυτούς που αναβοσβήνουν να επιλεγεί
- Προσπαθεί να εξάγει την εκχώρηση port με την χρήση ενός συνόλου κανόνων οι οποίοι αναζητούν στα ονόματα των port/socket και τους τύπους των port/socket (In, Out, ή I/O)
- Δημιουργεί την περιγραφή ιεραρχίας με το όνομα και τον αριθμό της υποσελίδας και αυτά τα μέρη της εκχώρησης port τα οποία μπορούν να βρεθούν αυτόματα
- Ανανεώνει την σελίδα ιεραρχίας

Αντικατάσταση από υποσελίδα. Για το τέλος, υπάρχει μία εντολή του editor που αντικαθιστά μία υποκατάσταση μετάβασης με ολόκληρο το περιεχόμενο της υποσελίδας του. Πάλι, αυτή η λειτουργία περιέχει πολλούς και πολύπλοκους υπολογισμούς και χειρισμούς, αλλά, και πάλι, όλοι αυτοί γίνονται από τον editor. Ο χρήστης απλά επιλέγει την υποκατάσταση μετάβασης, καλεί την εντολή και χρησιμοποιεί ένα απλό κουτί διαλόγου για να προσδιορίσει τις λεπτομέρειες της λειτουργίας (π.χ., αν θα πρέπει να διαγραφεί η υποσελίδα εφόσον άλλη υποκατάσταση μετάβασης δεν την χρησιμοποιεί).

Αυτές οι τρείς εντολές ιεραρχίας που περιγράφονται παραπάνω μπορούν να καλούνται με οποιαδήποτε σειρά. Ένας χρήστης με μία από πάνω-κάτω προσέγγιση θα άρχιζε τυπικά δημιουργώντας μία σελίδα όπου κάθε μετάβαση θα αντιπροσώπευε μία πιο πολύπλοκη δραστηριότητα. Τότε μία υποσελίδα δημιουργείται για κάθε δραστηριότητα. Ο πιο εύκολος τρόπος για να το κάνεις αυτό, είναι να χρησιμοποιήσεις την Move to Subpage εντολή. Έπειτα η υποσελίδα παίρνει αυτόματα τα σωστά port places, π.χ., το σωστό interface στην υποκατάσταση μετάβασης. Καθώς οι νέες υποσελίδες τροποποιούνται, προσθέτοντας places και μεταβάσεις, οι υποσελίδες μπορεί να γίνουν τόσο λεπτομερείς που επιπλέον επίπεδα υποσελίδων θα πρέπει να προστεθούν. Αυτό γίνεται ακριβώς με τον ίδιο τρόπο που δημιουργήθηκε και το πρώτο επίπεδο.

2.5.2 Ελαστικά Γραφικά

Για να είμαστε σε θέση να δημιουργήσουμε ευανάγνωστα CPN μοντέλα, CPN tools υποστηρίζει μία μεγάλη γκάμα από γραφικές σχηματικές παραμέτρους όπως σχήματα, σκιάσεις, όρια, κλπ. Το υποκείμενο τυπικό CPN μοντέλο (σε CPN ML του σχήματος 2.12) είναι ανεπηρέαστο από την γραφική εμφάνιση. Για παράδειγμα, ένα αντικείμενο που δημιουργήθηκε ως place παραμένει place για πάντα, ανεξάρτητα των γραφικών αλλαγών. Τα ελαστικά γραφικά στο CPN tools συνοδεύονται με μία λογική προεπιλογών, π.χ., το προεπιλεγμένο σχήμα για ένα place είναι η έλλειψη.

Το σχήμα 2.2 απεικονίζει την χρήση των ελαστικών γραφικών. Η κύρια ροή του αποστολέα είναι Accept ή TimeOut, Sending, SendDataFrame, και Waiting. Αυτό υποδεικνύεται από το παχύ όριο αυτών των places και των μεταβάσεων, και από τα παχιά τόξα ανάμεσα τους. Οι places που μοντελοποιούν buffers και μεταβλητές στον αποστολέα έχουν ένα λεπτό όριο. Όλα τα τόξα μοντελοποιούν άδεια σε αυτές τις μεταβλητές και τα buffers που έχουν λεπτό και διακεκομμένο όριο.

Ο χρήστης μπορεί να αλλάξει τις προεπιλεγμένες ρυθμίσεις για την γραφική εμφάνιση των αντικειμένων χρησιμοποιώντας system defaults και diagram defaults. Το πεδίο diagram defaults είναι ένα απλό διάγραμμα. Αν ο χρήστης θέσει το diagram defaults για ένα συγκεκριμένο τύπο αντικειμένου, π.χ., places, τότε όλοι οι places δημιουργούνται σε αυτό το διάγραμμα θα δημιουργηθούν χρησιμοποιώντας το diagram default. Το system defaults δουλεύει δια μέσω διαγραμμάτων και προσδιορίζει το αρχικό diagram default για νέα CPN διαγράμματα. Το system defaults κάνουν εύκολο για τον χρήστη να έχει συγκεκριμένο τρόπο με τον οποίο τα CPN διαγράμματα δημιουργούνται. Το diagram defaults κάνει απλό για ένα χρήστη να τροποποιήσει ένα CPN διάγραμμα που έχει δημιουργηθεί από άλλο χρήστη, και ακόμη να επιβεβαιώσει ότι η γραφική παρουσίαση του

τροποποιημένου και των προσθέτων αντικειμένων είναι συνεπή με τα υπόλοιπα, π.χ., το αυθεντικό κομμάτι του CPN διαγράμματος.

2.5.3 Συντακτικός έλεγχος και έλεγχος τύπων

Ο editor είναι συντακτικά κατευθυνόμενος με την εκτέλεση ενός αριθμού built-in περιορισμών. Αυτό αποτρέπει τους χρήστες από το να κάνουν συγκεκριμένα λάθη κατά τη διάρκεια κατασκευής ενός μοντέλου. Ως παράδειγμα, είναι αδύνατο να σχεδιάσουμε ένα τόξο ανάμεσα σε δύο μεταβάσεις ή ανάμεσα σε δύο places. Ωστόσο, είναι αδύνατο να βρεις όλα τα λάθη αποδοτικά με αυτό τον τρόπο. Ως εκ τούτου, υπάρχει ένας συντακτικός έλεγχος και ένας έλεγχος τύπων, οι οποίοι εμπλέκονται όταν ο χρήστης θέλει να βεβαιωθεί ότι το μοντέλο που δημιουργήθηκε αποτελεί ένα νόμιμο CP-net.

Ένα παράδειγμα συντακτικού λάθους μπορεί να είναι η έλλειψη ενός τύπου ενός place. Ένα παράδειγμα λάθους τύπου μπορεί να είναι μία arc expression που έχει τύπο διαφορετικό από τον τύπο του place που συνδέεται το τόξο. Διάφορα λάθη μπορούν να αναφερθούν ταυτόχρονα στη διάρκεια ενός συντακτικού ελέγχου.

Η αναφορά των λαθών βασίζεται στην δυνατότητα του hypertext. Δείχνουμε αυτό εξηγώντας πως αναφέρονται τα συντακτικά λάθη στην σελίδα Sender (φαίνεται στο σχήμα 2.2). Στην σελίδα ιεραρχίας (σχήμα 2.13) ένα κουτί λάθους θα εμφανιστεί. Το κουτί λάθους θα έχει ένα κείμενο που θα δηλώνει ότι υπάρχει λάθος στην σελίδα Sender, και εκεί θα υπάρχει hypertext link που θα συνδέεται με την σελίδα. Ακολουθώντας αυτό το link θα ανοίξουμε την σελίδα Sender και εμφανίζεται ένα άλλο κουτί λάθους με την περιγραφή του προβλήματος στη μορφή μήνυμα λάθους. Ως μέρος του μηνύματος ένα άλλο hypertext link παρέχεται, το οποίο δείχνει στο αντικείμενο, π.χ., place ή μετάβαση, όπου το λάθος εντοπίστηκε.

Σε πολλές περιπτώσεις, το να διορθώσεις τα λάθη περιλαμβάνει μόνο τοπικές αλλαγές. Για λόγους αποδοτικότητας, ο συντακτικός έλεγχος και ο έλεγχος τύπων είναι οριακοί. Αυτό σημαίνει ότι μόνο το τροποποιημένο κομμάτι του μοντέλου ελέγχεται ξανά – όχι ολόκληρο το μοντέλο. Για παράδειγμα, υποθέτουμε ότι και οι πέντε διαδικασίες του CPN μοντέλου που απεικονίζεται στο σχήμα 2.13 έχουν ελεγχθεί συντακτικά. Όταν ένα συντακτικό λάθος αναφορικά, λέει, η μετάβαση Accept στη σελίδα Sender έχει επισκευαστεί , μόνο αυτή η μετάβαση και τα περιβάλλοντα τόξα και places έχουν ελεγχθεί ξανά, όχι όλες οι πέντε διαδικασίες.

2.5.4 Εναλλαγή σε μορφή κειμένου

Τα CPN διαγράμματα των ιεραρχικών Cp-nets μπορούν να εισαχθούν ή να εξαχθούν σε μορφή κειμένου. Το σχήμα 2.14 δίνει ένα παράδειγμα της μορφής κειμένου που υποστηρίζει το CPN tools. Δείχνει πως η μετάβαση Transmit στη σελίδα UniChannel στο σχήμα 2.9 αναπαρίσταται σε μορφή κειμένου. Η μετάβαση Transmit αναπαρίσταται μα την χρήση trans block. Οι ετικέτες και τα attributes μέσα στο μπλοκ δίνουν πληροφορίες σχετικά με το όνομα της μετάβασης και της γραφικής της εμφάνισης όπως lineattributes, fillattributes, και textattributes. Ως εκ τούτου, η μορφή κειμένου δίνει πληροφορίες σχετικά με το ίδιο το CP-net, π.χ., μεταβάσεις, places, τόξα, και περιγραφές δικτύου, τόσο καλά όσο και η γραφική αναπαράσταση του Cp-net.

Η μορφή κειμένου βασίζεται στην SGML[15] (Standard Generalised Markup Language) πάνω στην οποία έχει βασιστεί ένας αριθμός markup γλωσσών όπως και η HTML (Hyper Text Markup Language). Αυτός είναι ο λόγος γιατί η μορφή κειμένου του σχήματος 2.14 μοιάζει παρόμοιο με HTML.

```
<trans id=id15>
  <text>Transmit</text>
  <lineattr type=Solid thick=2 colour=black>
  <fillattr pattern=None colour=black>
  <posattr x=-14.01390 y=1135.58337>
  <box h=254.00000 w=779.63892>
  <textattr font=Helvetica size=18 just=Centered
            colour=black bold=FALSE italic=FALSE
            underlin=FALSE outline=FALSE
            shadow=FALSE
            condense=FALSE extend=FALSE
            scbar=TRUE>
</trans>
```

Σχήμα 2.14. Παράδειγμα εναλλαγής σε μορφή κειμένου

2.6 Προσομοίωση CPN μοντέλων

Ο CPN προσομοιωτής υποστηρίζει την εκτέλεση των CPN μοντέλων. Παρέχει δύο βασικά είδη προσομοίωσης κατάλληλα για διάφορους σκοπούς όπως εξηγείται παρακάτω.

2.6.1 Εσωτερική προσομοίωση

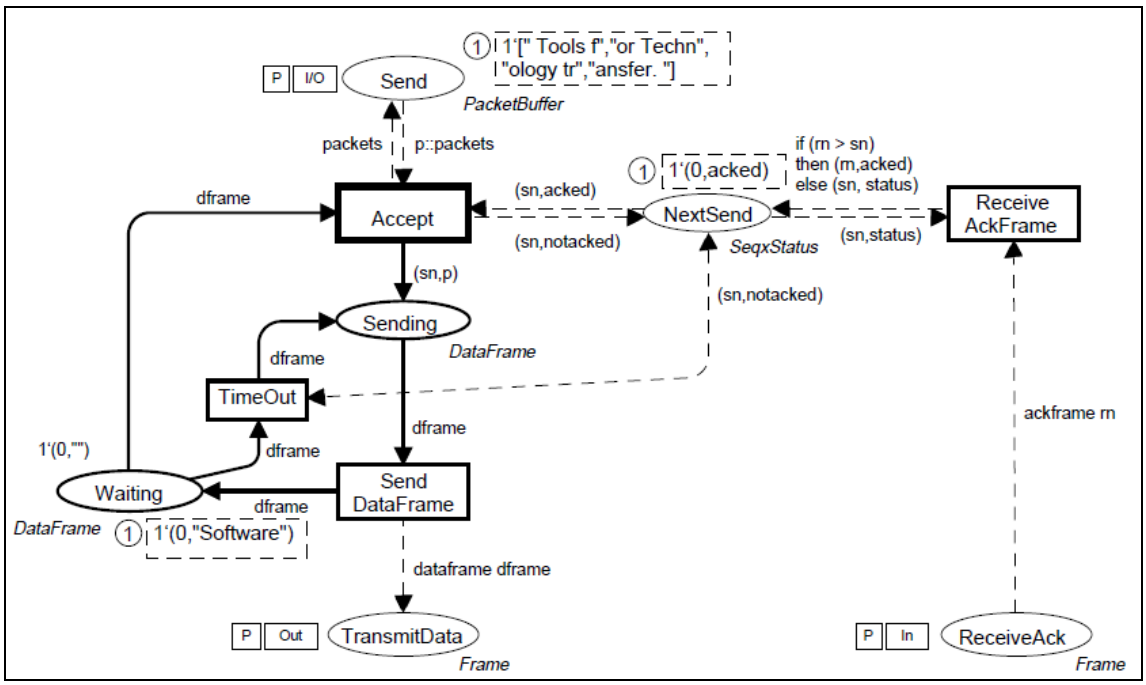
Καθώς τα ξεχωριστά κομμάτια του CP-net κατασκευάζονται, αυτά ερευνώνται και επαληθεύονται με τους κανόνες του CPN προσομοιωτή, όπως ακριβώς και ένας προγραμματιστής δοκιμάζει και επαληθεύει τα καινούργια μέρη ενός προγράμματος. Στις πρώτες φάσεις μίας διαδικασίας μοντελοποίησης, ο χρήστης συνήθως θέλει να κάνει μία λεπτομερή έρευνα της συμπεριφοράς του κάθε transition ξεχωριστά ή μικρών κομματιών του μοντέλου.

Για αυτό το σκοπό, ο προσομοιωτής προσφέρει ένα interactive mode. Σε αυτό ο χρήστης έχει τον πλήρη έλεγχο, θέτει σημεία διάσπασης, διαλέγει μεταξύ ενεργών δεσμευτικών στοιχείων, να αλλάζει την σήμανση ενός place, και να μελετά το token game σε βάθος. Αυτό το πλαίσιο δουλειάς μοιάζει με την απλού βήματος επαλήθευση σε μία συνηθισμένη γλώσσα προγραμματισμού. Ο σκοπός είναι βρεθεί πότε τα ξεχωριστά μέρη του δικτύου δουλεύουν όπως αναμένεται. Οι εσωτερικές προσομοιώσεις είναι, εκ φύσεως, πολύ αργές – κανένα ανθρώπινο ον δεν μπορεί να ερευνήσει παραπάνω από μερικές σημάνσεις ανά λεπτό.

Όπως υποδείχθηκε παραπάνω, ο σχεδιαστής είναι ικανός να επηρεάσει όλες τις λεπτομέρειες των σημάνσεων που έχουν επιτευχθεί. Ο χρήστης μπορεί να δει ένα σύνολο από ενεργά transitions και να διαλέξει τα δεσμευτικά στοιχεία με τα οποία αυτοί θα δράσουν. Στο σχήμα 2.15, φαίνεται ένα screenshot μίας εσωτερικής προσομοίωσης του stop-and-wait πρωτοκόλλου. Δείχνει τη σήμανση της σελίδας του αποστολέα.

Η παρούσα σήμανση παρουσιάζει: τον αριθμό των token σε ένα place που περιέχεται σε ένα μικρό κύκλο δίπλα στο place. Η απουσία του κύκλου υποδεικνύει ότι στο place δεν υπάρχουν token. Οι τιμές δεδομένων των token δείχνονται στο κουτί με ένα διακεκομμένο όριο που βρίσκεται δίπλα στον κύκλο. Αν το επιθυμεί, ο χρήστης μπορεί να κρύψει το κουτί, π.χ. αν οι τιμές δεδομένων είναι περιττές ή πολύ μεγάλες για να τυπωθούν. Ο transition Accept απεικονίζεται με ενισχυμένο πλαίσιο για να δείξει ότι είναι ενεργός με την παρούσα σήμανση. Συνήθως, ενεργά transitions και οι σημάνσεις των places απεικονίζονται χρησιμοποιώντας διαφορετικά χρώματα. Αυτό βελτιώνει αισθητά την αναγνωσιμότητα σε σύγκριση με το ασπρόμαυρο σχήμα 2.15.

Οι εσωτερικές προσομοιώσεις υποστηρίζονται σε διαφορετικές παραλλαγές. Είναι για παράδειγμα δυνατό να ορίσουμε ότι μόνο το token game συγκεκριμένων κομματιών του CPN μοντέλου πρέπει να παρατηρούνται, και είναι δυνατό να ελέγχουμε την ποσότητα της γραφικής ανατροφοδότησης σε κάθε βήμα. Οι εσωτερικές προσομοιώσεις δεν απαιτούν το μοντέλο να έχει τελειώσει, π.χ., ο χρήστης μπορεί να ξεκινήσει να ερευνά την συμπεριφορά κομματιών του μοντέλου και απευθείας να εφαρμόζει τα εσωτερικά δρώμενα στις εξερχόμενες σχεδιαστικές διεργασίες. Συχνά, ένα μοντέλο επαναπροσδιορίζεται βαθμιαία – από μία γενική περιγραφή σε μία πιο λεπτομερή.



Σχήμα 2.15. Screenshot από μία εσωτερική προσομοίωση

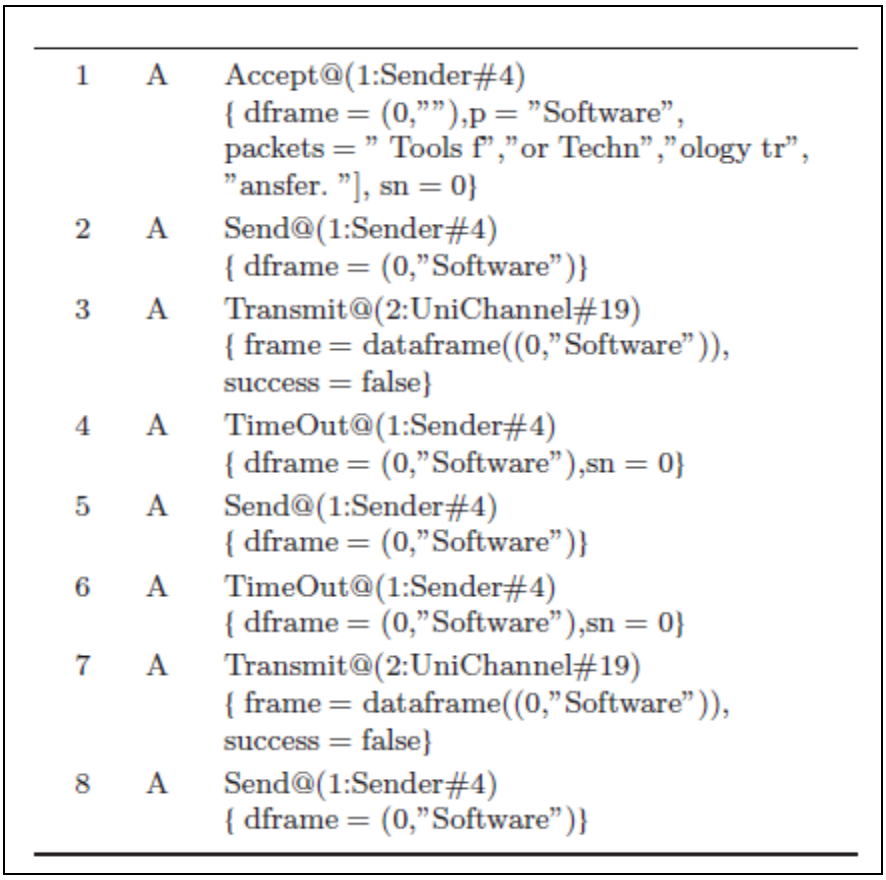
2.6.2 Αυτόματη προσομοίωση

Αργότερα σε μια διαδικασία μοντελοποίησης, το ενδιαφέρον μεταφέρεται από τα ξεχωριστά transition στη γενική συμπεριφορά του πλήρους μοντέλου. Το automatic mode του προσομοιωτή είναι κατάλληλο εδώ. Σε αυτή την περίπτωση, ο προσομοιωτής κάνει τυχαίες επιλογές (με την έννοια της μηχανής παραγωγής τυχαίων αριθμών) μεταξύ ενεργών δεσμευτικών στοιχείων. Με αυτό τον τρόπο, είναι δυνατό να πετύχουμε πολύ πιο γρήγορες προσομοιώσεις. Αυτό επιτυγχάνεται ακόμη και για μεγάλα μοντέλα, λόγω του ότι οι κανόνες ενεργοποίησης και δραστηριότητας των Petri nets είναι τοπικοί. Αυτό σημαίνει ότι, όταν ένα transition έχει δράσει, μόνο η ενεργοποίηση των γειτονικών transition πρέπει να επαναπροσδιοριστεί, π.χ., ο αριθμός των βημάτων ανά δευτερόλεπτο είναι ανεξάρτητος του μεγέθους του μοντέλου. Μια ολοκληρωμένη αυτόματη προσομοίωση εκτελείται με την ταχύτητα μερικών χιλιάδων βημάτων το δευτερόλεπτο (εξαρτώμενη από την φύση του CPN μοντέλου και από την δύναμη του υπολογιστή στον οποίο τρέχει ο CPN προσομοιωτής). Ο χρήστης συμμετέχει στον έλεγχο μίας αυτόματης προσομοίωσης με την έννοια των παραμέτρων τερματισμού. Οι παράμετροι τερματισμού κάνουν εφικτό, για παράδειγμα, να δοθεί ένα

ανώτατο όριο στον αριθμό των βημάτων που θα εκτελεστούν από την προσομοίωση.

Πριν και μετά από μία αυτόματη προσομοίωση, η παρούσα σήμανση και τα ενεργά transition απεικονίζονται όπως περιγράφηκε στο interactive mode. Ωστόσο, το token game δεν απεικονίζεται κατά τη διάρκεια μίας αυτόματης προσομοίωσης. Φυσικά, αυτό συνήθως μας παρέχει λιγότερες πληροφορίες από το επιθυμητό. Μία άμεση δυνατότητα να αποκτήσουμε πληροφορίες σχετικά με το "τι έγινε;" είναι να χρησιμοποιήσουμε την *αναφορά προσομοίωσης*. Είναι ένα αρχείο κειμένου που περιέχει όλες τις πληροφορίες σχετικά με όλες τις δεσμεύσεις των transition που έδρασαν. Το σχήμα 2.16 δείχνει μία αναφορά προσομοίωσης των 8 πρώτων βημάτων μίας αυτόματης προσομοίωσης του stop-and-wait πρωτοκόλλου.

Το Α ακολουθεί τον αριθμό του βήματος που δείχνει ότι η δέσμευση του transition εκτελέστηκε στο automatic mode. Η πληροφορία μετά το σύμβολο @ προσδιορίζει την έκφανση σελίδας, π.χ., το κομμάτι του CP-net στο οποίο ο "δράστης" transition ανήκει. Τέλος, για κάθε βήμα, δείχνεται η δέσμευση με την οποία έδρασε το κάθε transition. Για παράδειγμα, το βήμα 3 αντιστοιχεί σε μία δράση του transition Transmit στην δεύτερη έκφανση της σελίδας UniChannel με μία δέσμευση που αντιστοιχεί στο πρώτο πλαίσιο δεδομένων που χάνεται.



```
1  A  Accept@(1:Sender#4)
      { dframe = (0,""),p = "Software",
      packets = " Tools f","or Techn","ology tr",
      "ansfer. "], sn = 0}
2  A  Send@(1:Sender#4)
      { dframe = (0,"Software")}
3  A  Transmit@(2:UniChannel#19)
      { frame = dataframe((0,"Software")),
      success = false}
4  A  TimeOut@(1:Sender#4)
      { dframe = (0,"Software"),sn = 0}
5  A  Send@(1:Sender#4)
      { dframe = (0,"Software")}
6  A  TimeOut@(1:Sender#4)
      { dframe = (0,"Software"),sn = 0}
7  A  Transmit@(2:UniChannel#19)
      { frame = dataframe((0,"Software")),
      success = false}
8  A  Send@(1:Sender#4)
      { dframe = (0,"Software")}
```

Σχήμα 2.16. Μερικό παράδειγμα αναφοράς προσομοίωσης

2.6.3 Code segments

Είναι δυνατό να επισυνάψουμε ένα κομμάτι CPN ML κώδικα σε ξεχωριστά transition χρησιμοποιώντας τα όπως αποκαλούνται code segments. Όταν ένα transition δρα, το συγκεκριμένο code segment εκτελείται. Για παράδειγμα, μπορεί να γράφει ή να διαβάζει αρχεία κειμένου, να ανανεώνει γραφήματα ή ακόμη και να υπολογίζει τιμές που πρέπει να δεσμευτούν σε κάποιες μεταβλητές του transition. Με αυτό τον τρόπο, τα code segments παρέχουν ένα πολύ βολικό interface μεταξύ του CPN μοντέλου και του περιβάλλον του, π.χ., το σύστημα φακέλων. Όταν περιγράψουμε την οπτικοποίηση (visualisation) στον τομέα 6 και την ανάλυση επίδοσης (performance analysis0 στον τομέα 7, θα δώσουμε ένα αριθμό παραδειγμάτων χρήσης των code segments.

2.6.4 Αναβάθμιση με τον editor

Συχνά, τα αποτελέσματα μίας προσομοίωσης μας κάνει να θέλουμε να τροποποιήσουμε το μοντέλο. Μερικές από αυτές τις τροποποιήσεις μπορούν να γίνουν άμεσα: είναι δυνατό να κάνουμε μικρές αλλαγές καθώς παραμένουμε στον προσομοιωτή, π.χ., να αλλάξουμε μία arc expression. Άλλες τροποποιήσεις απαιτούν πιο περίπλοκες επανεξετάσεις/επαναπροσδιορισμούς από τον κώδικα του προσομοιωτή, οι οποίοι υποστηρίζονται στον editor, π.χ., είναι αδύνατο να προσθέσεις ή να αλλάξεις ένα type στον προσομοιωτή. Στο CPN tools, ο editor και ο προσομοιωτής στενά συνδεδεμένοι. Έτσι, είναι εύκολο και γρήγορο να μετακινούμαστε από τον προσομοιωτή πίσω στον editor, να επιδιορθώνεις ένα πρόβλημα, να ξαναμπαίνεις στον προσομοιωτή, και να συνεχίζεις την προσομοίωση.

2.7 Γραφική παρακολούθηση CPN μοντέλων

Η γραφική ανατροφοδότηση από τις εσωτερικές προσομοιώσεις και τις αναφορές προσομοίωσης από τις αυτόματες προσομοιώσεις είναι σε πολλές περιπτώσεις σε πολύ λεπτομερές επίπεδο απόκτησης και μετάφρασης των αποτελεσμάτων των προσομοιώσεων. Για να ανταπεξέλθουμε σε αυτό το πρόβλημα, το CPN tools περιλαμβάνει βιβλιοθήκες για γραφική παρακολούθηση (visualisation). Η γραφική παρακολούθηση κατευθύνεται από τον CPN προσομοιωτή και κάνει δυνατό να δημιουργούνται και να ανανεώνονται διάφορα είδη γραφημάτων κατά την διάρκεια της προσομοίωσης.

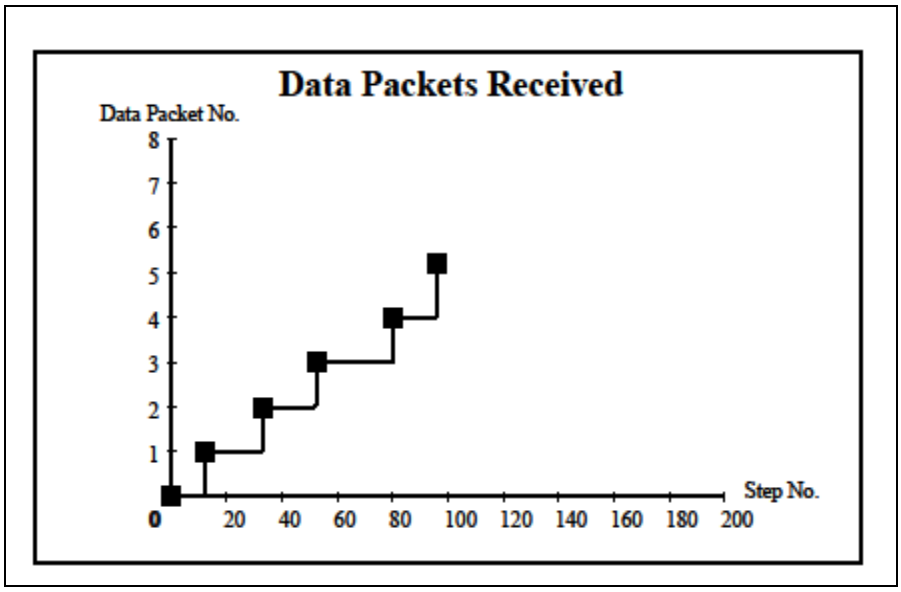
Υπάρχουν διάφοροι λόγοι για τους οποίους η γραφική παρακολούθηση της συμπεριφοράς του συστήματος κατά την διάρκεια μακρών προσομοιώσεων είναι επικερδής. Μπορούμε να αποκτήσουμε καλή γενική άποψη μίας μακράς προσομοίωσης. Εξάγοντας τις δραστηριότητες κλειδιά και αναπαριστώντας αυτές με ένα γραφικό τρόπο τα αποτελέσματα της προσομοίωσης μπορούν εύκολα να μεταφραστούν. Μπορούμε να δούμε πότε το CPN μοντέλο συμπεριφέρεται όπως αναμένεται. Αν δεν το κάνει, μπορούμε να δούμε πότε εμφανίζονται ανωμαλίες. Έπειτα μπορούμε να χρησιμοποιήσουμε την εσωτερική προσομοίωση ή την αναφορά προσομοίωσης για να κάνουμε μία πιο προσεκτική έρευνα αυτών των καταστάσεων. Επιπλέον, η γραφική παρακολούθηση έχει την δυνατότητα να κρύψει τα CP-nets και να παρέχει οπτική ανατροφοδότηση από προσομοιώσεις που είναι προσδιορισμένες για το domain της εφαρμογής. Αυτό κάνει δυνατό να μπορούμε να συζητήσουμε σχεδιαστικές ιδέες και αποτελέσματα με συναδέλφους και άλλους οι οποίοι έχουν γνώση του domain της εφαρμογής, αλλά δεν έχουν σχέση με τα CP-nets.

Σε αυτό το κομμάτι θα επικεντρωθούμε σε τρία είδη γραφικής ανατροφοδότησης για προσομοιώσεις: γραφήματα γραμμών και στηλών, γραφήματα σειράς μηνυμάτων, και ειδικά γραφικά εφαρμογής.

Η γραφική παρακολούθηση κατευθύνεται από code segments που σχετίζονται με transitions όπως είδαμε στον τομέα 5. Συνήθως ο χρήστης πρέπει να 2-5 γραμμές κώδικα CPN ML για κάθε code segment. Είναι σημαντικό να σημειώσουμε ότι η συμπεριφορά του CPN μοντέλου σε καμία περίπτωση δεν επηρεάζεται από την πρόσθετη γραφική παρακολούθηση, και μπορούν να προστεθούν ανεξάρτητα από άλλες ανατροφοδοτήσεις.

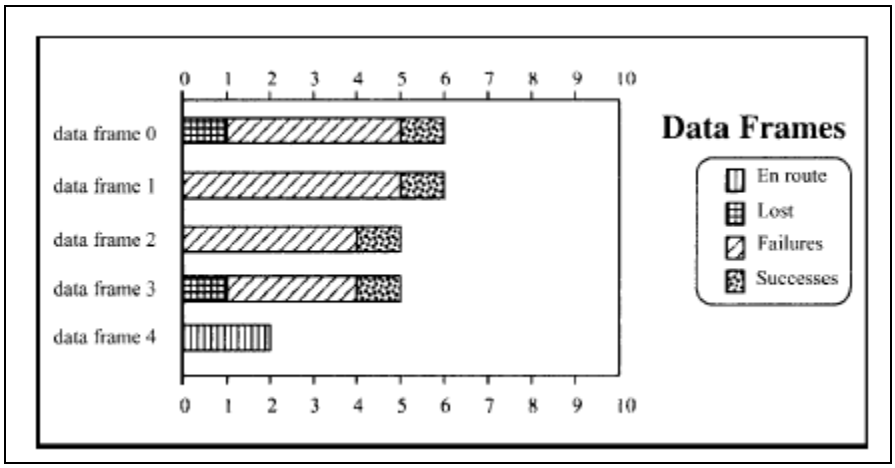
2.7.1 Γραφήματα γραμμών και στηλών

Για το stop-and-wait πρωτόκολλο μπορούμε να χρησιμοποιήσουμε τα τρία γραφήματα που φαίνονται στα σχήματα 2.17, 2.18 και 2.19. το γράφημα του σχήματος 2.17 είναι ένα γράφημα γραμμών που δείχνει πόσο γρήγορα το καθένα ξεχωριστό πακέτο δεδομένων παραλαμβάνεται επιτυχώς (σαν συνάρτηση του αριθμού βημάτων). Από το γράφημα γραμμών, μπορούμε να δούμε ότι το πρώτο πακέτο παραλήφθηκε μετά από περίπου 10 βήματα, το δεύτερο μετά από περίπου 30 βήματα, κλπ. Το γράφημα αυτό ανανεώνεται κάθε φορά που ένα πακέτο παραλαμβάνεται επιτυχώς. Αυτό επιτυγχάνεται με λίγες γραμμές κώδικα στο code segment του transition ReceiveAckFrame.



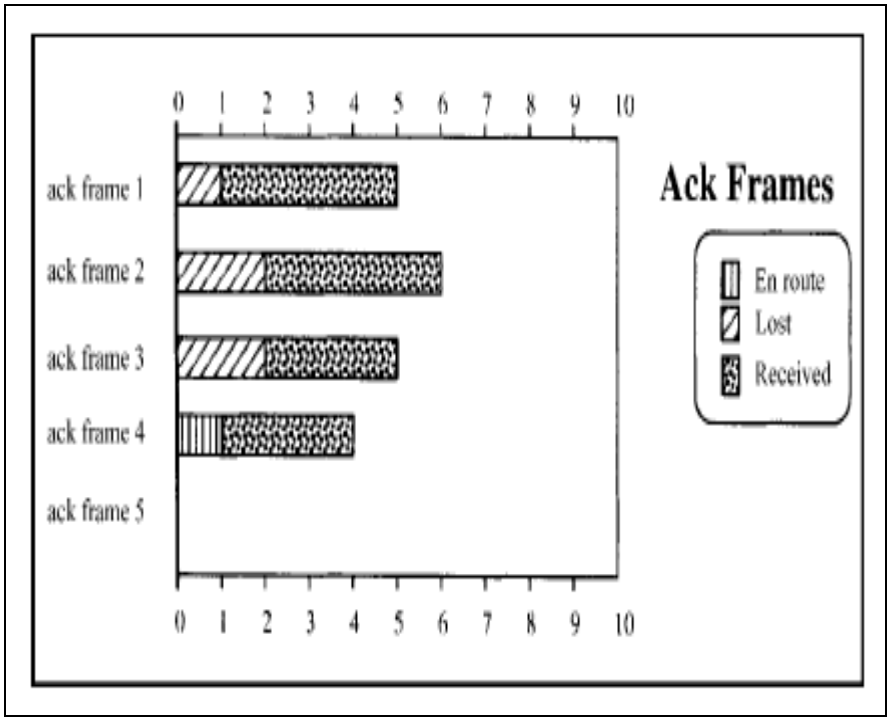
Σχήμα 2.17. Γράφημα γραμμών για την παραλαβή πακέτων δεδομένων

Το γράφημα στο σχήμα 2.18 είναι ένα γράφημα στηλών. Μας δείχνει πόσες φορές κάθε πλαίσιο δεδομένων στάλθηκε και με ποιό αποτέλεσμα. Από το γράφημα στηλών μπορούμε να δούμε ότι το πλαίσιο δεδομένων με τον αριθμό σειράς 0 έχει σταλεί 6 φορές, μία από αυτές χάθηκε, τέσσερις παραλήφθηκε ως αποτυχία (π.χ., εκτός σειράς) και την τελευταία παραλήφθηκε επιτυχώς. Ανάλογα, μπορούμε να δούμε ότι το πλαίσιο δεδομένων με τον αριθμό σειράς 1 στάλθηκε 6 φορές, ενώ τα πλαίσια δεδομένων 2 και 3 στάλθηκαν 5 φορές το καθένα. Τέλος, βλέπουμε ότι το πλαίσιο δεδομένων 4 έχει σταλεί 2 φορές, και τα 2 έχουν δρομολογηθεί (π.χ., το ένα στο place TransmitData και το άλλο στο ReceiveData).



Σχήμα 2.18. Γράφημα στηλών για την μετάδοση πλαισίων δεδομένων

Το γράφημα στο σχήμα 2.19 είναι παρόμοιο με το γράφημα στο σχήμα 2.18, εκτός από το ότι δείχνει την πρόοδο των πλαισίων επιβεβαίωσης. Τα δύο γραφήματα στηλών ανανεώνονται περιοδικά, με εσωτερικές παραμέτρους που ορίζονται από τον χρήστη, π.χ., κάθε 50 βήματα. Τα τρία γραφήματα μας δίνουν πολλές πολύτιμες πληροφορίες σχετικά με την συμπεριφορά του πρωτοκόλλου. Ως παράδειγμα, μπορείς άμεσα να δεις ότι οι αποτυχίες (π.χ., overtaking) συχνά προκαλούν περισσότερες αναμεταδόσεις παρά χαμένα πακέτα. Είναι επίσης εύκολο να δούμε ότι το να μεταδώσουμε επιτυχώς και τα 5 πακέτα χρειαζόμαστε παραπάνω από 90 βήματα, ενώ με τέλεια κανάλια επικοινωνίας (και χωρίς overtaking) είναι δυνατό να γίνει με 35 βήματα.



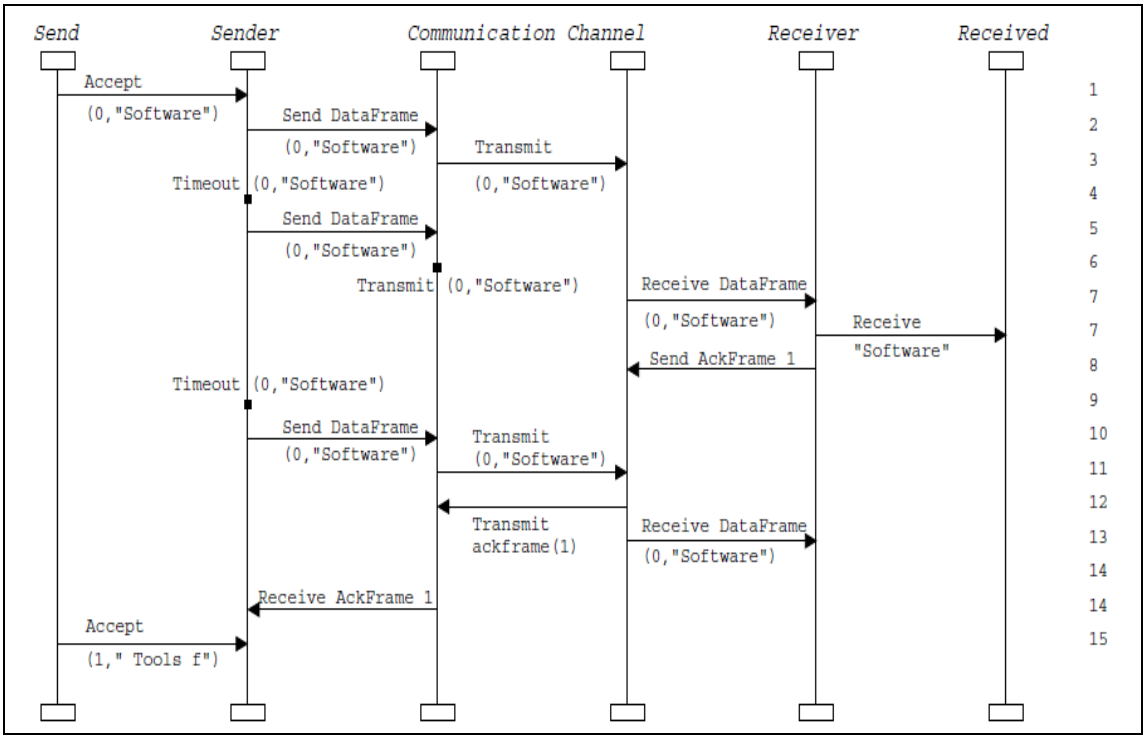
Σχήμα 2.19. Γράφημα στηλών για την μετάδοση πλαισίων επιβεβαίωσης

2.7.2 Γραφήματα σειράς μηνυμάτων

Τα γραφήματα σειράς μηνυμάτων (MSCs)[16] – γνωστά επίσης ως εντοπισμός γεγονότων, διαγράμματα ροής μηνυμάτων, ή διαγράμματα χρονικής σειράς – είναι μία ευρέως διαδεδομένη τυποποίηση για να προσδιορίζει παραδείγματα είτε φυσιολογικής είτε όχι εκτέλεσης του συστήματος που σχεδιάστηκε. Στο CPN tools, MSCs χρησιμοποιούνται για να "πιάσουν" ένα απλό τρέξιμο μίας προσομοίωσης, παρέχοντας μία γραφική απεικόνιση της εκτέλεσης του συστήματος. Οι ακριβείς πληροφορίες που περιέχονται στο MSC ορίζονται από τον χρήστη. Για το stop-and-wait πρωτόκολλο ένα MSC μπορεί να είναι όπως στο σχήμα 2.20. Οι κάθετες γραμμές συνήθως χρησιμοποιούνται στα MSCs για να αναπαραστήσουν διαδικασίες ή μέρη ενός συστήματος. Μικρά τετράγωνα πάνω στις κάθετες γραμμές και οριζόντια βέλη μεταξύ των κάθετων γραμμών συνήθως χρησιμοποιούνται για να περιγράψουν δράσεις ή γεγονότα ενός συστήματος.

Μία δράση του transition Accept δείχνεται με το οριζόντιο βέλος ανάμεσα στην πρώτη και δεύτερη κάθετη γραμμή. Το βέλος έχει ως ετικέτα το πακέτο δεδομένων που έγινε δεκτό για μετάδοση και τον αριθμό σειράς

που έχει το πακέτο. Ανάλογα, η δράση του transition SendDataFrame δείχνεται από ένα οριζόντιο βέλος ανάμεσα στην δεύτερη και τρίτη κάθετη γραμμή. Μία επιτυχής δράση του transition Transmit με ένα πλαίσιο δεδομένων υποδεικνύεται από ένα οριζόντιο βέλος μεταξύ της τρίτης και τέταρτης κάθετης γραμμής. Ωστόσο, αν το πλαίσιο δεδομένων χαθεί, παίρνουμε μόνο ένα μικρό τετράγωνο στην τρίτη γραμμή. Όμοια, μία δράση του transition TimeOut έχει επίσης ένα μικρό τετράγωνο στην δεύτερη γραμμή.



Σχήμα 2.20. MSC για το stop-and-wait πρωτόκολλο

Μία δράση του transition ReceiveDataFrame υποδεικνύεται με ένα οριζόντιο βέλος από την τέταρτη στην πέμπτη κάθετη γραμμή. Επιπλέον, αν το πλαίσιο δεδομένων παραλήφθηκε το οριζόντιο βέλος που αναμενόταν δημιουργείτε ανάμεσα στην πέμπτη και έκτη κάθετη γραμμή. Μία δράση του transition SendAckFrame υποδεικνύεται από ένα βέλος μεταξύ τέταρτης και πέμπτης κάθετης γραμμής. Το βέλος έχει ως ετικέτα τον αριθμό σειράς που αντιστοιχεί στο πλαίσιο επιβεβαίωσης. Οι δράσεις του Transmit με πλαίσιο επιβεβαίωσης υποδεικνύονται με παρόμοιο τρόπο με τις δράσεις του Transmit με πλαίσιο δεδομένων (όμως τα βέλη τώρα έχουν φορά από δεξιά προς τα αριστερά, ενώ τα τετράγωνα κουτιά τοποθετούνται στην

τέταρτη κάθετη γραμμή). Δράσεις του ReceiveAckFrame υποδεικνύονται από ένα βέλος από την τρίτη προς την δεύτερη κάθετη γραμμή και με ετικέτα που αντιστοιχεί στον αριθμό σειράς της επιβεβαίωσης. Οι αριθμοί δεξιά εξωτερικά του MSC αντιστοιχούν του βήματος της προσομοίωσης. Στο σχήμα 2.20 έχουμε ένα βέλος (ή ένα τετράγωνο) για κάθε βήμα της προσομοίωσης. Αυτό σημαίνει ότι το MSC περιέχει όλες τις πληροφορίες της αναφοράς προσομοίωσης. Ωστόσο, είναι προτιμότερο να καταγράφουμε μόνο κάποιες δραστηριότητες κλειδιά, π.χ., την μετάδοση πλαισίων δεδομένων και πλαισίων επιβεβαίωσης.

Για την δημιουργία των MSCs, code segments έχουν προσαρτηθεί στους transitions του CPN μοντέλου. Το σχήμα 2.21 δείχνει το code segment του transition Accept. MSC-graphics είναι μία αναφορά στο γράφημα σειράς μηνυμάτων, ενώ το mkst_col'DataFrame είναι μία προκαθορισμένη συνάρτηση που παρέχει μία αναπαράσταση συμβολοσειράς της τιμής της CPN ML (sn,p).Η συνάρτηση MSC.Message είναι μία από τις συναρτήσεις που παρέχονται από την MSC βιβλιοθήκη για την δημιουργία ενός οριζόντιου βέλους μεταξύ δύο κάθετων γραμμών στο MSC σύμφωνα με την εγγραφή που δόθηκε ως όρισμα. Οι περιγραφές **sender** και **receiver** χρησιμοποιούνται για να προσδιορίσουν την πηγή και τον προορισμό του οριζόντιου βέλους που θα δημιουργηθεί. Το βέλος παίρνει την ετικέτα που έχει το όνομα του transition (Accept) και το δεσμευτικό στοιχείο που τον έκανε να δράσει. Για τους άλλους transitions τα code segments είναι παρόμοια.

```
input (sn,p) ;
action
 MSC.Message (!MSCGraphics)
    {annotation = makestring (step ()),
     label = "Accept"^NEWLINE^
             (mkst_col´DataFrame (sn,p))^NEWLINE,
     sender = "Send",
     receiver = "Sender"
     } ;
```

Σχήμα 2.21. Code segment για τον transition Accept

2.7.3 Ειδικά γραφήματα εφαρμογής

Τα γραφήματα γραμμών, στηλών, και σειράς μηνυμάτων που παρουσιάστηκαν στους προηγούμενους τομείς χαρακτηρίζονται από το

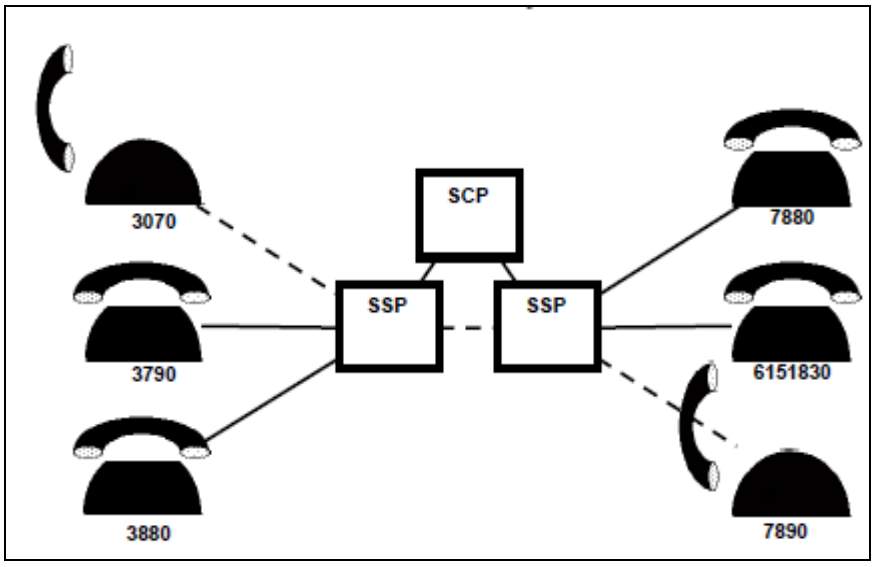
να είναι τυποποιημένοι τρόποι δημιουργίας γραφικής ανατροφοδότησης. Είναι όμως συχνά προτιμότερο να δημιουργούμε γραφικές ανατροφοδοτήσεις που βασίζονται σε έννοιες και αντικείμενα από το domain της συγκεκριμένης εφαρμογής. Η Mimic/CPN[17] βιβλιοθήκη υποστηρίζει τέτοια ειδικά γραφήματα εφαρμογών. Επιπρόσθετα, η βιβλιοθήκη κάνει δυνατό να είναι διαδραστική η γραφική ανατροφοδότηση και, με αυτό τον τρόπο, να έχουμε έναν υψηλού επιπέδου τρόπο ελέγχου και να παρέχει εισόδους για την προσομοίωση. Το stop-and-wait πρωτόκολλο είναι πραγματικά πολύ απλό να τεκμηριώσει τις δυνατότητες των ειδικών γραφημάτων εφαρμογής. Αντί αυτού όμως, θα δείξουμε πως τα CP-nets και τα ειδικά γραφήματα εφαρμογών εφαρμόζονται σε ένα έργο στη Deutsche Telekom για το σχεδιασμό υπηρεσιών σε ευφυή δίκτυα.

Τα ευφυή δίκτυα (Intelligent Networks) (IN) επεκτείνουν τα συμβατικά δίκτυα με την ικανότητα να συλλέγουν και να υπολογίζουν πληροφορίες επιπρόσθετα στις συμβατικές συναρτήσεις μετάδοσης και switching. Η ικανότητα να συλλέγουν και να υπολογίζουν πληροφορίες αναφέρεται ως ευφυΐα του δικτύου. Παράδειγμα IN υπηρεσίας είναι τα freephone, και οι υπηρεσίες card calling.

Το σχήμα 2.22 παρουσιάζει τα βασικά μέρη της αρχιτεκτονικής ενός ευφυούς δικτύου. Η βασική ιδέα είναι να διαχωρίσουμε τις κλασσικές συναρτήσεις μετάδοσης και switching από την συναρτησιακή λογική της IN υπηρεσία. Το service switching point (SSP) αντιπροσωπεύει φυσικές τοπικές τηλεφωνικές συναλλαγές ενώ τα service control points (SCP) αντιπροσωπεύουν κόμβους του δικτύου που περιέχουν την λογική της IN υπηρεσίας. Από αυτούς τους κόμβους οι τοπικές συναλλαγές (SSPs) μπορούν να ελέγχονται από απόσταση με το ένα σύστημα που ονομάζεται signaling system network (SS.7). Μία τοπική συναλλαγή περιέχει την call control function (CCF) η οποία αντιπροσωπεύει τις συναρτήσεις που περιέχονται σε μία τοπική συναλλαγή και δεν είναι έτοιμες για χειρισμό IN κλήσεων, και κάνει δυνατό να επικοινωνεί με την υπηρεσία IN λογικής που περιέχεται στο SCP. Το SCP αποτελείται από δύο κομμάτια: την service control function (SCF) η οποία περιέχει υπηρεσία IN λογικής και την service data function (SDF) που περιέχει την βάση δεδομένων για την IN.

Το κέρδος από τον διαχωρισμό των υπηρεσιών IN λογικής από τις βασικές συναρτήσεις μετάδοσης και switching είναι να μπορούμε να εισάγουμε νέες IN υπηρεσίες χωρίς να πρέπει να τροποποιήσουμε τις τοπικές συναλλαγές (SSPs). Ο στόχος του έργου στην Deutsche Telekom ήταν να διερευνηθεί πως οι βασικές διεργασίες κλήσης στους SSPs θα έπρεπε να εξοπλιστούν για να το επιτρέψουν αυτό. Για να το διερευνήσουν αυτό, ένα CPN μοντέλο της IN αρχιτεκτονικής κατασκευάστηκε. Το CPN μοντέλο αποτελούταν από περίπου 40 σελίδες. Περιλάμβανε μία αναπαράσταση των χρηστών του δικτύου, την υπηρεσία IN λογικής, και τις βασικές διεργασίες κλήσης των SSPs.

Για να παρέχονται είσοδοι στις προσομοιώσεις, το CPN μοντέλο εξοπλίστηκε με το ειδικό γράφημα εφαρμογής που δείχνεται στο σχήμα 2.23. Το ειδικό γράφημα εφαρμογής αναπαριστά ένα μικρό IN που αποτελείται από ένα service control point (SCP) που συνδέεται σε δύο service switching points (SSPs), που το καθένα έχει τρία τηλέφωνα συνδεδεμένα πάνω του. Έχοντας αρχίσει την προσομοίωση, ο χρήστης μπορεί να διαλέξει γραφικά αντικείμενα. Εξαρτώμενα από την κατάσταση του συστήματος, διαφορετικά είδη εσωτερικών δράσεων είναι δυνατά. Αν ένα τηλέφωνο είναι on-hook, είναι δυνατό να επιλέξει το τηλέφωνο το οποίο θα έχει ως αποτέλεσμα να αλλάξει την κατάσταση του σε off-hook (π.χ., το τηλέφωνο πάνω αριστερά στο σχήμα 2.23). Σε αυτή την κατάσταση ο χρήστης μπορεί να επιλέξει τον τηλέφωνο ή τον παραλήπτη του. Επιλέγοντας ένα παραλήπτη αναφέρεται σε μία on-hook λειτουργία. Επιλέγοντας το τηλέφωνο θα δώσει εντολή στο χρήστη να δώσει τα ψηφία που αντιστοιχούν στον αριθμό τηλεφώνου. Το σχήμα 2.23 δείχνει τα γραφήματα σε μία κατάσταση όπου δύο τηλέφωνα το 3070 και 7890 είναι ενεργά και έχει γίνει μία σύνδεση. Σε αυτή την περίπτωση ο χρήστης μπορεί να επιλέξει είτε να "κλείσει" ένα από τα ενεργά τηλέφωνα, ή να καλέσει τον παραλήπτη ενός από τα ανενεργά τηλέφωνα. Όπως μπορούμε να δούμε αυτή η απεικόνιση χρησιμοποιείται για να ελέγχεται η προσομοίωση και για να δοκιμάζονται διαφορετικά σενάρια, και επιτρέπει σε χρήστες που δεν έχουν σχέση με τα CP-nets να καταλάβουν την συμπεριφορά του συστήματος.



Σχήμα 2.23. Παράδειγμα ειδικών γραφημάτων εφαρμογής

Τα ειδικά γραφήματα εφαρμογής έχουν επίσης μία αναφορά και δυνατότητα επανάληψης. Αυτό κάνει δυνατό να εγγράφονται οι είσοδοι που παρέχονται από τον χρήστη και αργότερα να επαναλαμβάνονται αυτές οι είσοδοι, π.χ., αφού έχουν γίνει οι τροποποιήσεις στο εν λόγω CPN μοντέλο. Με αυτό τον τρόπο είναι δυνατό να δημιουργηθούν ένα σύνολο δεδομένων σεναρίων (περιπτώσεις χρήσης) για την δοκιμή του συστήματος που είναι υπό εξέταση.

III

Κεφάλαιο

Ανάλυση καταστάσεων(State Space Analysis)

3 Ανάλυση καταστάσεων(State Space Analysis)

3.1 Εισαγωγή

Ο αυξανόμενος αριθμός των αναπτυξιακών συστημάτων που σχεδιάζονται είναι συσχετισμένος με τα κατανεμημένα και ταυτόχρονα συστήματα. Υπάρχουν πολλά παραδείγματα, από τα συστήματα μεγάλης κλίμακας, στην περιοχή των τηλεπικοινωνιών και των εφαρμογών που βασίζονται στην τεχνολογία WWW, μέχρι μεσαίας ή μικρής κλίμακας συστήματα, στο πεδίο των ενσωματωμένων συστημάτων (embedded systems). Ένα συνηθισμένο κατανεμημένο ή ταυτόχρονο σύστημα αποτελείται από έναν αριθμό ανεξάρτητων αλλά σε επικοινωνία μεταξύ τους διεργασιών. Αυτό σημαίνει ότι η εκτέλεση τέτοιων συστημάτων μπορεί να διεξαχθεί με πολλούς διαφορετικούς τρόπους, π.χ., εξαρτάται από το πότε χάνονται μηνύματα, την ταχύτητα των διεργασιών που εμπλέκονται, και τον χρόνο που κάθε είσοδος λαμβάνεται από το περιβάλλον. Σαν αποτέλεσμα, τα κατανεμημένα και ταυτόχρονα συστήματα είναι, εκ φύσεως, πολύπλοκα και δύσκολα στο να τα σχεδιάσεις και να τα δοκιμάσεις. Αυτό έχει αποτελέσει την αρχή της ανάπτυξης μεθόδων οι οποίες υποστηρίζουν ανάλυση με την βοήθεια υπολογιστή, επικύρωση και επιβεβαίωση της συμπεριφοράς του ταυτόχρονου ή κατανεμημένου συστήματος.

Η βασική ιδέα πίσω από την μέθοδο state space είναι να κατασκευαστεί ένας προσανατολισμένος γράφος ο οποίος έχει ένα κόμβο για κάθε προσεγγίσιμη σήμανση και ένα βέλος για κάθε δράση ενός δεσμευτικού στοιχείου. Τα state spaces είναι επίσης γνωστά και ως occurrence graphs ή reachability graphs/trees. Το πρώτο από αυτά τα δύο ονόματα αντανακλά το γεγονός ότι το state space περιέχει όλες δυνατές σειρές δράσης ενός CP-net, ενώ τα επόμενα δύο ονόματα αντανακλούν ότι το state space περιέχει όλες τις προσεγγίσιμες σημάνσεις του Cp-net. Οι state spaces ενός CP-net μπορούν να κατασκευαστούν εντελώς αυτόματα, και από ένα κατασκευασμένο state space είναι δυνατό να απαντήσουμε σε ένα μεγάλο σύνολο ερωτήσεων σχετικά με την ανάλυση και την επαλήθευση του συστήματος βασισμένοι στην συμπεριφορά του συστήματος. Επιπλέον, αν το σύστημα δεν έχει μία επιθυμητή ιδιότητα έπειτα η state space μέθοδος μπορεί να παρέχει πληροφορίες για την επιδιόρθωση και απόκτηση ή κατάργηση της ιδιότητας.

Η εκφραστική δύναμη των CP-nets που έχει κληρονομηθεί από τα δίκτυα Place/Transitions[18] και η SML γλώσσα συνεπάγεται ότι ουσιαστικά

όλες οι ενδιαφέρουσες ερωτήσεις για την επαλήθευση σχετικά με τα CP-net είναι απροσδιόριστες. Ωστόσο, στην πράξη, πολλά CP-net έχουν μία πεπερασμένη state space κάνοντας την επαλήθευση δυνατή. Επιπρόσθετα σε αυτό, είναι δυνατό να αποδείξουμε ή να καταρρίψουμε ιδιότητες του συστήματος στηριζόμενοι σε μερικά state space, π.χ., πεπερασμένους υπογράφους του συνολικού state space.

3.2 State space των CP-nets

Πολύ συχνά, το πρώτο βήμα στην state space ανάλυση είναι να κάνουμε μερικές τροποποιήσεις ώστε να κάνουμε το CPN μοντέλο κατάλληλο για state space ανάλυση. Κατάλληλο συνήθως σημαίνει να τροποποιήσουμε το μοντέλο ώστε να έχει πεπερασμένο αριθμό καταστάσεων που θα πρέπει να είναι μέσα στα όρια του μεγέθους της υπολογιστικής δυνατότητας που μας παρέχεται.

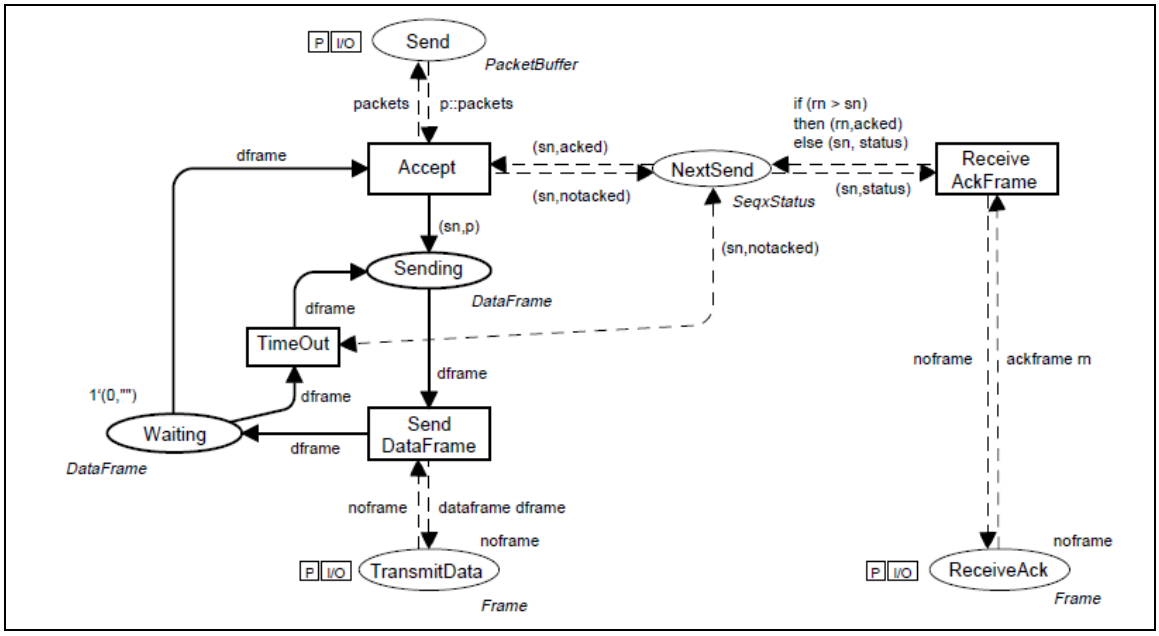
Για το stop-and-wait πρωτόκολλο τροποποιούμε το μοντέλο στους buffer πλαισίων μεταξύ του αποστολέα (sender), παραλήπτη (receiver), και του καναλιού επικοινωνίας (communication channel). Στο αυθεντικό CPN μοντέλο (βλέπε σχήμα 2.2) ένας αυθαίρετος αριθμός token πλαισίου μπορούν να είναι στους places TransmitData και ReceiveAck. Αυτό οδηγεί όμως στο να είναι οι buffers πλαισίων να είναι απεριόριστοι. Για να είναι πεπερασμένο το state space τροποποιούμε την μοντελοποίηση των buffers πλαισίων με τέτοιο τρόπο ώστε να έχουν πεπερασμένη χωρητικότητα (με ένα). Το σχήμα 3.2 δείχνει το τροποποιημένο CPN μοντέλο του αποστολέα του stop-and-wait πρωτοκόλλου, και το σχήμα 3.1 δείχνει τις τροποποιημένες δηλώσεις του τύπου Frame.

```
color Frame = union
        dataframe : DataFrame +
        ackframe  : AckFrame +
        noframe ;
```

Σχήμα 3.1. Τροποποιημένος ορισμός του τύπου Frame

Η τροποποίηση είναι ότι ο αποστολέας μπορεί να βάλει μόνο ένα πλαίσιο δεδομένων στο buffer πλαισίων TransmitData όταν είναι άδειο ανταποκρινόμενο σε ένα token με τιμή noframe που εντοπίζεται στον place. Ομοίως, όταν ο αποστολέας λαμβάνει ένα πλαίσιο επιβεβαίωσης, ένα token με τιμή noframe επιστρέφει στον place ReceiveAck υποδεικνύοντας ότι το buffer πλαισίων είναι τώρα άδειο. Αυτό συνεπάγεται ότι δεν είναι πλέον δυνατό για τα πλαίσια να υπερπηδά το ένα το άλλο. Ο παραλήπτης και το κανάλι επικοινωνίας έχουν τροποποιηθεί με παρόμοιο τρόπο σαν τον αποστολέα.

Το αρχικό κομμάτι για το state space του stop-and-wait πρωτοκόλλου δείχνεται στο σχήμα 3.3. Τα κυκλικά κουτιά με το λεπτό περίγραμμα είναι οι κόμβοι του state space. Κάθε ένα από αυτά σηματοδοτεί μία δυνατή σήμανση. Η σήμανση του κάθε place ξεχωριστά φαίνεται στα κουτιά με το διακεκομμένο περίγραμμα δίπλα στους κόμβους. Παρακάτω, επικεντρωνόμαστε στην σήμανση των places του αποστολέα, και για αυτό τα κουτιά αφορούν μόνο τη σήμανση των places του αποστολέα. Κατά σύμβαση παραλείπονται places με κενή σήμανση και επίσης έχει παραλειφθεί και ο place Send.



Σχήμα 3.2. Τροποποιημένος αποστολέας του stop-and-wait πρωτοκόλλου

Στην κορυφή του σχήματος 3.3, έχουμε ένα κόμβο με πιο λεπτή γραμμή περιγράμματος. Αυτός ο κόμβος αποτελεί την αρχική σήμανση. Το κείμενο εντός του κόμβου μας λέει ότι αυτός είναι ο κόμβος νούμερο 1 και ότι έχει μηδέν (0) predecessors (κόμβους που οδηγούν σε αυτόν) και ένα (1) successor (κόμβοι στους οποίους οδηγούμαστε μετά από αυτόν). Ανάλογα, βλέπουμε ότι ο κόμβος 2 έχει τρείς predecessors και ένα successor. Με σύμβαση χρησιμοποιούμε το σύμβολο M_n για να δηλώσουμε την σήμανση του κόμβου με αριθμό n. Κάθε βέλος αντιπροσωπεύει την ενεργοποίηση του δεσμευτικού στοιχείου που βρίσκεται δίπλα στο κουτί από κάθε βέλος. Στην M_1 το μόνο ενεργό δεσμευτικό στοιχείο είναι το transition Accept με την δέσμευση που βρίσκεται στο κουτί. Όταν, το δεσμευτικό στοιχείο ενεργεί, οδηγούμαστε στην σήμανση M_2, στην οποία πάλι έχουμε ένα ενεργό δεσμευτικό στοιχείο. Μία δράση του SendDataFrame με την μεταβλητή dframe δεσμευμένη στο (0, "Software") θα μας οδηγήσει στην σήμανση M_3. Στη σήμανση M_3 υπάρχουν τρία ενεργά δεσμευτικά στοιχεία τα οποία εξαρτώνται από τις δύο ενεργές δεσμεύσεις του transition Transmit (πλαίσιο δεδομένων χάθηκε ή επιτυχής μετάδοση) και τη μία ενεργή δέσμευση του transition TimeOut. Από το ποιά από τις τρείς αυτές δεσμεύσεις δρα καταλήγουμε σε μία από τις σημάνσεις M_4, M_5, ή M_6 . Ο κόμβος 6 έχει μία σημείωση (*) που ακολουθεί τον αριθμό του κόμβου. Αυτό δείχνει ότι οι successors αυτού του κόμβου δεν έχουν υπολογιστεί ακόμη.

Από το πρώτο κομμάτι του state space φαίνεται μπορούμε να δούμε ότι μία δράση του transition Transmit με την με την μεταβλητή success δεσμευμένη στο false στην σήμανση M_3 ακολουθουμένη από μία δράση του transition TimeOut οδηγεί μέσω της σήμανσης M_5 στην σήμανση M_2 . Η δράση των ίδιων δεσμευτικών στοιχείων με την αντίθετη σειρά οδηγεί επίσης από την M_3 στην M_2 , αλλά όχι με την M_4 ενδιάμεση σήμανση. Η ενεργή δέσμευση του TimeOut είναι ταυτόχρονα ενεργή με κάθε μία από τις δύο δεσμεύσεις του transition Transmit. Εξ ορισμού, βέλη τα οποία σχετίζονται με βήματα με πάνω από ένα δεσμευτικά στοιχεία δεν αποτελούν μέρος του state space. Τέτοια βέλη μπορούν να δίνουν πληροφορίες σχετικά με τον συγχρονισμό μεταξύ δεσμευτικών στοιχείων, αλλά δεν είναι απαραίτητα για την επιβεβαίωση των βασικών ιδιοτήτων της συμπεριφοράς του συστήματος.

3.3 Γένεση των state space

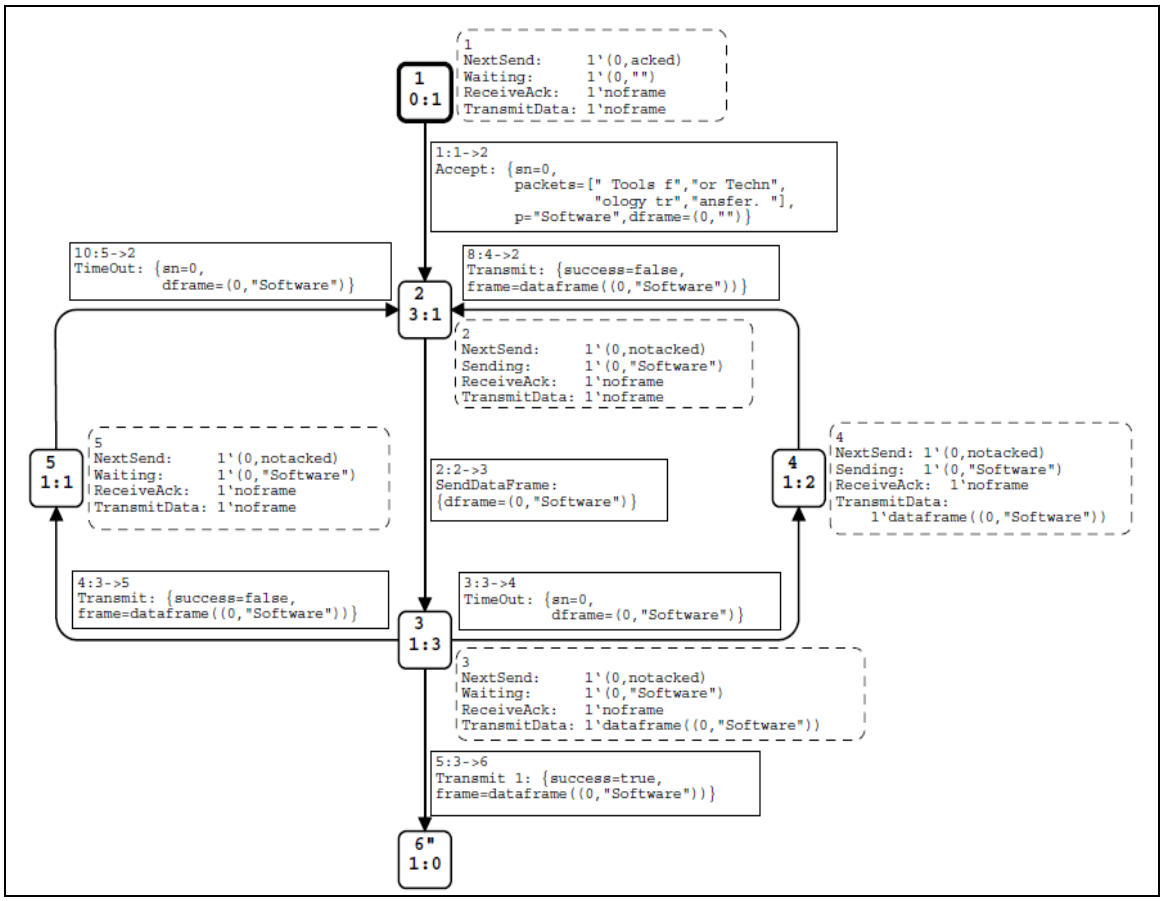
Η δημιουργία των state space γίνεται συνήθως χρησιμοποιώντας δύο τρόπους: τον διαδραστικό (interactive) και τον αυτόματο (automatic). Η σχέση ανάμεσα σε αυτούς τους δύο τρόπους είναι παρόμοια με την σχέση ανάμεσα στην διαδραστική και αυτόματη προσομοίωση.

3.3.1 Διαδραστική δημιουργία state space

Στην διαδραστική δημιουργία ο χρήστης προσδιορίζει μία σήμανση, και το αντίστοιχο εργαλείο (εδώ CPN tool) υπολογίζει τα ενεργά δεσμευτικά στοιχεία σε αυτή τη σήμανση, και τον successor που είναι αποτέλεσμα της δράσης του καθενός από αυτά τα δεσμευτικά στοιχεία. Το πρώτο κομμάτι του state space του stop-and-wait πρωτοκόλλου του σχήματος 3.3 το έχουμε λάβει με αυτό τον τρόπο. Η διαδραστική γένεση χρησιμοποιείται συνήθως σε σύνδεση με σχεδιαστικά κομμάτια του state space.

3.3.2 Αυτόματη δημιουργία state space

Τα state spaces είναι στις περισσότερες περιπτώσεις ενός μεγέθους το οποίο δεν είναι πρακτικό για να δημιουργήσουμε το πλήρες state space με τον διαδραστικό τρόπο. Για αυτό το λόγο είναι επίσης δυνατό να δημιουργήσουμε το state space με αυτόματο τρόπο. Εδώ, ο χρήστης απλά ξεκινά την δημιουργία και το state space αυτόματα υπολογίζεται πλήρως. Ο χρήστης μπορεί να ελέγξει την αυτόματη γένεση χρησιμοποιώντας τις δυνατότητες stop και branching που παρέχουν τα περισσότερα εργαλεία. Οι stop επιλογές επιτρέπουν στον χρήστη να προσδιορίσει κάτω από ποιές συνθήκες ο υπολογισμός θα πρέπει να τερματιστεί. Ένα παράδειγμα αυτού είναι να προσδιοριστεί ο ανώτερος αριθμός των κόμβων που θα υπολογιστούν. Οι branching επιλογές δίνουν τοπικό έλεγχο στον χρήστη πάνω από την γένεση successors, κάνοντας δυνατό ότι μόνο ένα συγκεκριμένο υποσύνολο των successors ενός κόμβου θα πρέπει να δημιουργηθεί. Ο έλεγχος της γένεσης είναι ιδιαιτέρως σημαντικός στις αρχικές φάσεις της state space ανάλυσης η οποία είναι συνήθως προσανατολισμένη στο να αναγνωριστεί η ρύθμιση του συστήματος η οποία είναι κατάλληλη για την state space ανάλυση. Οι stop και branching επιλογές κάνουν επίσης δυνατό να λάβουμε ένα μερικό state space. Το μερικό state space είναι ένα υποσύνολο του συνολικού state space. Σε πολλές περιπτώσεις δεν είναι απαραίτητο να δημιουργηθεί όλο το state space προκειμένου να εντοπίσουμε λάθη και να ελέγξουμε την ορθότητα ενός συστήματος.



Σχήμα 3.3 . Αρχικό κομμάτι του state space για το stop-and-wait πρωτόκολλο

3.4 Η αναφορά του state space

Όταν ένα state space έχει δημιουργηθεί, το επόμενο πράγμα που συνήθως κάνουμε είναι να μελετήσουμε την αναφορά state space. Αυτή είναι ένα αρχείο κειμένου που περιέχει απαντήσεις σε ένα σύνολο από προκαθορισμένες ιδιότητες οι οποίες μπορούν να χρησιμοποιηθούν σε οποιοδήποτε CPN μοντέλο (π.χ., γενικές ιδιότητες που μπορούν να τροποποιηθούν ανεξάρτητα από το σύστημα που είναι υπό εξέταση). Περιέχει πολύ σημαντικές πληροφορίες σχετικά με την συμπεριφορά του CPN μοντέλου. Μελετώντας την αναφορά ο χρήστης παίρνει μία πρώτη ρεαλιστική εικόνα, όσο αναφορά το αν το CPN μοντέλο δουλεύει όπως αναμενόταν. Αν το σύστημα περιέχει λάθη αυτά επηρεάζουν συχνά την αναφορά state space. Παρακάτω θα δούμε τα διάφορα μέρη της state space αναφοράς για το stop-and-wait πρωτόκολλο.

3.4.1 Στατιστικές πληροφορίες

Το πρώτο κομμάτι της state space αναφοράς περιέχει μερικές στατιστικές πληροφορίες σχετικά με το μέγεθος του state space. Οι στατιστικές πληροφορίες για το state space του stop-and-wait πρωτοκόλλου παρατίθενται στον πίνακα 5. μπορούμε να δούμε ότι έχει 1220 κόμβους και 3621 βέλη, για το πλήρες state space.

State Space		SCC Graph	
Nodes:	1220	Nodes:	742
Arcs:	3621	Arcs:	2622
Secs:	4	Secs:	1
Status:	Full		

Πίνακας 5. Στατιστικές πληροφορίες

Το στατιστικό κομμάτι περιέχει επίσης πληροφορίες για τον SCC-γράφο[19]. Ένας strongly connected component (SCC) είναι ένας μέγιστος υπογράφος στον οποίο είναι δυνατό να βρεθεί ένα μονοπάτι ανάμεσα σε κάθε ζευγάρι κόμβων, π.χ., ένα κομμάτι του state space στο οποίο όλοι οι κόμβοι είναι αμοιβαίως προσεγγίσιμοι από τον καθένα. Κάθε κόμβος στο state space ανήκει σε ένα ακριβώς SCC και ως εκ τούτου μπορούμε τα SCCs για να λάβουμε ένα πιο περιεκτικό γράφο ονομαζόμενο και SCC-γράφο. Ο SCC-γράφος έχει ένα κόμβο για κάθε SCC και ένα βέλος για κάθε δεσμευτικό στοιχείο που οδηγεί από ένα SCC σε κάποιο άλλο. Οι SCC μπορούν να υπολογιστούν με χρήση του αλγόριθμου Tarjan[20] ο οποίος έχει χρονική και χωρική πολυπλοκότητα γραμμική με το μέγεθος του state space. Ως εκ τούτου, είναι οικονομικό να κατασκευάζουμε SCC-γράφους οι οποίοι καταλήγουν να είναι πολύ χρήσιμοι για την διερεύνηση διαφόρων ειδών ιδιοτήτων συμπεριφοράς[10,19]. Από τον πίνακα 5 βλέπουμε ότι ο SCC-γράφος έχει 742 κόμβους και 2622 βέλη. Αυτό σημαίνει ότι υπάρχουν λιγότερα SCCs (742) από ότι state space κόμβοι (1220), και ως εκ τούτου τουλάχιστο ένα SCC καλύπτει παραπάνω από ένα state space κόμβο. Από αυτό συμπεραίνουμε ότι υπάρχει μία ατέρμονη σειρά δράσης, και έτσι δεν μπορούμε να είμαστε σίγουροι ότι το πρωτόκολλο τερματίζει. Για να σιγουρευτούμε για τερματισμό συνήθως βάζουμε περιορισμό στον αριθμό των επαναμεταδόσεων.

3.4.2 Ιδιότητες σε οριακές συνθήκες

Ένα άλλο κομμάτι της state space αναφοράς περιέχει πληροφορίες για τα όρια των ακεραίων (integer) και των multi-set. Ο πίνακας 6 δείχνει τα άνω και κάτω όρια των ακεραίων, π.χ., τον μέγιστο και τον ελάχιστο αριθμό των token που μπορούν να εντοπιστούν σε κάθε place ξεχωριστά σε κάθε προσεγγίσιμη σήμανση. Ο πίνακας παραθέτει μόνο τους places του παραλήπτη και του αποστολέα καθώς οι υπόλοιποι places του CP-net σχετίζονται μέσω port assignments στους places είτε του αποστολέα είτε του παραλήπτη. Αυτό σημαίνει ότι το όριο των ακεραίων αυτών των places μπορεί να συναχθεί από αυτό του πίνακα 6. Από τον πίνακα βλέπουμε ότι κάθε ένας από τους places Sending και Waiting έχει πάντα ένα ή κανένα token, ενώ όλοι οι υπόλοιποι places έχουν ακριβώς ένα.

Place	Upper Bound	Lower Bound
NextSend	1	1
ReceiveAck	1	1
Send	1	1
Sending	1	0
TransmitData	1	1
Waiting	1	0
NextReceive	1	1
ReceiveData	1	1
Received	1	1
TransmitAck	1	1

Πίνακας 6. Άνω και κάτω όρια ακεραίων

Ο πίνακας 7 δείχνει τα άνω όρια των multi-set για τους places στον αποστολέα. Εξ ορισμού, το άνω όριο ενός multi-set ενός place είναι το μικρότερο multi-set που είναι μεγαλύτερο από όλες τις προσεγγίσιμες σημάνσεις του place. Τα όρια ακεραίων μας δίνουν πληροφορίες για τον αριθμό των token, ενώ τα όρια των multi-set μας δίνουν πληροφορίες σχετικά με τις τιμές που μπορεί να έχουν τα token.

Από τα όρια των multi-set μπορούμε, για παράδειγμα, να δούμε ότι ο place TransmitData μπορεί να περιέχει και τα πέντε διαφορετικά πλαίσια δεδομένων, και ένα token με τιμή noframe που αναφέρεται στο buffer πλαισίου που είναι κενό. Βλέπουμε επίσης ότι καμία άλλη τιμή token δεν είναι δυνατή για αυτό το place. Πρέπει να προσέξουμε ότι το άνω multi-set όριο του TransmitData είναι ένα multi-set με έξι στοιχεία, παρόλο το άνω

όριο ακεραίου μας λέει ότι δεν μπορεί ποτέ να έχει παραπάνω από ένα token στον Transmitdata τη φορά. Ανάλογα, κάθε ένας από τους places Sending και Waiting μπορεί να περιέχουν όλα τα πέντε δυνατά πλαίσια δεδομένων. Ο place Waiting μπορεί να έχει επιπρόσθετα και το πλαίσιο δεδομένων (0,"") το οποίο είναι το πλαίσιο δεδομένων που εντοπίζεται στον place στην αρχική σήμανση.

Ωστόσο, τα όρια των multi-set μας δίνουν μερική παραπάνω καινούργια γνώση. Για παράδειγμα, ότι τα πλαίσια επιβεβαιώσεων δεν μεταφέρουν ποτέ τον αριθμό σειράς 0, όπως μπορούμε να δούμε από το άνω όριο του multi-set του place ReceiveAck.

Το κάτω multi-set όριο ενός place είναι το μεγαλύτερο multi-set το οποίο είναι μικρότερο από όλες τις προσεγγίσιμες σημάνσεις του place. Για όλους τους places του stop-and-wait πρωτοκόλλου το κάτω multi-set όριο είναι ένα κενό multi-set. Αυτό γίνεται διότι δε υπάρχουν places στο CPN μοντέλο στους οποίους ένα token με μία συγκεκριμένη τιμή να βρίσκεται μόνιμα εκεί.

NextSend	1‘(0,acked) + 1‘(0,notacked) + 1‘(1,acked) + 1‘(1,notacked) + 1‘(2,acked) + 1‘(2,notacked) + 1‘(3,acked) + 1‘(3,notacked) + 1‘(4,acked) + 1‘(4,notacked) + 1‘(5,acked)
ReceiveAck	1‘ackframe(1) + 1‘ackframe(2) + 1‘ackframe(3)+ 1‘ackframe(4) + 1‘ackframe(5)+ 1‘noframe
Sending	1‘(0,”Software”) + 1‘(1,” Tools f”) + 1‘(2,”or Techn”) + 1‘(3,”ology tr”) + 1‘(4,”ansfer. ”)
TransmitData	1‘dataframe((0,”Software”)) + 1‘dataframe((1,” Tools f”)) + 1‘dataframe((2,”or Techn”)) + 1‘dataframe((3,”ology tr”)) + 1‘dataframe((4,”ansfer. ”)) + 1‘noframe
Waiting	1‘(0,””) + 1‘(0,”Software”) + 1‘(1,” Tools f”) + 1‘(2,”or Techn”) + 1‘(3,”ology tr”) + 1‘(4,”ansfer. ”)
Send	1‘[”Software”,” Tools f”,”or Techn”, ”ology tr”,”ansfer. ”] + 1‘[” Tools f”,”or Techn”,”ology tr”, ”ansfer. ”] + 1‘[”or Techn”,”ology tr”,”ansfer. ”] + 1‘[”ology tr”,”ansfer. ”] + 1‘[”ansfer. ”] + 1‘[]

Πίνακας 7. Άνω όρια multi-set

3.4.3 Home και liveness ιδιότητες

Το επόμενο κομμάτι της state space αναφοράς δείχνεται στον πίνακα 8. Παρέχει πληροφορίες σχετικά με τις home και liveness ιδιότητες. Μία home σήμανση είναι μία σήμανση η οποία είναι προσεγγίσιμη από όλες τις προσεγγίσιμες σημάνσεις, π.χ., μία σήμανση η οποία πάντα μπορεί να προσεγγιστεί, ανεξάρτητα από το τι έχει συμβεί πριν. Βλέπουμε ότι

το πρωτόκολλο έχει μία απλή home σήμανση, M_{862}. Μία dead σήμανση είναι μία σήμανση με κανένα ενεργό transition. Βλέπουμε ότι το πρωτόκολλο έχει μία μόνο dead σήμανση, και αυτή η dead σήμανση είναι ταυτόσημη με την home σήμανση. Με μία πρώτη ματιά, κάποιος θα μπορούσε να πει ότι η ύπαρξη μίας dead σήμανσης παραπέμπει και στην ύπαρξη home σημάνσεων. Ωστόσο, εξ ορισμού, μία σήμανση είναι προσεγγίσιμη από τον ίδιο της τον εαυτό, και ως εκ τούτου μία dead σήμανση μπορεί να είναι και home σήμανση (υπό την προϋπόθεση ότι είναι η μόνη dead σήμανση). Αυτή είναι η περίπτωση του συστήματος του πρωτοκόλλου.

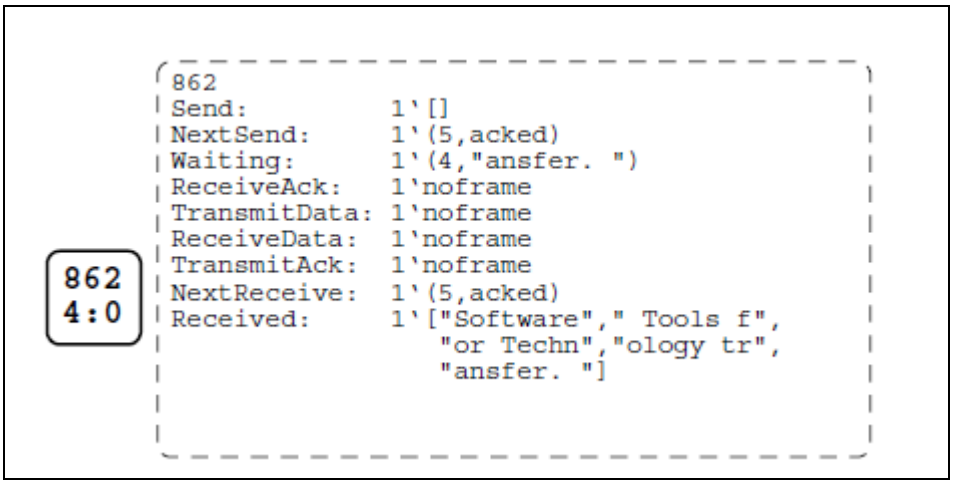
Home Markings:	[862]
Dead Markings:	[862]
Dead Transitions Instances:	None
Live Transitions Instances:	None

Πίνακας 8. Home και liveness ιδιότητες

Από τις liveness ιδιότητες βλέπουμε επίσης ότι δεν υπάρχουν καθόλου dead transitions. Οι Dead transitions είναι παρόμοιοι με τον dead κώδικα σε μία γλώσσα προγραμματισμού και σημαίνει ότι κάθε transition ενεργοποιείται τουλάχιστο σε μία προσεγγίσιμη σήμανση (το οποίο είναι μάλλον μία αδύναμη ιδιότητα). Επίσης βλέπουμε ότι δεν υπάρχουν live transitions. Ένας live transition είναι ένας transition ο οποίος μπορεί πάντα, ανεξάρτητα του τι συμβαίνει, να γίνει ενεργός ξανά. Όταν υπάρχουν dead σημάνσεις (όπως στην περίπτωση του πρωτοκόλλου μας), δεν μπορεί να υπάρχουν live transitions.

Τώρα θα ερευνήσουμε την σήμανση M_{862} με περισσότερη λεπτομέρεια. Το σχήμα 3.4 δείχνει την σήμανση M_{862}. Όμοια με το σχήμα 3.3 το κουτί με την διακεκομμένη γραμμή δίπλα στον κόμβο δίνει πληροφορίες σχετικά με την σήμανση των επιμέρους places. Το κουτί δείχνει τις σημάνσεις όλων των place στον αποστολέα και τον παραλήπτη που δεν έχουν κενή σήμανση. Η παρατήρηση του κουτιού αποκαλύπτει ότι η σήμανση M_{862} είναι η επιθυμητή τερματική κατάσταση του πρωτοκόλλου, στην οποία όλα τα πακέτα έχουν σταλθεί, έχουν ληφθεί στην σωστή σειρά, και τα buffer πλαισίων είναι κενά. Εφόσον η M_{862} είναι η μόνη dead σήμανση αυτό συνεπάγεται ότι το πρωτόκολλο είναι εν μέρει σωστό – αν το πρωτόκολλο τερματίζει, τότε τερματίζει στην επιθυμητή κατάσταση.

Εφόσον η σήμανση M_{862} είναι η μία home σήμανση το πρωτόκολλο έχει μία πιθανότητα να τερματίσει σωστά. Αυτό σημαίνει ότι είναι αδύνατο να φτάσουμε μία σήμανση από την οποία να δεν μπορούμε να τερματίσουμε το πρωτόκολλο με το σωστό αποτέλεσμα.



Σχήμα 3.4. Home και dead σήμανση

3.4.4 Ιδιότητες fairness

Το τελευταίο κομμάτι της state space αναφοράς φαίνεται στο σχήμα 2.9. Μας παρέχει πληροφορίες σχετικά με τις ιδιότητες fairness, π.χ., πόσο συχνά τα ξεχωριστά transition δρουν. Βλέπουμε ότι ο SendDataFrame, ο TimeOut, και ο Transmit 2 είναι impartial. Αυτό σημαίνει ότι ο καθένας από αυτούς δρα πάρα πολύ συχνά σε μία ατέρμονη σειρά δράσης. Ο Transmit 2 είναι ο transition της δεύτερης της σελίδας UniChannel. Αυτή είναι η έκφανση που αναφέρεται στο κανάλι δεδομένων. Ως εκ τούτου, σε κάθε μη ντετερμινιστική εκτέλεση του πρωτοκόλλου θα συνεχίσουμε να έχουμε λήξεις χρόνου (timeouts) και θα συνεχίσουμε να αποστέλλουμε και να μεταδίδουμε πλαίσια δεδομένων. Ο transition Accept είναι fair η οποία είναι μία αδύναμη ιδιότητα fairness σύμφωνα με την οποία ο transition δρα απείρως συχνά σε όλες τις ατέρμονες σειρές δράσης όπου είναι απείρως συχνά ενεργός.

Receive DataFrame	Fair
Send AckFrame	No Fairness
Accept	Fair
Receive AckFrame	No Fairness
Send DataFrame	Impartial
TimeOut	Impartial
Transmit 1	No Fairness
Transmit 2	Impartial

Πίνακας 9. ιδιότητες fairness

3.5 Πρότυπα ερωτήματα (Standard queries)

Επιπρόσθετα στη state space αναφορά, μας προσφέρεται ένα σύνολο από standard query συναρτήσεις οι οποίες επιτρέπουν στο χρήστη να κάνει μία πιο λεπτομερή διερεύνηση των πρότυπων ιδιοτήτων συμπεριφοράς. Πολλές από αυτές τις query συναρτήσεις επιστρέφουν αποτελέσματα τα οποία ήδη περιέχονται στην state space αναφορά όπως περιγράφηκε στην προηγούμενη ενότητα. Στην πραγματικότητα, η state space αναφορά εφαρμόζεται με την χρήση ενός υποσυνόλου των διαθέσιμων query συναρτήσεων.

Σαν παράδειγμα, το εργαλείο έχει δύο συναρτήσεις, την UpperInteger και την LowerInteger οι οποίες προσδιορίζουν τον μέγιστο/ελάχιστο αριθμό token που διαμένουν σε ένα σύνολο από places (όπου η state space αναφορά επικεντρώνεται μόνο σε επιμέρους places). Οι queries του σχήματος 3.5 χρησιμοποιούν αυτές τις δύο συναρτήσεις για να προσδιορίσουν τον μέγιστο/ελάχιστο αριθμό token στους places Sending και Waiting.

```
fun SendWaitMarking n =
    (Mark.Sender´Sending 1 n) +
    (Mark.Sender´Waiting 1 n) ;

UpperInteger (SendWaitMarking) ;
LowerInteger (SendWaitMarking) ;
```

Σχήμα 3.5. Παράδειγμα των standard query

Είναι εύκολο να χρησιμοποιήσουμε τις standard queries. Για τις queries στο σχήμα 3.5, το μόνο πράγμα που ο χρήστης έχει να κάνει, είναι να δηλώσει μία συνάρτηση που να χρησιμοποιείται ως όρισμα στην UpperInteger και/ή LowerInteger. Αυτή η συνάρτηση εκπροσωπεί μία σήμανση M_n (από ολόκληρο το CPN μοντέλο) μέσα σε ένα multi-set M_n (Sending) + M_n (Waiting). Το αποτέλεσμα και των δύο queries είναι 1, και ως εκ τούτου έχουμε αποδείξει ότι πάντα θα υπάρχει ένα token στον Sending ή ένα token στον Waiting, αλλά όχι ταυτόχρονα και στους δύο places (βλέπε σχήμα 3.1).

Ανάλογα υπάρχει μία συνάρτηση HomeSpace να προσδιορίσει πότε ένα συγκεκριμένο σύνολο σημάνσεων συγκροτεί ένα home χώρο, π.χ., πότε είναι πάντα δυνατό να προσεγγίσουμε μία από τις συγκεκριμένες σημάνσεις – ανεξάρτητα από το τι συνέβη προηγουμένως. Το όρισμα σε αυτή τη συνάρτηση είναι μία λίστα που περιέχει ένα σύνολο σημάνσεων υπό εξέταση.

3.6 Οπτικοποίηση των state spaces

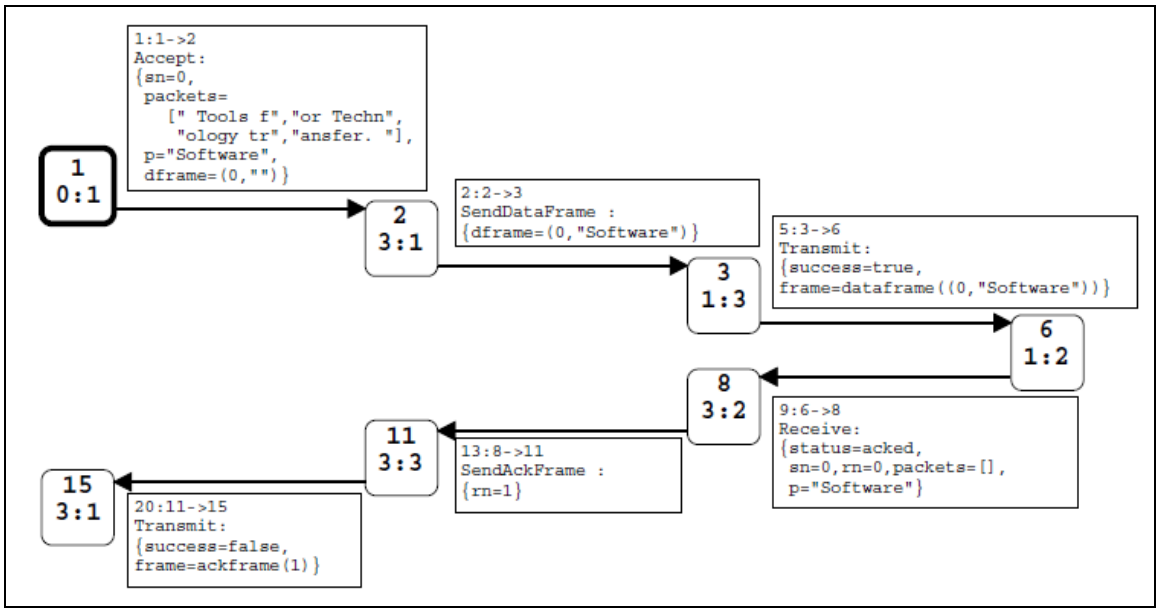
Από όταν τα state spaces γίνονται συχνά μεγάλα, σπάνια μας κάνει αίσθηση να τα σχεδιάσουμε ολόκληρα. Ωστόσο, τα αποτελέσματα των queries θα είναι συχνά ένα σύνολο κόμβων και/ή βελών που επεξεργάζονται ιδιότητες συγκεκριμένου ενδιαφέροντος, π.χ., ένα μονοπάτι στο state space που οδηγεί από μία σήμανση σε κάποια άλλη. Ένας καλός και γρήγορος τρόπος για να λάβουμε λεπτομερείς πληροφορίες σε ένα μικρό σύνολο κόμβων και βελών είναι να σχεδιάσουμε το κομμάτι του state space που μας ενδιαφέρει. Αυτό κάνει την οπτικοποίηση ιδιαιτέρως χρήσιμη στο να εντοπίσουμε λάθη σε ένα σύστημα υπό εξέταση.

Το state space μπορεί να σχεδιαστεί είτε με μικρά βήματα, π.χ., κόμβο με κόμβο ή βέλος με βέλος, ή σε τμήματα χρησιμοποιώντας αποτελέσματα από, για παράδειγμα, queries σαν είσοδο σε ένα αριθμό built-in σχεδιαστικών συναρτήσεων. Το σχήμα 3.3 έχει ληφθεί από την συνδυασμένη χρήση διαδραστικής γένεσης με σχεδιασμό.

Για να λάβουμε λεπτομερείς πληροφορίες σχετικά με τους σχεδιαστικούς περιορισμούς του state space, είναι δυνατό να επισυνάψουμε descriptors στους κόμβους και στα βέλη. Τα κουτιά που τοποθετούνται δίπλα στους κόμβους και τα βέλη του σχήματος 3.3 είναι παραδείγματα descriptors. Για τους κόμβους, οι descriptors συνήθως δείχνουν την σήμανση συγκεκριμένων places. Για τα βέλη, οι descriptors δείχνουν το δρώντα transition και τις δεσμεύσεις μερικών ή όλων των

μεταβλητών του. Οι descriptors έχουν ορατά αποτελέσματα αλλά ο χρήστης μπορεί να τροποποιήσει τους descriptors γράφοντας ένα script το οποίο προσδιορίζει τα περιεχόμενα και τη διάταξη του descriptor.

Για να γίνει αντιληπτή η χρήση του σχεδιασμού σε συνδυασμό με το debugging ερευνούμε τι θα συνέβαινε εάν τροποποιούσαμε το stop-and-wait πρωτόκολλο ώστε να στέλνει ένα πλαίσιο επιβεβαίωσης πίσω στον αποστολέα όταν ο παραλήπτης λαμβάνει μόνο το πλαίσιο δεδομένων το οποίο αναμένει. Το state space για αυτή την παραλλαγή του πρωτοκόλλου έχει 261 κόμβους και 609 βέλη. Έχει ακόμη μία dead σήμανση στην οποία όλα τα πακέτα δεδομένων έχουν σταλθεί, έχουν ληφθεί με την σωστή σειρά, και τα buffers πλαισίων είναι άδεια. Ωστόσο, αυτή η σήμανση δεν είναι πλέον home σήμανση. Αυτό υποδεικνύει ότι μπορούμε να καταλήξουμε σε μία σήμανση από την οποία δεν μπορούμε να τερματίσουμε σωστά το πρωτόκολλο. Για την διερεύνηση αυτού του προβλήματος, εκμεταλλευόμαστε την ανάδραση από την standard query συνάρτηση για τον έλεγχο του πότε μία σήμανση είναι home σήμανση. Αυτή η ανάδραση μας λέει ότι υπάρχει ένα SCC από το οποίο είναι δυνατό να καταλήξουμε σε μία dead σήμανση. Με τις έννοιες της standard query συνάρτησης, μπορούμε να δούμε ότι αυτό το SCC έχει μόνο οκτώ κόμβους και ότι ένας από αυτούς είναι ο κόμβος 15. χρησιμοποιώντας μία άλλη query συνάρτηση μπορούμε να βρούμε ένα από τα συντομότερα μονοπάτια που οδηγούν από την αρχική σήμανση (κόμβος 1) στον κόμβο 15. αυτό το μονοπάτι δείχνεται στο σχήμα 3.6, που έχει δημιουργηθεί με την χρήση των built-in συναρτήσεων. Από τους descriptors των βελών αυτού μονοπατιού είναι εύκολο να προσδιορίσουμε το πρόβλημα. Στην αρχή, επιτυχώς μεταδίδουμε το πρώτο πακέτο δεδομένων σύμφωνα με τα τρία πρώτα βέλη στο μονοπάτι. Μετά προσπαθούμε να μεταδώσουμε το πλαίσιο επιβεβαίωσης αλλά χάνεται. Ως εκ τούτου, όταν χάνουμε το πλαίσιο επιβεβαίωσης καταλήγουμε σε μία κατάσταση στην οποία δεν μπορούμε να τερματίσουμε το πρωτόκολλο σωστά. Το μόνο που μπορεί να συμβεί από αυτό το σημείο είναι να συνεχίσουμε να επαναμεταδίδουμε το πρώτο πλαίσιο δεδομένων. Αυτές οι επαναμεταδόσεις δεν μπορούν να δημιουργήσουν ένα πλαίσιο επιβεβαίωσης που να μας επιτρέπει να αρχίσουμε να μεταδίδουμε το επόμενο πλαίσιο δεδομένων.



Σχήμα 3.6. Σειρά δράσης ως debugging πληροφορία

3.7 Ενσωμάτωση με την προσομοίωση

Κατά την διάρκεια της μοντελοποίησης και του σχεδιασμού, ο χρήστης αρκετά συχνά μεταφέρεται από την state space ανάλυση στην προσομοίωση και πάλι πίσω. Για να υποστηριχθεί αυτό, η διαδικασία παραγωγής του state space είναι καλά συνδεδεμένη με τον προσομοιωτή.

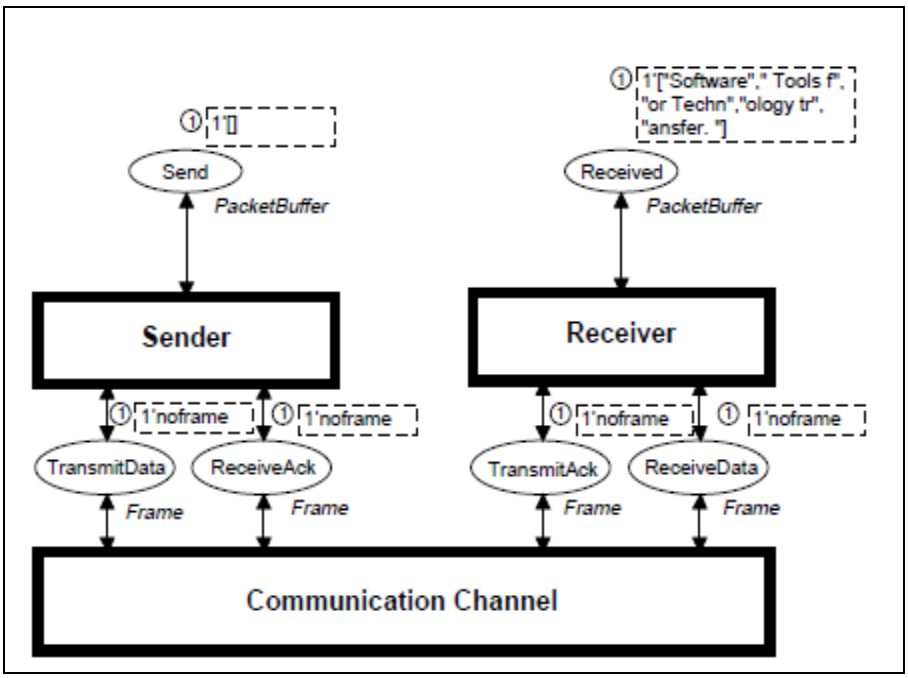
Όταν μία σήμανση μεταφέρεται από το εργαλείο γένεσης των state space στον προσομοιωτή, απεικονίζεται στο CPN διάγραμμα με τον συνηθισμένο τρόπο. Ο χρήστης μπορεί να επιθεωρεί τις σημάνσεις των επιμέρους places, και τις ενεργοποιήσεις των επιμέρους transitions. Είναι επίσης δυνατό να ξεκινήσει μία προσομοίωση από την μεταφερθείσα σήμανση. Ως παράδειγμα μπορούσαμε να έχουμε ελέγξει ότι η M_{862} είναι η επιθυμητή dead σήμανση μεταφέροντας την σήμανση από το εργαλείο στον προσομοιωτή και να επιβλέπουμε την σήμανση των επιμέρους places. Το σχήμα 3.7 δείχνει την σελίδα ανωτέρου επιπέδου του stop-and-wait πρωτοκόλλου μετά την μεταφορά της M_{862} στον προσομοιωτή.

Το να μεταφέρουμε την παρούσα σήμανση του προσομοιωτή στο εργαλείο υποστηρίζεται επίσης. Μία συνηθισμένη χρήση αυτού είναι να ερευνήσουμε όλες τις δυνατές σημάνσεις που προσεγγίζονται μέσα σε λίγα βήματα από την παρούσα σήμανση του προσομοιωτή. Σε αυτή την περίπτωση ο χρήστης μεταφέρει την σήμανση του προσομοιωτή στο

εργαλείο γένεσης και όλες οι successor σημάνσεις μπορούν να βρεθούν και να σχεδιαστούν όπως απεικονίζεται παρακάτω.

3.8 Προχωρημένα queries

Η state space αναφορά και οι standard queries είναι καλές για να παρέχουν μία γενική εικόνα της συμπεριφοράς του συστήματος. Ωστόσο, υπάρχουν επίσης και κάποιοι περιορισμοί. Πρώτα από όλα, πολλές ενδιαφέρουσες ιδιότητες του συστήματος δεν μπορούν εύκολα να διερευνηθούν με την χρήση των standard queries. Έπειτα, για συστήματα debugging, πιο επεξεργασμένες queries είναι συχνά απαραίτητες για να εντοπίσουμε την πηγή του προβλήματος. Γι' αυτό μία πιο γενική query γλώσσα που εφαρμόζεται πάνω από την Standard ML (SML) παρέχεται. Παρέχει πλεονεκτήματα στο να διερχόμαστε του state space με διαφορετικό τρόπο και έτσι να γράφονται non-standard και εξαρτώμενες από το μοντέλο queries. Πάνω από αυτή την γενική query γλώσσα, μία παραλλαγή της χρονικής λογικής CTL[21] παρέχεται. Σε αυτή την παραλλαγή της CTL, είναι δυνατό να διατυπώσουμε queries σχετικά με δεσμευτικά στοιχεία καθώς και σημάνσεις που αναφέρονται σε Petri nets που είναι ταυτόχρονα προσανατολισμένα στη δράση και στην κατάσταση.



Σχήμα 3.7. Home και dead σημάνσεις του πρωτοκόλλου

Όμοια με τις standard queries, μία non-standard query συνήθως αποτελείται από λίγες γραμμές κώδικα σε SML. Για να απεικονίσουμε αυτό θα χρησιμοποιήσουμε την query γλώσσα για να διερευνήσουμε μία ιδιότητα εξαρτώμενη από το σύστημα του stop-and-wait πρωτοκόλλου. Θέλουμε να ελέγξουμε ότι ο αποστολέας προηγείται το πολύ ενός πακέτου δεδομένων του αποστολέα, π.χ., ότι η πειθαρχεία του stop-and-wait παρατηρείται από το πρωτόκολλο. Αυτό μπορεί να αποδειχθεί με την non-standard query που παρατίθεται στο σχήμα 3.8. Η SWPredicate είναι μία διαδικασία, στη οποία δίνεται μία σήμανση, ελέγχει ότι το άθροισμα του μήκους των λιστών στους places Send και Received διαφέρει το πολύ κατά ένα από τον συνολικό αριθμό των πακέτων δεδομένων προς αποστολή. PredAllNodes είναι μία προκαθορισμένη query συνάρτηση, που επιστρέφει όλους τους κόμβους του state space που ικανοποιούν μία συγκεκριμένη διαδικασία. Στην query εμείς καταργούμε την SWPredicate και ως εκ τούτου ελέγχουμε το states space για κόμβους οι οποίοι παραβιάζουν την διαδικασία.

```
val Length = length o ms_to_col ;
val total = Length (Mark.Sender´Send 1 1)

fun SWPredicate n =
  let
    val sent = total - (Length (Mark.Sender´Send 1 n)) ;
    val received = (Length (Mark.Receiver´Receiver 1 n))
  in
    ((sent = received) orelse (sent = received + 1))
  end ;

val nodelist = PredAllNodes (not o SWPredicate) ;
```

Σχήμα 3.8. Παράδειγμα μίας non-standard query

Για το state space του stop-and-wait πρωτοκόλλου το αποτέλεσμα της query είναι μία κενή λίστα. Ως εκ τούτου όλοι οι κόμβοι στο state space ικανοποιούν την SWPredicate, και εμείς αποδείξαμε ότι το stop-and-wait πρωτόκολλο είναι πειθαρχημένο.

3.9 Προχωρημένη state space ανάλυση

Σε αυτή την παράγραφο αποδείξαμε ότι η μέθοδος state space είναι ένας αποτελεσματικός τρόπος για να εξερευνήσουμε την δυναμική συμπεριφορά ενός συστήματος. Η κατασκευή και η ανάλυση του state space είναι εντελώς αυτοματοποιημένη. Από την state space αναφορά ο χρήστης κερδίζει μεγάλη και σημαντική γνώση σχετικά με την δυναμική συμπεριφορά του συστήματος. Η state space μέθοδος, ωστόσο, έχει κάποια μειονεκτήματα.

Το πιο σοβαρό μειονέκτημα της state space μεθόδου είναι το ονομαζόμενο state explosion problem. Για πολλά συστήματα, το state space γίνεται πολύ μεγάλο που δεν μπορεί να κατασκευαστεί πλήρως ακόμη και για μικρές ρυθμίσεις του συστήματος. Ένας αριθμός τεχνικών για την ανακούφιση του state explosion problem. Αυτό επιτυγχάνεται κατασκευάζοντας ένα συμπυκνωμένο state space από το οποίο είναι ακόμη δυνατό να επιβεβαιώσουμε του ίδιου είδους ιδιότητες συμπεριφοράς όπως στο πλήρες συνηθισμένο state space. Τα συμπυκνωμένα state spaces είναι συνήθως τάξεις μεγέθους μικρότερα από τα πλήρη state space. Μερικές από τις τεχνικές εκμεταλλεύονται την συμμετρία[19,22,23,24,25,26] και την ισοδυναμία[19,27] που παρουσιάζονται σε πολλά σύγχρονα και κατανεμημένα συστήματα. Άλλες τεχνικές εκμεταλλεύονται την ανεξαρτησία και τον συγχρονισμό ανάμεσα στα δεσμευτικά στοιχεία[28,29,30,31], ή εκμεταλλεύονται την ιεραρχική δομή του CPN μοντέλου[32,33].

Ένα άλλο μειονέκτημα της μεθόδου state space είναι το γεγονός ότι πάντα κατασκευάζεται για μία συγκεκριμένη αρχική σήμανση, η οποία συχνά αναφέρεται σε μία μόνο από τις πολλές ρυθμίσεις του συστήματος. Είδαμε παραπάνω ότι το πρωτόκολλο δουλεύει όταν έχουμε πέντε πακέτα δεδομένων – θα δουλεύει όμως για παραπάνω πακέτα; Θεωρητικά δεν μπορούμε να είμαστε σίγουροι. Ωστόσο, στην πράξη η κατάσταση δεν είναι τόσο άσχημη, εφόσον τα περισσότερα λάθη στον σχεδιασμό συστημάτων τείνουν να δηλώνουν την ύπαρξη τους εύκολα σε μικρές ρυθμίσεις του συστήματος. Ως εκ τούτου, είναι πολύ πιθανό ότι το πρωτόκολλο μας θα δουλεύει και με περισσότερα από πέντε πακέτα.

IV

Κεφάλαιο

Μοντελοποίηση και σύνθεση Υπηρεσιών

4 Μοντελοποίηση και σύνθεση Υπηρεσιών

4.1 Εισαγωγή

Με σκοπό να επιβιώσουν του ανταγωνισμού που δημιουργήθηκε από την νέα online οικονομία, πολλές εταιρίες σπεύδουν ταχέως να μεταφέρουν τον κύριο πυρήνα δραστηριοτήτων τους στο διαδίκτυο κάνοντας χρήση της τεχνολογίας των Web services. Η τεχνολογία των Web services έχει γίνει τελευταία πολύ δημοφιλής. Χαρακτηριστικά παραδείγματα υπηρεσιών των Web services περιλαμβάνουν online κρατήσεις θέσεων, διαχείριση σχέσεων με τον πελάτη (CRM), χρεώσεις λογαριασμών, αλυσίδα προμηθειών, κλπ. Με τον όρο Web service (ή απλά service) εννοούμε μία αυτόνομη εφαρμογή λογισμικού ή συνιστώσα, π.χ., μια σημασιολογικά καλώς ορισμένη λειτουργία, μοναδικά αναγνωρίσιμη από ένα URL.

Η ικανότητα να προβάλλονται και να χρησιμοποιούνται αποδοτικά και αποτελεσματικά υπηρεσίες στο Web είναι ένα κρίσιμο βήμα προς τη κατεύθυνση της ανάπτυξης της νέας online οικονομίας. Οι υπάρχουσες επιχειρήσεις θα δημιουργήσουν συμμαχίες και θα αναβαθμίσουν τις υπηρεσίες τους για να μοιραστούν το κόστος, τις ικανότητες, και τους πόρους προσφέροντας μία αναβαθμισμένης αξίας υπηρεσία σχηματίζοντας έτσι αυτό που είναι γνωστό ως B2B υπηρεσία. Επιγραμματικά, μία B2B υπηρεσία είναι συγκέντρωση κυρίως υπεργολαβικών υπηρεσιών που δουλεύουν παράλληλα για να επιτύχουν τους επιχειρηματικούς στόχους της επιχείρησης. Ένα παράδειγμα μίας αναβαθμισμένης B2B υπηρεσίας είναι ένα σύστημα οικονομικής διαχείρισης που χρησιμοποιεί μισθοδοσίες, φορολογικές προεπιλογές, και διαχείριση μετρητών ως συνιστώσες.

Σήμερα, η ανάπτυξη B2B υπηρεσιών χρησιμοποιεί την λογική της ad-hoc ανάπτυξης, καταναλώνει χρόνο και έχει αρκετά μεγάλο κόστος[34, 35]. Αυτή η εργασία είναι προφανώς κουραστική και δύσκολα προσπελάσιμη λόγω της αστάθειας και του μεγέθους του Web. Καθώς οι υπηρεσίες είναι περισσότερο αυτόνομες και ετερογενείς, η δημιουργία μίας B2B υπηρεσίας είναι δυσκολότερη. Συμπερασματικά, η γρήγορη και δυναμική σύνθεση των υπηρεσιών είναι μία ουσιώδης απαίτηση των οργανισμών για να προσαρμόσουν τις επιχειρηματικές πρακτικές τους στη δυναμική φύση του Web.

Όπως αναφέρθηκε προηγουμένως, προκειμένου το B2B ηλεκτρονικό εμπόριο να μπορέσει να προηχθεί, υπάρχει η ανάγκη να αναλύονται, να συνθέτονται και να εξελίσσονται ημιαυτόματα Web services σε ένα λογικό

χρονικό πλαίσιο. Από τη μία υπάρχουν οι ad-hoc λύσεις και η έλλειψη ενός κανονικού μοντέλου για την μοντελοποίηση και διαχείριση των Web services, από την άλλη, οι ad-hoc λύσεις έχουν εμποδίσει σε μεγάλο βαθμό την ταχύτερη αξιοποίηση των B2B υπηρεσιών. Οι σύγχρονες τεχνολογίες βασισμένες σε πρότυπα όπως το Universal Description, Discovery, και Integration (UDDI), στη Web Service Description Language (WSDL), και στο Simple Object Access Protocol (SOAP) δεν είναι ικανές να μοντελοποιήσουν όλες τις παραμέτρους μια υπηρεσίας, ως εκ τούτου παρέχουν περιορισμένη υποστήριξη στην σύνθεση υπηρεσιών. Το SOAP είναι ένα πρότυπο για την ανταλλαγή μηνυμάτων XML-τυποποίησης πάνω από HTTP μεταξύ εφαρμογών. Η WSDL είναι μία γενικού σκοπού XML γλώσσα για την περιγραφή του τι κάνει μία Web service, που τη βρίσκουμε, και πώς να την εμπλέξουμε. Το UDDI είναι ένα πρότυπο για την κοινοποίηση πληροφοριών σχετικά με Web services σε μία παγκόσμια καταχώρηση καθώς και για την ανακάλυψη μίας Web service.

Στη συνέχεια προτείνουμε ένα βασισμένο σε Petri-net αλγεβρικό μοντέλο για την μοντελοποίηση της ροής ελέγχου των Web services. Το μοντέλο είναι αρκετά εκφραστικό ώστε να αντιληφθεί τη σχηματοποίηση πολύπλοκων συνδυασμών/υπηρεσιών και τις αντίστοιχες εξειδικεύσεις.

4.2 Web services ως Petri nets

Η χρήση εποπτικών τεχνικών μοντελοποίησης όπως τα Petri net στο σχεδιασμό πολύπλοκων Web services είναι δικαιολογημένος από πολλούς λόγους. Για παράδειγμα, οι οπτικές αναπαραστάσεις παρέχουν μία υψηλού επιπέδου ακριβή γλώσσα η οποία επιτρέπει να εκφράσουμε και να αιτιολογήσουμε έννοιες στο φυσικό τους επίπεδο αφαίρεσης.

Η συμπεριφορά μίας Web service είναι βασικά ένα μερικά διατεταγμένο σύνολο λειτουργιών. Γι' αυτό, είναι εύκολο να το μοντελοποιήσεις σε ένα Petri net. **Οι λειτουργίες μοντελοποιούνται από transitions και η κατάσταση της υπηρεσίας μοντελοποιείται από places**. **Τα βέλη μεταξύ places και transitions χρησιμοποιούνται για να προσδιορίσουν αλληλεπιδράσεις (σχέσεις αιτιατού αποτελέσματος).**

Μπορούμε να κατηγοριοποιήσουμε τις Web services σε α) υλικές υπηρεσίες (π.χ., παράδοση φυσικών προϊόντων), β) υπηρεσίες πληροφόρησης (δημιουργία, επεξεργασία, παροχή, και διαχείριση πληροφορίας), και γ) η σύνθεση και των δύο σε υλικές/πληροφορίας υπηρεσίες.

Υποθέτουμε ότι ένα Petri net, το οποίο αναπαριστά την συμπεριφορά μίας υπηρεσίας, περιέχει ένα place εισόδου και ένα place εξόδου. Ένα Petri net με ένα place εισόδου, για την απορρόφηση της

πληροφορίας, και ένα place εξόδου για την μετάδοση της πληροφορίας, θα διευκολύνουν τον προσδιορισμό των σύνθετων λειτουργιών και την ανάλυση όσο και την επιβεβαίωση συγκεκριμένων ιδιοτήτων (π.χ., reachability, liveness). Σε ένα δοθέντα χρόνο, μία Web service μπορεί να ανήκει σε μία από τις παρακάτω καταστάσεις: **NotInstantiated, Ready, Running, Suspended, ή Completed**[36]. Όταν μία Web service είναι στην κατάσταση Ready, αυτό σημαίνει ότι ένα token είναι στον αντίστοιχο place εισόδου, από την άλλη η κατάσταση Completed σημαίνει ότι ένα token είναι στον αντίστοιχο place εξόδου.

Ένα δίκτυο από Web services είναι ένα ονομασμένο place/transition δίκτυο, π.χ., ένα tuple SN = {P, T, W, I, o, I} όπου:

- P είναι ένα πεπερασμένο σύνολο από places,
- T είναι ένα πεπερασμένο σύνολο από transitions που αναπαριστούν τις λειτουργίες της υπηρεσίας,
- $W \subseteq (P\times T)\cup (T\times P)$ είναι ένα σύνολο από προσανατολισμένα τόξα (σχέση ροής),
- i είναι ο place εισόδου με $°i = \{x\in P\cup T | (x,i) \in W\}=\varnothing$,
- o είναι ο place εξόδου με $o° = \{x\in P\cup T | (o,x) \in W\}=\varnothing$, και
- $I:T\rightarrow A\cup\{\tau\}$ είναι μία συνάρτηση ονοματοδοσίας όπου A είναι ένα σύνολο από ονόματα λειτουργιών. Υποθέτουμε ότι $\tau \notin A$ και δηλώνει μία σιωπηλή λειτουργία.

Η σχέση ροής W μπορεί επίσης να μεταφραστεί ως μία συνάρτηση $W: (P\times T)\cup (T\times P) \rightarrow\{0,1\}$. Οι σιωπηλές λειτουργίες είναι πυροδοτήσεις transitions οι οποίες δεν μπορούν να παρατηρηθούν. Χρησιμοποιούνται για την διάκριση μεταξύ εξωτερικής και εσωτερικής συμπεριφοράς της υπηρεσίας. Αξίζει να προσέξουμε ότι το Service net, που ορίστηκε προηγουμένως, είναι ένα κανονικό(ordinary) Petri net, το οποίο σημαίνει, ότι υπάρχει το πολύ ένα προσανατολισμένο βέλος που συνδέει ένα place με ένα transition ή ένα transition με ένα place.

Μία Web service είναι ένα tuple S = (NameS, Desc, Loc, URL, CS, SN) όπου:

- NameS είναι το όνομα της υπηρεσίας, που χρησιμοποιείται ως μοναδικός της προσδιορισμός,
- Desc είναι η περιγραφή της υπηρεσίας που παρέχεται. Συνοψίζει το τι προσφέρει η υπηρεσία,
- Loc είναι ο server στον οποίο εντοπίζεται η υπηρεσία,
- URL είναι η διεύθυνση της Web service,
- CS είναι το σύνολο από τις συνιστώσες υπηρεσίες. Αν CS={NameS} τότε S είναι μία βασική υπηρεσία. Αλλιώς S είναι μία σύνθετη υπηρεσία, και
- SN={P, T, W, I, o, I} είναι το service net που μοντελοποιεί την δυναμική συμπεριφορά της υπηρεσίας.

Ο place i αναφέρεται ως η αρχική σήμανση της υπηρεσίας S. Η εκτέλεση της S ξεκινά όταν ένα token είναι στον place i και τερματίζει όταν ένα token φθάνει τον place o.

4.3 Συνθέτοντας Web services

Μία Web service έχει μία συγκεκριμένη εργασία να διεκπεραιώσει και μπορεί, είτα να την φέρνει σε πέρας μόνη της, είτε να βασίζεται σε άλλες Web services, σε αυτή την περίπτωση έχουμε **σύνθετες Web services**. Για παράδειγμα, μία εταιρία που ασχολείται με την πώληση βιβλίων μπορεί να εστιάσει σε αυτή την πτυχή ενώ γίνεται εξωτερική ανάθεση σε άλλες πτυχές όπως η πληρωμή και η αποστολή. Η σύνθεση δύο ή περισσοτέρων υπηρεσιών δημιουργεί μία νέα υπηρεσία που παρέχει ταυτόχρονα και την αυθεντική ξεχωριστή λογική συμπεριφορά και μία νέα συνεργατική συμπεριφορά για να φέρει εις πέρας μία σύνθετη εργασία. **Αυτό σημαίνει ότι οι υπάρχουσες υπηρεσίες είναι ικανές να συνεργαστούν παρόλο που η συνεργασία δεν είχε σχεδιαστεί εκ των προτέρων. Η σύνθεση υπηρεσιών μπορεί να είναι στατική (οι συνιστώσες της υπηρεσίας αλληλεπιδρούν η μία με την άλλη με ένα προκαθορισμένο τρόπο) ή δυναμική (ανακαλύπτουν η μία την άλλη και διαπραγματεύονται on the fly).**

Παρακάτω θα παρουσιάσουμε μία άλγεβρα που επιτρέπει την δημιουργία καινούργιων value-added Web services χρησιμοποιώντας τις υπάρχουσες. Η ακολουθία, η εναλλαγή, η επανάληψη, και η τυχαία ακολουθία είναι συνηθισμένες κατασκευές που προσδιορίζονται στη ροή ελέγχου.

4.3.1 Άλγεβρα των Web services

Στη συνέχεια περιγράφονται η σύνταξη και η ανεπίσημη σημασιολογία των τελεστών της άλγεβρας των Web services. Οι κατασκευές επιλέχθηκαν ώστε να επιτρέπουν τον συνδυασμό κοινών και προχωρημένων Web services. Το σύνολο μπορεί να οριστεί από την ακόλουθη γραμματική σε BNF-like σημείωση:

$$S ::= \varepsilon \mid X \mid S \odot S \mid S \oplus S \mid S \Diamond S \mid \mu S \mid S \mid\mid_C S \mid (S \mid S) \rightarrow S \mid [S(p,q) : S(p,q)] \mid Ref(S,a,S)$$

όπου:

- το ε αντιπροσωπεύει μία κενή υπηρεσία, π.χ., μία υπηρεσία η οποία δεν κάνει καμία λειτουργία.
- το Χ αντιπροσωπεύει μία σταθερά υπηρεσία, που χρησιμοποιείται ως ατομική ή βασική υπηρεσία.
- το $S_1 \odot S_2$ αντιπροσωπεύει μία σύνθετη υπηρεσία η οποία εκτελεί την υπηρεσία S_1 ακολουθούμενη από την υπηρεσία S_2, π.χ., είναι ένας τελεστής ακολουθίας.
- το $S_1 \oplus S_2$ αντιπροσωπεύει μία σύνθετη υπηρεσία που συμπεριφέρεται είτε ως την υπηρεσία S_1 είτε ως την υπηρεσία S_2. Μόλις μία από τις δύο εκτελέσει την πρώτη της λειτουργία η δεύτερη υπηρεσία αποσυνδέεται, π.χ., ο $\oplus$ είναι ένας τελεστής εναλλαγής (ή επιλογής).
- το $S_1 \Diamond S_2$ αντιπροσωπεύει μία σύνθετη υπηρεσία η οποία είτε εκτελεί την υπηρεσία S_1 ακολουθούμενη από την S_2, ή την S_2 ακολουθούμενη από την S_1, π.χ., ο $\Diamond$ είναι ένας τελεστής τυχαίας ακολουθίας.
- το μS αντιπροσωπεύει μία υπηρεσία η οποία εκτελεί ένα συγκεκριμένο αριθμό την υπηρεσία S, π.χ., το μ αντιπροσωπεύει ένα επαναληπτικό τελεστή.
- το $S_1 ||_C S_2$ αντιπροσωπεύει μία σύνθετη υπηρεσία η οποία εκτελεί τις υπηρεσίες S_1 και S_2 ανεξάρτητα τη μία από την άλλη με πιθανότητες επικοινωνίας πάνω στο σύνολο C των ζευγών των εκτελέσεων, έτσι, ο $||_C$ είναι ένας παράλληλος τελεστής με επικοινωνία.
- το $(S_1 | S_2) \rightarrow S_3$ αντιπροσωπεύει μία σύνθετη υπηρεσία η οποία περιμένει για την εκτέλεση της μίας υπηρεσίας (μεταξύ των S_1 και S_2) πριν ενεργοποιήσει την επόμενη υπηρεσία S_3, π.χ., ο $\rightarrow$ είναι ένας τελεστής διάκρισης. Εδώ οι S_1 και S_2 εκτελούνται παράλληλα και χωρίς επικοινωνία.
- το $[S_1(p_1,q_1) : S_n(p_n,q_n)]$ είναι μία σύνθετη υπηρεσία η οποία δυναμικά επιλέγει ένα πάροχο υπηρεσιών ανάμεσα από τις n διαθέσιμες $S_1, \ldots, S_n$ και την εκτελεί. Συμπεριφέρεται ως εξής: πρώτα μία αίτηση στέλνεται από ένα συνθέτη στους n διαθέσιμους παρόχους υπηρεσιών μίας δοθείσας εμπορικής κοινότητας μέσω των εισόδων των σημείων πρόσβασης $p_1, \ldots, p_n$. Μετά βασιζόμενος στις ληφθέντες απαντήσεις, από τις εξόδους των σημείων πρόσβασης $q_1, \ldots, q_n$, και σύμφωνα με τα δοθέντα κριτήρια κατάταξης (π.χ., η τιμή, ο χρόνος παράδοσης, ή ένας συνδυασμός και των δύο) ο καλύτερος πάροχος υπηρεσιών επιλέγεται. Τελικά οι απαραίτητες λειτουργίες εκτελούνται. Ο [:] είναι ένας τελεστής επιλογής.
- το $Ref(S_1, a, S_2)$ αντιπροσωπεύει μία σύνθετη υπηρεσία η οποία συμπεριφέρεται σαν την S_1 από τις λειτουργίες στην S_1 με ετικέτα a οι οποίες αντικαθιστώνται με μία μη κενή υπηρεσία S_2. Ο Ref είναι ένας τελεστής τελειοποίησης.

Η προτεινόμενη άλγεβρα επαληθεύει την closure ιδιότητα. Εγγυάται ότι στο κάθε αποτέλεσμα κάθε λειτουργίας στις υπηρεσίες είναι μία υπηρεσία στην οποία πάλι μπορούμε να εφαρμόσουμε τους αλγεβρικούς τελεστές. Έτσι είμαστε ικανοί να χτίσουμε πιο πολύπλοκες υπηρεσίες αθροίζοντας και χρησιμοποιώντας ξανά υπάρχουσες υπηρεσίες μέσω δηλωτικών εκφράσεων της άλγεβρας των υπηρεσιών.

4.3.2 Τυπικές σημασιολογίες

Παρακάτω δίνεται ο τυπικός ορισμός, σε όρους των Petri nets, από τους τελεστές σύνθεσης. Έστω S_i = ($NameS_i$, $Desc_i$, Loc_i, URL_i, CS_i, SN_i) με SN_i = (P_i, T_i, W_i, i_i, o_i, I_i) για i=1, . . ., n και n είναι οι Web services τέτοιο ώστε $P_i \cup P_j = \varnothing$ και $T_i \cup T_j = \varnothing$ για $i \neq j$.

Είναι σημαντικό να σημειώσουμε ότι η σύνθεση υπηρεσιών, όπως περιγράφεται και παρακάτω, εφαρμόζεται σε συντακτικά διαφορετικές υπηρεσίες. Αυτό γίνεται λόγω του ότι οι places και οι transitions των συνιστωσών υπηρεσιών πρέπει να κομματιαστούν για κατάλληλη σύνθεση. Ωστόσο, μία υπηρεσία μπορεί να συντεθεί με τον εαυτό της. Συνήθως, αυτό συμβαίνει όταν οι υπηρεσίες περιγράφουν μεταβλητές της ίδιας λειτουργίας (π.χ., κανονική εκτέλεση και εξαιρετικές καταστάσεις) ή, για παράδειγμα, αν ένας απλός προμηθευτής προσφέρει δύο διαφορετικά αγαθά, οι αιτήσεις μπορεί να χειριστούν ανεξάρτητα, σαν να ήταν από δύο διαφορετικούς προμηθευτές. Σε αυτή την περίπτωση, η επικάλυψη πρέπει να επιλυθεί πριν την σύνθεση. Αυτό μπορεί να γίνει μετονομάζοντας τα σύνολα P και T της μίας από τις δύο ίσες υπηρεσίες. Οι δύο υπηρεσίες παραμένουν ίσες όσο αφορά τον ισομορφισμό στους places και transitions. Επίσης, στην περίπτωση σιωπηλών λειτουργιών, αναπαριστούμε γραφικά τους αντίστοιχους transitions με μαύρα ορθογώνια.

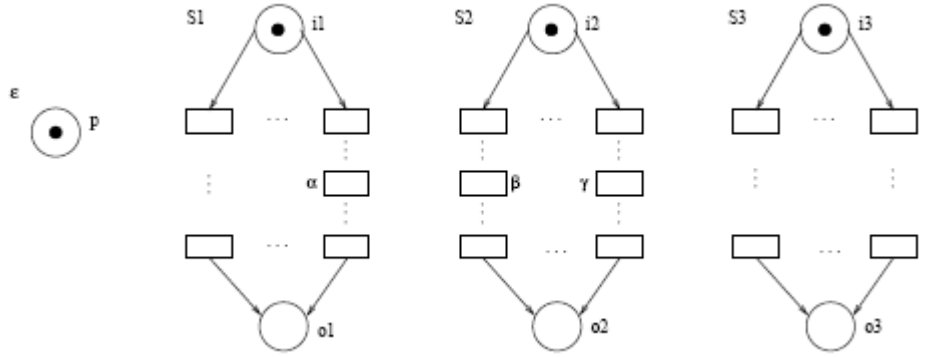
4.3.3 Βασικές δομές

<u>Κενή υπηρεσία</u>: Η κενή υπηρεσία ε είναι μία υπηρεσία η οποία δεν πραγματοποιεί καμία λειτουργία. Χρησιμοποιείται για τεχνικούς και θεωρητικούς λόγους.

Η κενή υπηρεσία ε ορίζεται ως ε=(NameS, Desc, Loc, URL, CS, SN) όπου :

- NameS = Empty,
- Desc = "Κενή Web service",
- Loc = Null, δηλώνει ότι δεν υπάρχει server για αυτή την υπηρεσία,
- URL = Null, δηλώνει ότι δεν υπάρχει URL για αυτή την ε,
- CS = {Empty}, και
- SN =({p}, ∅, ∅, p, p, ∅).

Γραφικά, η ε αντιπροσωπεύεται από το Petri net του σχήματος 4.1 που περιέχει μόνο ένα place.



Σχήμα 4.1. Υπηρεσίες ε, S_1, S_2, και S_3

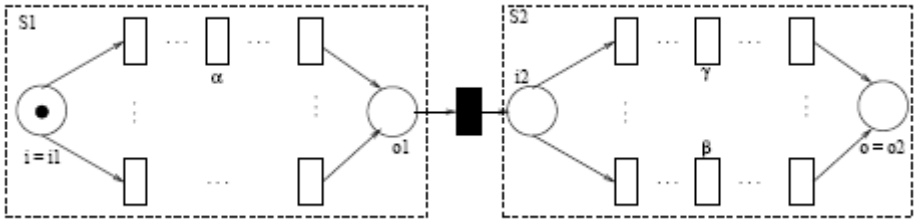
Ακολουθία: Ο τελεστής ακολουθίας επιτρέπει την εκτέλεση δύο υπηρεσιών S_1 και S_2 σε σειρά, δηλαδή η μία μετά την άλλη. Η S_1 πρέπει να ολοκληρωθεί πριν να μπορεί να ξεκινήσει η S_2. Αυτή είναι συνήθως η περίπτωση όπου μία υπηρεσία εξαρτάται από την έξοδο της προηγούμενης υπηρεσίας. Για παράδειγμα, η υπηρεσία Πληρωμή (Payment) εκτελείται μετά την ολοκλήρωση της υπηρεσίας Παράδοση (Delivery).

Η υπηρεσία $S_1 \odot S_2$ ορίζεται ως $S_1 \odot S_2$ = (NameS, Desc, Loc, URL, CS, SN) όπου :

- NameS είναι το όνομα της καινούργιας υπηρεσίας,
- Desc είναι η περιγραφή της καινούργιας υπηρεσίας,
- Loc είναι η τοποθεσία της καινούργιας υπηρεσίας (μπορεί να είναι στον ίδιο server με μία από τις δύο συνιστώσες υπηρεσίες),
- URL είναι η διεύθυνση της καινούργιας υπηρεσίας,
- $CS = CS_1 \cup CS_2$, και
- SN =(P, T, W, I, o, I) όπου:

- $P = P_1 \cup P_2$,
- $T = T_1 \cup T_2 \cup \{t\}$,
- $W = W_1 \cup W_2 \cup \{(o_1, t), (t, i_2)\}$,
- $i = i_1$,
- $o = o_2$, και
- $I = I_1 \cup I_2 \cup \{t, \tau\}$.

Γραφικά, δίνοντας τις S_1 και S_2 (βλέπε σχήμα 4.1), η $S_1 \odot S_2$ αντιπροσωπεύεται από το Petri net που δείχνεται στο σχήμα 4.2.



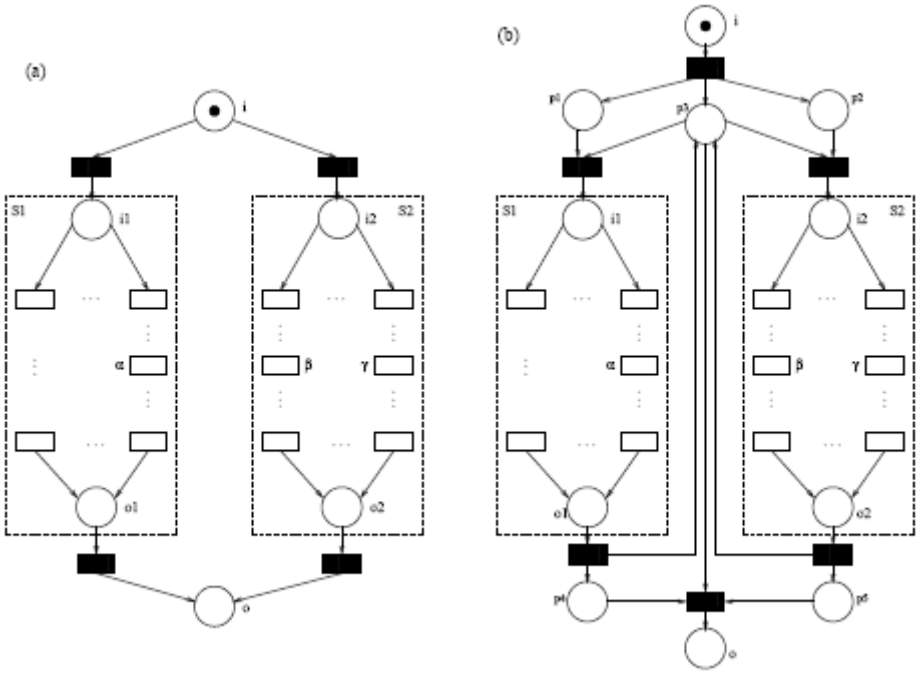
Σχήμα 4.2. Η υπηρεσία $S_1 \odot S_2$

Εναλλαγή: Ο τελεστής εναλλαγής επιτρέπει, έχοντας στην διάθεση μας δύο υπηρεσίες S_1 και S_2, να μοντελοποιήσουμε την εκτέλεση ή της S_1 ή της S_2,αλλά όχι και των δύο. Για παράδειγμα, η υπηρεσία access_claim ακολουθείται είτε από την υπηρεσία indemnify_customer ή από την υπηρεσία convoke_customer.

Η υπηρεσία $S_1 \oplus S_2$ ορίζεται ως $S_1 \oplus S_2$ = (NameS, Desc, Loc, URL, CS, SN) όπου :

- NameS είναι το όνομα της καινούργιας υπηρεσίας,
- Desc είναι η περιγραφή της καινούργιας υπηρεσίας,
- Loc είναι η τοποθεσία της καινούργιας υπηρεσίας,
- URL είναι η διεύθυνση της καινούργιας υπηρεσίας,
- $CS = CS_1 \cup CS_2$, και
- SN =(P, T, W, i, o, I) όπου:
  - $P = P_1 \cup P_2 \cup \{i, o\}$,
  - $T = T_1 \cup T_2 \cup \{t_{i1}, t_{i2}, t_{o1}, t_{o2}\}$,
  - $W = W_1 \cup W_2 \cup \{(i, t_{i1}), (i, t_{i2}), (t_{i1}, i_1), (t_{i2}, i_2), (o_1, t_{o1}), (o_2, t_{o2}), (t_{o1}, o), (t_{o2}, o)\}$, και
  - $I = I_1 \cup I_2 \cup \{(t_{i1}, \tau), (t_{i1}, \tau), (t_{o1}, \tau), (t_{o2}, \tau)\}$.

Γραφικά, δίνοντας τις S_1 και S_2 (βλέπε σχήμα 4.1), η $S_1 \oplus S_2$ αντιπροσωπεύεται από το Petri net που δείχνεται στο σχήμα 4.3(α).



Σχήμα 4.3. Υπηρεσίες (a) $S_1 \oplus S_2$ και (b) $S_1 \Diamond S_2$

Τυχαία ακολουθία: Ο τελεστής τυχαίας ακολουθίας προσδιορίζει την εκτέλεση δύο υπηρεσιών οι οποίες δεν πρέπει να εκτελούνται ταυτόχρονα, αυτό σημαίνει δηλαδή, όταν έχουμε δύο υπηρεσίες S_1 και S_2, έχουμε την S_1 ακολουθούμενη από την S_2 ή την S_2 ακολουθούμενη από την S_1. Υποθέτουμε, για παράδειγμα, ότι υπάρχουν δύο αγαθά, η απορρόφηση ενός απλού αγαθού δεν είναι χρήσιμη εκτός εάν οι υπόλοιποι από τους συνδέσμους μπορούν και αυτοί να απορροφηθούν. Επιπλέον, χωρίς μία προθεσμία, δεν υπάρχει κέρδος να εκτελέσουμε τις δύο αιτήσεις παράλληλα, και κάνοντας αυτό μπορεί να οδηγηθούμε σε μη απαραίτητα κόστη αν ένας από τους συνδέσμους δεν είναι διαθέσιμος ή είναι δυσεύρετος. Για αυτό, η βέλτιστη εκτέλεση είναι απαραίτητα μία τυχαία σειριακή παράταξη από αιτήσεις σε προμηθευτές.

Η υπηρεσία $S_1 \Diamond S_2$ ορίζεται ως $S_1 \Diamond S_2$ = (NameS, Desc, Loc, URL, CS, SN) όπου :

- NameS είναι το όνομα της σύνθετης υπηρεσίας,
- Desc είναι η περιγραφή της σύνθετης υπηρεσίας,
- Loc είναι ο server όπου εντοπίζεται η σύνθετη υπηρεσία,
- URL είναι η διεύθυνση της σύνθετης υπηρεσίας,
- $CS = CS_1 \cup CS_2$, και

- SN =(P, T, W, i, o, l) όπου:
  - $P = P_1 \cup P_2 \cup \{i, o, p_1, p_2, p_3, p_4, p_5\}$,
  - $T = T_1 \cup T_2 \cup \{t_i, t_1, t_2, t_3, t_4, t_o\}$,
  - $W = W_1 \cup W_2 \cup \{(i, t_i), (t_i, p_1), (t_i, p_2), (t_i, p_3), (p_1, t_1), (p_2, t_2), (p_3, t_1), (p_3, t_2), (p_3, t_o), (t_1, i_1), (t_2, i_2), (o_1, t_3), (o_2, t_4), (t_3, p_3), (t_4, p_3), (t_3, p_4), (t_4, p_5), (p_4, t_o), (p_5, t_o), (t_o, o)\}$, και
  - $l = l_1 \cup l_2 \cup \{(t_i, \tau), (t_1, \tau), (t_2, \tau), (t_3, \tau), (t_4, \tau), (t_o, \tau)\}$.

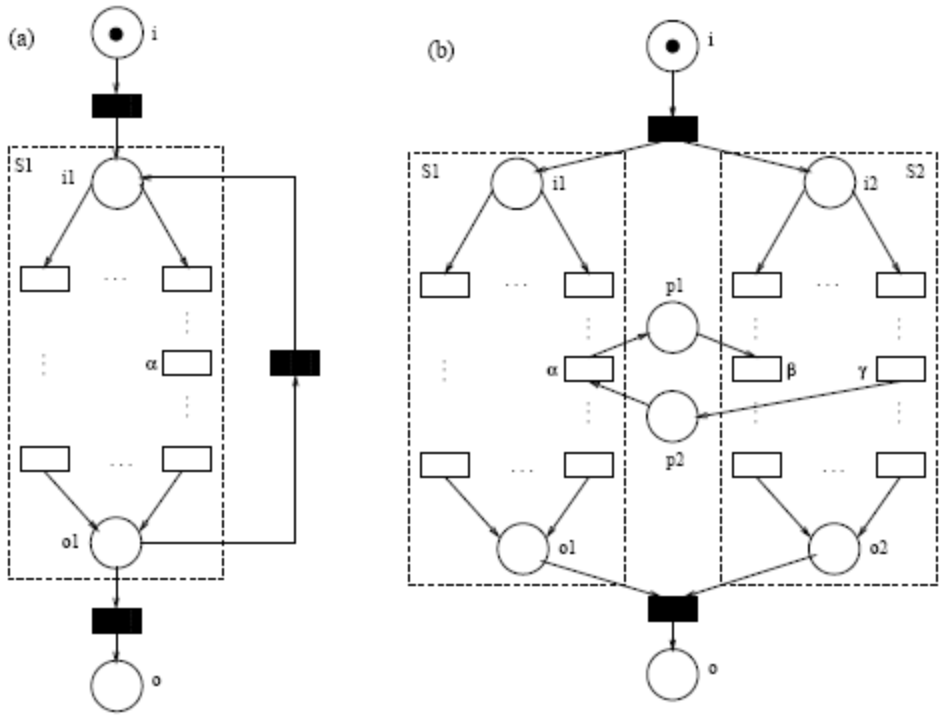
Γραφικά, δίνοντας τις S_1 και S_2 (βλέπε σχήμα 1), η $S_1 \Diamond S_2$ αντιπροσωπεύεται από το Petri net που δείχνεται στο σχήμα 4.3(b).

Επανάληψη: Ο τελεστής επανάληψης μοντελοποιεί την εκτέλεση μίας υπηρεσίας ακολουθούμενη ένα συγκεκριμένο αριθμό από επαναλήψεις της ίδιας. Τυπικά παραδείγματα όπου η επανάληψη είναι απαραίτητη είναι η επικοινωνία κα ο έλεγχος ποιότητας όπου οι υπηρεσίες εκτελούνται πάνω από μία φορά.

Η υπηρεσία μS_1 ορίζεται ως μS_1 = (NameS, Desc, Loc, URL, CS, SN) όπου :

- NameS είναι το όνομα της καινούργιας υπηρεσίας,
- Desc είναι η περιγραφή της καινούργιας υπηρεσίας,
- Loc είναι η τοποθεσία της καινούργιας υπηρεσίας,
- URL είναι η διεύθυνση της καινούργιας υπηρεσίας,
- $CS = CS_1$, και
- SN =(P, T, W, i, o, l) όπου:
  - $P = P_1 \cup \{i, o\}$,
  - $T = T_1 \cup \{t_i, t_o, t\}$,
  - $W = W_1 \cup \{(i, t_i), (t_i, i_1), (o_1, t_o), (t_o, o), (o_1, t), (t, i_1)\}$, και
  - $l = l_1 \cup \{(t_i, \tau), (t_{i1}, \tau), (t_o, \tau), (t, \tau)\}$.

Γραφικά, δίνοντας την S_1 (βλέπε σχήμα 4.1), η μS_1 αντιπροσωπεύεται από το Petri net που δείχνεται στο σχήμα 4.4(a).



Σχήμα 4.4. Οι υπηρεσίες (a) μS_1 και (b) $S_1 ||_C S_2$

4.3.4 Προχωρημένες δομές

Παραλληλισμός με επικοινωνία: Ο τελεστής παραλληλισμού αντιπροσωπεύει την ταυτόχρονη εκτέλεση δύο υπηρεσιών. Ταυτόχρονες υπηρεσίες μπορεί να συγχρονίζονται και να ανταλλάσουν πληροφορίες. Για παράδειγμα, σε μία υπηρεσία Online κατάστημα υπολογιστών, μετά την λήψη μίας παραγγελίας ενός υπολογιστή από ένα πελάτη, δύο παράλληλες υπηρεσίες ενεργοποιούνται: η Order Monitor και η Order Processor.

Έστω $C = \{(\alpha, \beta) \mid (\beta, \alpha) \in T_1 \times T_2 \cup T_2 \times T_1\}$ είναι ένα σύνολο από στοιχεία επικοινωνίας. Η υπηρεσία $S_1 ||_C S_2$ ορίζεται ως $S_1 ||_C S_2 = (NameS, Desc, Loc, URL, CS, SN)$ όπου :

- NameS είναι το όνομα της σύνθετης υπηρεσίας,
- Desc είναι η περιγραφή της σύνθετης υπηρεσίας,
- Loc είναι ο server όπου εντοπίζεται η σύνθετη υπηρεσία,
- URL είναι η διεύθυνση της σύνθετης υπηρεσίας,
- $CS = CS_1 \cup CS_2$, και

- SN =(P, T, W, i, o, l) όπου:
  - $P = P_1 \cup P_2 \cup \{i, o\} \cup \{p_i / (\alpha_i, \beta_i) \in C\}$,
  - $T = T_1 \cup T_2 \cup \{t_i, t_o\}$,
  - $W = W_1 \cup W_2 \cup \{(i, t_i), (t_i, i_1), (t_i, i_2), (o_1, t_o), (o_2, t_o), (t_o, o)\} \cup \{(\alpha_i, p_i), (p_i, \beta_i) / (\alpha_i, \beta_i) \in C\}$, και
  - $l = l_1 \cup l_2 \cup \{(t_i, \tau), (t_o, \tau)\}$.

Γραφικά, δίνοντας τις S_1 και S_2 (βλέπε σχήμα 4.1), και $C = \{(\alpha, \beta), (\gamma, \alpha)\}$, η $S_1 \mid\mid_C S_2$ αντιπροσωπεύεται από το Petri net που δείχνεται στο σχήμα 4.4(b).

Διάκριση: Οι υπηρεσίες διαδικτύου είναι αναξιόπιστες, έχουν μία σχετικά μεγάλη πιθανότητα να αποτύχουν ή να είναι απαράδεκτα αργές. Οι καθυστερήσεις μόνο λίγων δευτερολέπτων μπορεί να έχουν ως αποτέλεσμα για τους παρόχους την απώλεια σημαντικών χρηματικών ποσών ή να απογοητεύσουν τους πελάτες τους. Διαφορετικοί πάροχοι υπηρεσιών μπορεί να παρέχουν τις ίδιες ή παρόμοιες υπηρεσίες. Για αυτό, πρέπει να είναι δυνατό να συνδυάσουμε μη αξιόπιστες υπηρεσίες για να λάβουμε πιο αξιόπιστες υπηρεσίες. Ο τελεστής διάκρισης χρησιμοποιείται, για παράδειγμα, να τοποθετήσει περιττές παραγγελίες σε διαφορετικούς παρόχους που προσφέρουν την ίδια υπηρεσία για να αυξήσει την αξιοπιστία. Η πρώτη που θα εκτελέσει την απαιτούμενη υπηρεσία ενεργοποιεί την μεταγενέστερη υπηρεσία και οι υπόλοιπες καθυστερημένες αποκρίσεις αγνοούνται για την υπόλοιπη επεξεργασία της σύνθετης υπηρεσίας[37].

Η υπηρεσία $(S_1 \mid S_2) \rightarrow S_3$ ορίζεται ως $(S_1 \mid S_2) \rightarrow S_3$ = (NameS, Desc, Loc, URL, CS, SN) όπου :

- NameS είναι το όνομα της καινούργιας υπηρεσίας,
- Desc είναι η περιγραφή της καινούργιας υπηρεσίας,
- Loc είναι η τοποθεσία της καινούργιας υπηρεσίας,
- URL είναι η διεύθυνση της καινούργιας υπηρεσίας,
- $CS = CS_1 \cup CS_2 \cup CS_3$, και
- SN =(P, T, W, i, o, l) όπου:
  - $P = P_1 \cup P_2 \cup P_3 \cup \{i, o, p_1, p_2\}$,
  - $T = T_1 \cup T_2 \cup T_3 \cup \{t_i, t_1, t_2, t_3, t_o\}$,
  - $W = W_1 \cup W_2 \cup W_3 \cup \{(i, t_i), (t_i, i_1), (t_i, i_2), (t_i, p_2), (o_1, t_1), (o_2, t_2), (t_1, p_1), (t_2, p_1), (p_1, t_3), (p_1, t_o), (p_2, t_3), (t_3, i_3), (o_3, t_o), (t_o, o)\}$, και
  - $l = l_1 \cup l_2 \cup l_3 \cup \{(t_i, \tau), (t_1, \tau), (t_2, \tau), (t_3, \tau), (t_o, \tau),\}$.

Γραφικά, δίνοντας τις S_1, S_2, και S_3 (βλέπε σχήμα 4.1), η $(S_1 | S_2) \rightarrow S_3$ αντιπροσωπεύεται από το Petri net που δείχνεται στο σχήμα 4. 5(a).

Επιλογή: Στηριζόμενη σε ένα μόνο πάροχο θέτει την εταιρία στο έλεος του. Για να μειωθεί ο κίνδυνος, μία εταιρία πρέπει να διατηρεί σχέσεις με πολλαπλούς παρόχους. Αυτοί οι πάροχοι μπορεί, π.χ., να έχουν διαφορετικές τιμές, να προτείνουν διαφορετικές ημερομηνίες και φορές παράδοσης, και να έχουν διαφορετική αξιοπιστία. Η κατασκευή της επιλογής επιτρέπει να διαλέξουμε την καλύτερη παροχή υπηρεσίας, χρησιμοποιώντας ένα κριτήριο κατάταξης, ανάμεσα σε διάφορους ανταγωνιστικούς παρόχους για αναθέσει μία συγκεκριμένη λειτουργία.

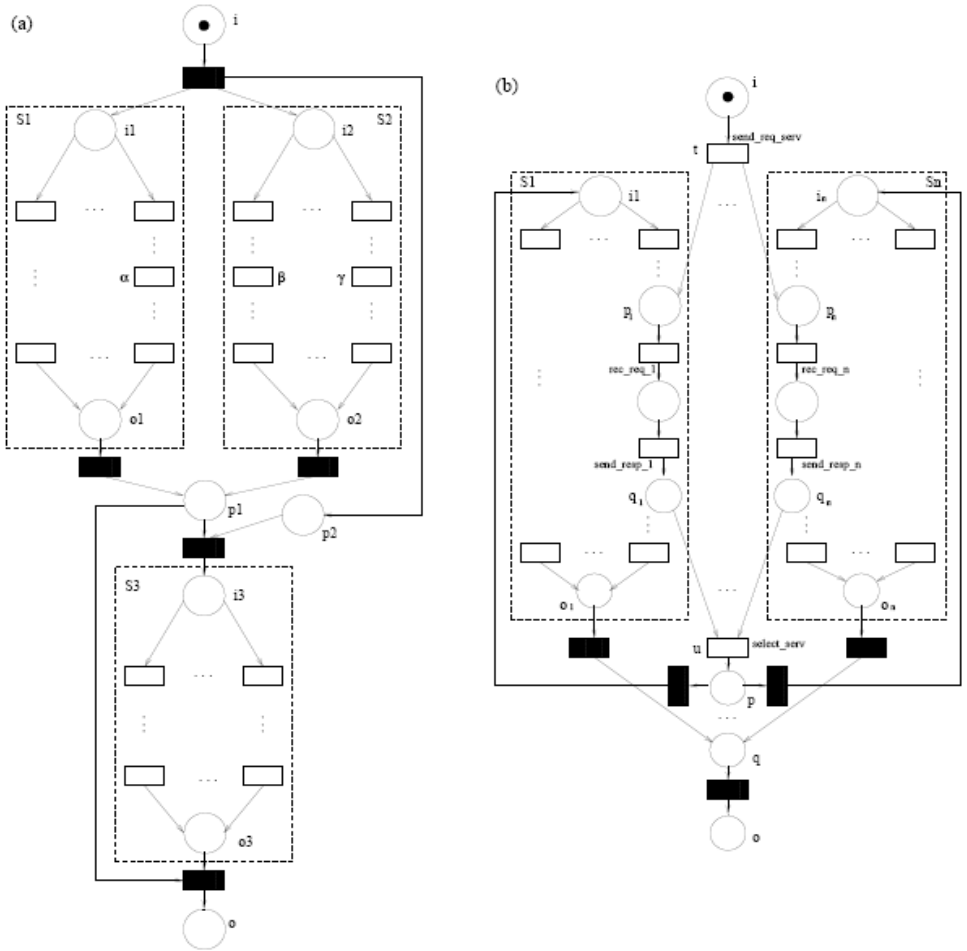
Έστω $p_i, q_i \in P_i$ είναι τα σημεία πρόσβασης της υπηρεσίας από Si για i = 1, . ., n. Η υπηρεσία $[S_1(p_1,q_1) : S_n(p_n,q_n)]$ ορίζεται ως $[S_1(p_1,q_1) : S_n(p_n,q_n)] = (NameS, Desc, Loc, URL, CS, SN)$ όπου :

- NameS είναι το όνομα της σύνθετης υπηρεσίας,
- Desc είναι η περιγραφή της σύνθετης υπηρεσίας,
- Loc είναι ο server όπου εντοπίζεται η σύνθετη υπηρεσία,
- URL είναι η διεύθυνση της σύνθετης υπηρεσίας,
- $CS = \bigcup_{i=1}^{n} CS_i$, και
- SN =(P, T, W, i, o, I) όπου:
  - $P = \bigcup_{i=1}^{n} P_i \cup \{i, o, p, q\}$,
  - $T = \bigcup_{i=1}^{n} T_i \cup \{t, u, v\} \cup \{t_i, t'_i \mid 1 \le i \le n\}$,
  - $W = \bigcup_{i=1}^{n} W_i \cup \{(i, t), (u, p), (q, v), (v, o)\} \cup \{(t, p_j),$ $(q_j, u), (p, t'_j), (t'_j, i_j), (o_j, t_j), (t_j, q) \mid 1 \le j \le n\}$, και
  - $I = \bigcup_{i=1}^{n} I_i \cup \{(t, send\_req\_serv), (u, select\_serv), (u, \tau)\} \cup$ $\{(t_i, \tau), (t'_i, \tau) \mid 1 \le i \le n\}$.

Γραφικά, δίνοντας $S_1, \ldots, S_n$, και $p_i, q_i \in P_i$ για i = 1, . ., n, η $[S_1(p_1,q_1) : S_n(p_n,q_n)]$ αντιπροσωπεύεται από το Petri net που δείχνεται στο σχήμα 4.5(b). Για να επιτρέψουμε την ανακάλυψη της υπηρεσίας, υποθέτουμε ότι οι πάροχοι υπηρεσιών τοποθετούν τις υπηρεσίες τους σε μία συγκεκριμένη κοινότητα εμπορίου[34, 35] και η πληροφορία σχετικά με τις διαθέσιμες υπηρεσίες είναι γνωστή. Επίσης υποθέτουμε ότι κάθε υπηρεσία που παρέχεται περιέχει δύο διακριτά μέρη, ένα μέρος για την επεξεργασία την

αίτηση της υπηρεσίας και ένα άλλο μέρος για να εκτελέσει την υπηρεσία αυτή. Το ζευγάρι των λειτουργιών send_req_serv και select_serv αντιπροσωπεύει την παραπάνω ορισμένη στρατηγική επιλογής που επιλέχθηκε, μεταξύ άλλων, από τον σχεδιαστή της σύνθετης υπηρεσίας.

Η απόφαση επιλογής μπορεί επίσης να βασίζεται σε αυτόματες διαπραγματεύσεις και δημοπρασίες (π.χ., σφραγισμένες προσφορές και ανοικτές δημοπρασίες) όπου οι μετέχοντες, οι οποίοι είναι, ο συνθέτης και οι πάροχοι υπηρεσιών, είναι αυτοματοποιημένοι. Κάνοντας αυτό, οι αυτοματοποιημένες online δημοπρασίες γίνονται ένα βασικό δομικό στοιχείο για την απόφαση επιλογής.



Σχήμα 4.5. Οι υπηρεσίες (a) $(S_1 \mid S_2) \rightarrow S_3$ και (b) $[S_1(p_1,q_1) : S_n(p_n,q_n)]$

Τελειοποίηση: Η κατασκευή τελειοποίησης, στην οποία λειτουργίες αντικαθίστανται από πιο λεπτομερής μη κενές υπηρεσίες, χρησιμοποιείται για να εισάγουμε πρόσθετες συνιστώσες υπηρεσιών σε μία υπηρεσία. Η τελειοποίηση είναι ο μεταμόρφωση ενός σχεδίου από μία υψηλού επιπέδου

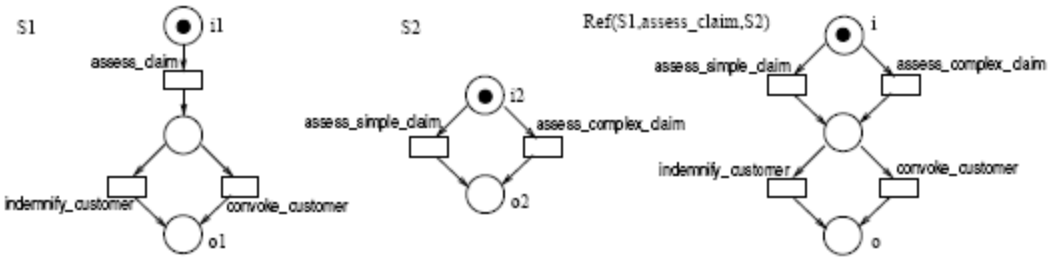
abstract μορφή σε μία χαμηλότερου επιπέδου πιο συμπαγή μορφή και ως εκ τούτου επιτρέποντας την ιεραρχική μοντελοποίηση.

Έστω $a \in A$. Η υπηρεσία $Ref(S_1, a, S_2)$ ορίζεται ως $Ref(S_1, a, S_2) = (NameS, Desc, Loc, URL, CS, SN)$ όπου :

- NameS είναι το όνομα της βελτιωμένης υπηρεσίας,
- Desc είναι η περιγραφή της βελτιωμένης υπηρεσίας,
- Loc είναι η τοποθεσία της βελτιωμένης υπηρεσίας,
- URL είναι η διεύθυνση της βελτιωμένης υπηρεσίας,
- $CS = \begin{cases} CS1 \cup CS2 & \text{if } a \in l1(T1) \\ CS1 & \text{otherwise} \end{cases}$, και
- $SN = (P, T, W, i, o, l)$ όπου:
  - $P = P_1 \cup \{<p,t> \mid p \in P_2 \setminus \{i_2, o_2\}, t \in l_1^{-1}(\alpha)\}$,
  - $T = T_1 \setminus l_1^{-1}(a) \cup \{<x,t> \mid x \in T_2, t \in l_1^{-1}(\alpha)\}$,
  - $W(x,y) = \begin{cases} W_1(x,y) & \text{if } x,y \in P_1 \cup T_1 \setminus l_1^{-1}(\alpha) \\ W_2(x',y') & \text{if } x=<x',t>,\ y=<y',t> \quad \text{for } t \in l_1^{-1}(\alpha) \\ W_1(x,t) & \text{if } y=<y',t>,\ x \in {}^\bullet t \quad \text{for } t \in l_1^{-1}(\alpha) \\ & \quad \text{και } y' \in i_2^\bullet \\ W_1(t,y) & \text{if } x=<x',t>,\ y \in t^\bullet \quad \text{for } t \in l_1^{-1}(\alpha) \\ & \quad \text{και } x' \in {}^\bullet o_2 \\ 0 & \text{otherwise} \end{cases}$,
  - $i = i_1$,
  - $o = o_1$, και
  - $l(t) = \begin{cases} l_1(t) & \text{if } t \in T_1 \setminus l_1^{-1}(a) \\ l_2(x) & \text{if } t=<x,t'> \text{ for } x \in T_2 \text{ και } t' \in l_1^{-1}(\alpha) \end{cases}$.

Το σχήμα 4.6 δείχνει ένα παράδειγμα μίας βελτιωμένης υπηρεσίας όπου η S_2 (παίζοντας το ρόλο είτε της assess_simple_claim, είτε της assess_complex_claim) είναι μία τελειοποίηση (π.χ., εξειδίκευση) της γενικής λειτουργίας assess_claim στην S_1.

Μία άλλη ενδιαφέρουσα χρήση του τελεστή τελειοποίησης είναι ο συνδυασμός του με τον τελεστή επιλογής για βέλτιστες εξωτερικές λειτουργίες. Στην $Ref(S_1, assess\_claim, [A_1:A_n])$, για παράδειγμα, η assess_claim διενεργείται από την καταλληλότερη υπηρεσία διαλεγμένη ανάμεσα σε n διαθέσιμους αξιολογητές.



Σχήμα 4.6.Ένα παράδειγμα τελειοποίησης

Η σύνθετη Web service που δημιουργήθηκε με την χρήση των παραπάνω κατασκευών είναι μία υπηρεσία δικτύου η οποία έχει δύο ειδικούς places, οι οποίοι αντιστοιχούν στην έναρξη και τον τερματισμό της επεξεργασίας της σύνθετης Web service. Αυτή η ενδιαφέρουσα ιδιότητα σχετίζεται με την δομή του υποκείμενου Petri net.

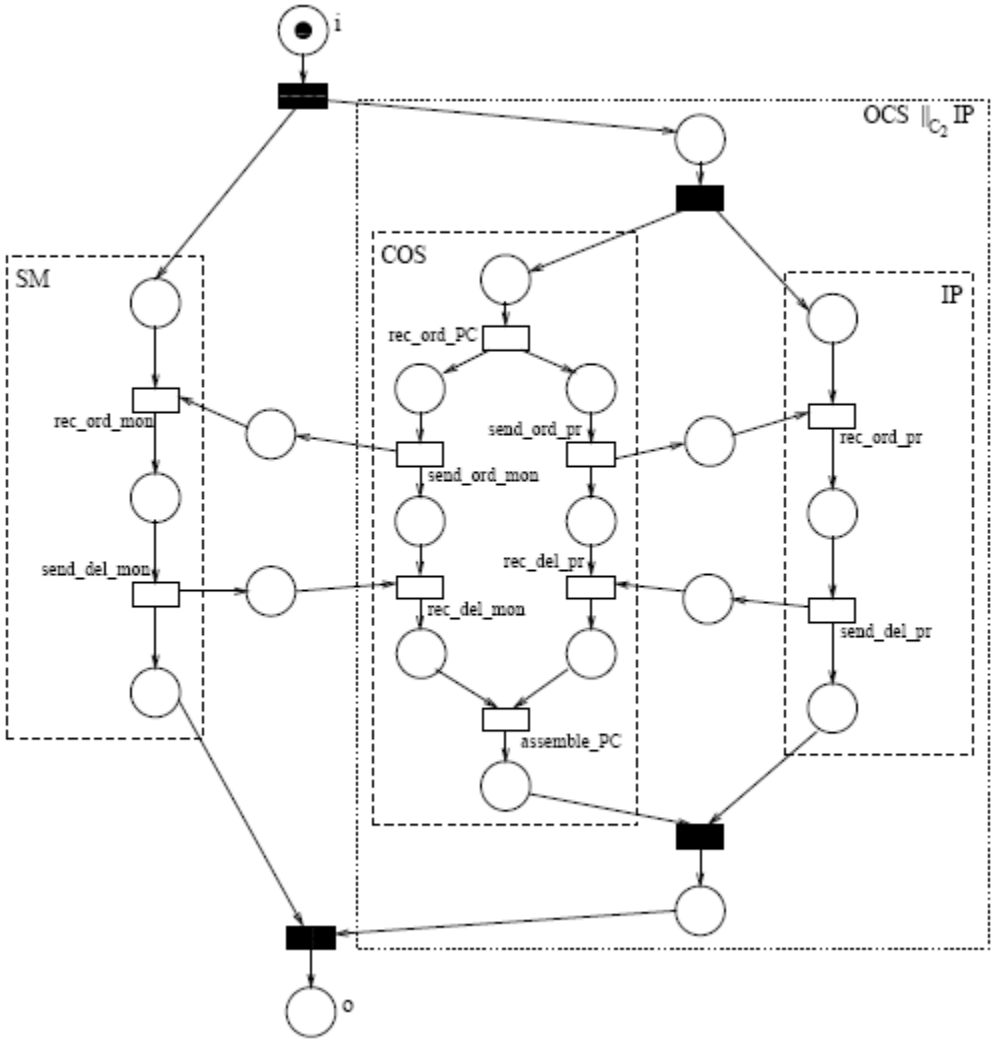
Η υπηρεσία δικτύου μίας σύνθετης Web service S λαμβάνεται χρησιμοποιώντας τις παραπάνω ορισμένες κατασκευές που περιέχουν ένα place εισόδου i και ένα place εξόδου o.

Αυτό είναι άμεση συνέπεια του ορισμού της Web service και των ορισμών των σύνθετων κατασκευών που ορίστηκαν προηγουμένως.

Παρακάτω, δίνουμε ένα παράδειγμα μίας σύνθετης υπηρεσίας μοντελοποιημένη ως Petri net για απεικονίσουμε μερικές από τις παραπάνω ορισμένες δομές.

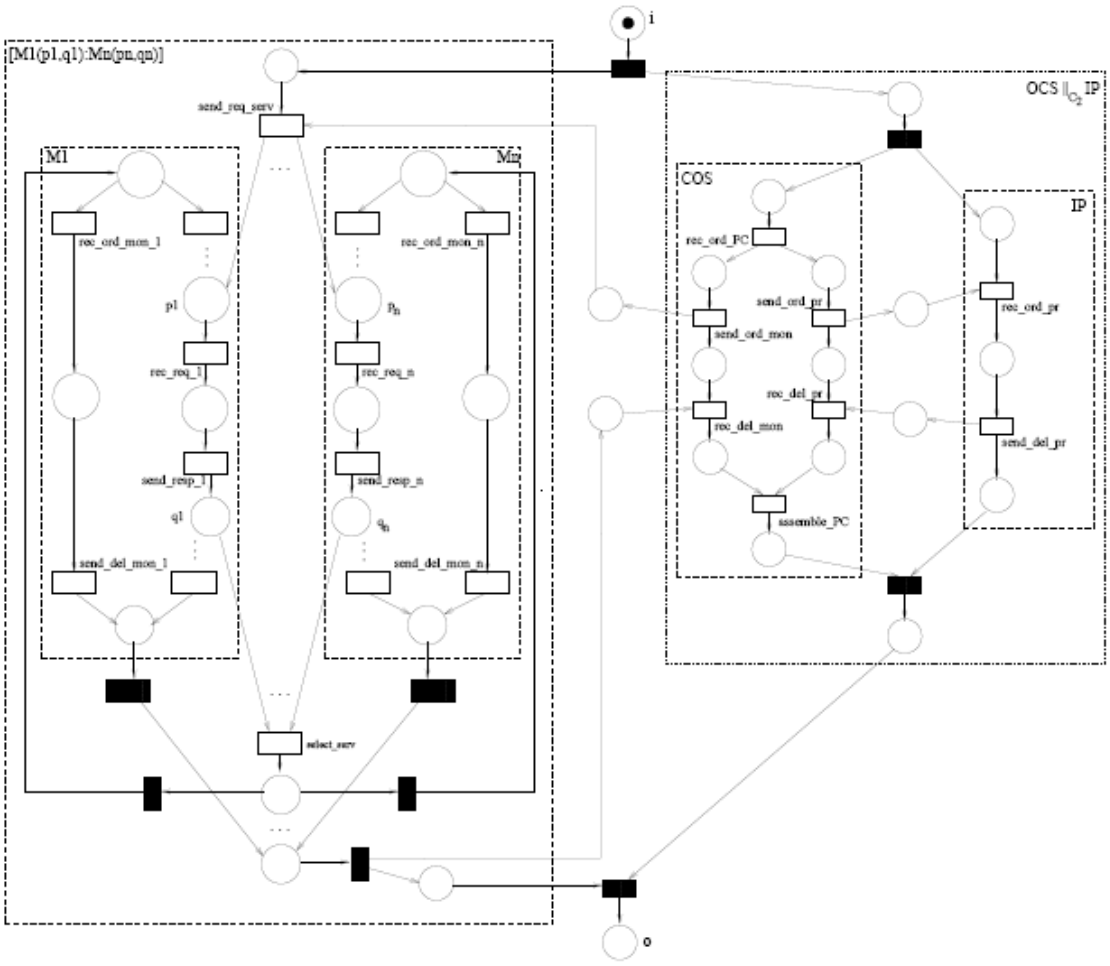
Παράδειγμα. Το σχήμα 4.7 δείχνει μία Web service που συντίθεται από τρείς βασικές υπηρεσίες, την OCS που αντιπροσωπεύει την Online Computer Store και την SM και την IP, που αντιπροσωπεύουν αντίστοιχα την Sony Monitors και την Intel Processors. Όταν ληφθεί μία παραγγελία rec_ord_PCγια ένα υπολογιστή από ένα πελάτη, η OCS αρχίζει, παράλληλα, τις εξωτερικά ανατεθειμένες υπηρεσίες SM για να παραγγείλει μία οθόνη και IP για να παραγγείλει ένα επεξεργαστή εκτελώντας τις λειτουργίες send_ord_mon και send_ord_pr αντίστοιχα. Το σύνολο με τα στοιχεία επικοινωνίας είναι C_1={(send_ord_mon, rec_ord_mon), (send_del_mon, rec_del_mon)} και C_2 = {(send_ord_pr, rec_ord_pr), (send_del_pr, rec_del_pr)}. Μόλις τα απαιτούμενα αντικείμενα ληφθούν, η OCS διενεργεί την assemble_PC λειτουργία.

Για λόγους απλούστευσης και σαφήνειας, δεν αντιπροσωπεύονται όλες οι λειτουργίες του σεναρίου (π.χ., παράδοση και χρέωση) και ετικέτες χρησιμοποιούνται αντί ονομάτων για τους transitions.



Σχήμα 4.7. Η υπηρεσία SM ||$_{C1}$ (OCS ||$_{C2}$ IP)

Ας υποθέσουμε τώρα ότι, στη θέση μίας βασικής υπηρεσίας Sony Monitors, χρησιμοποιούμε μία σύνθετη υπηρεσία Monitors. Ο πάροχος οθονών μπορεί να επιλεχθεί δυναμικά ανάμεσα στις διαθέσιμες υπηρεσίες (π.χ., M_1, . . . , M_n) οι οποίες είναι μέλη της Monitors (βλέπε σχήμα 4.8). Είναι C_3 = {(send_ord_mon, send_req_serv), (τ, rec_del_mon)}.



Σχήμα 4.8. Η υπηρεσία $[M_1(p_1,q_1):M_n(p_n,q_n)]$ $||_{C3}$ (OCS $||_{C2}$ IP)

4.3.5 Αλγεβρικές ιδιότητες

Η σημασιολογία μας μπορεί να χρησιμοποιηθεί για να αποδείξουμε αλγεβρικές ιδιότητες των κατασκευών. Σαν μια γενική παρατήρηση, μπορεί να είναι η περίπτωση όπου ο σχεδιαστής παράγει μία πολύπλοκη Web service συνθέτοντας ένα σύνολο από υπάρχουσες Web services χρησιμοποιώντας τους αλγεβρικούς τελεστές. Οι αλγεβρικές ιδιότητες μπορούν τότε να χρησιμοποιηθούν για να μεταμορφώσουν και να βελτιστοποιήσουν σύνθετες Web services βασισμένες σε συνιστώσες Web services.

Μία περιγραφή των αλγεβρικών ιδιοτήτων των τελεστών που εισήχθησαν παραπάνω δίνεται στον πίνακα 1 (όπου $\prod_{i=1}^{n} S_i$ είναι στη θέση

του $[S_1(p_1,q_1):S_n(p_n,q_n)]$). Οι αλγεβρικοί τελεστές ικανοποιούν κάποιες συνηθισμένες ιδιότητες, όπως η προσεταιριστική, και η αντιμεταθετική.

$$S_1 \odot (S_2 \odot S_3) = (S_1 \odot S_2) \odot S_3 \quad (1)$$
$$\varepsilon \odot S = S \quad (2)$$
$$S \odot \varepsilon = S \quad (3)$$
$$S_1 \oplus S_2 = S_2 \oplus S_1 \quad (4)$$
$$S_1 \oplus (S_2 \oplus S_3) = (S_1 \oplus S_2) \oplus S_3 \quad (5)$$
$$S \oplus S = S \quad (6)$$
$$(S_1 \oplus S_2) \odot S_3 = (S_1 \odot S_3) \oplus (S_2 \odot S_3) \quad (7)$$
$$S_1 \Diamond S_2 = (S_1 \odot S_2) \oplus (S_2 \odot S_1) \quad (8)$$
$$\mu\varepsilon = \varepsilon \quad (9)$$
$$S_1 \parallel_C S_2 = S_2 \parallel_C S_1 \quad (10)$$
$$S_1 \parallel_\varnothing (S_2 \parallel_\varnothing S_3) = (S_1 \parallel_\varnothing S_2) \parallel_\varnothing S_3 \quad (11)$$
$$S \parallel_\varnothing \varepsilon = S \quad (12)$$
$$(S_1|S_2) \rightsquigarrow S_3 = (S_2|S_1) \rightsquigarrow S_3 \quad (13)$$
$$(S_1|\varepsilon) \rightsquigarrow S_2 = S_1 \parallel_\varnothing S_2 \quad (14)$$
$$(S_1|S_2) \rightsquigarrow \varepsilon = S_1 \parallel_\varnothing S_2 \quad (15)$$
$$\forall\{i_1,...,i_n\} = \{1,...,n\} \quad \prod_{i\in\{i_1,...,i_n\}} S_i = \prod_{i=1}^{n} S_i \quad (16)$$
$$\text{If } S_j = \varepsilon \text{ then } \prod_{i=1}^{n} S_i = \prod_{i=1, i\neq j}^{n} S_i \quad (17)$$
$$Ref(S_1, a, S_2) = S_1 \text{ if } a \notin \ell_1(T_1) \quad (18)$$
$$S_1 \Diamond S_2 = S_2 \Diamond S_1 \text{ (from (8) and (4))} \quad (19)$$
$$S \Diamond S = S \odot S \text{ (from (8) and (6))} \quad (20)$$
$$S \Diamond \varepsilon = S \text{ (from (8), (2), (3), and (6))} \quad (21)$$

Πίνακας 1. Επιθυμητές ιδιότητες της άλγεβρας υπηρεσιών

Οι απλές ιδιότητες μπορούν εύκολα να παραχθούν, για παράδειγμα οι (2), (3), (9), και (12). Ένα εύρος άλλων εξισώσεων μπορεί να παραχθεί, το καθένα από μία εύκολη ανάλυση περίπτωσης.

Μία ενδιαφέρουσα παρατήρηση είναι ότι η σημασιολογία μας εξισώνει τις υπηρεσίες $S_1 \Diamond S_2$ και $(S_1 \odot S_2) \oplus (S_2 \odot S_1)$ (ιδιότητα 8). Αυτό σημαίνει ότι η τυχαία ακολουθία μπορεί να εκφραστεί με την χρήση της ακολουθίας και της εναλλαγής. Ωστόσο, είναι ακόμη χρήσιμο να περιέχουμε τον τελεστή τυχαίας ακολουθίας στην άλγεβρα εφόσον η εφαρμογή του θα είναι αναποτελεσματική για την υπηρεσία $(S_1 \odot S_2) \oplus (S_2 \odot S_1)$.

Κωστέλλος Μαριος-Χρηστος
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

4.4 Ανάλυση των Web services

Μία σύνθετη Web service είναι ένα σύστημα που αποτελείται από διάφορες εννοιολογικά αυτόνομες αλλά συνεργαζόμενες μονάδες[39]. Είναι δύσκολο να προσδιορίσουμε το πώς αυτό το σύστημα θα έπρεπε να συμπεριφέρεται και να εξασφαλίσουμε ότι συμπεριφέρεται όπως απαιτήθηκε από τον προσδιορισμό. Ο λόγος είναι ότι ακόμη και αν δώσουμε την ίδια είσοδο και ίδια αρχική κατάσταση, το σύστημα μπορεί να έχει διαφορετικές εξόδους.

Οι Web services αλληλεπιδρούν η μία με την άλλη για να κάνουν μία συγκεκριμένη εργασία παρόλο που δημιουργήθηκαν και ανεπτύχθησαν από ανεξάρτητους παρόχους υπηρεσιών. Τα κομμάτια εργασίας μέσα στη συγκεκριμένη εργασία είναι στενά συνδεδεμένα. Για παράδειγμα, ένα Online Computer Store μπορεί να χρειαστεί υπηρεσίες που παρέχονται από τις Web services Monitor και Processor. Η επιχειρησιακή λογική των δύο αυτών Web services είναι ανεξάρτητη η μία από την άλλη, για μία συγκεκριμένη απαίτηση υπολογιστή, τα δύο κομμάτια εργασίας, που είναι η παραγγελία της οθόνης και η παραγγελία του επεξεργαστή, σχετίζονται. Από την στιγμή που Web services που περιέχουν λάθη μπορεί να οδηγήσουν σε εξαγριωμένους πελάτες και απώλεια υπεραξίας, κάνει τόσο σημαντική την ανάλυση των Web services πριν αυτές μπουν σε λειτουργία. Ο στόχος είναι να παρέχουμε μηχανισμό που να υποστηρίζει σωστά την σύνθεση των Web services.

Οι ιδιότητες που πρέπει να επαληθευθούν μπορεί να είναι γενικές, όπως η απουσία αδιεξόδων και livelocks, ή εξειδίκευση στην εφαρμογή, όπως αν ένας πελάτης συνεχίζει να "κλείνει" και να ακυρώνει την κράτηση επ' αόριστον, σε μία Online Ticket Sales Web service, ένα εξελιγμένο σύστημα θα πρέπει να αρνηθεί αν εξυπηρετήσει ένα τέτοιο πελάτη μετά από ένα συγκεκριμένο αριθμό αιτήσεων κράτησης ή ακύρωσης. Η ορθότητα των υπηρεσιών είναι ζωτική για τις εταιρίες. Ο σωστός τερματισμός, είναι μία από τις βασικές ιδιότητες που θα πρέπει να ικανοποιεί μία ορθή Web service. Η επιβεβαίωση της ιδιότητας της ορθότητας βασίζεται στο πότε το υποκείμενο Petri net είναι live και δεσμευμένο.

Οι ισοδυναμίες συμπεριφοράς είναι χρήσιμες στην επαλήθευση καθώς θέτουν την σημασιολογική βάση για την απόφαση ότι η συμπεριφορά δύο Web services μπορεί να θεωρηθεί ότι είναι η "ίδια". Μπορούν επίσης να χρησιμοποιηθούν ως εργαλείο για την μείωση της προσπάθειας επαλήθευσης αντικαθιστώντας το Petri net μιας υπηρεσίας με ένα μικρότερο (σε μέγεθος), αλλά "ισοδύναμο". Η bisimulation ισοδυναμία μεταξύ δύο Web services είναι μία σχέση μεταξύ των εξελίξεων

τους τέτοια ώστε για κάθε εξέλιξη της μίας από τις υπηρεσίες υπάρχει μία αντίστοιχη εξέλιξη της άλλης υπηρεσίας τέτοια ώστε οι εξελίξεις να παρατηρούνται ως ‘‘ισοδύναμες’’ και να οδηγούν σε υπηρεσίες οι οποίες είναι και αυτές bisimilar. Αυτός ο χαρακτηρισμός της συμπεριφοράς των Web services χρησιμοποιώντας την έννοια της bisimulation βοηθά τον σχεδιαστή υπηρεσιών να βελτιστοποιήσει σύνθετες υπηρεσίες, π.χ., αλλάζοντας τις συνιστώσες Web services με άλλες ισοδύναμες. Ακόμη ένα κίνητρο για την βελτιστοποίηση των υπηρεσιών. Για να ενισχύσει την ανταγωνιστικότητα ένας πάροχος υπηρεσιών μπορεί να τροποποιήσει την υπηρεσία του κατά τη βολή του πελάτη και αυτή η τροποποιημένη υπηρεσία πρέπει να συμμορφώνεται με την αυθεντική.

V
Κεφάλαιο

Ένα σενάριο σύνθεσης υπηρεσιών

5. Ένα σενάριο σύνθεσης υπηρεσιών

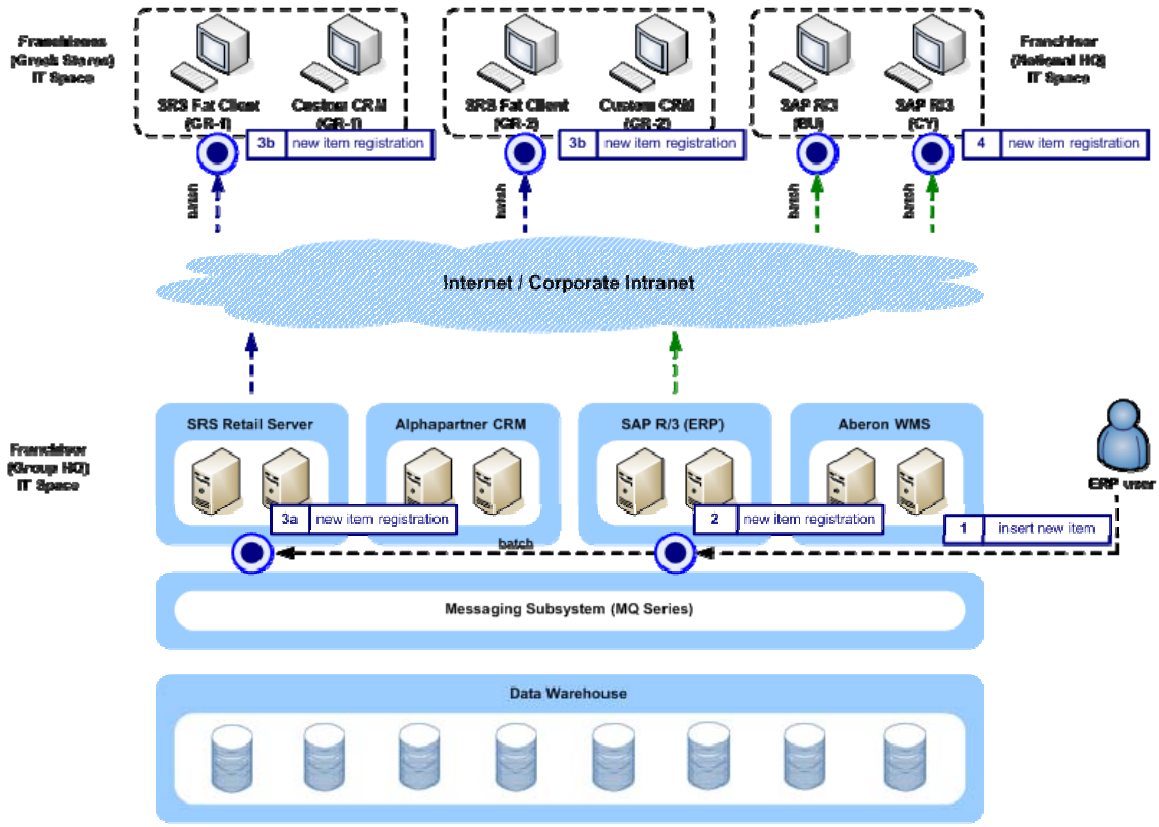
5.1 Περιγραφή σεναρίου

Στο παρών κεφάλαιο παρουσιάζεται ένα σενάριο σύνθεσης μιας σύνθετης υπηρεσίας από επιμέρους υπηρεσίες[38] που είναι εγκατεστημένες στα πληροφοριακά συστήματα μιας επιχείρησης που ασχολείται με το εμπόριο. Το σενάριο εμπλέκει υποδομές του franchisor (π.χ., έδρα ομάδας) και τις υποδομές των franchisees (π.χ., καταστήματα λιανικής πώλησης), δηλαδή στα πλαίσια του σεναρίου υποθέτουμε ότι υπάρχει μια επιχείρηση που έχει κεντρικό κατάστημα (που ελέγχει προμήθειες και παραγγελίες συνολικά) και πολλά υποκαταστήματα λιανικής πώλησης. Το σενάριο το οποίο θα μας απασχολήσει είναι ο ανεφοδιασμός προϊόντος (restocking) από το δίκτυο της λιανικής πώλησης προς την έδρα και η τροφοδοσία αυτού.

Σύμφωνα με το σενάριο τέσσερις διαφορετικοί τύποι επιχειρηματικών συστημάτων συμμετέχουν: ένα ERP , πολλά συστήματα λιανικής (retail clients), ένας εξυπηρετητής λιανικής (retail server) και ένα σύστημα διαχείρισης αποθήκης (Warehouse Management System).
Το ERP σύστημα είναι υπεύθυνο για τη διατήρηση των κεντρικών, εταιρικών επιχειρηματικών υποδομών για την διαδικασία ‘‘Εισαγωγής νέου αντικειμένου’’ κτλ. Ο retail server και οι retail clients συνεργάζονται με σκοπό την ενημέρωση των κεντρικών για πιθανές ελλείψεις ή παραγγελίες. Τέλος το WMS είναι υπεύθυνο για την ενημέρωση του ERP από τις αποθήκες.

Κάθε προϊόν πρέπει να δημιουργηθεί ουσιαστικά μαζί με τα συνοδευτικά χαρακτηριστικά του (π.χ., είναι φυσικό αντικείμενο;, θα υποβληθεί σε εκπτώσεις;, ποιο είναι το κόστος του μοντέλου;). Μετά από αυτό τον αυστηρό ορισμό το προϊόν εισάγεται στο εταιρικό ERP. Στη συνέχεια, τα καταστήματα λιανικής πώλησης θα πρέπει να ενημερωθούν για κάθε νέο αντικείμενο που εισήχθηκε στον κεντρικό κατάλογο προϊόντων του εταιρικού ERP.

Περιγράφοντας την παραπάνω διαδικασία (εισαγωγή νέου αντικειμένου) γίνεται αντιληπτό ότι κάθε μια επιμέρους διαδικασία που λαμβάνει μέρος στην διαδικασία ‘εισαγωγής νέου αντικειμένου’ έχει δημιουργηθεί ως μια ξεχωριστή web service.



Σχήμα 5.1. Το σενάριο "εισαγωγή νέου αντικειμένου"

Υποθέτουμε τώρα ότι θέλουμε να υλοποιήσουμε το σενάριο του ανεφοδιασμού χρησιμοποιώντας παραδοσιακές τεχνικές. Υποθέτοντας ότι όλα τα μετέχοντα συστήματα είναι βασισμένα σε web services, οι υλοποιητές που θα συμμετείχαν στην εφαρμογή θα έπρεπε να διενεργήσουν τα ακόλουθα:

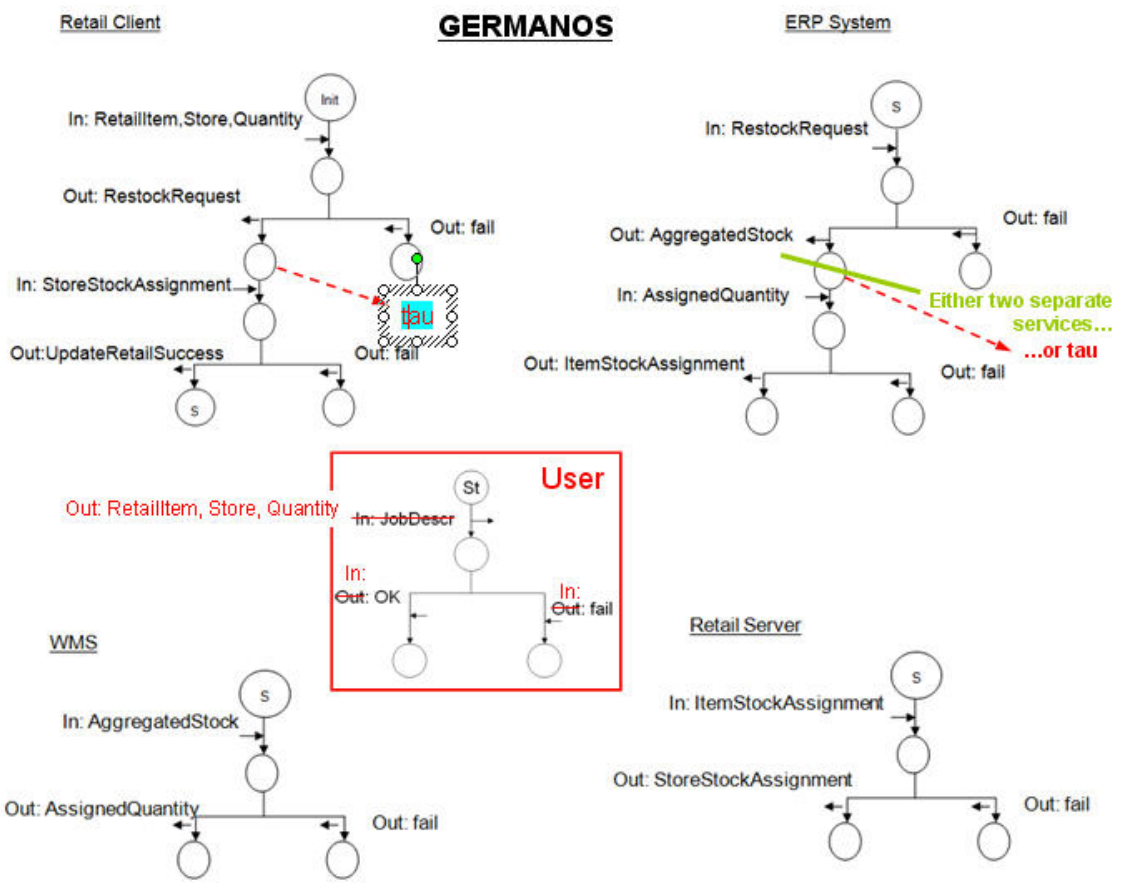
- Να αναγνωρίσουν για κάθε σύστημα την κατάλληλη υπηρεσία που είναι σχετική με την επιθυμητή λειτουργικότητα. Φυσικά αυτό δεν θα γίνει αυτόματα,
- Να εξετάσουν τις εισόδους (ορίσματα) και τις εξόδους των μετεχόντων συστημάτων,
- Η ομάδα των developers του ενός συστήματος θα πρέπει να επικοινωνεί με την ομάδα των developers του άλλου συστήματος με σκοπό την ευθυγράμμιση των στοιχείων που ανταλλάσσονται μεταξύ τους, και
- Τελικά, η ομάδα των developers του ενός συστήματος θα πρέπει να επικοινωνεί με την ομάδα των developers του άλλου συστήματος με σκοπό να διαχειριστούν εξαιρέσιμες διαδρομές της εκτέλεσης.

Ωστόσο αν όλα τα συστήματα που μετέχουν στο σενάριο διέθεταν ένα μοντέλο συμπεριφοράς για τις υπηρεσίες που διαθέτουν, θα μπορούσαν ημιαυτόματα να γνωρίζουν αν μια πιο σύνθετη συμπεριφορά μπορεί να συντεθεί από τις αρχικές. Στην επόμενη παράγραφο δίνουμε το μοντέλο των επιμέρους υπηρεσιών.

5.2 Μοντέλο Σεναρίου

Παρακάτω παρατίθεται εποπτικά ένα υποθετικό σενάριο της επιχειρηματικής υπηρεσίας ανεφοδιασμού (restocking). Η διαδικασία ανεφοδιασμού μιας επιχείρησης είναι η διαδικασία που ακολουθείται από την επιχείρηση, ώστε να μεταδοθούν οι παραγγελίες από τα καταστήματα λιανικής πώλησης στο κεντρικό σύστημα της εταιρίας και η μεταφορά των παραγγελιών αυτών πίσω στα καταστήματα, βάση της πολιτικής της εταιρίας.

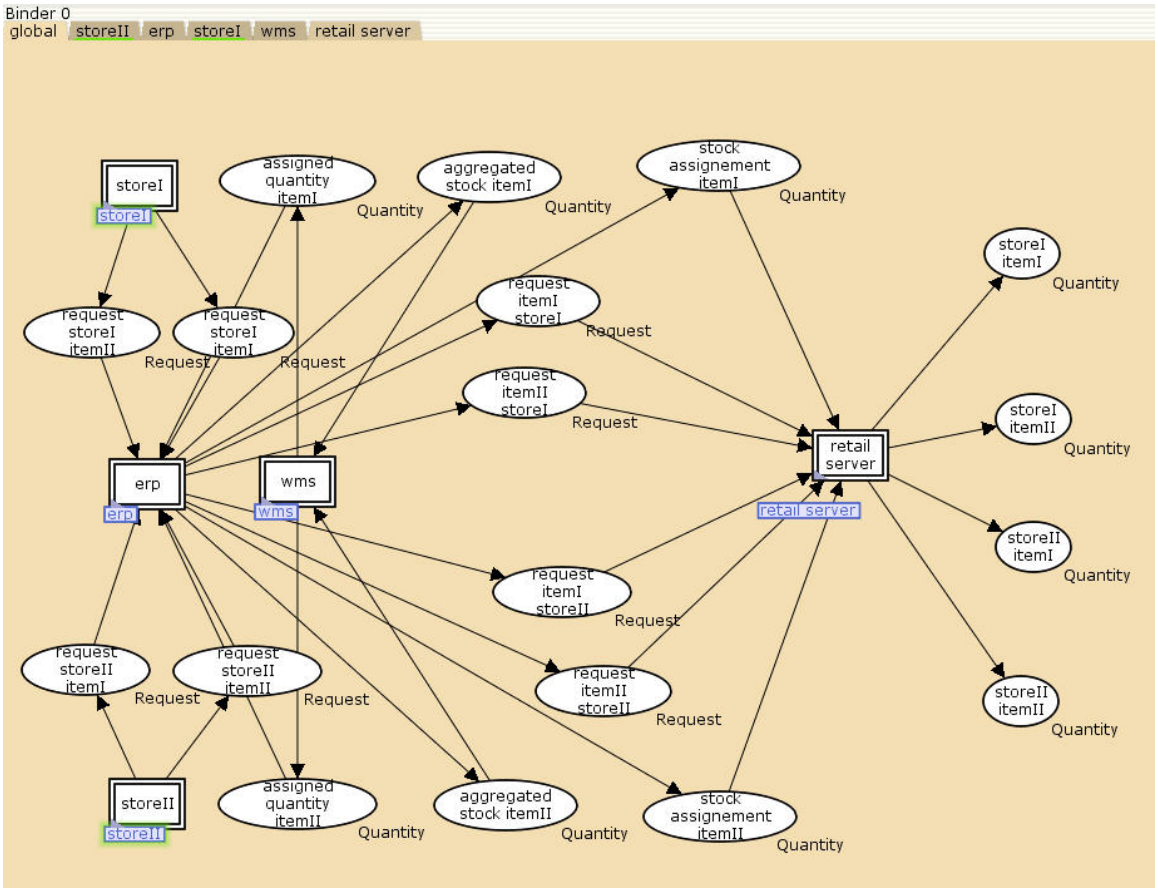
Στην διαδικασία ανεφοδιασμού συμμετέχουν 4 συστήματα, τα οποία είναι ο retail client, ο retail server, το ERP system, και το WMS. Ο retail client είναι το σύστημα που μεταφέρει την παραγγελία από το κάθε κατάστημα λιανικής πώλησης στο ERP σύστημα. Το ERP σύστημα είναι αυτό που συγκεντρώνει της παραγγελίες και τις μεταφέρει συνολικά στο WMS. Το WMS είναι το σύστημα το οποίο δέχεται τον όγκο της συνολικής παραγγελίας και αποφασίζει την ποσότητα που τελικά θα διατεθεί στο σύνολο των καταστημάτων λιανικής πώλησης (ενημερώνοντας το ERP). Τέλος, υπάρχει ο retail server, ο οποίος δέχεται την ποσότητα που του στέλνει το ERP και την διαμοιράζει στα καταστήματα λιανικής πώλησης με βάση της αρχικές τους παραγγελίες.



Σχήμα 5.2. Το ERP του GERMANOS σχηματικά

Δεδομένης της ύπαρξης των Web services που υπάρχουν στα επιμέρους συστήματα σκοπός μας είναι να ανιχνεύσουμε την ύπαρξη ενός σύνθετου Web service, με ημιαυτόματο τρόπο, που αναχρησιμοποιεί την λειτουργικότητα των υφιστάμενων Web services. Προκειμένου να το πετύχουμε αυτό θα μοντελοποιήσουμε τα υφιστάμενα Web services με χρήση Petri nets. Μετά την μοντελοποίηση της κάθε υπηρεσίας ξεχωριστά, θα εξομοιώσουμε την λειτουργία του συστήματος συνολικά και κρίνοντας με βάση τις επιθυμητές καταστάσεις που θα βρεθεί το σύστημα μετά την εξομοίωση θα αποφανθούμε για την δυνατότητα σύνθεσης των επιμέρους υπηρεσιών.

Στη συνέχεια παραθέτουμε την υλοποίηση αυτού του μοντέλου με την χρήση Colored Petri net και το εργαλείο CPN tools, όπου υπάρχει ένα καθολικό σχεδιάγραμμα, που δίνει την συνολική εικόνα, το οποίο όμως αναλύεται και αυτό σε επιμέρους τμήματα.

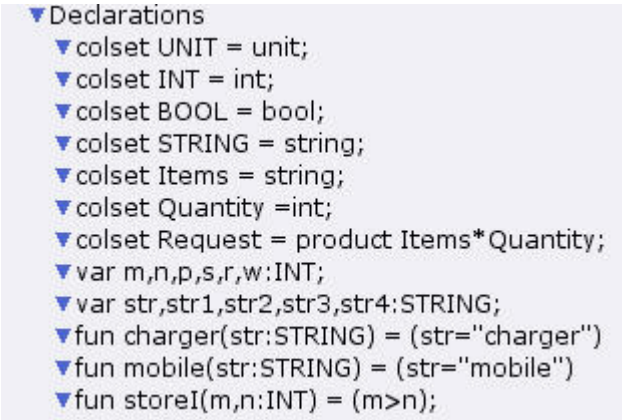


Σχήμα 5.3. Η αναπαράσταση με CPN του σχήματος 5.2

Όπως είδαμε σε προηγούμενα κεφάλαια σχετικά με την χρήση και την σημασία των CPN, το παραπάνω CPN αποτελείται από μία κεντρική σελίδα (global) και πέντε υποκαταστάσεις μετάβασης (storeI, storeII, wms, erp, retail server). Στη συνέχεια θα αναλυθούν οι έξι αυτές υποκαταστάσεις μετάβασης και η λειτουργία του CPN, με βάση αυτά που ειπώθηκαν σε προηγούμενα κεφάλαια. Πρώτα όμως πρέπει να αναλύσουμε τις μεταβλητές που χρησιμοποιούνται στο CPN καθώς και την γενική σελίδα του CPN.

Στη γενική σελίδα του CPN (σχήμα 5.3) φαίνονται οι υποκαταστάσεις μετάβασης storeI, storeII, wms, erp, και retail server. Οι πρώτες δύο αποτελούν ουσιαστικά το σύστημα retail client του σχήματος 5.2, δηλαδή είναι δύο καταστήματα λιανικής πώλησης, τα οποία δίνουν παραγγελίες στο κεντρικό σύστημα. Στη συνέχεια αυτά τροφοδοτούν το σύστημα erp το οποίο εκτελεί την λειτουργία του, δηλαδή να αθροίσει τις επιμέρους παραγγελίες. Το erp τροφοδοτεί με τη σειρά του το wms, το οποίο δέχεται ως είσοδο το σύνολο της παραγγελίας και δίνει στην έξοδο την ποσότητα που θα διατεθεί συνολικά στα καταστήματα λιανικής πώλησης. Στη

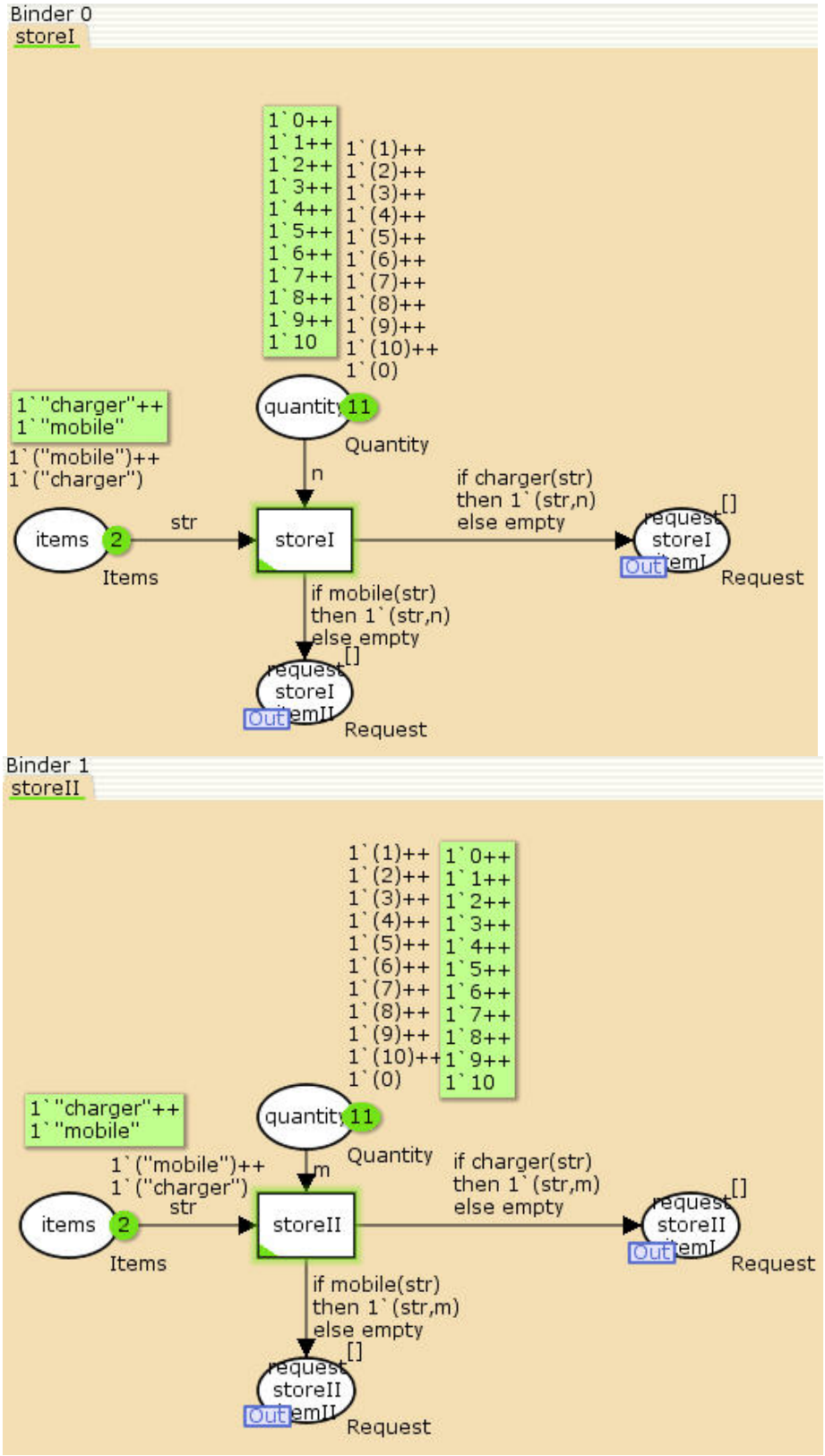
συνέχεια ανατροφοδοτείται το erp από το wms, το οποίο με την σειρά του τροφοδοτεί το σύστημα retail server το οποίο μοιράζει την συνολική ποσότητα, που είναι διαθέσιμη για το σύνολο των καταστημάτων λιανικής πώλησης, στα καταστήματα λιανικής πώλησης με βάση την αρχική τους παραγγελία.



Σχήμα 5.4. Οι δηλώσεις μεταβλητών του CPN ERP GERMANOS

Στη συνέχεια οι μεταβλητές του CPN είναι οι εξής.Οι τέσσερις πρώτες μεταβλητές είναι οι πρότυπες μεταβλητές του CPN tool,δηλαδή η UNIT, η INT, η BOOL, και η STRING. Στη συνέχεια ορίζεται η μεταβλητή Items, η οποία είναι τύπου string και αντιπροσωπεύει τα προϊόντα που χρησιμοποιούνται στο ERP. Η μεταβλητή Quantity, η οποία είναι τύπου int και αντιπροσωπεύει την ποσότητα των προϊόντων. Η μεταβλητή Request, η οποία είναι συνδυασμός του Items και του Quantity και αντιπροσωπεύει την αίτηση για την ποσότητα του κάθε προϊόντος από τα καταστήματα λιανικής πώλησης στο ERP. Οι μεταβλητές m, n, p, s, r, και w, που είναι τύπου int και είναι βοηθητικές για την εκτέλεση του CPN, όπως και οι str, str1, str2, str3, και str4, που είναι τύπου string και είναι και αυτές βοηθητικές. Στη συνέχεια ορίζονται οι συναρτήσεις charger και mobile, οι οποίες δέχονται ως όρισμα την μεταβλητή str και είναι true αν str=charger και str=mobile αντίστοιχα. Ορίζεται και η συνάρτηση storeI, με ορίσματα τις μεταβλητές m, και n, η οποία είναι true όταν είναι m>n.

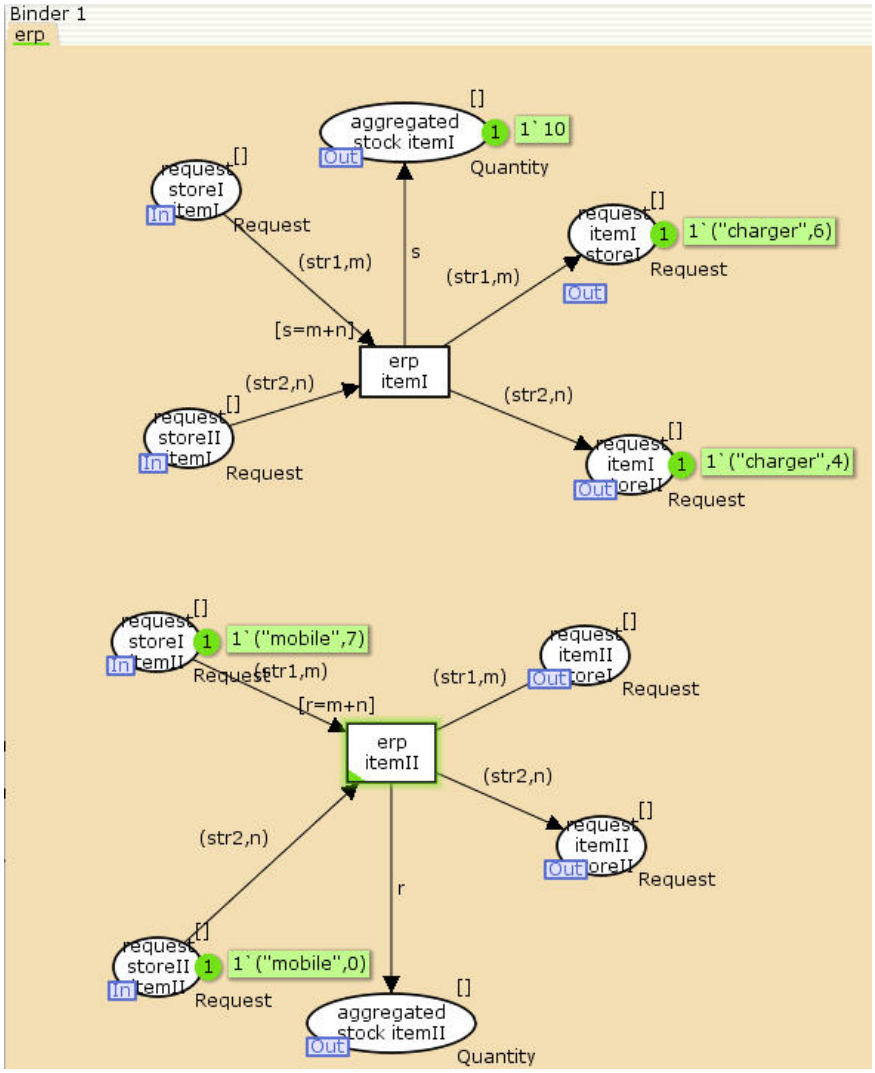
Πρώτα εξετάζουμε τις υποκαταστάσεις μετάβασης storeI και storeII, οι οποίες είναι πανομοιότυπες. Η εσωτερική δομή τους φαίνεται στο παρακάτω σχήμα.



Σχήμα 5.5. Οι υποκαταστάσεις μετάβασης storeI και storeII

Οι storeI και storeII αποτελούνται από δύο places εισόδου (items και quantity) η κάθε μία, από ένα transition με την αντίστοιχη ετικέτα, και από δύο places εξόδου, οι οποίοι είναι και οι έξοδοι της υποκατάστασης μετάβασης. Ο transition δέχεται από τους places εισόδου τις αιτήσεις του κάθε καταστήματος, δηλαδή προϊόν και ποσότητα, και στη συνέχεια απομονώνει σε τύπο Request την αίτηση ως συνδυασμό της ποσότητας και του προϊόντος, για κάθε προϊόν χωριστά. Έπειτα αποστέλλονται στους places εξόδου οι αιτήσεις και από εκεί εκχωρούνται στην κύρια σελίδα του CPN.

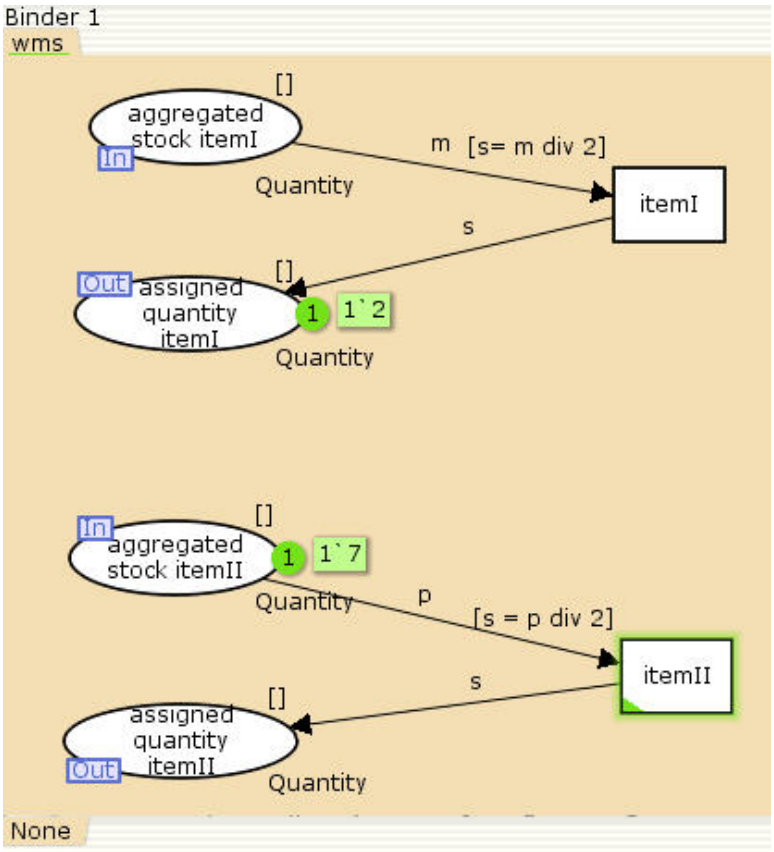
Από τις εξόδους των υποκαταστάσεων μετάβασης storeI και storeII τροφοδοτείται η είσοδος της υποκατάστασης μετάβασης erp, η οποία αποτελεί την αναπαράσταση της μονάδας erp και παρακάτω αναλύεται η εσωτερική της δομή.



Σχήμα 5.6. Η εσωτερική δομή της υποκατάστασης μετάβασης erp

Η εσωτερική δομή της υποκατάστασης μετάβασης erp αποτελείται από δύο transitions,τα οποία δέχονται ως είσοδο, τους places που είναι οι έξοδοι των storeI και storeII. Στην erp οι δύο transitions επεξεργάζονται ο καθένας τα στοιχεία για κάθε προϊόν και αθροίζει την συνολική ποσότητα που απαιτείτε από το κάθε προϊόν και για τα δύο καταστήματα (storeI, storeII). Οι έξοδοι του κάθε transition της erp είναι τρεις places (για λόγους συντομίας αναφέρουμε μόνο για τον ένα transition, τον erp itemI, όμοια είναι και για τον erp itemII), ο aggregated stock itemI, ο request itemI storeI, και ο request itemI storeII. Ο πρώτος είναι η έξοδος προς την επόμενη υποκατάσταση μετάβασης και είναι η συνολική ποσότητα που απαιτούν τα καταστήματα. Οι άλλοι δύο είναι έξοδοι προς την τελική υποκατάσταση μετάβασης, και είναι οι απαιτήσεις των καταστημάτων για κάθε προϊόν.

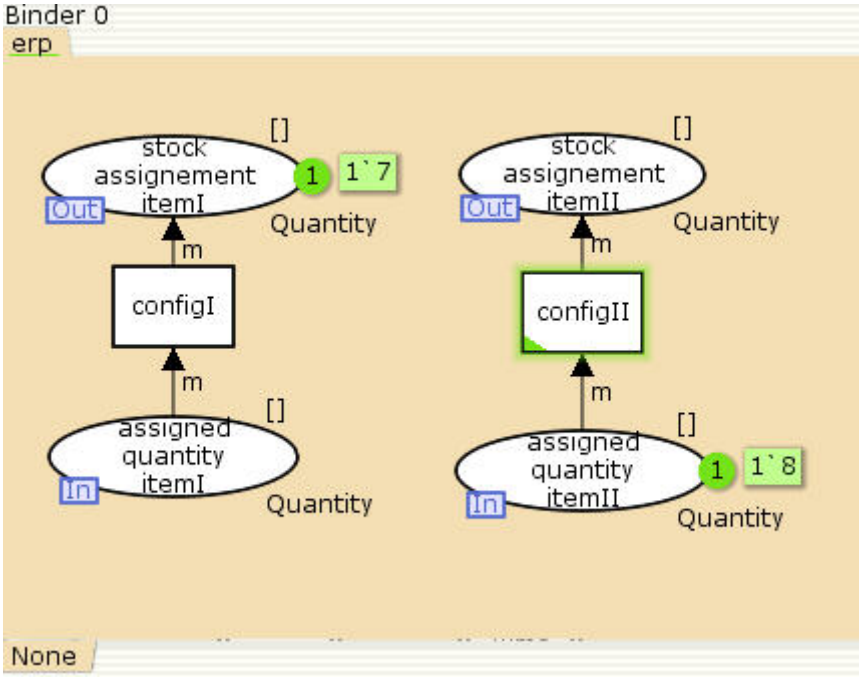
Στη συνέχεια ακολουθεί η υποκατάσταση μετάβασης wms ,η οποία δέχεται ως εισόδους τους places aggregated stock itemI και aggregated stock itemII, και τις επεξεργάζεται όπως δείχνεται στην εσωτερική της δομή.



Σχήμα 5.7. Η εσωτερική δομή της υποκατάστασης μετάβασης wms

Στην wms εγκρίνονται οι συνολικές ποσότητες των προϊόντων που θα δοθούν τελικά στα καταστήματα. Η wms αποτελείται από δύο transitions(itemI και itemII),μία για κάθε προϊόν, από δύο places εισόδου, οι οποίοι προέρχονται από την προηγούμενη υποκατάσταση μετάβασης erp, και από δύο places εξόδου με ετικέτες assigned quantity itemI/itemII. Ο ρόλος του transition, αλλά και της wms, είναι να ρυθμίσει τις τελικές συνολικές ποσότητες του κάθε προϊόντος που τελικά θα δοθούν στα καταστήματα, με βάση την πολιτική που επιθυμεί η εταιρία, π.χ., εδώ επιστρέφει στα καταστήματα την μισή ποσότητα από κάθε προϊόν, αυτό φαίνεται από τα *s=p div 2* και *s=m div 2*.

Στη συνέχεια η εκχωρούμενη ποσότητα (assigned quantity) του κάθε προϊόντος δίνεται ως έξοδος της υποκατάστασης μετάβασης wms στην γενική σελίδα global, μέσω των places εξόδου assigned quantity itemI/itemII. Στη συνέχεια ακολουθεί πάλι η υποκατάσταση μετάβασης erp, αυτή τη φορά με είσοδο τις εξόδους της wms.

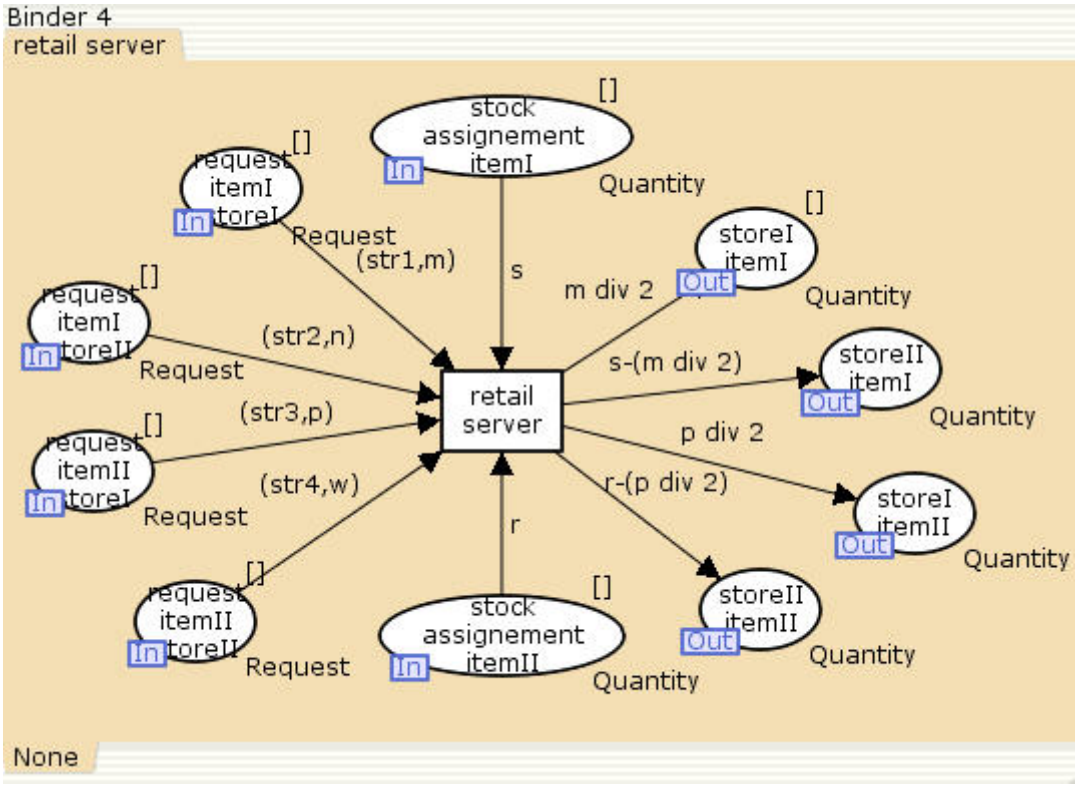


Σχήμα 5.8. Η εσωτερική δομή του δεύτερου μέρους της erp

Εδώ επιβεβαιώνονται οι ποσότητες των προϊόντων που εκχωρούνται από την wms, μέσω των transitions configI και configII. Αυτό το δεύτερο μέρος της erp έχει ως έξοδο τους places stock assignment itemI/itemII.

Μετά και από το δεύτερο στάδιο της erp ακολουθεί η υποκατάσταση μετάβασης retail server, η οποία αποδίδει την ποσότητα του κάθε προϊόντος που θα πάρει κάθε κατάστημα. Αποτελείται από έξι εισόδους,

τους places request itemI storeI, request itemI storeII, request itemII storeI, και request itemII storeII (δηλαδή τις τέσσερις από τις έξι εξόδους του πρώτου σταδίου της erp), αλλά και τους places stock assignment itemI/itemII (δηλαδή τις εξόδους του δεύτερου σταδίου της erp). Οι έξοδοι της retail server είναι οι τέσσερις places storeI itemI, storeII itemI, storeI itemII, και storeII itemII, οι οποίοι είναι και οι τελικές έξοδοι του συστήματος και αντιπροσωπεύουν την τελική ποσότητα από κάθε προϊόν που θα λάβουν τα καταστήματα.



Σχήμα 5.9. Η εσωτερική δομή της υποκατάστασης μετάβασης retail server

VI
Κεφάλαιο

Συμπεράσματα

6. Συμπεράσματα

Σε αυτή την εργασία διερευνήσαμε την δυνατότητα επαλήθευσης σύνθετων υπηρεσιών με την χρήση των Petri nets. Μοντελοποιήσαμε την διαδικασία ανεφοδιασμού μίας αλυσίδας καταστημάτων λιανικής πώλησης, περιγράφοντας αναλυτικά τα επιμέρους συστήματα που παίρνουν μέρος στην εκτέλεση της υπηρεσίας.

Για να ερευνήσουμε την αποτελεσματικότητα της υπηρεσίας που δημιουργήσαμε χρησιμοποιήσαμε τη ανάλυση καταστάσεων (state space analysis) και μοντελοποιήσαμε τα επιμέρους συστήματα σε ξεχωριστά CPn το καθένα.

Λόγω της μεγάλης εκφραστικής δύναμης των Petri nets είναι εύκολο να αντιληφθούμε τον τρόπο με τον οποίο συνεργάζονται τα ανεξάρτητα συστήματα. Ταυτόχρονα αντιλαμβανόμαστε ότι είναι δυνατό να συντεθούν αυτά τα τέσσερα συστήματα σε μία σύνθετη υπηρεσία, η οποία θα εκτελεί σωστά και αποδοτικά την λειτουργία της.

Για να παρατηρήσουμε τα αποτελέσματα της σύνθετη υπηρεσία χρησιμοποιήσαμε την ανάλυση καταστάσεων, η οποία αποτελείται από όλες τις δυνατές καταστάσεις που μπορεί να παρέλθει το σύστημα και μετά από ποια σήμανση επιτυγχάνεται η παρούσα σήμανση και με ποιά δράση και σε ποια δέσμευση.

Αυτός ο τρόπος ανάλυσης αποτελεί ένα αποτελεσματικό και αποδοτικό τρόπο, όταν πρόκειται για συστήματα με μικρούς γράφους ανάλυσης καταστάσεων. Στο παράδειγμα μας όμως , από τη αναφορά της ανάλυσης καταστάσεων βλέπουμε ότι έχουμε παραπάνω από 16000 κόμβους, γεγονός που δυσκολεύει την διεξοδική έρευνα των καταστάσεων. Με αποτέλεσμα να απαιτεί μεγάλη υπολογιστική ισχύ και χρόνο.

Παραπομπές

[1].C.A. Petri. Kommunikation mit automaten. Technical report, Doctoral Thesis, University of Bonn, 1962. (Available in English as: *Communication with automata*, Technical Report RADC-TR-65-377, Rome Air Development Center, Griffiss NY, 1966)

[2] T. Agerwala. Putting Petri nets to work. *IEEE Computer*, pages 85–94, December 1979.

[3] W.M.Chow, E.A.McNair, and C.H.Sauer. Analysis of manufacturing systems by Research Queueing Package. *IBM Journal of Research and Development*, 29:330–341,
1985

[4] W.Whitt. Blocking when service is required from several facilities simultaneously. *AT&T Technical Journal*, 64:1807–1856, 1985.

[5] J.Martinez and M.Silva. A simple fast algorithm to obtain all invariants of a generalized Petri net. In *Proceedings 2-nd European Workshop on Application and Theory of Petri Nets*. Springer-Verlag, 1981.

[6] J.L. Peterson. *Petri net theory and the modeling of systems*. Prentice Hall, Englewood Cliffs, 1981.

[7] G. Ciardo. Toward a definition of modeling power for stochastic Petri net models. In *Proceedings International Workshop on Petri Nets and Performance Models*, pages 54–62, Madison, 1987. IEEE Computer Society Press no. 796.

[8] J. Bechta Dugan, A. Bobbio, G. Ciardo, and K. Trivedi. The design of a unified package for the solution of stochastic Petri net models. In *Proceedings International Workshop on Timed Petri Nets*, pages 6–13, Torino (Italy), 1985. IEEE Comp Soc Press no. 674.

[9] M. Ajmone Marsan, G. Balbo, and G. Conte. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2:93–122, 1984.

[10] Jensen, K.: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 2, Analysis Methods. Monographs in Theoretical Computer Science. Berlin, Heidelberg, New York: Springer-Verlag, 2nd corrected printing 1997, ISBN: 3-540-58276-2

[11] Jensen, K.: An Introduction to the Theoretical Aspects of Coloured Petri Nets. In: de Bakker, J.W., de Roever, W.P., Rozenberg, G. (eds.): A Decade of Concurrency. LNCS 803. Berlin, Heidelberg, New York: Springer-Verlag, 1994, pp. 230-272

[12] Bertsekas, D., Gallager, R.: Data Networks. Prentice-Hall, 1992

[13] Loiseaux, G., Graf, S., Sifakis, J., Bouajjani, A., Bensalem, S.: Property Preserving Abstractions for the Veri_cation of Concurrent Systems. Formal Methods in System Design 6, 1995

[14] Milner, R., Harper, R., Tofte, M.: The De_nition of Standard ML. MIT Press, 1990

[15] Goldfarb, C.F., Rubinsky, Y.: The SGML handbook. Clarendon Press, Oxford, UK, 1990

[16] ITU (CCITT). Recommendation z.120: Msc. Technical report, International Telecommunication Union, 1992

[17] Rasmussen, J.L., Singh, M.: Mimic/CPN. A Graphical Simulation Utility for Design/CPN. User's Manual. Available from http://www.daimi.au.dk/designCPN

[18] Reisig, W.: Petri nets, volume 4 of EACTS Monographs on Theoretical Computer Science. Berlin, Heidelberg, New York: Springer-Verlag, 1985

[19] Jensen, K.: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 2, Analysis Methods. Monographs in Theoretical Computer Science. Berlin, Heidelberg, New York: Springer-Verlag, 2nd corrected printing 1997, ISBN: 3-540-58276-2

[20] Gibbons, A.: Algorithmic Graph Theory. Cambridge University Press, 1985

[21] Cheng, A., Christensen, S., Mortensen, K.H.: Model Checking Coloured Petri Nets Exploiting Strongly Connected Components. In: Spathopoulos, M.P., Smedinga, R., Koz_ak, P. (eds.): Proceedings of the International Workshop on Discrete Event Systems, WODES96. Institution of Electrical Engineers, Computing and Control Division, Edinburgh, UK, 1996

[22] Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic Verification of Finite State Concurrent Systems using Temporal Logic. ACM Transactions on Programming Languages and Systems 8(2): 244-263, 1986

[23] Emerson, E.A., Prasad Sistla, A.: Symmetry and Model Checking. Formal Methods in System Design 9, 1996

[24] Ip, C.N., Dill, D.L.: Better Verification Through Symmetry. Formal Methods in System Design 9, 1996

[25] Jensen, K.: Condensed State Spaces for Colored Petri Nets. Formal Methods in System Design 9, 1996

[26] Jorgensen, J.B., Kristensen, L.M.: Computer Aided Verification of Lamport's Fast Mutual Exclusion Algorithm Using Coloured Petri Nets and Occurrence Graphs with Symmetries. To Appear in IEEE Transactions on Parallel and Distributed Systems, 1999

[27] Jorgensen, J.B., Kristensen, L.M.: Verification of Coloured Petri Nets Using State Spaces with Equivalence Classes. In: Farwer, B., Moldt, D., Stehr, M.O. (eds.): Proceedings of Workshop on Petri nets in System Engineering Modeling, Verification, and Validation. Department of Computer Science, University of Hamburg, 1997, pp. 20-31, Report no. 205

[28] Godefroid, P.: Using partial orders to improve automatic verification methods. In: Proceedings of CAV' 90. LNCS 531. Berlin, Heidelberg, New York: Springer-Verlag, 1990, pp. 175-186

[29] Kristensen, L.M., Valmari, A.: Finding Stubborn Sets of Coloured Petri Nets Without Unfolding. In: Desel, J., Silva, M. (eds.): Proceedings of ICATPN' 98. LNCS 1420. Berlin, Heidelberg, New York: Springer-Verlag, 1998, pp. 104-123

[30] Peled, D.: Combining Partial Order Reductions with On-thefly Model Checking. Formal Methods in System Design, 1996

[31] Valmari, A.: Compositionality in State Space Verification Methods. In: Billington, J., Reisig, W. (eds.): Proceedings of ICATPN' 96. LNCS 1091. Berlin, Heidelberg, New York: Springer-Verlag, 1996

[32] Christensen, S., Petrucci, L.: Modular State Space Analysis of Coloured Petri Nets. In: De Michelis, G., Diaz, M. (eds.): Proceedings of ICATPN' 96. LNCS 935. Berlin, Heidelberg, New York: Springer-Verlag, 1996

[33] Valmari, A.: Compositionality in State Space Verification Methods. In: Billington, J., Reisig, W. (eds.): Proceedings of ICATPN' 96. LNCS 1091. Berlin, Heidelberg, New York: Springer-Verlag, 1996

[34] Benatallah, B., Medjahed, B., Bouguettaya, A., Elmagarmid, A. & Beard, J. (2000), Composing and MaintainingWeb-based Virtual Enterprises, in `Proceedings of theWorkshop on Technologies for E-Services (in Cooperation with VLDB'00)', Cairo, Egypt.

[35] Benatallah, B., Dumas, M., Sheng, Q. & Ngu, A. (2002), Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services, in `Proceedings of the 18th International Conference on Data Engineering (ICDE'02)', IEEE Computer Society, California, USA, pp. 297-308.

[36] Schuster, H., Georgakopoulos, D., Cichocki, A.& Baker, D. (2000), Modeling and Composing Service-based and Reference Process-based Multi-enterprise Processes, in `Proceedings of the 12th Conference on Advanced Information Systems Engineering (CAiSE'00)', Stockholm, Sweden.

[37] Aalst, W. v. d., Hofstede, A. t., Kiepuszewski, B. & Barros, A. (2002), Workow Patterns, Technical Report FIT-TR-2002-02, Queensland University of Technology, Brisbane, Australia.

[38] Thatte, S. (2001), XLANG: Web Services for Business Process Design, Microsoft Corporation. http://www.gotdotnet.com/team/xml wsspecs /xlang-c/default.htm/

[39] Hofner, Y., Ludwig, H., Gulcu, C. & Grefen, P. (2000), Architecture for Cross-Organizational Business Processes, Research report, IBM, Zurich, Switzerland.