ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ (MSc)
στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Έλεγχος Χρονικής Ακεραιότητας Multimedia Εφαρμογών»

Τσερκέζογλου Σταύρος

M3970015

ΑΘΗΝΑ, ΦΕΒΡΟΥΑΡΙΟΣ 1999

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ (MSc)
στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Έλεγχος Χρονικής Ακεραιότητας Multimedia Εφαρμογών»

Τσερκέζογλου Σταύρος

M3970015

Επιβλέπων Καθηγητής: Μιχάλης Βαζιργιάννης

Ευχαριστίες

Συνήθως πολλοί θεωρούν το τμήμα των ευχαριστιών σαν ένα τμήμα μιας εργασίας, το οποίο απλά πρέπει να υπάρχει για τυπικούς λόγους. Στην περίπτωση της παρούσας εργασίας οι ευχαριστίες δεν αποτελούν το τυπικό της μέρος. Η συνεισφορά των ατόμων που αναφέρονται ήταν σημαντική, καθ' όλη την περίοδο στην οποία διήρκησε η εκπόνηση της εργασίας.

Καταρχήν πρέπει να ευχαριστήσω τον επιβλέποντα καθηγητή αυτής της εργασίας τον κύριο Μιχάλη Βαζιργιάννη. Ολόκληρη η εργασία στηρίχτηκε σε μία ιδέα, ένα τμήμα της οποίας αποτελεί δικό του γέννημα. Η εμπειρία του, ειδικά στο ζήτημα των εφαρμογών multimedia και στο ζήτημα του ελέγχου της χρονικής ακεραιότητας, μου ήταν πολύτιμη. Ακόμα πιο σημαντικό θεωρώ όμως το γεγονός ότι, παρά το φόρτο εργασίας του, ήταν πάντα πρόθυμος να συζητήσουμε τα προβλήματα που παρουσιάστηκαν κατά την υλοποίηση της μεθοδολογίας και έτοιμος να συνεισφέρει με την εμπειρία του στην επίλυσή τους ή να με φέρει σε επαφή με άτομα που μπορούσαν να με βοηθήσουν.

Ειδική μνεία πρέπει να γίνει στην κυρία Barbara Pernici, ένα από τα τρία άτομα που μαζί με τον κύριο Βαζιργιάννη είχαν την αρχική ιδέα για την ανάπτυξη της μεθοδολογίας που χρησιμοποίησα για τον έλεγχο της χρονικής ακεραιότητας. Η υλοποίηση του συστήματος Later την οποία πρόθυμα μου έστειλε ήταν πολύτιμη, ειδικά την πρώτη περίοδο στην οποία σχεδίαζα το μοντέλο της υλοποίησης της μεθοδολογίας.

Τέλος θα ήταν λάθος μου να μην αναφερθώ στο προσωπικό του εργαστηρίου Πληροφοριακών Συστημάτων και Βάσεων Δεδομένων για την προθυμία τους να με βοηθήσουν αλλά και την ανοχή που έδειξαν τις ημέρες που εργάστηκα στο εργαστήριο.

Περιεχόμενα

Σύνοψη

Κατά τη διαδικασία συγγραφής μιας multimedia εφαρμογής είναι δύσκολο να εξασφαλιστεί η χρονική ακεραιότητα του σεναρίου που περιγράφεται. Το πρόβλημα γίνεται περισσότερο έντονο αν πρόκειται για πολύπλοκες εφαρμογές που περιλαμβάνουν αλληλεπίδραση με το χρήστη. Αποτέλεσμα των χρονικών ασυνεπειών είναι η απροσδόκητη συμπεριφορά ορισμένων από τα αντικείμενα της εφαρμογής ή της εφαρμογής στο σύνολό της.

Σκοπός της παρούσας εργασίας είναι να υλοποιήσει ένα εργαλείο που θα δίνει τη δυνατότητα στους συγγραφείς multimedia εφαρμογών να διαπιστώνουν την ακεραιότητα μιας εφαρμογής και να εντοπίζουν πιθανά προβλήματα, κατά τη διάρκεια της διαδικασίας συγγραφής. Το εργαλείο στηρίζεται στη μεθοδολογία Scenario Integrity Checking. Η μεθοδολογία με τη σειρά της βασίζεται στο μοντέλο Interactive Multimedia Document, που χρησιμοποιείται για να αναπαραστήσει εφαρμογές που περιλαμβάνουν user interaction. Η μέθοδος που χρησιμοποιείται για τον έλεγχο της χρονικής ακεραιότητας έχει τις ρίζες της στη θεωρία των δικτύων χρονικών περιορισμών.

Η μεθοδολογία, και επομένως και το εργαλείο που αναπτύχθηκε, αποτελείται από ένα σύνολό διακριτών βημάτων. Σκοπός των βημάτων είναι να εντοπίσουν τους χρονικούς περιορισμούς που συνδέουν τα γεγονότα της εφαρμογής και να τους αναπαραστήσουν χρησιμοποιώντας δίκτυα χρονικών περιορισμών. Στη συνέχεια, σε κάθε δίκτυο εφαρμόζεται η μεθοδολογία Simple Temporal Problem που εντοπίζει χρονικές ασυνέπειες και κατασκευάζει το ελάχιστο δίκτυο της εφαρμογής.

Το κείμενο που ακολουθεί δεν αποτελεί το εγχειρίδιο χρήσης της εφαρμογής. Στόχος του είναι να περιγράψει τα μοντέλα που χρησιμοποιήθηκαν, να αναλύσει το θεωρητικό τους υπόβαθρο και τέλος να παρουσιάσει τον τρόπο με τον οποίο έγινε η υλοποίηση της μεθοδολογίας Scenario Integrity Checking.

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

Βασική λειτουργία των υπολογιστών είναι ο χειρισμός και η παρουσίαση πληροφορίας. Στα πρώτα χρόνια της ύπαρξής τους, η μορφή με την οποία παρουσιαζόταν η πληροφορία ήταν η απλούστερη δυνατή: απλό κείμενο. Το επόμενο βήμα ήταν η παρουσίαση πληροφοριών με τη μορφή εικόνας. Για αρκετά χρόνια οι δυνατότητες παρουσίασης των υπολογιστών ήταν περιορισμένες σε αυτές τις δύο μορφές πληροφορίας, ενώ σε ορισμένες μόνο περιπτώσεις υπολογιστών και εφαρμογών (κυρίως παιχνίδια και προγράμματα σύνθεσης για home computers) γινόταν εκμετάλλευση της δυνατότητας των υπολογιστών να παράγουν ήχο.

Η αλλαγή αυτής της φιλοσοφίας, με την είσοδο νέων μορφών παρουσίασης πληροφορίας, ήταν ένα γεγονός που πολλοί προέβλεπαν ότι θα συνέβαινε. Ωστόσο κανείς δεν μπορούσε να προσδιορίσει την χρονική στιγμή και το αίτιο που θα την προκαλούσε. Ακόμα και σήμερα δεν μπορούμε να προσδιορίσουμε αυστηρά τον τρόπο με τον οποίο έγινε αποδεκτή η τεχνολογία των multimedia. Η τεχνολογία για να παρουσιάζονται ήχοι, εικόνες, animation, video στον υπολογιστή μας είχε ήδη αναπτυχθεί, ωστόσο δεν υπήρχαν εφαρμογές που να την εκμεταλλεύονται. Όπως συμβαίνει στην μέχρι τώρα ιστορία της τεχνολογίας των υπολογιστών, το λογισμικό ακολουθεί σε εξέλιξη το υλικό.

Η εμφάνιση των πρώτων εφαρμογών multimedia ακολούθησε χρονικά την αποδοχή των Windows σαν το λειτουργικό σύστημα που απευθύνεται σε όλους. Η επιτυχία τους ωστόσο οφείλεται στην πτώση των τιμών των υπολογιστών, γεγονός που τους έκανε προσιτούς στο ευρύ κοινό. Σε ένα κοινό που δεν βλέπει τον υπολογιστή σαν εργαλείο εργασίας, αλλά σαν ένα μέσο για να ξοδεύει τον ελεύθερό του χρόνο.

Σήμερα η τεχνολογία των multimedia έχει εισαχθεί σε όλες σχεδόν τις εφαρμογές που παρουσιάζονται σε γραφικό περιβάλλον. Άλλες λιγότερο και άλλες περισσότερο, οι σύγχρονες εφαρμογές χρησιμοποιούν ήχο, video, animation και άλλες

μορφές παρουσίασης για να κάνουν την εμφάνιση της πληροφορίας περισσότερο προσιτή στον κοινό χρήστη. Ωστόσο ο χώρος στον οποίο η τεχνολογία των multimedia βρίσκει πλήρη εκμετάλλευση είναι ο χώρος των εφαρμογών που προσπαθούν να αντικαταστήσουν την εκτύπωση σε χαρτί σαν μορφή παρουσίασης πληροφορίας. Χιλιάδες εφαρμογές έχουν αναπτυχθεί ως σήμερα, για να μεταφέρουν στον υπολογιστή τη γνώση που μέχρι τώρα ήταν "στριμωγμένη" στο χαρτί.

Οι εφαρμογές αυτές παρουσιάζουν το πλεονέκτημα ότι η παρουσίαση της πληροφορίας είναι περισσότερο ζωντανή, ενσωματώνει ήχο και κινούμενη εικόνα και επιπρόσθετα δίνεται η δυνατότητα στο χρήστη να συμμετέχει σε μεγαλύτερο βαθμό, από το να γυρίζει απλά τις σελίδες ενός βιβλίου ή ενός περιοδικού. Ωστόσο ο βαθμός της αλληλεπίδρασης του χρήστη με την εφαρμογή είναι ακόμα περιορισμένος. Συνήθως οι χρήστης μπορεί μόνο να τροποποιεί τη σειρά με την οποία παρουσιάζεται η πληροφορία ή να ζητά πρόσθετη πληροφόρηση για τα σημεία που τον ενδιαφέρουν. Πολλοί ωστόσο θα ήθελαν να επεμβαίνουν, σε ακόμα μεγαλύτερο βαθμό, στον τρόπο με τον οποίο εμφανίζονται τα αντικείμενα μιας εφαρμογής. Για παράδειγμα θα ήταν ενδιαφέρον να διακόπτουν την εμφάνιση ενός video που δεν τους ενδιαφέρει, να επανεμφανίζουν ένα άλλο video ή να αυξάνουν την ταχύτητα παρουσίασής του για να εντοπίσουν ένα σημείο που τους ενδιαφέρει.

Ο τρόπος με τον οποίο κατασκευάζονται οι σημερινές εφαρμογές δεν παρέχει αυτές τις δυνατότητες. Ωστόσο έχει προταθεί ένας αριθμός μοντέλων που καταφέρνουν να παρέχουν έναν μεγάλο αριθμό δυνατοτήτων αλληλεπίδρασης. Ένα από τα πιο ενδιαφέροντα μοντέλα είναι το μοντέλο Interactive Multimedia Document (IMD). Χρησιμοποιώντας το μοντέλο ο συγγραφέας (author) μιας εφαρμογής ορίζει το σενάριο εκτέλεσης μιας multimedia εφαρμογής, που περιλαμβάνει πλήθος multimedia αντικειμένων. Το σενάριο ορίζει τον τρόπο με τον οποίο θα παρουσιαστούν τα επιμέρους αντικείμενα, τη σχέση που τα συνδέει αλλά και τον τρόπο με τον οποίο αλληλεπιδρούν κατά τη διάρκεια της εκτέλεσης της εφαρμογής. Επιπρόσθετα, το σενάριο ορίζει τον τρόπο με τον οποίο ο χρήστης μπορεί να επέμβει και να τροποποιήσει τη ροή εκτέλεσης της εφαρμογής.

Βασική μονάδα στον ορισμό ενός IMD σεναρίου είναι τα γεγονότα (events). Τα γεγονότα χρησιμοποιούνται για να προσδιορίσουν τον τρόπο εκτέλεσης μιας εφαρμογής. Ουσιαστικά ένα IMD σενάριο ορίζει τον τρόπο με τον οποίο αντιδρά η εφαρμογή όταν δέχεται συγκεκριμένα γεγονότα. Συνολικά το μοντέλο μπορεί να θεωρηθεί σαν ένα event-based μοντέλο.

Οι εφαρμογές που μοντελοποιούνται με τη χρήση του IMD μπορούν να έχουν αυξημένη πολυπλοκότητα. Η πολυπλοκότητα των εφαρμογών σχετίζεται με τον αριθμό των αντικειμένων που λαμβάνουν μέρος στην παρουσίαση, τους μετασχηματισμούς που πραγματοποιούνται σε αυτά, τις σχέσεις που συνδέουν τα αντικείμενα και βέβαια το βαθμό της αλληλεπίδρασής τους με το χρήστη. Ο πιο σημαντικός από τους τέσσερις αυτούς παράγοντες είναι ο τελευταίος. Πράγματι, οι ενέργειες του χρήστη -για να είμαστε πιο ακριβείς, τα γεγονότα που προκαλούνται από τις ενέργειες του χρήστη- τροποποιούν τη ροή εκτέλεσης της εφαρμογής, δημιουργώντας διαφορετικά μονοπάτια εκτέλεσης (execution paths). Ωστόσο υπάρχει μία κατηγορία εφαρμογών multimedia που δεν περιλαμβάνει αλληλεπίδραση με το χρήστη. Πρόκειται για τις

προκαθορισμένες (preorchistrated) εφαρμογές, οι οποίες ορίζουν ένα και μόνο μονοπάτι εκτέλεσης και φυσικά έχουν μειωμένη πολυπλοκότητα.

Το μοντέλο IMD μπορεί να χρησιμοποιηθεί για να ορίσει τόσο preorchistrated εφαρμογές όσο και εφαρμογές που περιλαμβάνουν αλληλεπίδραση με το χρήστη, ανεξάρτητα από το βαθμό πολυπλοκότητάς τους. Ωστόσο όσο αυξάνει ο βαθμός πολυπλοκότητας, είναι δύσκολο για τον συγγραφέα της εφαρμογής να διατηρήσει μια λεπτομερής εικόνα του τρόπου με τον οποίο εξελίσσεται η παρουσίαση των αντικειμένων της. Υπάρχει η πιθανότητα, το σενάριο να περιέχει χρονικές ασυνέπειες, με αποτέλεσμα ορισμένα τμήματα της εφαρμογής να μην παρουσιαστούν στο χρήστη ή η εφαρμογή να έχει απροσδόκητη συμπεριφορά. Για παράδειγμα τι θα συμβεί αν ο χρήστης προσπαθήσει να σταματήσει την παρουσίαση ενός video το οποίο έχει ολοκληρωθεί ή αν προσπαθήσουμε να παίξουμε ταυτόχρονα δύο ήχους;

Σκοπός της παρούσας εργασίας είναι η υλοποίηση ενός εργαλείου που θα αναλαμβάνει τον έλεγχο της χρονικής ακεραιότητας ενός σεναρίου (temporal integrity - temporal consistency). Το εργαλείο χρησιμοποιείται κατά τη διαδικασία συγγραφής, δέχεται ένα σενάριο, το αναλύει και εντοπίζει πιθανές χρονικές ασυνέπειες. Η χρησιμότητα ενός εργαλείου για τον έλεγχο της ακεραιότητας ενός απλού σεναρίου είναι μάλλον περιορισμένη, αφού ο ίδιος ο συγγραφέας μπορεί εύκολα να εντοπίσει χρονικά λάθη. Ωστόσο.σε εφαρμογές με αυξημένη πολυπλοκότητα, είναι πολύ δύσκολο να προσδιοριστούν όλα τα πιθανά μονοπάτια εκτέλεσης και να εξεταστεί αξιόπιστα η χρονική τους ακεραιότητα. Σε τέτοιες εφαρμογές υπάρχει η ανάγκη για την ύπαρξη ενός εργαλείου που θα αναλαμβάνει να εντοπίσει τις χρονικές ασυνέπειες του σεναρίου.

Ο χώρος των χρονικών προβλημάτων (temporal problems) αποτελεί έναν από τους πιο ενδιαφέροντες τομείς της τεχνητής νοημοσύνης (artificial intelligence). Τα ζητήματα που ανήκουν στο χώρο αυτό χαρακτηρίζονται από αυξημένη πολυπλοκότητα. Ο έλεγχος της χρονικής ακεραιότητας, τον οποίο υλοποιεί η εφαρμογή που κατασκευάσαμε στηρίζεται σε μία μεθοδολογία που έχει προταθεί από τους Mirbel, Pernici, Selli και Vazirgianni [MPSV98].

Η προσέγγιση που προτείνεται βασίζεται σε προηγούμενες εργασίες που έχουν γίνει για τη μοντελοποίηση IMD εφαρμογών [VB97, VTS98]. Σύμφωνα με αυτές, δομική μονάδα ενός IMD σεναρίου είναι το tuple. Το tuple αποτελεί μια αυτόνομη μονάδα η οποία ορίζει τον τρόπο με την οποίο θα αντιδράσει η εφαρμογή, όταν δεχθεί ένα γεγονός. Κάθε tuple ορίζει έναν αριθμό πράξεων (Temporal Access Control actions, TAC) που εκτελούνται στα αντικείμενα της εφαρμογής καθώς και τα γεγονότα που θα προκαλέσουν την εκκίνηση και την διακοπή του.

Όσον αφορά την μεθοδολογία που υιοθετείται για την επίλυση των χρονικών προβλημάτων, στηρίζεται στη εργασία των Dechter, Meiri και Pearl για την εξέταση απλών χρονικών προβλημάτων (Simple Temporal Problems, STP) [DMP91]. Κάθε σενάριο μεταφράζεται σε έναν αριθμό δικτύων χρονικών περιορισμών (Temporal Constraint Networks, TCN) που αναπαριστά τους περιορισμούς που υπάρχουν μεταξύ των γεγονότων ενός μονοπατιού εκτέλεσης. Στη συνέχεια, τα δίκτυα χρησιμοποιούνται για να εντοπιστούν οι χρονικές ασυνέπειες.

Σκοπός των κεφαλαίων που ακολουθούν είναι να παρουσιάσουν το μοντέλο και τη μεθοδολογία που χρησιμοποιείται, πριν αναλυθεί ο τρόπος με τον οποίο έγινε η

υλοποίηση του εργαλείου ελέγχου της χρονικής ακεραιότητας. Είναι λάθος να θεωρήσουμε την εργασία σαν το εγχειρίδιο χρήσης της εφαρμογής. Σε κανένα σημείο της δεν υπάρχει αναφορά στον τρόπο με τον οποίο χρησιμοποιείται η εφαρμογή, αφού άλλωστε δεν είναι αυτός ο στόχος της. Στόχος της είναι να παρουσιάσει το θεωρητικό υπόβαθρο στο οποίο στηρίχτηκε η υλοποίηση του εργαλείου ελέγχου της χρονικής ακεραιότητας.

Εκτός από αυτή την σύντομη εισαγωγή, η εργασία περιλαμβάνει τέσσερα ακόμα κεφάλαια, καθώς και ένα παράρτημα:

Το κεφάλαιο 2 περιλαμβάνει μια λεπτομερής παρουσίαση του μοντέλου IMD, στο οποίο βασίζεται η μεθοδολογία. Παρουσιάζονται όλα τα στοιχεία του μοντέλου, τα πλεονεκτήματα και τα μειονεκτήματά του ενώ γίνεται και μία σύγκριση με τα άλλα μοντέλα που έχουν προταθεί. Επίσης το κεφάλαιο περιλαμβάνει ένα παράδειγμα μιας απλής IMD εφαρμογής, με στόχο να γίνει περισσότερο κατανοητή η φιλοσοφία του μοντέλου.

Το κεφάλαιο 3 αναλύει τη μεθοδολογία Scenario Integrity Checking στην οποία βασίζεται το εργαλείο. Η μεθοδολογία αποτελείται από τέσσερα διακριτά βήματα, τα οποία και θα παρουσιαστούν αναλυτικά με την πρόσθετη βοήθεια ενός απλού παραδείγματος. Επίσης το κεφάλαιο περιλαμβάνει μια σύντομη παρουσίαση άλλων μεθοδολογιών ελέγχου της χρονικής ακεραιότητας καθώς και τη σύγκρισή τους με τη μεθοδολογία που χρησιμοποιείται.

Το κεφάλαιο 4 παρουσιάζει την υλοποίηση της μεθοδολογίας Scenario Integrity Checking. Φυσικά είναι αδύνατο να γίνει πλήρης ανάλυση της υλοποίησης που έχει πραγματοποιηθεί. Παρουσιάζονται τα βασικά της στοιχεία καθώς και οι αλγόριθμοι που χρησιμοποιήθηκαν σε κάθε ένα από τα τέσσερα βήματα.

Το κεφάλαιο 5 κλείνει το βασικό τμήμα της εργασίας παρουσιάζοντας τα συμπεράσματα που προέκυψαν και δίνοντας κάποιες βασικές κατευθύνσεις για μελλοντικές επεκτάσεις.

Το παράρτημα Α τέλος παρουσιάζει τις αποδείξεις κάποιων μαθηματικών θεωρημάτων που δεν κρίθηκε σκόπιμο να ενσωματωθούν στο κύριο μέρος της εργασίας.

ΚΕΦΑΛΑΙΟ 2

Το Μοντέλο Interactive Multimedia Document

Ο χώρος των εφαρμογών multimedia γνωρίζει πολύ μεγάλη άνθηση την τελευταία δεκαετία. Έχουν παρουσιαστεί πολλά εμπορικά πακέτα για την ανάπτυξη εφαρμογών multimedia και ακόμα περισσότερες εφαρμογές. Κοινό χαρακτηριστικό των εφαρμογών που κατασκευάζονται είναι η πλήρης εκμετάλλευση των διαφορετικών τύπων αντικειμένων multimedia (ήχου, εικόνας, κινούμενης εικόνας, video, κειμένου κ.ο.κ) αλλά και η ελλιπής αλληλεπίδραση των αντικειμένων. Επιπρόσθετα η ροή εκτέλεσης της εφαρμογής είναι αυστηρά καθορισμένη και οι δυνατότητες που δίνονται στο χρήστη να επέμβει σε αυτή είναι περιορισμένες.

Βασική αιτία για αυτές τις αδυναμίες των συνηθισμένων εφαρμογών multimedia είναι η έλλειψη ενός μοντέλου που θα υποστηρίζει την αλληλεπίδραση των αντικειμένων που συμμετέχουν στην εφαρμογή αλλά και θα περιγράφει την επίδραση διαφόρων γεγονότων και ενεργειών του χρήστη στην ροή εκτέλεσής της. Τα τελευταία χρόνια έχουν παρουσιαστεί αρκετά μοντέλα που προσπαθούν να καλύψουν αυτές τις αδυναμίες με στόχο να αυξήσουν την λειτουργικότητα των εφαρμογών multimedia και να δώσουν τη δυνατότητα στους χρήστες τους να επεμβαίνουν στη ροή εκτέλεσης.

Τα μοντέλα μπορούν να χωριστούν σε δύο μεγάλες κατηγορίες. Η πρώτη κατηγορία περιλαμβάνει μοντέλα με περιορισμένες δυνατότητες επίδρασης του χρήστη στη ροή εκτέλεσης της εφαρμογής. Για το λόγο αυτό οι εφαρμογές που μοντελοποιούν ονομάζονται *προκαθορισμένες* (*preorchistrated*). Η δεύτερη κατηγορία περιλαμβάνει λιγότερα μοντέλα. Βασικό χαρακτηριστικό τους είναι ότι στηρίζονται στην έννοια των γεγονότων που αναγνωρίζονται από την εφαρμογή και μεταβάλουν την ροή εκτέλεσής της.

Σε ένα από τα μοντέλα της δεύτερης κατηγορίας, το μοντέλο *Interactive Multimedia Document* (IMD) *scenario model*, στηρίζεται η παρούσα εργασία. Σκοπός

του κεφαλαίου αυτού είναι να παρουσιάσει όλες τις πλευρές του μοντέλου IMD, όσον αφορά τις δυνατότητές του για χρονική μοντελοποίηση μιας multimedia εφαρμογής. Η εργασία δεν εξετάζει τη χωρική σχέση των αντικειμένων μιας εφαρμογής και επομένως δεν υπάρχει λόγος να επιμείνουμε στην παρουσίαση των χωρικών χαρακτηριστικών του μοντέλου. Η παράγραφος 2.1 αποτελεί μια σύντομη εισαγωγή στις εφαρμογές multimedia με στόχο να αποδείξει την αναγκαιότητα μοντελοποίησής τους. Η παράγραφος 2.2 παρουσιάζει με συντομία τα πιο σημαντικά από τα μοντέλα που έχουν ως τώρα παρουσιαστεί για την μοντελοποίηση multimedia εφαρμογών. Η παράγραφος 2.3 είναι αφιερωμένη στην παρουσίαση του μοντέλου IMD. Τέλος στην παράγραφο 2.4 παρουσιάζεται ένα παράδειγμα μιας εφαρμογής multimedia και ο τρόπος με τον οποίο περιγράφεται χρησιμοποιώντας το μοντέλο IMD.

2.1 Εισαγωγή στις εφαρμογές multimedia

Οι εφαρμογές multimedia μπορούν να παρουσιάζουν εξαιρετική πολυπλοκότητα, ανάλογα με τον αριθμό και το είδος των αντικειμένων που συμμετέχουν σε αυτές. Τα αντικείμενα μπορούν να αναπαριστούν ήχο, εικόνα, κινούμενη εικόνα, απλό κείμενο ή οποιοδήποτε άλλο είδος μπορεί να χειριστεί ένας υπολογιστής. Φυσικά τα αντικείμενα δεν παρουσιάζονται απλά στον χρήστη άλλα έχουν την δυνατότητα να αλλάζουν μορφή, να αλληλεπιδρούν μεταξύ τους, να αντιδρούν με διαφορετικό τρόπο στις επιλογές του χρήστη. Πρόκειται δηλαδή για ενεργά αντικείμενα και όχι για αντικείμενα με παθητικό χαρακτήρα. Από το χαρακτηριστικό αυτό μπορεί να προκύψει ένας άτυπος ορισμός μιας multimedia εφαρμογής [VS96]:

Ορισμός 2.1 *Μια εφαρμογή multimedia (multimedia application, MAP) μπορεί να θεωρηθεί σαν μια σκηνή στην οποία παρουσιάζεται ένα σύνολο αντικειμένων που μετασχηματίζονται, αλληλεπιδρούν και δέχονται την επίδραση των πράξεων του χρήστη για τους σκοπούς της εφαρμογής.*

Είναι φανερό ότι σε μια εφαρμογή με μεγάλο αριθμό αντικειμένων είναι πολύ δύσκολο να περιγραφεί πλήρως η λειτουργικότητα της εφαρμογής αλλά και όλα τα πιθανά σενάρια που μπορούν να ακολουθηθούν, ανάλογα με τις επιλογές του χρήστη. Αυτή η αυξημένη πολυπλοκότητα αντικατοπτρίζεται και στη διαδικασία ανάπτυξης-συγγραφής (*authoring*) μιας multimedia εφαρμογής. Μάλιστα το πρόβλημα είναι ακόμα μεγαλύτερο αφού συνήθως το άτομο που έχει την ευθύνη ανάπτυξης της εφαρμογής δεν έχει σημαντικές τεχνικές γνώσεις. Είναι επομένως απαραίτητο να υπάρχει ένα μοντέλο υψηλού επιπέδου που να δίνει τη δυνατότητα σε άτομα με ελλιπείς τεχνικές γνώσεις να περιγράφουν με ακρίβεια και να αναπτύσσουν πολύπλοκες multimedia εφαρμογές.

Μείζονος σημασίας ζήτημα στην αποτελεσματική αναπαράσταση multimedia εφαρμογών αποτελεί η περιγραφή των χρονικών και χωρικών σχέσεων που συνδέουν τα αντικείμενα της εφαρμογής αλλά και η αντίδραση της εφαρμογής στα γεγονότα (events) που δημιουργούνται. Επομένως μια εφαρμογή μπορεί να θεωρηθεί σαν ένα περιβάλλον που περιλαμβάνει ένα μεγάλο αριθμό γεγονότων τα οποία καθορίζουν τη ροή εκτέλεσης της εφαρμογής. Για παράδειγμα γεγονότα μιας εφαρμογής μπορούν να

θεωρηθούν η ολοκλήρωση ενός video, η λήξη ενός μετρητή χρόνου, η επαφή δύο αντικειμένων στο χώρο ή το πάτημα ενός πλήκτρου από το χρήστη. Κάθε ένα από αυτά τα γεγονότα γίνεται αντιληπτό από την εφαρμογή και ο συγγραφέας μπορεί να ορίσει την αντίδραση που θα επιφέρει. Για παράδειγμα το πάτημα ενός πλήκτρου από το χρήστη της εφαρμογής μπορεί να εμφανίσει στην οθόνη ένα νέο αντικείμενο. Επιπρόσθετα ο συγγραφέας της εφαρμογής θα πρέπει να μπορεί να ορίζει σύνθετα γεγονότα ώστε να εκφράζει πιο πολύπλοκες υποθέσεις. Για παράδειγμα την ταυτόχρονη δημιουργία δύο διαφορετικών γεγονότων.

Η πλήρης και αποτελεσματική μοντελοποίηση εφαρμογών multimedia παραμένει ένα ανοιχτό ζήτημα. Σύμφωνα με τις παραπάνω απόψεις ένα μοντέλο που θα περιγράφει μια εφαρμογή multimedia που περιλαμβάνει αλληλεπίδραση (*interaction*) με το χρήστη θα πρέπει να ικανοποιεί απαραίτητα τα παρακάτω χαρακτηριστικά:

- ακριβής περιγραφή των αντικειμένων που συμμετέχουν στην εφαρμογή των μετασχηματισμών τους καθώς και των χρονικών και χωρικών σχέσεων που τα συνδέουν
- ακριβής περιγραφή της λειτουργικότητας της εφαρμογής, δηλαδή της ροής εκτέλεσής της και της επίδρασης των γεγονότων σε αυτή
- υποστήριξη σύνθετων γεγονότων

Τα μοντέλα που έχουν αναπτυχθεί ικανοποιούν κάποια από τα παραπάνω χαρακτηριστικά. Το κοινό τους χαρακτηριστικό είναι ότι περιγράφουν με ακρίβεια τα αντικείμενα που συμμετέχουν σε μια εφαρμογή, περιγράφουν τη λειτουργικότητα της εφαρμογής ωστόσο τα περισσότερα δεν υποστηρίζουν σύνθετα γεγονότα ούτε καθορίζουν την επίδραση των ενεργειών του χρήστη στη λειτουργικότητα της εφαρμογής.

Το μοντέλο στο οποίο στηρίζεται αυτή η εργασία και το οποίο θα περιγραφεί αναλυτικά στις παραγράφους που ακολουθούν ικανοποιεί σε μεγάλο βαθμό και τις τρεις βασικές προϋποθέσεις που τέθηκαν. Πρόκειται για το Interactive Multimedia Document scenario model (IMD scenario model). Το μοντέλο υιοθετεί την έννοια του σεναρίου (scenario) που στην ουσία είναι μια περιγραφή της λειτουργικότητας μια multimedia εφαρμογής. Βασικά υλικά ενός σεναρίου είναι:

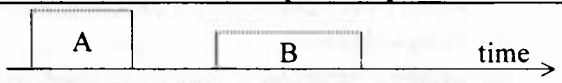
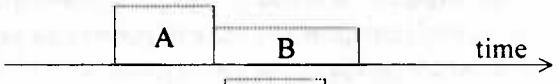
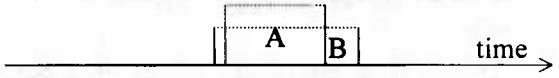
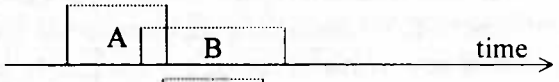
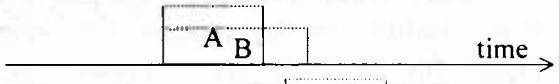
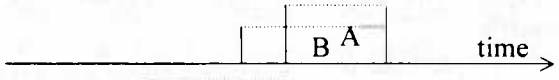
- τα αντικείμενα (actors) που συμμετέχουν στην εφαρμογή [Vaz96]
- οι πράξεις (actions) που εκτελούνται στα αντικείμενα [LG93]
- τα γεγονότα (events) που επιδρούν στην ροή εκτέλεσης της εφαρμογής [VB97]

Το σενάριο χωρίζεται σε μικρότερα τμήματα (scenario tuples) που ορίζουν την αντίδραση της εφαρμογής σε ένα απλό η σύνθετο γεγονός και επομένως καθορίζουν την ροή εκτέλεσης. Το μοντέλο στηρίζεται σε προηγούμενες εργασίες που έχουν γίνει [VH93, VH95, VM93, VS96, VTS98] και έχει επεκταθεί ώστε να υποστηρίζει σε μεγαλύτερο βαθμό τις προϋποθέσεις που τέθηκαν.

2.2 Μοντέλα εφαρμογών multimedia

Κατά το παρελθόν ο όρος συγχρονισμός (*synchronization*) έχει χρησιμοποιηθεί κατά κόρον για να περιγράψει την χρονική ακολουθία των αντικειμένων που συμμετέχουν σε μια multimedia εφαρμογή [LG93]. Η περιγραφή μιας εφαρμογής θα πρέπει να περιλαμβάνει ωστόσο τόσο την χρονική όσο και την χωρική σύνθεση (*composition*) των αντικειμένων που συμμετέχουν σε μια εφαρμογή.

Πολλά από τα υπάρχοντα μοντέλα για την περιγραφή multimedia εφαρμογών βασίζονται στο σύνολο των 13 χρονικών σχέσεων [Ham72, All83]: *before, meets, during, overlaps, starts, ends, equal* και τα αντίστροφα αυτών[1]. Ο πίνακας 2.1 περιγράφει τις εφτά αυτές σχέσεις χρησιμοποιώντας δύο αντικείμενα με συγκεκριμένες χρονικές διάρκειες (για παράδειγμα ένα video).

Χρονική Σχέση	Αναπαράσταση
A *before* B	
A *meets* B	
A *during* B	
A *overlaps* B	
A *starts* B	
A *ends* B	
A *equal* B	

Πίνακας 2.1: Οι χρονικές σχέσεις μεταξύ δύο αντικειμένων.

Οι σχέσεις αυτές μπορούν να εκφράσουν τη χρονική σχέση μεταξύ δύο αντικειμένων, ωστόσο δεν είναι κατάλληλες για να περιγράψουν τη χρονική τους σύνθεση αφού μπορούν να δημιουργήσουν χρονικές αντιφάσεις. Πιο συγκεκριμένα δύο είναι τα βασικά προβλήματα που προκύπτουν όταν προσπαθούμε να χρησιμοποιήσουμε το παραπάνω σύνολο σχέσεων [DK95]:

- Οι σχέσεις έχουν σχεδιαστεί για να εκφράσουν την σχέση μεταξύ χρονικών διαστημάτων συγκεκριμένης διάρκειας. Στην περίπτωση των εφαρμογών multimedia μια σχέση μεταξύ δύο αντικειμένων πρέπει να ισχύει ανεξαρτήτως της διάρκειας του κάθε αντικειμένου. Για παράδειγμα ας πάρουμε τη σχέση A equal B. Αν η διάρκεια του αντικειμένου A μικρύνει, η

[1] Η σχέση equal δεν έχει αντίστροφη σχέση.

σχέση δεν ισχύει πια και μεταβαίνουμε στη σχέση A starts B. Ακόμα πιο χαρακτηριστική είναι η περίπτωση της σχέσης A before B. Ανάλογα με τη διάρκεια του αντικειμένου A ισχύουν οι παραπάνω σχέσεις: A meets B, A ovelaps B, B during A.

- Ο περιγραφικός χαρακτήρας των σχέσεων δεν εκφράζει τη σχέση αιτίας και αποτελέσματος

Τα υπόλοιπα μοντέλα για την αναπαράσταση της χρονική σύνθεση αντικειμένων μπορούν να καταταγούν [DK95] σε δύο κατηγορίες: *point-based* μοντέλα και *interval-based* μοντέλα. Στα μοντέλα της πρώτης κατηγορίας οι βασικές μονάδες για την έκφραση των σχέσεων μεταξύ των αντικειμένων είναι τα σημεία στο χρόνο και το χώρο. Κάθε γεγονός σχετίζεται με ένα σημείο στο χρόνο εκτέλεσης της εφαρμογής. Στη συνέχεια τα χρονικά αυτά σημεία συνδέονται χρησιμοποιώντας ένα σύνολο σχέσεων (για παράδειγμα αναφέρουμε τις σχέσεις *precede, simultaneous* και *after*) και σχηματίζουν μια περιγραφή της εφαρμογής. Τα interval-based μοντέλα θεωρούν τα αντικείμενα που συμμετέχουν σε μια εφαρμογή σαν απλά χρονικά διαστήματα, τα οποία στη συνέχεια διατάσσονται στο χρόνο χρησιμοποιώντας και πάλι ένα σύνολο σχέσεων. Τα μοντέλα που ανήκουν στην κατηγορία αυτή χρησιμοποιούν κυρίως το υποσύνολα ή υπερσύνολα των σχέσεων που ήδη περιγράψαμε [Ham72, All83].

Ένας αρκετά ενδιαφέρον μηχανισμός για τη χρονική σύνθεση αντικειμένων παρουσιάστηκε στο [DK95]. Το μοντέλο λαμβάνει υπόψη του τη χρονική σχέση μεταξύ των αντικειμένων και καταλήγει στον ορισμό ενός συνόλου χρονικών τελεστών (*operators*). Ωστόσο το μοντέλο αυτό δεν παρουσιάζει δυνατότητα επίδρασης του χρήστη, επομένως ανήκει στην κατηγορία των preorchistrated μοντέλων.

Το μοντέλο που παρουσιάζεται στο [HFK95] υποστηρίζει ενέργειες που εκτελούνται από το χρήστη και μεταβάλλουν την ροή της εφαρμογής. Χρησιμοποιεί μια point-based προσέγγιση και παρέχει δυνατότητες για τον ορισμό σύγχρονων και ασύγχρονων χρονικών συνθέσεων. Για την αναπαράστασή τους χρησιμοποιεί ένα δέντρο, κάθε υποδέντρο του οποίου αναπαριστά ένα διαφορετικό μονοπάτι που μπορεί να ακολουθήσει η εφαρμογή, ανάλογα με τις ενέργειες που εκτελεί ο χρήστης.

Άλλες προσεγγίσεις χρησιμοποιούν το σύνολο των σχέσεων του Allen για να ορίσουν ένα σχήμα multimedia βάσης δεδομένων. Για παράδειγμα οι Little και Ghafoor προτείνουν την υιοθέτηση Petri Nets στο μοντέλο OCPN (Object Composition Petri Nets) που παρουσιάζουν [LG93]. Το μοντέλο αυτό όχι μόνο δεν υποστηρίζει user interaction αλλά υποθέτει ότι οι διάρκεια παρουσίασης κάθε αντικειμένου θα είναι προκαθορισμένη. Άλλες προσεγγίσεις [Kin94] βασίζονται σε ένα ισχυρό μαθηματικό υπόβαθρο με στόχο την αυστηρή περιγραφή του μοντέλου. Ωστόσο σε τέτοιες προσεγγίσεις η περιγραφή των εφαρμογών είναι ελλιπής και δύστροπη και δυσκολεύει τη διαδικασία ανάπτυξης.

Στο [Han96] παρουσιάζεται ένα μοντέλο που στηρίζεται στην έννοια του συγχρονισμού. Το μοντέλο καλύπτει πολλά ζητήματα όπως ιεραρχικός συγχρονισμός, αναπαράσταση της ροής εκτέλεσης της εφαρμογής με χρήση γράφων, επιλογή διαφορετικών σεναρίων εκτέλεσης ανάλογα με τις ενέργειες του χρήστη, υποστήριξη γεγονότων, πρόβλεψη χρόνου εκτέλεσης της εφαρμογής και άλλα. Δεν υπάρχουν

χρονικοί περιορισμοί στα γεγονότα ωστόσο το μοντέλο δεν υποστηρίζει σύνθεση γεγονότων.

Ένα ακόμα μοντέλο που υποστηρίζει εν μέρει user interaction παρουσιάστηκε στο [SD96]. Το μοντέλο στηρίζεται επίσης στην έννοια του συγχρονισμού. Ωστόσο είναι από τα λίγα μοντέλα που προτείνει την έννοια των καταστάσεων των αντικειμένων (*object states*) που συμμετέχουν στην παρουσίαση της εφαρμογής. Το μοντέλο προτείνει 5 διακριτές καταστάσεις στις οποίες μπορεί να βρίσκεται ένα αντικείμενο: *Idle, Ready, In-Process, Finished, Complete*. Παρά το γεγονός ότι το μοντέλο δεν υποστηρίζει γεγονότα, μπορεί να χειριστεί καθορισμένες ενέργειες που γίνονται από τον χρήστη. Πιο συγκεκριμένα υποστηρίζει ενέργειες που σχετίζονται με το πάτημα πλήκτρων και ενέργειες αποφυγής αντικειμένων (*user skip*) όπως *forward* και *backward*.

Μια άλλη κατηγοριοποίηση μοντέλων για την αναπαράσταση multimedia εφαρμογών [Blak96] ορίζει μια νέα κατηγορία που στηρίζεται στην έννοια των γεγονότων. Δύο από τα πιο σημαντικά μοντέλα που στηρίζονται στα γεγονότα είναι τα HyTime[Buf96, ISO92] και HyperODA.

Τα γεγονότα στο μοντέλο HyTime δημιουργούνται κατά την παρουσίαση ή την ολοκλήρωση της παρουσίασης ενός αντικειμένου και εμπεριέχουν εκτός των άλλων τα χαρακτηριστικά της παρουσίασης. Το HyTime προτείνει ένα πλήρες μοντέλο για τον χρονικό και χωρικό προσδιορισμό των γεγονότων που καλείται FCS (*Finite Coordinate Space*). Το μοντέλο FCS παρέχει ένα εργαλείο στο συγγραφέα multimedia εφαρμογών ώστε να ορίσει με ακρίβεια τη θέση κάθε αντικειμένου στο χρόνο και το χώρο καθώς και τη σχέση του με τα υπόλοιπα αντικείμενα.

Παρόμοια είναι και η προσέγγιση του HyperODA. Τα γεγονότα σχετίζονται κυρίως με την εμφάνιση ενός αντικειμένου στην σκηνή (scene) της εφαρμογής ή την ολοκλήρωση της εμφάνισης. Επιπρόσθετα ορίζονται και αντικείμενα-μετρητές τα οποία επίσης προκαλούν events σε καθορισμένες χρονικές στιγμές. Τα δύο αυτά μοντέλα μπορούν να χρησιμοποιηθούν για να αναπαραστήσουν την χρονική σύνθεση αντικειμένων. Ωστόσο "*...η υποστήριξη γεγονότων - και κατά συνέπεια ενεργειών του χρήστη - είναι ελλιπής, δεν υποστηρίζεται σύνθεση γεγονότων και επιπρόσθετα δεν υπάρχει δυνατότητα υποστήριξης μιας scripting γλώσσας*" [Buf96].

Το μοντέλο MHEG [ISO93] είναι ένα ακόμα από τα μοντέλα που υποστηρίζουν user interaction. Ωστόσο οι δυνατότητες του είναι περιορισμένες αφού στην πραγματικότητα οι ενέργειες που υποστηρίζονται επιλέγονται από ένα καθορισμένο σύνολο. Η επιλογή καθορίζει τη συνέχεια της εκτέλεσης του σεναρίου. Το μοντέλο δίνει και μια άλλη δυνατότητα αλληλεπίδρασης, που ονομάζεται *modification interaction*. Ο χρήστης έχει την δυνατότητα να επιλέξει κατά τη διάρκεια εκτέλεσης της εφαρμογής συγκεκριμένα χαρακτηριστικά που αφορούν την εμφάνιση των αντικειμένων. Για παράδειγμα μπορεί να επιλέξει μέσα από έναν κατάλογο το επόμενο video που θα παρουσιαστεί ή να επιλέξει την ένταση στην οποία ακούγεται ένας ήχος.

2.3 Το μοντέλο Interactive Multimedia Document

Ένα σύστημα για να μπορέσει να υποστηρίξει πολύπλοκες multimedia εφαρμογές πρέπει να παρέχει όχι μόνο δυνατότητες μοντελοποίησης υψηλού επιπέδου αλλάκαι να υποστηρίζει δυνατότητες user interaction. Το μοντέλο θα πρέπει να υποστηρίζει την χρονική και χωρική σύνθεση (*spatiotemporal composition*) των multimedia αντικειμένων, δυνατότητες ορισμού της επίδρασης των ενεργειών του χρήστη στην ροή εκτέλεσης του σεναρίου, δυνατότητες ορισμού της αλληλεπίδρασης μεταξύ των αντικειμένων καθώς και δυνατότητες συγχρονισμού της παρουσίασης των αντικειμένων τόσο στο χώρο όσο και στο χρόνο.

Από την προηγούμενη παράγραφο πρέπει να έγινε φανερό ότι τα υπάρχοντα μοντέλα πάσχουν κυρίως στην υποστήριξη αλληλεπίδρασης. Αυτό ακριβώς είναι το σημείο στο οποίο διαφέρει το μοντέλο Interactive Multimedia Document Scenario Model το οποίο θα περιγράψουμε στην παράγραφο αυτή. Εκτός από τις δυνατότητες σύνθεσης και συγχρονισμού το IMD υποστηρίζει *εξωτερικές ενέργειες* (*external interaction*) που προέρχονται από το χρήστη κατά τη διάρκεια της εκτέλεσης της εφαρμογής. Με τον όρο εξωτερικές ενέργειες δεν εννοούμε μόνο το συνηθισμένο πάτημα ενός πλήκτρου του πληκτρολογίου ή ενός από τα κουμπιά που εμφανίζονται στην οθόνη της εφαρμογής. Ο όρος καλύπτει ενέργειες όπως scrolling, drag and drop καθώς και ενέργειες που επιδρούν άμεσα στην κατάσταση ενός αντικειμένου της εφαρμογής (για παράδειγμα start actor, stop actor).

Επιπρόσθετα το μοντέλο υποστηρίζει εσωτερικές ενέργειες (*internal interaction*) που παρέχουν τη δυνατότητα ορισμού της αλληλεπίδρασης μεταξύ των αντικειμένων της εφαρμογής. Για παράδειγμα ο συγγραφέας ενός σεναρίου μπορεί να ορίσει ότι το τέλος της παρουσίασης ενός video προκαλεί την εμφάνιση ενός κειμένου και ταυτόχρονα ενός ήχου.

Είναι φανερό ότι το μοντέλο στηρίζεται στην έννοια των γεγονότων (*events*) τα οποία προκαλούνται από εσωτερικές ή εξωτερικές ενέργειες και καθορίζουν την εξέλιξη της εφαρμογής. Τα γεγονότα συνδέονται με τα αντικείμενα που παρουσιάζονται κατά την εκτέλεση της εφαρμογής και ονομάζονται στην ορολογία του IMD, *actors*. Ένα αντικείμενο μιας εφαρμογής μπορεί να βρίσκεται σε μία από τέσσερις διαφορετικές καταστάσεις εκτέλεσης:

- *Active*, που υποδηλώνει ότι το αντικείμενο είναι ενεργό. Η έννοια "ενεργός" μεταφράζεται με διαφορετικό τρόπο ανάλογα με το είδος του αντικειμένου. Για παράδειγμα ένα video είναι ενεργό όσο παίζεται στην οθόνη.

- *Idle*, που είναι η αντίθετη κατάσταση της Active. Όλα τα αντικείμενα είναι αρχικά σε κατάσταση Idle.

- *Paused-Suspended*, που είναι μια ενδιάμεση κατάσταση. Το αντικείμενο συνεχίζει είναι ενεργό αλλά η εμφάνισή του έχει ακυρωθεί. Και πάλι η κατάσταση αυτή έχει διαφορετική σημασία για τα διαφορετικά αντικείμενα. Θέτοντας ένα video σε κατάσταση paused σταματούμε την εξέλιξή του.

- Θέτοντας ένα κουμπί σε κατάσταση paused το απενεργοποιούμε, επομένως ο χρήστης δεν έχει τη δυνατότητα να το πατήσει.

Η μετάβαση των αντικειμένων από τη μία κατάσταση στην άλλη γίνεται μέσω των ενεργειών (*actions*). Το μοντέλο ορίζει ένα σύνολο ενεργειών καθώς και τις μεταβάσεις που αυτές προκαλούν.

Το μοντέλο τέλος βασίζεται σε επιμέρους αυτόνομες μονάδες οι οποίες ορίζουν την συμπεριφορά της εφαρμογής στα διάφορα γεγονότα που δέχεται. Οι μονάδες αυτές ονομάζονται *scenario tuples* και στην ουσία είναι αυτές που καθορίζουν την ροή εκτέλεσης της εφαρμογής.

Αυτές είναι εν συντομία οι τέσσερις βασικές έννοιες (events, actors, actions, tuples) στις οποίες στηρίζεται το μοντέλο IMD και που θα μας απασχολήσουν στις επόμενες παραγράφους. Όπως προαναφέραμε, το μοντέλο υποστηρίζει τόσο τις χρονικές όσο και τις χωρικές σχέσεις των αντικειμένων. Ωστόσο θα δώσουμε περισσότερη προσοχή στην ανάλυση των χρονικών σχέσεων αφού αυτές θα μας απασχολήσουν στα κεφάλαια που ακολουθούν.

2.3.1 Μοντελοποίηση της αλληλεπίδρασης με τη χρήση γεγονότων

Η έννοια των γεγονότων δεν είναι φυσικά καινούργια στο χώρο της πληροφορικής. Τα γεγονότα έχουν οριστεί και χρησιμοποιηθεί σε πολλούς τομείς έρευνας. Για παράδειγμα στον τομέα των Ενεργών Βάσεων Δεδομένων (Active Databases) [CT95, CM93] ένα γεγονός ορίζεται σαν ένα στιγμιαίο συμβάν το οποίο ενδιαφέρει την βάση δεδομένων. Το συμβάν προκαλείται από κάποια ενέργεια που προκλήθηκε μια συγκεκριμένη χρονική στιγμή και μπορεί να είναι απλό ή σύνθετο (δηλαδή να αφορά περισσότερα από ένα αντικείμενα).

Από την άλλη πλευρά η έννοια των γεγονότων έχει ξαναχρησιμοποιηθεί στον χώρο των multimedia έχοντας διαφορετικούς ορισμούς και χρήσεις. Σύμφωνα με τους Schloss και Wynblatt [SW95] ένα γεγονός ορίζεται σαν μια σύνθεση αντικειμένων στο χρόνο. Αυτό σημαίνει ότι ένα γεγονός δεν είναι στιγμιαίο αλλά έχει μια συγκεκριμένη χρονική διάρκεια.

Η προσέγγιση που ακολουθούμε υιοθετεί την έννοια των στιγμιαίων συμβάντων [HFK95, VS96]. Ωστόσο σε σύγκριση με την έννοια των στιγμιαίων συμβάντων που υιοθετείται στον τομέα των ενεργών βάσεων έχουν γίνει κάποιες επεκτάσεις. Κάθε γεγονός συμβαίνει μια συγκεκριμένη χρονική στιγμή (*temporal instance*). Επιπρόσθετα υπάρχουν γεγονότα που συνδέονται και με χωρικά δεδομένα (*spatial instance*). Για παράδειγμα ένα γεγονός μπορεί να συνδέεται με τη σχετική θέση ενός αντικειμένου μέσα στο χώρο εκτέλεσης της εφαρμογής.

Μια άλλη σημαντική διαφορά σε σχέση με το χώρο των ενεργών βάσεων δεδομένων αφορά το πεδίο ενδιαφέροντος των γεγονότων. Κάθε αντικείμενο μπορεί να προκαλέσει έναν μεγάλο αριθμό γεγονότων (για παράδειγμα γεγονός μπορεί να θεωρηθεί ή εκκίνηση, η παύση, η ολοκλήρωση του). Επιπρόσθετα εκτός από τα γεγονότα που προκαλούνται από αντικείμενα υπάρχουν και γεγονότα που προκαλούνται από το χρήστη (για παράδειγμα από το πληκτρολόγιο). Επομένως για

κάθε εφαρμογή υπάρχει ένας τεράστιος αριθμός γεγονότων που μπορούν να προκληθούν. Το μοντέλο IMD περιορίζει το "ενδιαφέρον" του σε ένα μικρό αριθμό γεγονότων. Αυτό το υποσύνολο είναι που αφορά την εφαρμογή και μπορεί να μεταβάλει τη ροή εκτέλεσής της. Τα συμπεράσματα από την παραπάνω συζήτηση μπορούν να συνοψιστούν στον παρακάτω ορισμό [VB97].

Ορισμός 2.2 *Ένα γεγονός σε μια multimedia εφαρμογή προκαλείται από μια εσωτερική ή εξωτερική ενέργεια και συνδέεται με χρονικά και χωρικά δεδομένα.. Το γεγονός αναγνωρίζεται και χειρίζεται κατάλληλα από την εφαρμογή.*

Ένα γεγονός προκαλείται εξαιτίας μιας ενέργειας που συμβαίνει σε ένα από τα αντικείμενα της εφαρμογής. Στα αντικείμενα συμπεριλαμβάνονται και τα αντικείμενα του συστήματος δηλαδή το πληκτρολόγιο και το ποντίκι. Κάθε γεγονός συμβαίνει σε ένα συγκεκριμένο χρονικό σημείο της εκτέλεσης της εφαρμογής. Επιπρόσθετα κάποια γεγονότα συνδέονται και με μια θέση[2] μέσα στο χώρο εκτέλεσης της εφαρμογής. Ο συνδυασμός των δεδομένων που χαρακτηρίζουν ένα γεγονός καλείται *χωροχρονική υπογραφή (spatiotemporal signature)* και είναι μοναδικά για κάθε γεγονός.

Εκτός από τον ορισμό των γεγονότων, οι Vazirgiannis και Boll [VB97] ορίζουν ένα μοντέλο για την κατηγοριοποίηση και τη σύνθεση των γεγονότων. Σε αυτό έχουν στηριχθεί όλες οι υλοποιήσεις του IMD μοντέλου. Στις επόμενες παραγράφους δεν θα περιγράψουμε την πλήρες μοντέλο αλλά τα τμήματα του στα οποία έχει στηριχθεί το μοντέλο το οποίο υλοποιούμε.

2.3.2 Κατηγοριοποίηση γεγονότων

Τα γεγονότα χωρίζονται σε δύο βασικές κατηγορίες: στα *απλά γεγονότα* (*simple events*) και στα *σύνθετα γεγονότα* (*complex events*). Στα απλά γεγονότα κατατάσσονται τα γεγονότα που προκαλούνται από μια απλή ενέργεια που εκτελείται σε ένα αντικείμενο[3]. Όπως θα δούμε στην υποπαράγραφο που ακολουθεί τα απλά γεγονότα μπορούν να χωριστούν περαιτέρω σε υποκατηγορίες ανάλογα με το χαρακτήρα της ενέργειας αλλά και το αντικείμενο το οποίο τη δέχτηκε. Τα σύνθετα γεγονότα είναι τα γεγονότα που προκύπτουν από τη σύνθεση απλών γεγονότων. Όπως συμβαίνει και με τα απλά γεγονότα τα σύνθετα γεγονότα μπορούν να χωριστούν σε μικρότερες υποκατηγορίες ανάλογα με τον τρόπο με τον οποίο γίνεται η σύνθεση των επιμέρους απλών γεγονότων.

---

[2] Η θέση αφορά ένα δύο σημεία που ορίζουν έναν παραλληλόγραμμο χώρο μέσα στον οποίο συνέβει το γεγονός. Φυσικά τα δύο σημεία μπορούν να συμπίπτουν οπότε η θέση αφορά ένα σημείο. Για παράδειγμα ένα γεγονός που συνδέεται με το πάτημα του αριστερού πλήκτρου το ποντικιού μπορεί να να συνδεθεί με το σημείο στο οποίο βρίσκεται ο δείκτης του ποντικιού.

[3] Στη συνέχεια όταν χρησιμοποιούμε τον όρο αντικείμενο και αναφερόμαστε στο αντικείμενο που σχετίζεται με ένα γεγονός, η έννοια του αντικειμένου δεν περιορίζεται στα αντικείμενα που συμμετέχουν στην multimedia εφαρμογή (actors). Συμπεριλαμβάνει και τα αντικείμενα του συστήματος όπως το πληκτρολόγιο και το ποντίκι. Αν θέλουμε να αναφερθούμε σε actors θα χρησιμοποιούμε τον όρο αντικείμενο της εφαρμογής.

Απλά γεγονότα

Τα απλά γεγονότα συνδέονται με μια ενέργεια που εκτελείται σε ένα αντικείμενο. Για παράδειγμα ένα απλό γεγονός αφορά το πάτημα του δεξιού κουμπιού του ποντικιού, ή την απενεργοποίηση ενός κουμπιού της εφαρμογής. Κάθε γεγονός χαρακτηρίζεται επίσης και από την χωροχρονική του υπογραφή. Τα απλά γεγονότα μπορούν να χωριστούν στις παρακάτω υποκατηγορίες, ανάλογα με τα χαρακτηριστικά του αντικειμένου με το οποίο συνδέονται αλλά και του ίδιου του γεγονότος:

- *Γεγονότα που προκαλούνται από το χρήστη* (*user events*). Στην κατηγορία αυτή ανήκουν τα γεγονότα που προκαλούνται από ενέργειες του χρήστη που πραγματοποιούνται μέσω των συσκευών εισόδου του συστήματος (πληκτρολόγιο, ποντίκι, touch screen, voice input). Στην υλοποίηση που ακολουθήσαμε λαμβάνονται υπόψη μόνο τα events που προκαλούνται από το πληκτρολόγιο (πάτημα συγκεκριμένων πλήκτρων) και από το ποντίκι (left click, right click, double-click).

- *Γεγονότα που προκαλούνται από την αλληλεπίδραση των αντικειμένων της εφαρμογής* (*intra-object events*). Τα γεγονότα αυτής της κατηγορίας προκαλούνται από την μεταβολή της κατάστασης ενός αντικειμένου της εφαρμογής. Για παράδειγμα εκτελώντας την ενέργεια *start* σε ένα αντικείμενο της εφαρμογής το αντικείμενο μεταβαίνει σε κατάσταση Active. Αυτή η αλλαγή προκαλεί το αντίστοιχο γεγονός. Ειδική αναφορά πρέπει να γίνει στα γεγονότα που προκαλούνται από χρονομετρητές σε καθορισμένες χρονικές στιγμές. Παρά το γεγονός ότι δεν συνδέονται με αλλαγή στην κατάσταση του μετρητή (ο μετρητής συνεχίζει να βρίσκεται σε κατάσταση Active) τα γεγονότα αυτού του τύπου κατατάσσονται σε αυτή την κατηγορία.

- *Γεγονότα που σχετίζονται με την ίδια την εφαρμογή* (*application events*). Η κατηγορία αυτή περιλαμβάνει όλα τα γεγονότα που δεν σχετίζονται με κάποιο συγκεκριμένο αντικείμενο αλλά με την κατάσταση της εφαρμογής στο σύνολό της. Όπως συμβαίνει και με τα επιμέρους αντικείμενα, η εφαρμογή μπορεί να βρίσκεται σε μία από τις τρεις καταστάσεις: *Active, Idle, Suspended*. Η υλοποίησή μας δεν λαμβάνει υπόψη τις μεταβάσεις που σχετίζονται με την κατάσταση Suspended. Στην ουσία υιοθετεί δύο μηνύματα. Ένα μήνυμα που προκαλείται με την εκκίνηση της εφαρμογής (*StartApp*) και την οδηγεί στην κατάσταση Active και ένα μήνυμα που σχετίζεται με την ολοκλήρωση της εφαρμογής (*ExitApplication*) και την οδηγεί ξανά σε κατάσταση Idle. Επιπρόσθετα στην κατηγορία αυτή συμπεριλαμβάνονται και τα γεγονότα που συνδέονται με έναν ειδικό χρονομετρητή. Πρόκειται για τον χρονομετρητή της εφαρμογής (*ApplicationTimer*) που εκκινήται ταυτόχρονα με την εφαρμογή και παραμένει ενεργός καθ' όλη τη διάρκειά της.

- *Γεγονότα συγχρονισμού* (*synchronization events*). Όπως αναφέραμε στην εισαγωγή της παρουσίασης του μοντέλου η βασική μονάδα εκτέλεσης του σεναρίου είναι το tuple. Όπως θα δούμε στη συνέχεια, ένα tuple εκκινήται και σταματά, όπως ακριβώς συμβαίνει και με τα αντικείμενα της εφαρμογής. Επίσης όπως μπορούμε να συνδέσουμε ένα γεγονός με την εκκίνηση ή την παύση ενός αντικειμένου μπορούμε να συνδέσουμε ένα γεγονός και με την εκκίνηση ή ολοκλήρωση της εκτέλεσης ενός tuple. Τα ειδικά αυτά συμβάντα ονομάζονται γεγονότα συγχρονισμού και μπορούν να χρησιμοποιηθούν όπως χρησιμοποιούνται και τα απλά γεγονότα.

Σύνθετα γεγονότα

Τα απλά γεγονότα συνδέονται με ένα από τα συμβάντα που λαμβάνουν μέρος κατά την εκτέλεση της εφαρμογής. Ωστόσο δεν μπορούν να χρησιμοποιηθούν για να μοντελοποιήσουν σύνθετα συμβάντα. Για παράδειγμα ας υποθέσουμε ότι θέλουμε να εμφανίσουμε ένα συγκεκριμένο κείμενο όταν ο χρήστης πατήσει τρεις φορές το πλήκτρο ESC. Κάθε πάτημα του πλήκτρου αποτελεί ένα απλό γεγονός (που σύμφωνα με την κατηγοριοποίηση που κάναμε ανήκει στην κατηγορία user interaction). Ωστόσο το *τριπλό πάτημα* του πλήκτρου δεν μπορεί να μοντελοποιηθεί σαν ένα απλό γεγονός. Είναι επομένως απαραίτητο ένα μοντέλο για την περιγραφή multimedia εφαρμογών να δίνει τη δυνατότητα σύνθεσης των απλών γεγονότων.

Η σύνθεση των γεγονότων στο μοντέλο IMD στηρίζεται στο μοντέλο που παρουσιάστηκε στο [VB97]. Το μοντέλο αυτό παρουσιάζει δύο απόψεις της σύνθεσης γεγονότων:

- *Αλγεβρική σύνθεση* (*algebraic composition*). Πρόκειται για τη σύνθεση γεγονότων που βασίζεται σε ένα σύνολο τελεστών και συναρτήσεων.
- *Χωροχρονική σύνθεση* (*spatiotemporal composition*). Η χωροχρονική σύνθεση αντικατοπτρίζει την χρονική και χωρική σχέση μεταξύ των γεγονότων.

Στην υλοποίησή μας δεν μας απασχόλησε η χωροχρονική σύνθεση. Ασχοληθήκαμε μόνο με την αλγεβρική σύνθεση και αυτή είναι που θα μας απασχολήσει στην επόμενη υποπαράγραφο. Πριν όμως αναλύσουμε τον τρόπο με τον οποίο γίνεται η σύνθεση των απλών γεγονότων πρέπει να ορίσουμε δύο βασικές έννοιες, την *έννοια του χωροχρονικού σημείου αναφοράς* και την έννοια του *χρονικού διαστήματος*.

Ορισμός 2.3 *Το χωροχρονικό σημείο αναφοράς.(spatiotemporal reference point) ορίζει το σημείο στο χώρο και στο χρόνο στο οποίο εκκινήται η εφαρμογή. Το σημείο αυτό που θα συμβολίζεται στο εξής ως* $\boldsymbol{\theta}$ *χρησιμοποιείται σαν σημείο αναφοράς για τις χωροχρονικές υπογραφές των γεγονότων της εφαρμογής.*

Στην ουσία το χωροχρονικό σημείο ορίζει την αρχή του χρόνου για μια εφαρμογή. Έτσι μιλώντας για ένα γεγονός που συνέβη τη χρονική στιγμή 12, υπονοούμε ότι το γεγονός συνέβη 12 χρονικές μονάδες μετά από την εκκίνηση της εφαρμογής. Συνήθως σαν

χρονική μονάδα χρησιμοποιούμε το δευτερόλεπτο, αλλά αυτή η παραδοχή δεν είναι δεσμευτική για το μοντέλο. Έμμεσα δώσαμε έναν ακόμα ορισμό, αυτόν του χρονικού σημείου:

Ορισμός 2.4 *Σαν χρονικό σημείο (temporal instance) ενός γεγονότος ορίζουμε το χρόνο που έχει περάσει από το χωροχρονικό σημείο αναφοράς τη χρονική στιγμή που συνέβη το γεγονός.*

Με τη βοήθεια αυτού του ορισμού δίνουμε και τον ορισμό του χρονικού διαστήματος:

Ορισμός 2.5 *Ορίζουμε σαν χρονικό διάστημα (temporal interval) t_int μεταξύ δύο γεγονότων e_1 και e_2 την χρονική απόσταση μεταξύ των χρονικών σημείων των δύο γεγονότων: t_interval :== (e_1, e_2).*

2.3.3 Αλγεβρική σύνθεση γεγονότων

Όπως προαναφέραμε θεωρούμε ότι η σύνθεση των απλών γεγονότων είναι μια δυνατότητα που πρέπει απαραίτητα να παρέχεται στο χρήστη ώστε να του δίνεται η δυνατότητα να ορίζει πολύπλοκες εφαρμογές. Έχουμε χωρίσει τη σύνθεση γεγονότων σε δύο βασικές κατηγορίες. Στη σύνθεση που προκύπτει χρησιμοποιώντας ένα σύνολο τελεστών και στη σύνθεση που προκύπτει χρησιμοποιώντας ένα σύνολο συναρτήσεων. Και στις δύο περιπτώσεις τα γεγονότα που χρησιμοποιούνται σαν ορίσματα είναι απλά συμβάντα. Αυτό σημαίνει ότι δεν μπορούμε να χρησιμοποιήσουμε σαν ορίσματα σύνθετα γεγονότα. Επίσης πρέπει να αναφέρουμε έναν περιορισμό που αφορά τη χρήση των δύο ειδικών συμβάντων. Τόσο το γεγονός StartApp όσο και το γεγονός ExitApplication *δεν* μπορούν να χρησιμοποιηθούν για τη δημιουργία σύνθετων γεγονότων.

Σύνθεση με χρήση τελεστών

Οι τελεστές που υποστηρίζονται από την υλοποίηση του μοντέλου είναι οι τελεστές *σύζευξης*, *διάζευξης* και *άρνησης*:

- *Τελεστής διάζευξης:* "|". Πρόκειται για τον τελεστή που υλοποιεί τον αλγεβρικό τελεστή *OR*. Ας υποθέσουμε πάλι ότι έχουμε δύο γεγονότα e_1 και e_2.Το σύνθετο γεγονός "$e_1 \mid e_2$" θα συμβεί αν συμβεί ένα από τα γεγονότα e_1 ή e_2. Ο τελεστής μπορεί να εφαρμοστεί "αναδρομικά". Δηλαδή το σύνθετο γεγονός "$e_1 \mid e_2 \mid ... \mid e_n$" θα συμβεί όταν ένα από τα n απλά γεγονότα e_1, e_2 ,..., e_n συμβεί κατά την εκτέλεση της εφαρμογής.

- *Τελεστής σύζευξης:* ";". Πρόκειται για τον τελεστή που υλοποιεί τον αλγεβρικό τελεστή *AND*. Ας υποθέσουμε ότι έχουμε δύο γεγονότα e_1 *και* e_2. Το σύνθετο γεγονός "e_1 ; e_2" θα συμβεί μόνο αν συμβούν *ταυτόχρονα* τα δύο γεγονότα. Ο τελεστής σύζευξης μπορεί επίσης να εφαρμοστεί αναδρομικά.

- *Τελεστής άρνησης*: *NOT*. Και πάλι πρόκειται για τον τελεστή που υλοποιεί τον αντίστοιχο αλγεβρικό τελεστή. Το σύνθετο γεγονός NOT e_1 ισχύει αν δεν συμβεί σε κάποια στιγμή το απλό γεγονός e_1.

Σύνθεση με χρήση συναρτήσεων

Οι συναρτήσεις μπορούν να θεωρηθούν και αυτοί σαν τελεστές με τη διαφορά ότι ενεργούν σε έναν μεταβλητό αριθμό απλών γεγονότων. Οι συναρτήσεις που ορίζονται από το μοντέλο είναι οι εξής:

- *Συνάρτηση ANY(k; e_1, e_2,..., e_n)*. Το σύνθετο γεγονός που ορίζει η συνάρτηση θα συμβεί όταν τουλάχιστον *k* από τα n ($k > 0$, $n > 0$, $n \geq k$) διαφορετικά μεταξύ τους στιγμιαία γεγονότα e_1, e_2,..., e_n, εγερθούν στη διάρκεια εκτέλεσης της εφαρμογής. Πρέπει να σημειώσουμε ότι τα γεγονότα δεν είναι απαραίτητο να συμβούν ταυτόχρονα, ενώ δεν παίζει ρόλο και η σειρά με την οποία εγείρονται. Το χρονικό σημείο του σύνθετου γεγονότος που ορίζει η συνάρτηση θα είναι το χρονικό σημείο του τελευταίου γεγονότος που συνέβη.

- *Συνάρτηση ANYNEW(k; e_1, e_2,..., e_n)*. Το σύνθετο γεγονός που ορίζει η συνάρτηση *ANYNEW* θα συμβεί όταν κατά τη διάρκεια εκτέλεσης της εφαρμογής εγερθούν ταυτόχρονα τουλάχιστον *k* από τα *n* στιγμιαία γεγονότα e_1, e_2,..., e_n. Η διαφορά με την συνάρτηση *ANY* αφορά την απαίτηση για ταυτόχρονη έγερση των *k* γεγονότων. Αντίθετα η συνάρτηση *ANY* τα γεγονότα μπορεί να έχουν συμβεί σε οποιαδήποτε στιγμή εκτέλεσης της εφαρμογής.

- *Συνάρτηση SEQ(e_1, e_2,..., e_n)*. Το γεγονός που ορίζει η συνάρτηση θα συμβεί αν συμβούν όλα τα γεγονότα της λίστας και μάλιστα με την καθορισμένη σειρά. Όπως συμβαίνει και με τις δύο προηγούμενες συναρτήσεις τα *n* στιγμιαία γεγονότα της λίστας πρέπει να είναι διαφορετικά μεταξύ τους.

- *Συνάρτηση TIMES(k; e_1)*. Το σύνθετο γεγονός *TIMES(k; e_1)* θα συμβεί όταν το απλό γεγονός e_1 συμβεί *k* συνεχόμενες φορές. Το χρονικό σημείο του σύνθετου γεγονότος θα είναι το χρονικό σημείο της τελευταίας έγερσης του γεγονότος e_1. Φυσικά μεταξύ των *k* διαφορετικών εγέρσεων του γεγονότος e_1 μπορούν να συμβούν και άλλα γεγονότα της εφαρμογής.

- *Συνάρτηση IN(e_1; t_int)*. Το σύνθετο γεγονός που ορίζει η συνάρτηση θα συμβεί αν το γεγονός e_1 συμβεί εντός του χρονικού διαστήματος *t_int*. Το χρονικό διάστημα ορίζεται σαν η σύνθεση δύο χρονικών σημείων: (t_1, t_2). Το χρονικό σημείο του σύνθετου γεγονότος είναι το σημείο στο οποίο θα συμβεί το απλό γεγονός e_1.

- *Συνάρτηση NOT(e_1; t_int)*. Το σύνθετο γεγονός που ορίζει η συνάρτηση *NOT* διαφέρει από το γεγονός που ορίζει ο τελεστής *NOT*. Το γεγονός θα

συμβεί αν το γεγονός e_1 δεν συμβεί εντός του χρονικού διαστήματος *t_int*. Το χρονικό διάστημα ορίζεται και πάλι σαν η σύνθεση δύο χρονικών σημείων: (t_1, t_2). Το χρονικό σημείο του σύνθετου γεγονότος είναι το σημείο t_2.

- *Συνάρτηση S_CON(e_1, e_2,..., e_n).* Η συνάρτηση S_*CON* μοιάζει σε λειτουργία με τη συνάρτηση *SEQ*. Υπάρχει ωστόσο μια βασική διαφορά. Στη συνάρτηση *SEQ* μεταξύ των n απλών γεγονότων μπορούσε να παρεμβληθεί η έγερση άλλων γεγονότων της εφαρμογής που δεν συμπεριλαμβάνονται στη λίστα. Αντίθετα μεταξύ των γεγονότων της συνάρτησης *S_CON* δεν πρέπει να παρεμβληθεί κανένα άλλο γεγονός, ακόμα και αν πρόκειται για την δεύτερη έγερση ενός εκ των γεγονότων της λίστας.

Πρέπει σημειώσουμε ότι η συνάρτηση ANYNEW συμπίπτει με τον τελεστή σύζευξης, στην περίπτωση που ο αριθμός των γεγονότων της λίστας συμπίπτει με τον αριθμό των γεγονότων που πρέπει να συμβούν (k = n). Ωστόσο για λόγους πληρότητας έχουν υλοποιηθεί και οι δύο τελεστές.

Ορισμός αλγεβρικής σύνθεσης

Κλείνοντας την περιγραφή είναι χρήσιμο να παρουσιάσουμε τον τυπικό ορισμό ενός γεγονότος, σύνθετου ή απλού. Για την πλήρη κατανόηση των ορισμάτων που χρησιμοποιούνται ο αναγνώστης μπορεί να ανατρέξει στις παραγράφους που προηγήθηκαν. Το σχήμα 2.1 περιέχει το τυπικό ορισμό ενός γεγονότος.

```
# Αλγεβρική σύνθεση γεγονότων
Event :== simpleEvent [{operator simpleEvent}]
Event := "NOT" simpleEvent
Event := eventFunction
operator :== "||" | ";"
eventFunction :== ANY(parameter, eventList) |
                  ANYNEW(parameter, eventList) |
                  SEQ(eventList) |
                  TIMES (parameter, simpleEvent) |
                  IN(simpleEvent, tempInterval) |
                  TIMES (parameter, simpleEvent) |
                  IN(simpleEvent, tempInterval) |
                  NOT(simpleEvent, tempInterval) |
                  SEQ(eventList)
parameter :== INTEGER
eventList := simpleEvent [{simpleEvent}]
```

Σχήμα 2.1: Ο τυπικός ορισμός της αλγεβρικής σύνθεσης γεγονότων.

2.3.4 Χωροχρονική σύνθεση των αντικειμένων της εφαρμογής

Δύο από τις βασικές έννοιες του μοντέλου IMD είναι η έννοια του αντικειμένου της εφαρμογής και της ενέργειας η οποία πραγματοποιείται σε αυτό και μεταβάλει την κατάστασή του. Στην παράγραφο αυτή θα μας απασχολήσει ο τρόπος με τον οποίο γίνεται η σύνθεση των ενεργειών στο χρόνο. Το μοντέλο ορίζει με ακρίβεια και τον τρόπο με τον οποίο οι ενέργειες συνθέτονται στον χώρο ωστόσο δεν θα αναφερθούμε αναλυτικά στην χωρική σύνθεση.

Στην ουσία το μοντέλο της χρονικής σύνθεσης που ακολουθείται στηρίζεται στο μοντέλο χωροχρονικής σύνθεσης που παρουσιάστηκε στο [VTS98]. Οι επεκτάσεις που έχουν πραγματοποιηθεί έχουν στόχο τον καλύτερο χειρισμό των χρονικών σχέσεων. Οι δύο βασικές ενέργειες που εφαρμόζονται στα αντικείμενα της εφαρμογής είναι ή εκκίνηση και η διακοπή της παρουσίασης του.. Επιπρόσθετα θα χρησιμοποιήσουμε άλλες δύο βασικές ενέργειες: την *παύση* της παρουσίασης (*pause*) του αντικειμένου και την *επανεκκίνησή* του (*resume*). Συνοψίζοντας το μοντέλο χρησιμοποιεί τέσσερις ενέργειες:

- *Εκκίνηση (start)*. Πρόκειται για την ενέργεια που ξεκινά την παρουσίαση ενός αντικειμένου της εφαρμογής. Μετά την επιτυχημένη εκκίνηση το αντικείμενο βρίσκεται σε κατάσταση Active.

- *Διακοπή (stop)*. Η ενέργεια αυτή σταματά την παρουσίαση του αντικειμένου και το επαναφέρει σε κατάσταση Idle.

- *Παύση (pause)*. Η ενέργεια αυτή έχει σαν αποτέλεσμα την προσωρινή διακοπή της παρουσίασης του αντικειμένου. Το αντικείμενο μεταβαίνει σε κατάσταση Suspended.

- *Επανεκκίνησή*[4] (*resume*). Η ενέργεια αυτή πραγματοποιείται ενώ το αντικείμενο βρίσκεται σε κατάσταση Suspended και έχει σαν αποτέλεσμα την συνέχιση της παρουσίασής του από το σημείο στο οποίο είχε διακοπεί με την ενέργεια pause. Το αντικείμενο μεταβαίνει ξανά σε κατάσταση Active.

Για την αναπαράσταση αυτών των ενεργειών και την χρονική σύνθεση των αντικειμένων στα οποία εφαρμόζονται θα χρησιμοποιήσουμε ένα σύνολο τελεστών που παρουσιάστηκαν από τους Little και Ghafoor [LG93]. Πρόκειται για τους τελεστές TAC (*Temporal Access Control*):

- ">", που αναπαριστά την πράξη start
- "!", που αναπαριστά την πράξη stop

[4] Ο όρος επανεκκίνηση θα χρησιμοποιείται για να αποδοθεί ο αγγλικός όρος resume. Δεν θα πρέπει να συγχέουμε την επανεκκίνηση με τον αγγλικό όρο restart που σημαίνει ότι το αντικείμενο θα ξαναπαρουσιαστεί από την αρχή ξεκινώντας από κατάσταση idle.

- "||", που αναπαριστά την πράξη pause και
- "\>" που αναπαριστά την πράξη resume.

Επομένως ο ορισμός για τους τελεστές TAC είναι ο εξής:

TAC_operator :== ">" | "!" | "||" | "\>"

Κάθε μία από τις ενέργειες αυτές προκαλεί την έγερση ενός γεγονότος. Για παράδειγμα για ένα αντικείμενο A, η ενέργεια "A>" (ξεκίνα την παρουσίαση του αντικειμένου A) προκαλεί την έγερση του γεγονότος "A>". Επιπρόσθετα για τα αντικείμενα που έχουν περιορισμένη χρονική διάρκεια (για παράδειγμα ένα video ή ένας ήχος) ορίζουμε ένα ακόμα γεγονός που συνδέεται με την ολοκλήρωση της παρουσίασης του αντικειμένου. Συνοψίζοντας έχουμε πέντε γεγονότα που συνδέονται με την κατάσταση της παρουσίασης ενός αντικειμένου:

- ">", που προκαλείται από την πράξη start
- "!", που προκαλείται από την πράξη stop
- "<", που προκαλείται από την ολοκλήρωση της παρουσίασης του αντικειμένου
- "||", που προκαλείται από την πράξη pause και
- "\>" που προκαλείται από την πράξη resume.

Ο ορισμός για τα γεγονότα είναι ο ακόλουθος:

t_event :== ">" | "!" | "<" | "||" | "\>"

Πρέπει να σημειωθεί η διαφορά των γεγονότων που προέρχονται από την πράξη της διακοπής και την ολοκλήρωση της παρουσίασης ενός αντικειμένου. Το γεγονός "A!" προκαλείται όταν εκτελεστεί η αντίστοιχη ενέργεια στο αντικείμενο A. Το γεγονός "A<" δημιουργείται όταν ολοκληρωθεί η παρουσίαση του αντικειμένου.

Τώρα μπορούμε να ορίσουμε τον τρόπο με τον οποίο γίνεται η χρονική σύνθεση των αντικειμένων. Έστω δύο αντικείμενα A και B. Η έκφραση:

A TAC_operator time_interval B TAC_operator

ορίζει τη σύνθεση των ενεργειών που εκτελούνται στα δύο αντικείμενα. Επομένως ο ορισμός της χρονικής σύνθεσης των αντικειμένων είναι ο ακόλουθος:

tempComposition :==
object TAC_operator [{time_interval object TAC_operator}]

όπου time_interval είναι το μήκος του χρονικού διαστήματος που μεσολαβεί μεταξύ των δύο διαδοχικών ενεργειών.

Για παράδειγμα έστω τρία αντικείμενα A, B και C Η έκφραση:

A> 12 B> 4 A! 0 C>

αποδίδει την εξής σύνθεση ενεργειών (υποθέτουμε ότι ο χρόνος μετράται σε δευτερόλεπτα): "Ξεκίνα την παρουσίαση του αντικειμένου Α. Μετά από 12 δευτερόλεπτα ξεκίνα την παρουσίαση του αντικειμένου Β. Μετά από 4 δευτερόλεπτα σταμάτα την παρουσίαση του αντικειμένου Α και ταυτόχρονα ξεκίνα την παρουσίαση του αντικειμένου C.

Πριν κλείσουμε την παρουσίαση του μοντέλου σύνθεσης των αντικειμένων της εφαρμογής, είναι απαραίτητο να παρουσιάσουμε δύο ακόμα σύμβολα που χρησιμοποιούνται από το μοντέλο που έχουμε υλοποιήσει. Το σύμβολο "$" υποδηλώνει το χρονικό σημείο στο οποίο εκκινήται ένα από τα tuples της εφαρμογής και μπορεί να χρησιμοποιηθεί σε μία σύνθεση αντικειμένων. Με το συμβολισμό "d_A" αναπαριστούμε την χρονική διάρκεια του αντικειμένου Α που λαμβάνει μέρος στην παρουσίαση της εφαρμογής.

2.3.5 Μοντελοποίηση εφαρμογής

Μέχρι τώρα περιγράψαμε τις τρεις από τις τέσσερις βασικές έννοιες που χρησιμοποιεί το μοντέλο IMD. Με τη χρήση των αντικειμένων, των γεγονότων και των ενεργειών είμαστε σε θέση να περιγράψουμε πλήρως μια multimedia εφαρμογή. Ωστόσο δεν έχουμε ακόμα παρουσιάσει τον τρόπο με τον οποίο συνδέονται οι τρεις αυτές έννοιες.

Το μοντέλο IMD χρησιμοποιεί την έννοια του *σεναρίου* (*scenario*). Μια εφαρμογή περιγράφεται από ένα σενάριο. Το σενάριο καθορίζει τη συμπεριφορά της εφαρμογής όταν δέχεται τα γεγονότα που προκαλούνται κατά την παρουσίαση των αντικειμένων. Με τον τρόπο αυτό το σενάριο ορίζει τη ροή εκτέλεσης της εφαρμογής. Κάθε σενάριο αποτελείται από μια ομάδα αυτόνομων τμημάτων που ονομάζονται scenario tuples. Ένα tuple μπορεί να θεωρηθεί σαν ένα αντικείμενο που εκκινήται και σταματά όταν εγερθεί ένα γεγονός. Με την εκκίνηση ενός tuple εκτελείται μια λίστα ενεργειών που ορίζονται σε αυτό. Τέλος η ίδια η εκκίνηση και η διακοπή της εκτέλεσης ενός tuple προκαλεί γεγονότα που καλούνται γεγονότα συγχρονισμού.

Οι πρώτες προσπάθειες για τον ορισμό των tuples βρίσκονται στα [VM93, VS96]. Στο μοντέλο που υλοποιούμε ένα tuple αποτελείται από τα εξής στοιχεία:

- *Το γεγονός εκκίνησης* (*Start Event*). Πρόκειται για το γεγονός (σύνθετο η απλό) το οποίο προκαλεί την εκτέλεση του tuple.

- *Το γεγονός διακοπής* (*Stop Event*). Πρόκειται για το γεγονός το οποίο προκαλεί την διακοπή της εκτέλεσης του tuple.

- *Την λίστα ενεργειών* (Action List). Αντικειμενικός σκοπός της εκτέλεσης ενός tuple είναι η εκτέλεση συγκεκριμένων ενεργειών στα αντικείμενα της εφαρμογής. Στην ουσία η λίστα ενεργειών αποτελεί μια έκφραση χρονικής σύνθεσης αντικειμένων της εφαρμογής, όπως αυτή περιγράφηκε στην παράγραφο 2.3.4.

- *Τα γεγονότα συγχρονισμού* (*Synchronization Events*). Τόσο η εκκίνηση ενός tuple όσο και η διακοπή του μπορούν να συνδεθούν με συγκεκριμένα γεγονότα που ονομάζονται γεγονότα συγχρονισμού.

Το σχήμα 2.2 παρουσιάζει τον τυπικό ορισμό του σεναρίου με βάση τα στοιχεία που αναλύσαμε παραπάνω. Ο ορισμός του τύπου Event περιέχεται στο σχήμα 2.1.

```
Scenario :== scenarioTuple [{, scenarioTuple}]
scenarioTuple :==   startEvent "," stopEvent "," actionList ","
                    startSynchEvent "," stopSynchevent
startEvent :== Event
stopEvent :== Event
actionList :== tempComposition
tempComposition :== object TAC_operator [{timeInterval object TAC_operator}]
TAC_operator :== ">" | "!" | "||" | "\>"
timeInterval := INTEGER
startSynchEvent :== simpleEvent
stopSynchEvent :== simpleEvent
```

Σχήμα 2.2: Ο τυπικός ορισμός του σεναρίου μιας εφαρμογής.

2.4 Ένα παράδειγμα εφαρμογής του μοντέλου IMD

Στις παραγράφους που προηγήθηκαν αναλύσαμε το μοντέλο IMD και παρουσιάσαμε τις δυνατότητές του στην περιγραφή multimedia εφαρμογών. Στην παράγραφο αυτή θα παρουσιάσουμε ένα παράδειγμα μιας εφαρμογής καθώς και τον τρόπο με τον οποίο αναπαρίσταται με τη χρήση του μοντέλου. Η πολυπλοκότητα του παραδείγματος είναι άνω του μετρίου και επιλέχθηκε γιατί περιλαμβάνει αλληλεπίδραση με το χρήστη αλλά και σύνθεση απλών γεγονότων. Το παράδειγμα αφορά μια multimedia παρουσίαση των αρχαίων Ολυμπιακών Αγώνων.

Ξεκινάμε με την παρουσίαση των αντικειμένων της εφαρμογής. Η εφαρμογή χρησιμοποιεί 6 διαφορετικούς τύπους αντικειμένων:

- *Κουμπιά* (*buttons*) τα οποία μπορεί να επιλέξει ο χρήστης
- *Ήχους* (*waves*)
- Εικόνες (BMPs)
- *Κινούμενη εικόνα* με τη μορφή *video (AVI)*
- *Κείμενο (text)* και
- *Χρονομετρητές* (*timers*) που χρησιμοποιούνται για να ορίσουν γεγονότα που θα συμβούν σε καθορισμένο χρόνο

Αυτοί είναι οι έξι βασικοί τύποι που υποστηρίζει και η υλοποίηση μας. Οι λεπτομέρειες για τα 7 αντικείμενα που λαμβάνουν μέρος στην εφαρμογή παρουσιάζονται στο σχήμα 2.3.

ACTOR IMG1 Type of actor = BMP FileName = AF204.BMP Height = 1500 Width = 2500 Xcoord = 5500 Ycoord = 500 Layer = 1 PreEffect = Blinds Slow PostEffect = None	ACTOR INTRO Type of actor = WAVE FileName = INTRO.WAV Start = 0 Duration = 47 Volume = 100 Looping = No Direction = Forward Effect = None
ACTOR EXITBTN Type of actor = BUTTON Type of button = PushButton Caption = Exit Height = 500 Width = 1000 Xcoord = 7000 Ycoord = 6000 Tranparent = FALSE FontFace = MS Sans Serif FontSize = 14 Color = 255, 0, 0 FontStyle = Regular	ACTOR TITLE Type of actor = BUTTON Type of button = Label Caption = Olympic Games Height = 1200 Width = 5000 Xcoord = 1500 Ycoord = 0 Tranparent = TRUE FontFace = Hellas Times FontSize = 28 Color = 255, 255, 0 FontStyle = Bold, Italic
ACTOR YMNOS Type of actor = TXT FileName = YMNOS.TXT Height = 2500 Width = 2500 Xcoord = 4500 Ycoord = 4000 Tranparent = FALSE Layer = 1 BorderStyle = Raised FontFace = HellasArc FontSize = 10 Color = 128, 0, 128 FontStyle = Italic	ACTOR KAVALAR Type of actor = AVI FileName = KAVALAR.AVI Start = 291 Duration = 20 Scale Factor = 3 Volume = 70 Looping = No Direction = Forward Height = 3500 Width = 2500 Xcoord = 500 Ycoord = 1400
ACTOR TIMER1 Type of actor = TIMER Type of timer = sec(150)	

Σχήμα 2.3: Τα αντικείμενα της multimedia εφαρμογής Olympic Games.

Όπως παρατηρούμε για κάθε αντικείμενο υπάρχουν δεδομένα τόσο για τηνεμφάνισή του όσο και για την τοποθέτησή του στο χώρο και στο χρόνο. Η ανάλυση της σημασιολογίας του κάθε πεδίου είναι πέρα από τα όρια αυτής της εργασίας.

Η ροή της εφαρμογής καθορίζεται με τη χρήση γεγονότων. Η εφαρμογή χρησιμοποιεί 6 γεγονότα για να ορίσει την ακολουθία εκτέλεσης των tuples του σεναρίου:

- _DoubleClick: Το γεγονός εγείρεται όταν ο χρήστης πατήσει δύο φορές το αριστερό πλήκτρο του ποντικιού.
- _KeyEsc: Το γεγονός εγείρεται όταν ο χρήστης της εφαρμογής πατήσει το πλήκτρο Escape.
- _IntroStop: Το γεγονός εγείρεται όταν ο ήχος INTRO ολοκληρώσει την παρουσίασή του.
- _ExitEvent: Το γεγονός εγείρεται όταν ο χρήστης πατήσει το κουμπί EXITBTN.
- _Timer10: Το γεγονός θα συμβεί όταν ο χρονομετρητής TIMER φθάσει το δέκατο δευτερόλεπτο της παρουσίασής του.
- _AppTimerEvent: Όπως έχουμε ήδη αναφέρει με την εκκίνηση κάθε εφαρμογής εκκινήται και ένας ειδικός χρονομετρητής (ApplicationTimer) που μετρά το χρόνο τον οποίο διαρκεί η παρουσίαση της εφαρμογής. Το γεγονός _AppTimerEvent θα εγερθεί όταν η εφαρμογή φθάσει στο1 λεπτό και 10 δευτερόλεπτα παρουσίασης.

Παρατηρούμε ότι ο πρώτος χαρακτήρας του ονόματος κάθε γεγονότος είναι ο χαρακτήρας "_". Αυτή είναι μια παραδοχή που χρησιμοποιείται και στην υλοποίηση του μοντέλου χωρίς ωστόσο περιοριστικό χαρακτήρα. Στο σχήμα 2.4 παρουσιάζονται τα γεγονότα της εφαρμογής.

EVENT _DoubleClick Subject = Mouse Action = DoubleClick	EVENT _KeyEsc Subject = Keyboard Action = Escape
EVENT _IntroStop Subject = INTRO Action = Close	EVENT _ExitEvent Subject = EXITBTN Action = Click
EVENT _Timer10 Subject = TIMER1 Action = time_instance(10)	EVENT _AppTimerEvent Subject = ApplicationTimer Action = time_instance(70)

Σχήμα 2.4: Τα γεγονότα της multimedia εφαρμογής Olympic Games.

Έχοντας ορίσει τα γεγονότα και τα αντικείμενα της εφαρμογής μένει να περιγράψουμε την ίδια την εφαρμογή και να την αναπαραστήσουμε χρησιμοποιώντας το μοντέλο IMD. Η εφαρμογή έχει τρία στάδια παρουσίασης. Το πρώτο στάδιο ξεκινά με την εκκίνηση της εφαρμογής (δηλαδή με το event StartApp). Αμέσως μετά την εκκίνηση της εφαρμογής παρουσιάζεται το αντικείμενο TITLE, που περιέχει τον τίτλο της εφαρμογής. Ταυτόχρονα ακούγεται και ο ήχος INTRO. Μετά από τρία δευτερόλεπτα εμφανίζεται στην οθόνη η εικόνα IMG1. Η ακολουθία αυτή της εμφάνισης των τριών αντικειμένων μπορεί να διακοπεί όταν ο χρήστης προκαλέσει ένα από τα γεγονότα _DoubleClick, _KeyEsc ή ολοκληρωθεί η παρουσίαση του αντικειμένου INTRO.

Αν συμβεί το γεγονός _IntroStop εκτός από την διακοπή της εκτέλεσης του πρώτου σταδίου προκαλείται και η εκκίνηση της εκτέλεσης του δεύτερου σταδίου. Η εκκίνηση έχει σαν αποτέλεσμα την εμφάνιση του κειμένου YMNOS, την εκκίνηση του video KAVALAR καθώς και την εκκίνηση του χρονομετρητή TIMER1. Επίσης μετά από 6 δευτερόλεπτα εμφανίζεται στην οθόνη και το κουμπί EXITBTN. Για να σταματήσει η εκτέλεση του δεύτερου σταδίου πρέπει ο χρήστης να πατήσει το κουμπί EXITBTN αφού όμως έχουν περάσει τουλάχιστον 10 δευτερόλεπτα από την εκκίνηση του tuple και επομένως και του χρονομετρητή TIMER1. Δηλαδή το tuple διακόπτεται από την ακολουθία γεγονότων Timer10 και _ExitEvent. Η διακοπή του tuple έχει σαν αποτέλεσμα την έγερση του γεγονότος συγχρονισμού _e1

Το τρίτο και τελευταίο στάδιο έχει σαν αποτέλεσμα την διακοπή της εφαρμογής. Η εκκίνησή του γίνεται μόλις συμβεί το γεγονός συγχρονισμού _e1. Ωστόσο το γεγονός _e1 θα συμβεί μόνο αν ο χρήστης πατήσει το κουμπί EXITBTN. Αν ο χρήστης δεν προκαλέσει το αντίστοιχο η εφαρμογή θα μείνει μονίμως ενεργή. Για το λόγο αυτό η εκκίνηση του tuple μπορεί να γίνει και από το γεγονός _AppTimerEvent. Επομένως η παρουσίαση δεν μπορεί να κρατήσει πάνω από 1 λεπτό και 10 δευτερόλεπτα.

Κάθε ένα από τα τρία στάδια της εφαρμογής αναπαρίσταται από ένα tuple. Το σύνολο των τριών tuples σχηματίζει το σενάριο της εφαρμογής. Το σχήμα 2.5 παρουσιάζει τα τρία tuples του σεναρίου που περιγράψαμε.

```
TUPLE Stage1
    Start_Event = StartApp
    Stop_Event = ANY(1; _DoubleClick; _KeyEsc; _IntroStop)
    Action_List = TITLE> 0 INTRO> 3 IMG1>
    Start_Synch_Event = None
    Stop_Synch_Event = None
TUPLE Stage2
    Start_Event = _IntroStop
    Stop_Event = SEQ(_Timer10; _ExitEvent)
    Action_List = KAVALAR> 0 YMNOS> TIMER1> 6 EXITBTN>
    Start_Synch_Event = None
    Stop_Synch_Event = _e1
TUPLE ExitTuple
    Start_Event = _e1 | _AppTijmerEvent
    Stop_Event = None
    Action_List = ExitApplication
    Start_Synch_Event = None
    Stop_Synch_Event = None
```

Σχήμα 2.5: Τα tuples της multimedia εφαρμογής Olympic Games.

ΚΕΦΑΛΑΙΟ 3

Χρονική Ακεραιότητα Multimedia Εφαρμογών

Η περιγραφή μιας multimedia εφαρμογής αποτελεί το πρώτο στάδιο της διαδικασίας ανάπτυξης. Σε αυτό ο συγγραφέας περιγράφει τον τρόπο με τον οποίο θα παρουσιαστούν τα αντικείμενα της εφαρμογής κατά την εκτέλεσή της. Στο προηγούμενο κεφάλαιο ασχοληθήκαμε με τα μοντέλα που έχουν αναπτυχθεί για την περιγραφή multimedia εφαρμογών.

Ένα από τα προβλήματα που παρουσιάζονται κατά την διαδικασία συγγραφής αφορά την χρονική ακεραιότητα της εφαρμογής που περιγράφεται. Ειδικά όταν πρόκειται για εφαρμογές οι οποίες περιλαμβάνουν αλληλεπίδραση με το χρήστη το πρόβλημα γίνεται περισσότερο πολύπλοκο αφού δεν είναι δυνατόν να προσδιοριστούν οι χρονικές στιγμές στις οποίες θα πραγματοποιηθούν οι ενέργειες του χρήστη. Το αποτέλεσμα είναι ότι υπάρχει η πιθανότητα κατά την εκτέλεση της εφαρμογής να παρουσιαστεί ανεπιθύμητη συμπεριφορά από ορισμένα από τα αντικείμενα της εφαρμογής, ειδικά αν πρόκειται για αντικείμενα που συνδέονται με χρονικούς περιορισμούς.

Το βασικό θέμα που μας απασχολεί στην εργασία αυτή αφορά την παρουσίαση και υλοποίηση μιας μεθοδολογίας που θα ελέγχει την χρονική ακεραιότητα των multimedia εφαρμογών, κατά τη διαδικασία συγγραφής τους. Σκοπός του κεφαλαίου αυτού είναι η αναλυτική παρουσίαση της μεθοδολογίας *Scenario Integrity Checking (SIC)* που προτάθηκε από τους Mirbel, Pernici, Sellis και Vazirgiannis [MPSV98]. Η μεθοδολογία στηρίζεται στο μοντέλο Interactive Multimedia Document που ήδη έχουμε παρουσιάσει και εφαρμόζεται κατά τη διαδικασία ανάπτυξης μιας multimedia εφαρμογής. Αποτελεί επομένως ένα εργαλείο που παρέχεται στο συγγραφέα της εφαρμογής και του δίνει τη δυνατότητα να εντοπίσει τις χρονικές ασυνέπειες και να τις διορθώσει κατά τη διάρκεια της διαδικασίας ανάπτυξης.

Το κεφάλαιο ξεκινά με μια εισαγωγή στη μεθοδολογία. Παράλληλα παρουσιάζει κάποιες από τις βασικές έννοιες που θα μας συνοδεύσουν στην υπόλοιπη εργασία. Στην παράγραφο 3.2 αναλύεται η έννοια της χρονικής ακεραιότητας multimedia εφαρμογών. Η μεθοδολογία SIC στηρίζεται στην αναπαράσταση χρονικών περιορισμών που σχετίζονται με τα αντικείμενα της εφαρμογής με τη χρήση Δικτύων Χρονικών Περιορισμών. Στην παράγραφο 3.3 αναλύονται τα Δίκτυα Χρονικών Περιορισμών. Οι επόμενες παράγραφοι είναι αφιερωμένες στην παρουσίαση της μεθοδολογίας SIC. Στην παράγραφο 3.4 παρουσιάζεται η γενική αρχιτεκτονική της μεθοδολογίας. Η μεθοδολογία αποτελείται από τέσσερα διακριτά βήματα τα οποία αναλύονται σε ισάριθμες παραγράφους. Τέλος στην παράγραφο 3.9 παρουσιάζονται οι διαφορετικές προσεγγίσεις που έχουν αναπτυχθεί ως σήμερα για τον έλεγχο της ακεραιότητας multimedia εφαρμογών και συγκρίνονται με τη μεθοδολογία SIC.

3.1 Εισαγωγή στη μεθοδολογία Scenario Integrity Checking

Το πρόβλημα με την *χρονική ακεραιότητα* (*temporal integrity*) των multimedia εφαρμογών ξεκινά από την φιλοσοφία με την οποία αντιμετωπίζουν οι συγγραφείς τη διαδικασία ανάπτυξης. Ο *συγγραφέας* (*author*) συγκεντρώνει κάθε φορά την προσοχή του σε ξεχωριστά τμήματα της παρουσίασης. Το αποτέλεσμα είναι ότι δεν έχει μια συνολική εικόνα της παρουσίασης. Έτσι ενώ τα επιμέρους τμήματα είναι *χρονικά συνεπή*[1] (*consistent*) είναι πιθανόν η εφαρμογή στο σύνολό της να παρουσιάζει χρονικές ασυνέπειες (*inconsistencies*).

Οι ασυνέπειες αυτές δεν γίνονται αντιληπτές παρά μόνο κατά την *εκτέλεση* (*rendering*) της εφαρμογής: Κάποια τμήματα της παρουσίασης δεν εμφανίζονται στην οθόνη, κάποια από τα αντικείμενα έχουν λανθασμένη συμπεριφορά ή οι ενέργειες του χρήστη δεν έχουν καμία επίδραση στην ροή εκτέλεσης της εφαρμογής.

Η μεθοδολογία Scenario Integrity Checking (SIC) έχει σαν σκοπό τον εντοπισμό χρονικών ασυνεπειών, κατά τη διαδικασία της *συγγραφής* (*authoring*) του σεναρίου. Η μεθοδολογία στηρίζεται στο μοντέλο Interactive Multimedia Document scenario model. Χρησιμοποιώντας τη μεθοδολογία ο συγγραφέας μπορεί να αποφασίσει αν η σειρά των ενεργειών που προσδιορίζουν τα tuples του σεναρίου είναι συνεπής, έχοντας σαν δεδομένο ένα σύνολο χρονικών περιορισμών που αφορούν τα αντικείμενα της παρουσίασης.

Αυτοί οι περιορισμοί μπορούν είτε να έχουν γενικό χαρακτήρα (για παράδειγμα "... ένα αντικείμενο που βρίσκεται σε κατάσταση Idle δεν μπορεί να δεχτεί ενέργεια Stop") είτε να αφορούν τη συγκεκριμένη παρουσίαση (για παράδειγμα "... η παρουσίαση του video A δεν θα πρέπει να συμπίπτει χρονικά με την παρουσίαση του ήχου B " ή "... δεν επιτρέπεται να υπάρχουν δύο ήχοι που να βρίσκονται ταυτόχρονα σε κατάσταση Idle").

Η λειτουργικότητα μιας μεθοδολογίας ελέγχου της χρονικής ακεραιότητας δεν γίνεται τόσο φανερή αν πρόκειται για απλές εφαρμογές, με μικρό αριθμό αντικειμένων και περιορισμένη *αλληλεπίδραση με το χρήστη* (*user interaction*). Ο συγγραφέας μπορεί

[1] Στη συνέχεια οι όροι χρονική συνέπεια και χρονική ακεραιότητα θα θεωρούνται ταυτόσημοι.

να εντοπίσει τα διαφορετικά *μονοπάτια εκτέλεσης* της εφαρμογής (*scenario paths*) και να ελέγξει την χρονική τους συνέπεια. Σε πολύπλοκες όμως εφαρμογές είναι δύσκολο για τον συγγραφέα να εντοπίσει όλα τα πιθανά μονοπάτια εκτέλεσης και ακόμα πιο δύσκολο να ελέγξει με αξιοπιστία την συνέπειά τους. Είναι επομένως απαραίτητο να παρέχεται στον συγγραφέα ένα αξιόπιστο εργαλείο με το οποίο θα μπορεί να αποφασίσει την χρονική συνέπεια της εφαρμογής που σχεδιάζει ή των εφαρμογών που ήδη έχει αναπτύξει.

Η μεθοδολογία SIC έχει τα εξής βασικά χαρακτηριστικά:

- Βασίζεται στον μετασχηματισμό του σεναρίου σε ένα δίκτυο το οποίο αναπαριστά τους χρονικούς περιορισμούς μεταξύ των αντικειμένων της εφαρμογής. Η διαδικασία στηρίζεται σε ένα σύνολο κανόνων για την κατασκευή του δικτύου χρονικών περιορισμών (*Temporal Constraint Network*). Κατά το μετασχηματισμό λαμβάνονται επίσης υπόψη οι επιδράσεις των ενεργειών του χρήστη στη ροή εκτέλεσης της εφαρμογής.

- Από τον μετασχηματισμό προκύπτει ένας αριθμός δικτύων που αναπαριστούν τα διαφορετικά μονοπάτια εκτέλεσης της εφαρμογής. Σε κάθε μονοπάτι εφαρμόζονται τεχνικές ελέγχου της χρονικής ακεραιότητας [BCTP94, BCTPL97, BCTPQ97, DMP91] που αποφασίζουν ποια μονοπάτια είναι συνεπή και προσδιορίζουν τα προβλήματα που παρουσιάζονται στα ασυνεπή μονοπάτια.

- Κατά την εφαρμογή των τεχνικών επαλήθευσης κατασκευάζεται μια μορφή αναπαράστασης ενός συνεπούς μονοπατιού που ονομάζεται *ελάχιστο δίκτυο* (*minimal network*). Το δίκτυο αυτό μπορεί να χρησιμοποιηθεί για να απαντηθούν ερωτήσεις που αφορούν τα χρονικά χαρακτηριστικά της παρουσίασης των αντικειμένων της εφαρμογής[2].

Η μεθοδολογία αποτελείται από τέσσερα διακριτά βήματα. Το πρώτο βήμα αφορά την εύρεση των διαφορετικών μονοπατιών της εφαρμογής. Όπως αναφέραμε ανάλογα με τις ενέργειες που εκτελεί ο χρήστης η εφαρμογή μπορεί να ακολουθήσει διαφορετικές διαδρομές εκτέλεσης. Το αποτέλεσμα του πρώτου βήματος είναι ο σχηματισμός ενός διαγράμματος μετάβασης καταστάσεων (State Transition Diagram) που περιγράφει τον τρόπο με τον οποίο αντιδρά η εφαρμογή στα διάφορα γεγονότα που δέχεται. Στο δεύτερο βήμα τα tuples του σεναρίου εξετάζονται και μετασχηματίζονται σε επιμέρους δίκτυα (Constraint Network Fragments) που θα χρησιμοποιηθούν για να συνθέσουν τα δίκτυα περιορισμών της εφαρμογής. Στο τρίτο βήμα γίνεται η σύνθεση των επιμέρους δικτύων. Η σύνθεση βασίζεται στα διαφορετικά μονοπάτια εκτέλεσης που ανακαλύφθηκαν κατά το βήμα 1. Τέλος στο βήμα 4 σε κάθε ένα από τα δίκτυα

[2] Στην παρουσίαση της μεθοδολογίας δεν θα ασχοληθούμε με την υποστήριξη των χρονικών ερωτήσεων. Ο ενδιαφερόμενος αναγνώστης μπορεί να ανατρέξει στη σχετική βιβλιογραφία [MPSV98, BCTP94, BCTPL97].

περιορισμών που έχουν προκύψει εφαρμόζεται η μεθοδολογία Simple Temporal Problem (STP) [DMP91]. Το αποτέλεσμα είναι ο διαχωρισμός των μονοπατιών σε χρονικά συνεπή (consistent paths) και χρονικά ασυνεπή μονοπάτια (inconsistent paths).

3.2 Χρονική ακεραιότητα multimedia εφαρμογών

Σκοπός της μεθοδολογίας είναι να βοηθήσει τον συγγραφέα να αναπτύξει εφαρμογές χωρίς προβλήματα χρονικής ακεραιότητας. Πως όμως ορίζεται η χρονική ακεραιότητα; Πότε ένα μονοπάτι μιας εφαρμογής θα θεωρείται συνεπές και πότε όχι; Τι προβλήματα προκύπτουν σε μια εφαρμογή που δεν είναι συνεπής; Αυτά τα ερωτήματα θα μας απασχολήσουν σε αυτή την παράγραφο.

Η χρονική ακεραιότητα αφορά την ορθότητα της χρονικής ακολουθίας των ενεργειών που εκτελούνται στα αντικείμενα της εφαρμογής, σύμφωνα με τα χαρακτηριστικά του κάθε αντικειμένου αλλά και τα χαρακτηριστικά της *σκηνής* (*scene*) στην οποία συμμετέχει. Ένα σενάριο θα θεωρείται χρονικά συνεπές εάν:

1. η ακολουθία των ενεργειών που εκτελούνται σε κάθε αντικείμενο κατά την εκτέλεση του σεναρίου ικανοποιεί τους περιορισμούς του σεναρίου και
2. η σύνθεση μεταξύ των αντικειμένων της εφαρμογής, όπως ορίζεται από τα tuples του σεναρίου, ικανοποιεί τους περιορισμούς του σεναρίου

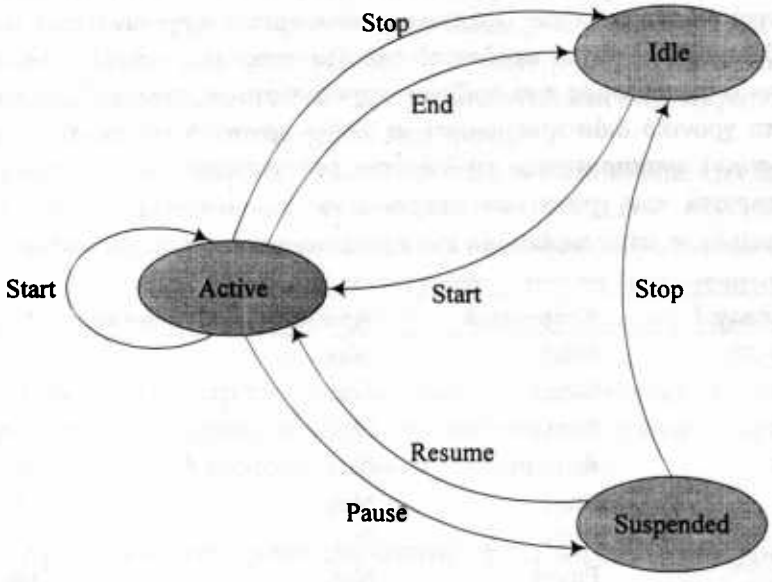
Με βάση τις δύο αυτές βασικές προϋποθέσεις, η χρονική ακεραιότητα μιας εφαρμογής εξετάζεται σε δύο επίπεδα: Στο επίπεδο των αντικειμένων που λαμβάνουν μέρος στην παρουσίαση και στο επίπεδο του σεναρίου στο σύνολό του.

3.2.1 Ακεραιότητα multimedia αντικειμένων

Κατά την παρουσίαση του μοντέλου IMD αναφερθήκαμε (παρ. 2.3) στις τρεις διαφορετικές καταστάσεις στις οποίες μπορεί να βρίσκεται ένα αντικείμενο μιας multimedia εφαρμογής: Idle, Active, Suspended. Επίσης αναφερθήκαμε στις τέσσερις βασικές ενέργειες που εκτελούνται στα αντικείμενα της εφαρμογής: Start (“>”), Stop(“!”), Pause(“||”), Resume(“\>”) και στις μεταβάσεις που προκαλούν στην κατάσταση ενός αντικειμένου.

Ωστόσο υπάρχουν περιορισμοί στις πράξεις που μπορούν να εκτελεστούν όταν ένα αντικείμενο βρίσκεται σε μια κατάσταση. Για παράδειγμα αν ένα αντικείμενο βρίσκεται σε κατάσταση Suspended, δεν μπορεί να εκτελεστεί η πράξη Pause. Αντίθετα εκτελώντας την πράξη Stop το αντικείμενο μεταπίπτει σε κατάσταση Idle, ενώ εκτελώντας την πράξη Resume το αντικείμενο ξαναγίνεται ενεργό (κατάσταση Active). Το σχήμα 3.1 παρουσιάζει όλες τις επιτρεπτές μεταβάσεις μεταξύ των τριών καταστάσεων των αντικειμένων. Στο διάγραμμα εκτός από τις τέσσερις βασικές πράξεις έχει συμπεριληφθεί και το γεγονός της ολοκλήρωσης της παρουσίασης ενός χρονικά περιορισμένου αντικειμένου (*End*).

Μια απλή μελέτη του διαγράμματος μετάβασης καταστάσεων που παρουσιάζεται στο σχήμα 3.1 φανερώνει τους περιορισμούς που υπάρχουν στην εκτέλεση των ενεργειών σε ένα αντικείμενο. Επομένως ένα πρώτο συμπέρασμα όσον αφορά τους κανόνες για την ακεραιότητα των multimedia αντικειμένων αφορά τους περιορισμούς στην ακολουθία των ενεργειών που εκτελούνται σε ένα αντικείμενο. Ο πίνακας 3.1 συνοψίζει αυτούς τους περιορισμούς.



Σχήμα 3.1: Διάγραμμα μετάβασης καταστάσεων για τις ενέργειες του IMD μοντέλου.

Υπάρχουν ωστόσο και άλλοι περιορισμοί που σχετίζονται μεν με την ακολουθία των ενεργειών που εκτελούνται σε ένα αντικείμενο αλλά εισάγουν επιπρόσθετα και την έννοια του χρόνου. Ας θεωρήσουμε την έκφραση:

A operator1 time A operator2 (3.1)

στην οποία A είναι ένα από τα αντικείμενα της εφαρμογής, operator1 και operator2 είναι κάποιες από τις τέσσερις πράξεις (>, !, ||, \>) και time είναι το χρονικό διάστημα που μεσολαβεί μεταξύ των δύο πράξεων και το οποίο μπορεί να λάβει και αρνητικές

Ενέργεια	Περιορισμός
Start	Κανένας
Stop	Το αντικείμενο πρέπει να βρίσκεται σε κατάσταση Active ή Suspended
Pause	Το αντικείμενο πρέπει να βρίσκεται σε κατάσταση Active
Resume	Το αντικείμενο πρέπει να βρίσκεται σε κατάσταση Suspended

Πίνακας 3.1: Οι περιορισμοί που ισχύουν για τις ενέργειες που εκτελούνται στα αντικείμενα μιας εφαρμογής.

τιμές. Ένα πρώτος γενικός περιορισμός θα μπορούσε να είναι: "...*το χρονικό διάστημα time θα πρέπει να είναι τέτοιο ώστε η δεύτερη πράξη να μην πραγματοποιείται πριν από την εκκίνηση της εφαρμογής*".

Ο πίνακας 3.2 παρουσιάζει το σύνολο των περιορισμών που πρέπει να πληρούν οι ενέργειες που πραγματοποιούνται στα αντικείμενα της εφαρμογής. Ο πίνακας στηρίζεται στην έκφραση 3.1. Η πρώτη στήλη αφορά τον τελεστή operator1, η δεύτερη τον τελεστή operator2 ενώ η τέταρτη στήλη αφορά τους περιορισμούς που αφορούν το χρονικό διάστημα time. Κάποιοι συνδυασμοί ενεργειών είναι αποδεκτοί, κάποιοι συνδυασμοί δεν είναι αποδεκτοί ενώ για τους υπόλοιπους συνδυασμούς πρέπει να ισχύει ο περιορισμός που ορίζεται στην αντίστοιχη στήλη. Πρέπει να υπενθυμίσουμε ότι το χρονικό διάστημα μπορεί να λάβει αρνητικές τιμές. Τέλος με το σύμβολο d (*duration*) αναπαριστούμε τη διάρκεια του αντικειμένου. Το σύμβολο r (*remaining*) αναπαριστά τον χρόνο που απομένει για την ολοκλήρωση της παρουσίασης ενός αντικειμένου, στην περίπτωση που έχει εκτελεστεί μια πράξη Pause.

Ενέργεια 1	**Ενέργεια 2**	**Αποδεκτός Συνδυασμός**	**Περιορισμός**
Start	Start	Ναι	Κανένας
Start	Stop	Ναι	0 < time < d
Start	Pause	Ναι	0 < time < d
Start	Resume	Ναι	-d < time < 0
Stop	Start	Ναι	-d < time < 0
Stop	Stop	Όχι	
Stop	Pause	Ναι	time < 0
Stop	Resume	Ναι	-d < time < 0
Pause	Start	Ναι	-d < time < 0
Pause	Stop	Ναι	time > 0
Pause	Pause	Όχι	
Pause	Resume	Ναι	time > 0
Resume	Start	Ναι	Κανένας
Resume	Stop	Ναι	0 < time < r
Resume	Pause	Ναι	0 < time < r
Resume	Resume	Όχι	

Πίνακας 3.2: Οι περιορισμοί που ισχύουν για όλους τους συνδυασμούς ενεργειών που μπορούν να εκτελεστούν σε ένα αντικείμενο της εφαρμογής.

3.3 Επίλυση χρονικών προβλημάτων με τη χρήση δικτύων χρονικών περιορισμών

Η εξέταση της έννοιας της ακεραιότητας των multimedia αντικειμένων μας οδηγεί σε ένα βασικό συμπέρασμα όσον αφορά την ακεραιότητα του σεναρίου στο σύνολό του. Στην ουσία η ακεραιότητα ενός σεναρίου συνίσταται στην ικανοποίηση ενός συνόλου

χρονικών περιορισμών που αφορούν τις ενέργειες που εκτελούνται στα αντικείμενα της εφαρμογής.

Παρόμοια προβλήματα έχουν παρουσιαστεί σε πολλά διαφορετικά πεδία έρευνας όπως ο προγραμματισμός εργασιών (task scheduling), η επαλήθευση προγραμμάτων (program verification) αλλά και στον τομέα της παράλληλης επεξεργασίας. Έχουν προταθεί πολλές μέθοδοι για την αναπαράσταση χρονικών περιορισμών. Μεταξύ αυτών αναφέρουμε την άλγεβρα διαστημάτων του Allen [All83], την άλγεβρα χρονικών σημείων των Vilain και Kautz [VKV89] καθώς και τις μελέτες των Montanari [Mon74], Mackworth [Mac77], Freuder [Freu78] και Dechter [DP87]. Οι περισσότερες από τις προτάσεις αυτές συνοδεύονται από έναν αλγόριθμο για τον έλεγχο της ταυτόχρονης ικανοποίησης των περιορισμών.

Η μέθοδος στην οποία στηρίζεται το μοντέλο που υιοθετούμε έχει προταθεί από τους Dechter και Pearl [DP91] και συνοδεύεται επίσης από έναν αλγόριθμο για τον έλεγχο της ικανοποίησης των περιορισμών. Το μοντέλο μπορεί να επιλύσει απλά αλλά και σύνθετα προβλήματα χρονικών περιορισμών. Δεν θα μας απασχολήσουν τα σύνθετα προβλήματα αφού το πρόβλημα που ορίζεται από τους περιορισμούς που τίθενται από τα αντικείμενα του σεναρίου είναι ένα απλό χρονικό πρόβλημα (Simple Temporal Problem, STP). Στη συνέχεια θα αναλύσουμε εν συντομία το μοντέλο STP, και τον αλγόριθμο που χρησιμοποιεί για την επίλυση των χρονικών προβλημάτων, χρησιμοποιώντας ένα απλό παράδειγμα χρονικών περιορισμών.

3.3.1 Χρονικά προβλήματα χρονικές ανισότητες και δίκτυα χρονικών περιορισμών

Ας υποθέσουμε δύο γεγονότα που συμβαίνουν τις χρονικές στιγμές X_i και X_j. Η διπλή χρονική ανισότητα (*bound of difference*):

$$a \leq X_j - X_i \leq b, b \geq a \geq 0 \tag{3.2}$$

εκφράζει έναν χρονικό περιορισμό μεταξύ των δύο γεγονότων. Συγκεκριμένα η ανισότητα ματαφράζεται ως εξής: Το γεγονός X_j πρέπει να συμβεί τουλάχιστον a χρονικές μονάδες μετά από το γεγονός X_i αλλά δεν πρέπει να συμβεί αργότερα από b χρονικές μονάδες.

Ένα χρονικό πρόβλημα αποτελείται από ένα σύνολο τέτοιων ανισοτήτων. Η αναπαράσταση του χρονικού προβλήματος με τη χρήση ενός συνόλου ανισοτήτων είναι ακριβής, ωστόσο παρουσιάζει δυσκολίες στην άμεση κατανόησή της.

Ένας πιο εύγλωττος τρόπος παρουσίασης υιοθετεί την χρησιμοποίηση ενός κατευθυνόμενου γράφου. Κάθε *κόμβος* (*node*) του γράφου αναπαριστά τα γεγονότα που μας ενδιαφέρουν. Οι κόμβοι συνδέονται μεταξύ τους με *τόξα* (*arcs*) τα οποία χαρακτηρίζονται από ένα διάστημα. Το διάστημα αυτό εκφράζει τα όρια της διπλής ανισότητας (2). Για παράδειγμα οι δύο κόμβοι που αναπαριστούν τα γεγονότα X_i και X_j συνδέονται, σύμφωνα με αυτόν τον τρόπο αναπαράστασης, με ένα τόξο με διάρκεια [a, b]. Το σύνολο των κόμβων του γράφου σχηματίζει ένα *δίκτυο* (*network*). Αυτό το δίκτυο αποτελεί το *δίκτυο χρονικών περιορισμών* (*temporal constraint network, TCN*)

του προβλήματος. Επομένως ένα δίκτυο TCN δεν είναι τίποτα άλλο παρά μια διαφορετική αναπαράσταση ενός συνόλου χρονικών ανισοτήτων που χαρακτηρίζουν ένα χρονικό πρόβλημα.

Ας δούμε τώρα ένα απλό παράδειγμα ενός χρονικού προβλήματος και ας παρουσιάσουμε τον τρόπο με τον οποίο αναπαρίσταται με ένα δίκτυο TCN.

Παράδειγμα 3.1 *Θέλουμε να εξετάσουμε την συνέπεια του εξής σεναρίου που αφορά δύο υποτιθέμενους συνάδελφους, τον Γιώργο και τον Νίκο. Ο Γιώργος πηγαίνει στο γραφείο είτε με το αυτοκίνητό του, είτε χρησιμοποιώντας το λεωφορείο. Στην πρώτη περίπτωση χρειάζεται 30 έως 40 λεπτά ενώ στη δεύτερη τουλάχιστον 1 ώρα. Ο Νίκος χρησιμοποιεί πάντα το αυτοκίνητό του για να πάει στο γραφείο (40 έως 50 λεπτά). Σήμερα ο Γιώργος έφυγε από το σπίτι του μεταξύ 7:10 και 7:20. Ο Νίκος έφθασε στη δουλειά μεταξύ 8:00 και 8:10. Επίσης είναι γνωστός ένας ακόμα περιορισμός: Ο Γιώργος έφθασε στη δουλειά 10 έως 20 λεπτά αφού ο Νίκος έφυγε από το σπίτι του. Το ερώτημα που τίθεται είναι αν το παραπάνω σενάριο είναι συνεπές. Με άλλα λόγια αν οι χρονικοί περιορισμοί που θέτει μπορούν να ικανοποιηθούν ταυτόχρονα.*

Στο παράδειγμα αυτό έχουμε 4 γεγονότα, που συνδέονται βέβαια με μια χρονική στιγμή κατά την οποία συμβαίνουν.

- Γεγονός E_1: Ο Γιώργος έφυγε από το σπίτι του
- Γεγονός E_2: Ο Γιώργος έφθασε στο γραφείο
- Γεγονός E_3: Ο Νίκος έφυγε από το σπίτι του
- Γεγονός E_4: Ο Νίκος έφθασε στο γραφείο

Επιπρόσθετα πρέπει να έχουμε ένα σημείο αναφοράς. Δηλαδή ένα σημείο από το οποίο θα μετρώνται όλοι οι χρόνοι. Αυτό είναι το σημείο-γεγονός E_0 που συνδέεται με τη χρονική στιγμή 7:00.

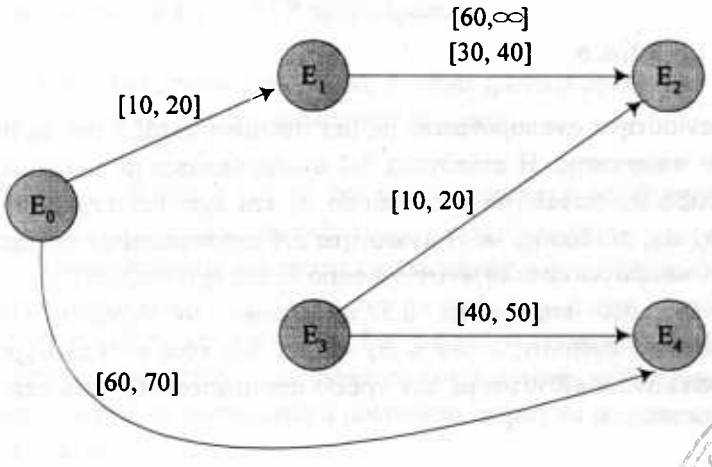
Το παράδειγμα ορίζει ένα σύνολο περιορισμών που εκφράζονται με αντίστοιχες ανισότητες (όλοι οι χρόνοι μετρώνται σε λεπτά)

- Ο Γιώργος έφυγε από το σπίτι μεταξύ 7:10 και 7:20: $10 \leq E_1 - E_0 \leq 20$
- Ο Γιώργος πηγαίνει στη δουλειά με αυτοκίνητο: $30 \leq E_2 - E_1 \leq 40$
- Ο Γιώργος πηγαίνει στη δουλειά με λεωφορείο: $E_2 - E_1 \geq 60$
- Ο Νίκος πηγαίνει στη δουλειά με αυτοκίνητο: $40 \leq E_4 - E_3 \leq 50$
- Ο Γιώργος έφθασε στη δουλειά 10 έως 20 λεπτά αφού ο Νίκος έφυγε από το σπίτι του: $10 \leq E_2 - E_3 \leq 20$
- Ο Νίκος έφθασε στη δουλειά μεταξύ 8:00 και 8:10: $60 \leq E_4 - E_0 \leq 70$

Το παράδειγμα αποτελεί ένα *πρόβλημα ικανοποίησης χρονικών περιορισμών* (*Temporal Constraint Satisfaction Problem, TCSP*). Όπως βλέπουμε τα γεγονότα E_2 και E_1 συνδέονται με δύο δυνατούς περιορισμούς. Κάθε ένα από τα δύο σενάρια που σχηματίζονται αποτελεί ένα *απλό χρονικό πρόβλημα* (*Simple Temporal Problem*). Τα

απλά χρονικά προβλήματα έχουν το χαρακτηριστικό ότι για κάθε δύο γεγονότα υπάρχει το πολύ μία ανισότητα που τα συνδέει.

Κάθε σενάριο αναπαρίσταται από ένα διαφορετικό δίκτυο TCN που αποτελείται από 5 κόμβους. Για λόγους ευκολίας χρησιμοποιούμε ένα μόνο δίκτυο στο οποίο η σύνδεση μεταξύ των κόμβων E_2 και E_1 έχει δύο διαστήματα. Το δίκτυο παρουσιάζεται στο σχήμα 3.2.



Σχήμα 3.2: Το δίκτυο χρονικών περιορισμών του παραδείγματος 3.1

3.3.2 Επίλυση ενός απλού χρονικού προβλήματος

Ένα απλό χρονικό πρόβλημα αναπαρίσταται όπως είδαμε με ένα σύνολο χρονικών ανισοτήτων. Σκοπός της επίλυσης του προβλήματος είναι να διαπιστωθεί αν είναι δυνατόν να ικανοποιούνται ταυτόχρονα όλες οι χρονικές ανισότητες. Επομένως η επίλυση ενός χρονικού προβλήματος συνίσταται στην επίλυση ενός συνόλου χρονικών ανισοτήτων.

Η επίλυση χρονικών ανισοτήτων είναι ένα πρόβλημα που έχει απασχολήσει πολλές φορές την επιστημονική κοινότητα. Μπορεί να επιλυθεί από την εκθετική μέθοδο που παρουσίασε ο Dantzig [Dan62] ή χρησιμοποιώντας τον αλγόριθμο του Khachiyan [Kha79]. Και οι δύο μέθοδοι παρουσιάζουν υψηλή πολυπλοκότητα. Ευτυχώς ο τύπος των ανισοτήτων που αποτελούν ένα απλό χρονικό πρόβλημα επιτρέπει την αναπαράστασή τους με ένα κατευθυνόμενο γράφο, στον οποίο μπορεί στη συνέχεια να εφαρμοστεί ένας *αλγόριθμος εύρεσης συντομότερων μονοπατιών* (*all shortest paths algorithm*) [AS80, LS83, LW83, Sho81].

Οι ανισότητες του προβλήματος STP αναπαρίστανται με ένα δεύτερο κατευθυνόμενο γράφο που αποτελεί μια άλλη μορφή του δικτύου χρονικών περιορισμών. Ο γράφος αυτός καλείται *γράφος αποστάσεων* (*distance graph*). Έχει το ίδιο σύνολο κόμβων με τον δίκτυο TCN, αλλά κάθε περιορισμός μεταξύ δύο κόμβων-γεγονότων δεν αναπαρίσταται με μια αλλά με δύο συνδέσεις, με διαφορετική βέβαια σημασία. Ας πάρουμε και πάλι την διπλή ανισότητα:

$$a \leq X_j - X_i \leq b, b \geq a \geq 0$$

που στην ουσία αποτελείται από δύο απλές ανισότητες:

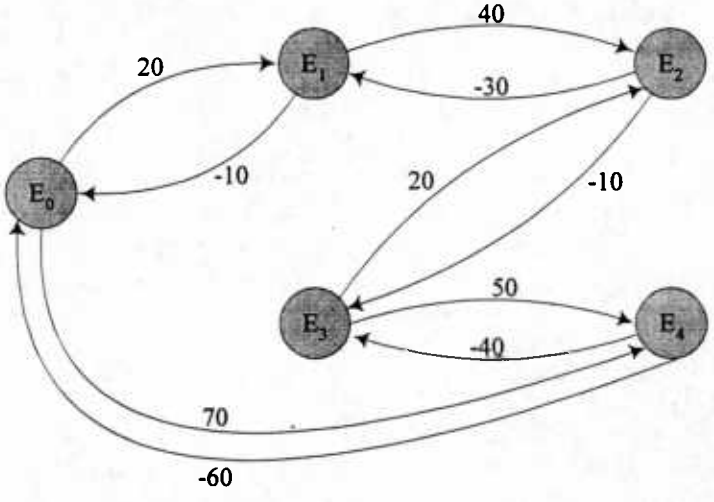
$$X_j - X_i \geq a \quad \textbf{(3.3)}$$

και

$$X_j - X_i \leq b \quad (3.4)$$

Κάθε ανισότητα αναπαρίσταται με μια σύνδεση μεταξύ των αντίστοιχων κόμβων του γράφου απόστασης. Η ανισότητα 3.3 αναπαρίσταται με μία σύνδεση που ξεκινά από τον κόμβο X_j, καταλήγει στον κόμβο X_i και έχει διάρκεια, που καλείται και *βάρος (weight)* της σύνδεσης, -a. Η ανισότητα 3.4 αναπαρίσταται με μια σύνδεση που ξεκινά από τον κόμβο X_i, καταλήγει στον κόμβο X_j και έχει διάρκεια b.

Αν στο παράδειγμα 3.1 επιλέξουμε το σενάριο στο οποίο ο Γιώργος χρησιμοποιεί αυτοκίνητο ($30 \leq E_2 - E_1 \leq 40$) τότε το απλό χρονικό πρόβλημα που προκύπτει αναπαρίσταται με τον γράφο αποστάσεων G_d που απεικονίζεται στο σχήμα 3.3.



Σχήμα 3.3: Ο γράφος αποστάσεων για ένα από τα σενάρια του παραδείγματος 3.1.

Κάθε μονοπάτι από έναν κόμβο i σε έναν κόμβο j του γράφου G_d, ικανοποιεί τον εξής περιορισμό για την απόσταση $X_j - X_i$:

$$X_j - X_i \leq \sum_{j=1}^{k} a_{i_{j-1}, i_j}, \; i_0 = i, i_1, \ldots, i_k = j \quad (3.5)$$

όπου a_{ij} είναι η απόσταση που συνδέει τους κόμβους X_i και X_j στον γράφο αποστάσεων. Αν υπάρχουν περισσότερα από ένα μονοπάτια που να συνδέουν τους δύο κόμβους και αν παραστήσουμε με d_{ij} το συντομότερο μονοπάτι τότε ισχύει:

$$X_j - X_i \leq d_{ij} \tag{3.6}$$

Βασιζόμενοι σε αυτή την παρατήρηση μπορούμε να διατυπώσουμε το εξής θεώρημα για την χρονική ακεραιότητα ενός STP προβλήματος:

Θεώρημα 3.1 *Ένα απλό χρονικό πρόβλημα, T, είναι χρονικά συνεπές αν και μόνο αν ο γράφος αποστάσεων του G_d, δεν έχει αρνητικούς κύκλους*[3].

Απόδειξη 3.1 Ας υποθέσουμε ότι σε ένα συνεπές χρονικό πρόβλημα υπάρχει ένας αρνητικός κύκλος C_n στον γράφο αποστάσεων, που αποτελείται από τους κόμβους $i_1 = i, i_2, \ldots, i_k = i$. Αθροίζοντας τις ανισότητες κατά μήκος του μονοπατιού C_n προκύπτει (ορισμός αρνητικού κύκλου): $X_i - X_i < 0$ που φυσικά είναι άτοπο. Αντίστροφα τώρα. Έστω ότι σε ένα γράφο αποστάσεων δεν υπάρχει αρνητικός κύκλος. Τότε το συντομότερο μονοπάτι μεταξύ δύο οποιοδήποτε κόμβων είναι ορθά ορισμένο. Για κάθε ζευγάρι κόμβων X_i και X_j τα συντομότερα μονοπάτια πρέπει να ικανοποιούν τις σχέσεις: $d_{0j} \leq d_{0i} + a_{ij}$. Επομένως:

$$d_{0j} - d_{0i} \leq a_{ij} \tag{3.7}$$

Αυτό σημαίνει ότι το διάνυσμα $(d_{01}, d_{02}, \ldots, d_{0n})$ είναι μια λύση του, χρονικά ακέραιου STP προβλήματος. □

Πόρισμα 3.2 *Έστω G_d ο γράφος αποστάσεων ενός συνεπούς STP προβλήματος. Δύο συνεπή σενάρια δίνονται από τα διανύσματα:*

$$S_1 = (d_{01}, d_{02}, \ldots, d_{0n}) \tag{3.8}$$

και

$$S_2 = (-d_{10}, -d_{20}, \ldots, -d_{n0}) \tag{3.9}$$

τα οποία αναθέτουν σε κάθε απόσταση μεταξύ των κόμβων X_0 (κόμβος αναφοράς) και X_i την μέγιστη και ελάχιστη τιμή του συντομότερου μονοπατιού.

[3] Κάθε μονοπάτι που ξεκινά από έναν κόμβο και καταλήγει ξανά στον ίδιο κόμβο ονομάζεται κύκλος. Σε έναν γράφο στον οποίο κάθε σύνδεση μεταξύ δύο κόμβων χαρακτηρίζεται από ένα βάρος (*weight*) , κάθε μονοπάτι και επομένως και ένας κύκλος χαρακτηρίζεται από μια απόσταση (*distance*) που προκύπτει από το άθροισμα των βαρών των κόμβων που διατρέχει το μονμοπάτι. Αν η απόσταση του κύκλου είναι αρνητική ο κύκλος καλείται αρνητικός.

Απόδειξη 3.2 Από την απόδειξη 3.1 προκύπτει ότι όντως το διάνυσμα S_1 αποτελεί μία από τις λύσεις του προβλήματος. Όσον αφορά το διάνυσμα S_2, σημειώνουμε ότι για κάθε ζευγάρι κόμβων ισχύει η σχέση: $d_{i0} \leq d_{j0} + a_{ij}$. Επομένως:

$$(-d_{j0}) - (-d_{i0}) \leq a_{ij} \tag{3.10}$$

που σημαίνει ότι και το διάνυσμα S_2 αποτελεί μια λύση του STP προβλήματος. □

Από τα μέχρι τώρα αποτελέσματα της συζήτησής μας συμπεραίνουμε ότι ένα STP πρόβλημα μπορεί να αναπαρασταθεί με τη χρήση αρχικά ενός δικτύου χρονικών περιορισμών και στη συνέχεια με τη χρήση ενός γράφου αποστάσεων. Το πρόβλημα είναι χρονικά συνεπές αν ο γράφος αποστάσεων δεν περιέχει αρνητικούς κύκλους. Στην περίπτωση αυτή δύο από τις λύσεις του προβλήματος προκύπτουν αναθέτοντας σε κάθε απόσταση μεταξύ των κόμβων X_0 (κόμβος αναφοράς) και X_i την μέγιστη και ελάχιστη τιμή του συντομότερου μονοπατιού που τους συνδέει.

Αυτό που μένει είναι η παρουσίαση ενός αλγόριθμου που θα βρίσκει την λύση ενός STP προβλήματος. Το θεώρημα 3.3 παρέχει τη βάση για έναν τέτοιο αλγόριθμο. Η απόδειξή του βρίσκεται στο παράρτημα Α.

Θεώρημα 3.3 (Αποσύνθεση STP προβλήματος) *Ένα συνεπές χρονικό πρόβλημα, Τ, μπορεί να αποσυντεθεί χρησιμοποιώντας τις ανισότητες που εκφράζονται στο διάγραμμα αποστάσεων που το αναπαριστά.*

Με βάση το θεώρημα της αποσύνθεσης, η λύση ενός STP προβλήματος μπορεί να βρεθεί αν αναθέσουμε σε κάθε μεταβλητή οποιαδήποτε τιμή που ικανοποιεί τις ανισότητες του γράφου απόστασης όπως έχουν διαμορφωθεί από προηγούμενες αναθέσεις. Το θεώρημα της αποσύνθεσης αποδεικνύει ότι μια τέτοια λύση μπορεί να βρεθεί, ανεξάρτητα από τη σειρά με την οποία γίνονται οι αναθέσεις. Ένα ακόμα συμπέρασμα που προκύπτει από το θεώρημα 3.3 αφορά το διάστημα στο οποίο ανήκουν οι λύσεις για κάθε μεταβλητή του γράφου:

Θεώρημα 3.4 *Έστω ένα συνεπές χρονικό πρόβλημα,Τ. Το διάστημα $[-d_{i0}, d_{0i}]$ περιέχει όλες τις δυνατές λύσεις για την μεταβλητή X_i.*

Απόδειξη 3.4 Σύμφωνα με το θεώρημα 3.3, η ανάθεση $X_0 = 0$, μπορεί να επεκταθεί αναθέτοντας οποιαδήποτε τιμή $u \in [-d_{i0}, d_{0i}]$ στην μεταβλητή X_i. Αυτή η ανάθεση από την άλλη πλευρά μπορεί να επεκταθεί σε μια πλήρης λύση. Επομένως η τιμή u αποτελεί μια από τις λύσεις του προβλήματος. □

Κλείνουμε την θεωρητική στήριξη της μεθοδολογίας επίλυσης STP προβλημάτων με την παρουσίαση της έννοιας του *ελάχιστου δικτύου* (*minimal network*). Πρόκειται για το γράφο που προκύπτει στη σύνδεση κάθε ζεύγους κόμβων X_i, X_j χρησιμοποιήσουμε το μήκος του ελάχιστου μονοπατιού που συνδέει τους δύο κόμβους.

Πόρισμα 3.5 *Έστω ένα συνεπές STP πρόβλημα, Τ. Ο γράφος αποστάσεων οι αποστάσεις του οποίου ορίζονται από τη σχέση:*

$$\forall\ i,j,\ M_{ij} = \{[-d_{ji},\ d_{ij}]\} \tag{3.11}$$

αποτελεί το ελάχιστο δίκτυο του προβλήματος Τ.

Απόδειξη 3.5 Η απόδειξη βρίσκεται στο παράρτημα Α. □

Συνοψίζοντας τα τελικά συμπεράσματα της συζήτησής μας, για την επίλυση ενός STP προβλήματος αρκεί να σχηματίσουμε τον γράφο αποστάσεων του προβλήματος και στη συνέχεια να υπολογίσουμε τα συντομότερα μονοπάτια μεταξύ όλων των κόμβων του γράφου. Αν κάποιο μονοπάτι που ξεκινά από έναν κόμβο και καταλήγει στον ίδιο κόμβο έχει αρνητικό βάρος, το πρόβλημα δεν είναι χρονικά συνεπές. Στην αντίθετη περίπτωση για κάθε δύο κόμβους X_i, X_j το διάστημα λύσεων βρίσκεται στο διάστημα $[-d_{ji},\ d_{ij}]$ όπου d_{ij} είναι το μήκος του συντομότρου μονοπατιού από τον κόμβο X_j στον κόμβο X_i και d_{ji} το αντίστοιχο μήκος του μονοπατιού από τον κόμβο X_i στον κόμβο X_j.

Θα κλείσουμε την παρουσίαση της μεθοδολογίας επίλυσης ενός STP προβλήματος με την εφαρμογή της στο πρόβλημα του παραδείγματος 3.1. Αρχικά χρησιμοποιούμε το γράφο του σχήματος 3.3 ($30 \le E_2 - E_1 \le 40$). Ο πίνακας 3.3 παρουσιάζει τα μήκη των συντομότερων μονοπατιών μεταξύ δύο οποιονδήποτε κόμβων του γράφου. Για παράδειγμα το γεγονός E_2 (ο Γιώργος έφθασε στο γραφείο) θα συμβεί το πολύ 50 λεπτά μετά το σημείο αναφοράς (7:00) άλλα όχι λιγότερο από 40 λεπτά. Ο πρώτος περιορισμός βρίσκεται στη θέση E_0- E_2 του πίνακα και ο δεύτερος στη θέση E_2 - E_0.

	E_0	E_1	E_2	E_3	E_4
E_0	0	20	50	30	70
E_1	-10	0	40	20	60
E_2	-40	-30	0	10	30
E_3	-20	-10	20	0	50
E_4	-60	-50	-20	-40	0

Πίνακας 3.3: Τα μήκη των συντομότερων μονοπατιών του γράφου αποστάσεων του σχήματος 3.3.

Χρησιμοποιώντας το πόρισμα 3.2, μια λύση του προβλήματος μπορεί να παραχθεί αν επιλέξουμε για κάθε μεταβλητή την μέγιστή της απόσταση από το σημείο αναφοράς E_0. Επομένως μια πιθανή λύση είναι η:

$$S_1 = \{X_1 = 20,\ X_2 = 50,\ X_3 = 30,\ X_4 = 70\} \tag{3.12}$$

Σύμφωνα με τη λύση αυτή ο Γιώργος έφυγε από το σπίτι του στις 7:20 και έφθασε στη δουλειά στις 7:50. Ο Νίκος έφυγε από το σπίτι στις 7:30 και έφθασε στη δουλειά στις 8:10.

Όσον αφορά το ελάχιστο δίκτυο, που στην ουσία είναι και αυτό που μας ενδιαφέρει προκύπτει εύκολα από τον πίνακα 3.2. Για παράδειγμα για το διάστημα των λύσεων των χρονικών αποστάσεων μεταξύ των κόμβων E_2 και E_4 βλέπουμε ότι το ελάχιστο μονοπάτι από τον κόμβο E_2 στον κόμβο E_4 έχει μήκος 30 ενώ για την αντίστροφη κατεύθυνση το μήκος είναι -20. Άρα το διάστημα λύσεων είναι [-(-20), 30] ή [20, 30], όπως ορίζει η σχέση (3.11). Ο πίνακας 3.4 περιέχει το ελάχιστο δίκτυο του παραδείγματος 3.1. Πρέπει να παρατηρήσουμε την συμμετρία που χαρακτηρίζει το δίκτυο. Αν το διάστημα λύσεων μεταξύ των κόμβων X_i και X_j είναι το [a, b], το διάστημα λύσεων μεταξύ των κόμβων X_j και X_i είναι το [-b, a].

	E_0	E_1	E_2	E_3	E_4
E_0	[0, 0]	[10, 20]	[40, 50]	[20, 30]	[60, 70]
E_1	[-20, -10]	[0, 0]	[30, 40]	[10, 20]	[50, 60]
E_2	[-50, -40]	[-40, -30]	[0, 0]	[-20, -10]	[20, 30]
E_3	[-30, -20]	[-20, -10]	[10, 20]	[0, 0]	[40, 50]
E_4	[-70, -60]	[-60, -50]	[-30, -20]	[-50, -40]	[0, 0]

Πίνακας 3.4: Το ελάχιστο δίκτυο του γράφου αποστάσεων του σχήματος 3.3.

Όσον αφορά το δεύτερο σενάριο, στο οποίο ο Γιώργος χρησιμοποιεί αυτοκίνητο και επομένως τα γεγονότα E_2 και E_1 συνδέονται με την ανισότητα $E_2 - E_1 \geq 60$, εύκολα διαπιστώνουμε ότι περιέχει αρνητικούς κύκλους. Επομένως το σενάριο αυτό είναι χρονικά ασυνεπές και το αντίστοιχο STP πρόβλημα δεν έχει λύση.

3.3.3 Αλγόριθμος για την επίλυση ενός απλού χρονικού προβλήματος

Η διαδικασία επίλυσης ενός απλού χρονικού προβλήματος, σύμφωνα με τη συζήτηση που προηγήθηκε μπορεί να χωριστεί σε τέσσερα βήματα:

1. Αναπαράσταση του προβλήματος με ένα σύνολο χρονικών ανισοτήτων
2. Αναπαράσταση των ανισοτήτων με τη χρήση ενός δικτύου χρονικών περιορισμών
3. Αναπαράσταση του δικτύου χρονικών περιορισμών από ένα κατευθυνόμενο γράφο αποστάσεων
4. Εύρεση των συντομότερων μονοπατιών μεταξύ των κόμβων του γράφου απόστασης

Τα τρία πρώτα βήματα εξαρτώνται από το χρονικό πρόβλημα το οποίο προσπαθούμε να επιλύσουμε. Όσον αφορά το τέταρτο βήμα μπορούμε να εφαρμόσουμε έναν από τους αλγορίθμους για την εύρεση των συντομότερων μονοπατιών σε έναν κατευθυνόμενο γράφο. Ο πιο γνωστός αλγόριθμος διατυπώθηκε από τους Floyd και Warshall [Dro95, Ev79, Tar81]. Η πολυπλοκότητα του αλγορίθμου είναι $O(n^3)$, όπου n είναι το πλήθος

των κόμβων του γράφου. Εντοπίζει τους αρνητικούς κύκλους ελέγχοντας την τιμή των διαγωνίων στοιχείων d_{ii}. Επομένως ο αλγόριθμος μπορεί σε πολυωνυμικό χρόνο να αποφασίσει την ακεραιότητα ενός STP προβλήματος και να υπολογίσει το ελάχιστο δίκτυο. Ο αλγόριθμος παρουσιάζεται στο σχήμα 3.4.

```
for i = 1 to n do d[i, i] = 0;
for k = 1 to n do
    for i, j = 1 to n do
        d[i, j] = min(d[i, j], d[i,k] + d[k, j]);
```

Σχήμα 3.4: Ο αλγόριθμος των Floyd-Warshall για την εύρεση των συντομότερων μονοπατιών σε ένα γράφο.

Όπως βλέπουμε το πρώτο βήμα αφορά το μηδενισμό των διαγωνίων στοιχείων. Στη συνέχεια διατρέχονται όλοι οι κόμβοι του δικτύου. Για κάθε τρεις κόμβους *i*, *j*, *k* ο αλγόριθμος ελέγχει το μήκος του μονοπατιού που συνδέει άμεσα τους κόμβους *i* και *j* και το συγκρίνει με το μονοπάτι που χρησιμοποιεί σαν ενδιάμεσο κόμβο τον κόμβο *k*. Σαν απόσταση του μονοπατιού d_{ij} καταχωρεί το συντομότερο από τα δύο μονοπάτια.

Ο αλγόριθμος χαρακτηρίζεται από δύο στοιχεία. Το πρώτο αφορά την ευκολία με την οποία υπολογίζεται η πολυπλοκότητά του (εξαιτίας του τριπλού βρόγχου που αποτελεί τον κορμό του αλγορίθμου) και την ευκολία με την οποία μπορεί να υλοποιηθεί. Το δεύτερο χαρακτηριστικό αφορά την αποδοτικότητά του. Όταν οι γράφοι είναι πλήρεις ο αλγόριθμος παρουσιάζει υψηλή αποδοτικότητα, σε σχέση με άλλους αλγορίθμους εύρεσης συντομότερων μονοπατιών. Όσο όπως μειώνεται ο αριθμός των συνδέσεων μεταξύ των κόμβων του γράφου τόσο μειώνεται και η αποδοτικότητα του αλγορίθμου.

3.4 Η αρχιτεκτονική της μεθοδολογίας SIC

Το ζήτημα των χρονικών προβλημάτων δεν είναι φυσικά άσχετο με το πρόβλημα της ακεραιότητας μιας multimedia εφαρμογής. Όπως προαναφέραμε η ακεραιότητα μιας εφαρμογής συνίσταται στην ικανοποίηση ενός συνόλου χρονικών περιορισμών που αφορούν τα αντικείμενα που λαμβάνουν μέρος στην multimedia παρουσίαση. Φυσικά η εφαρμογή μπορεί να ακολουθήσει πολλά διαφορετικά μονοπάτια εκτέλεσης εξαιτίας των ενεργειών του χρήστη. Όπως είδαμε στην ανάλυση του παραδείγματος 3.1, κάθε διαφορετικό μονοπάτι εκτέλεσης αποτελεί ένα απλό χρονικό πρόβλημα.

Επομένως για να ελέγξουμε την ακεραιότητα μιας multimedia εφαρμογής πρέπει να εντοπίσουμε τα διαφορετικά μονοπάτια εκτέλεσης και στη συνέχεια για κάθε μονοπάτι να εφαρμόσουμε τα 4 βήματα που παρουσιάσαμε στην υποπαράγραφο 3.3.3. Αυτή είναι η γενική φιλοσοφία που ακολουθείται από την μεθοδολογία SIC. Το βασικό ζήτημα που αντιμετωπίζει είναι ο τρόπος με τον οποίο θα μεταφράσει τους περιορισμούς που χαρακτηρίζουν τα αντικείμενα της εφαρμογής σε ένα σύνολο

ανισοτήτων, που στη συνέχεια θα χρησιμοποιηθεί για να κατασκευαστεί ο γράφος αποστάσεων κάθε μονοπατιού.

Η μεθοδολογία μπορεί να συνοψιστεί ως εξής. Είσοδο της διαδικασίας ελέγχου αποτελεί ένα IMD σενάριο, που περιγράφει μια multimedia εφαρμογή. Για κάθε tuple υπολογίζονται οι χρονικές ανισότητες που συνδέουν τις ενέργειες που εκτελεί στα αντικείμενα της εφαρμογής. Οι ανισότητες μετασχηματίζονται σε ένα δίκτυο χρονικών περιορισμών που καλείται *Constraint Network Fragment* (*CNF*).

Στη συνέχεια το σενάριο εξετάζεται στο σύνολό του για να προσδιοριστούν τα διαφορετικά μονοπάτια εκτέλεσης (scenario paths). Το αποτέλεσμα αυτής της διαδικασίας είναι ο προσδιορισμός του διαγράμματος μετάβασης καταστάσεων (state transition diagram) της εφαρμογής. Μια κατάσταση καθορίζεται από το σύνολο των tuples τα οποία είναι ενεργά κατά την διάρκειά της. Η μετάβαση από τη μία κατάσταση στην άλλη προκύπτει εξαιτίας ενός γεγονότος που δέχεται η εφαρμογή. Το γεγονός έχει σαν αποτέλεσμα την διακοπή εκτέλεσης κάποιων από τα ενεργά tuples ή/και την εκκίνηση κάποιων ανενεργών tuples.

Το επόμενο στάδιο αφορά τη χρήση των αποτελεσμάτων των δύο προηγούμενων σταδίων. Για κάθε μονοπάτι εκτέλεσης συλλέγονται τα CNFs των διαφορετικών tuples που συμμετέχουν σε αυτό. Χρησιμοποιώντας ένα σύνολο κανόνων τα CNFs συνδέονται παράγοντας ένα η περισσότερα δίκτυα χρονικών περιορισμών για κάθε μονοπάτι. Το συνολικό αυτό δίκτυο καλείται *σύνθετο δίκτυο περιορισμών* (*composite constraint network*, *CN*) και αποτελεί την είσοδο για το τελευταίο βήμα της μεθοδολογίας. Το βήμα αυτό αντιστοιχεί στο βήμα 4 της υποπαραγράφου 3.3.3. Εφαρμόζοντας τον αλγόριθμο συντομότερων μονοπατιών μπορούμε να συμπεράνουμε ποια από τα σύνθετα δίκτυα περιορισμών είναι χρονικά συνεπή και ποια όχι. Εκτός από τον τελικό έλεγχο της ακεραιότητας, επιμέρους έλεγχοι εκτελούνται τόσο κατά την κατασκευή των CNFs όσο και κατά τη σύνθεσή τους σε σύνθετα δίκτυα περιορισμών. Η γενική αρχιτεκτονική της μεθοδολογίας απεικονίζεται στο σχήμα 3.5.

Είναι φανερό ότι η μεθοδολογία χωρίζεται σε τέσσερα διακριτά στάδια:

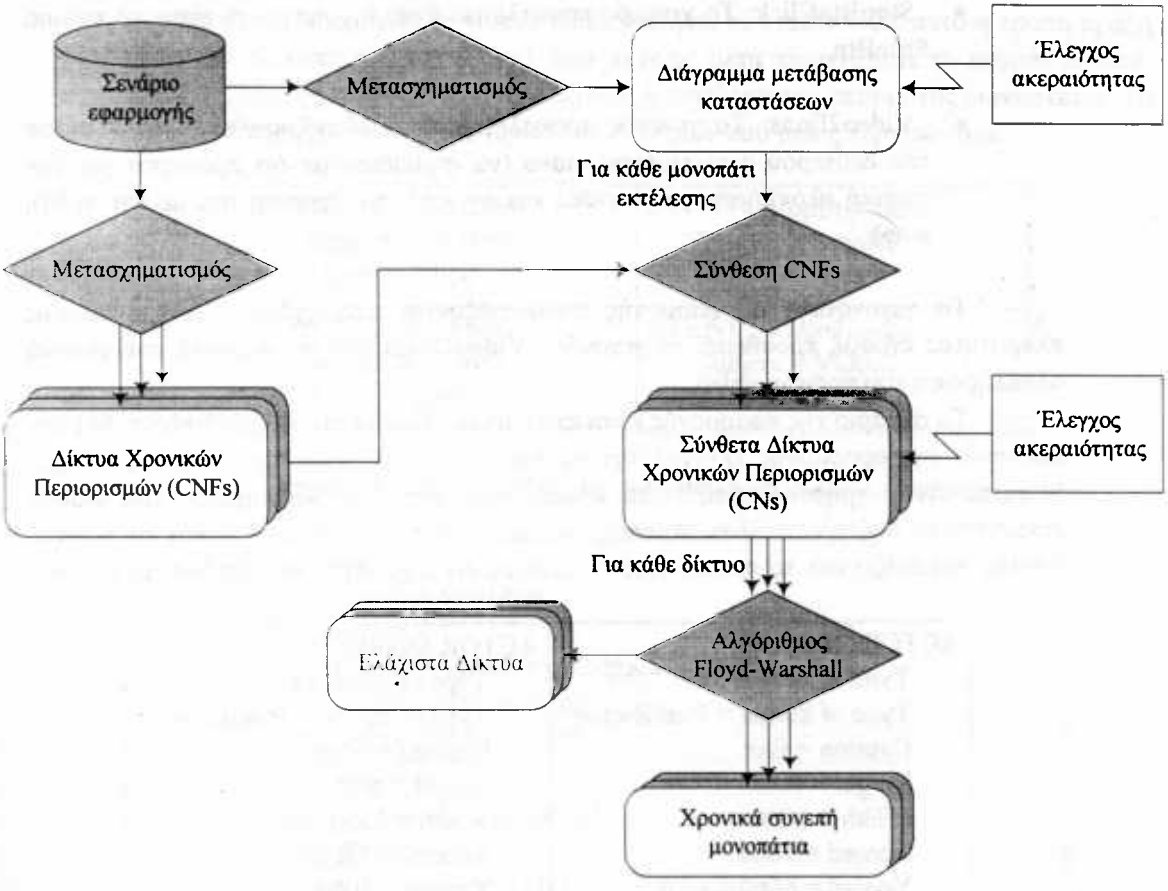
Στάδιο 1. Δημιουργία διαγράμματος μετάβασης καταστάσεων και εύρεση μονοπατιών εκτέλεσης

Στάδιο 2. Κατασκευή Δικτύων Χρονικών Περριορισμών

Στάδιο 3. Σύνθεση Δικτύων Χρονικών Περιορισμών

Στάδιο 4. Έλεγχος συνολικής ακεραιότητας

Κάθε στάδιο βασίζει τη λειτουργία του σε ένα σύνολο κανόνων. Στη συνέχεια του κεφαλαίου θα αφιερώσουμε μία παράγραφο σε κάθε ένα από τα τέσσερα στάδια. Θα αναλύσουμε αυτούς τους κανόνες και θα δούμε πως εφαρμόζονται σε μια απλή multimedia εφαρμογή.



Σχήμα 3.5: Η γενική αρχιτεκτονική της μεθοδολογίας Scenario Integrity Checking.

3.4.1 Ένα απλό παράδειγμα multimedia εφαρμογής

Το παράδειγμα που θα χρησιμοποιήσουμε για να επιδείξουμε την μεθοδολογία επιλέχτηκε ώστε να είναι σχετικά απλό - ώστε να μπορεί να γίνει κατανοητή η μεθοδολογία από τον αναγνώστη, αλλά να περιλαμβάνει αλληλεπίδραση με το χρήστη και ένα ασυνεπές μονοπάτι.

Η εφαρμογή του παραδείγματος Example1 αποτελείται από τέσσερα αντικείμενα. Δύο αντικείμενα αφορούν κουμπιά με τα οποία εξασφαλίζεται η αλληλεπίδραση με το χρήστη. Τα άλλα δύο αντικείμενα είναι δύο video Video1, και Video2 με διαφορετική χρονική διάρκεια. Το πρώτο έχει διάρκεια 10 δευτερολέπτων ενώ το δεύτερο διαρκεί 15 δευτερόλεπτα. Το σχήμα 3.6 παρουσιάζει τα 4 αντικείμενα της εφαρμογής.

Το σενάριο χρησιμοποιεί τρία γεγονότα για να ορίσει τη ροή εκτέλεσης της εφαρμογής:

- _ExitBtnClick: Το γεγονός προκαλείται όταν ο χρήστης πατήσει το κουμπί ExitBtn.

- _StopBtnClick: Το γεγονός προκαλείται όταν ο χρήστης πατήσει το κουμπί StopBtn.

- _Video2Ends: Το γεγονός προκαλείται όταν ολοκληρωθεί η παρουσίαση του δεύτερου αντικειμένου video (να σημειώσουμε ότι πρόκειται για την φυσική ολοκλήρωση του video και όχι από την διακοπή του με την πράξη stop).

Τα γεγονότα της εφαρμογής παρουσιάζονται στο σχήμα 3.7. Για λόγους πληρότητας έχουμε προσθέσει το γεγονός _Video1Ends που αναπαριστά την φυσική ολοκλήρωση του πρώτου video.

Το σενάριο της εφαρμογής είναι πολύ απλό. Χωρίζεται σε τρία στάδια. Με την εκκίνηση της εφαρμογής εκκινούνται τα δύο video και εμφανίζεται και το κουμπί StopBtn. Αν ο χρήστης πατήσει το κουμπί πριν από την ολοκλήρωση του video2 εκκινήται το δεύτερο στάδιο. Αμέσως σταματά τόσο το video1 όσο και το video2. Επίσης εμφανίζει και το κουμπί εξόδου ExitBtn. Αν ο χρήστης δεν πατήσει το κουμπί

ACTOR ExitBtn Type of actor = BUTTON Type of button = PushButton Caption = Exit Height = 500 Width = 1000 Xcoord = 7000 Ycoord = 6000 Tranparent = FALSE FontFace = MS Sans Serif FontSize = 14 Color = 255, 0, 0 FontStyle = Regular	ACTOR StopBtn Type of actor = BUTTON Type of button = PushButton Caption = Stop Height = 500 Width = 1000 Xcoord = 7500 Ycoord = 3000 Tranparent = FALSE FontFace = MS Sans Serif FontSize = 14 Color = 255, 0, 0 FontStyle = Regular
ACTOR Video1 Type of actor = AVI FileName = KAVAL1.AVI Start = 0 Duration = 10 Scale Factor = 1 Volume = 70 Looping = No Direction = Forward Height = 3500 Width = 2500 Xcoord = 500 Ycoord = 1400	ACTOR Video2 Type of actor = AVI FileName = KAVAL2.AVI Start = 145 Duration = 15 Scale Factor = 1 Volume = 70 Looping = No Direction = Forward Height = 3500 Width = 2500 Xcoord = 500 Ycoord = 4000

Σχήμα 3.6: Τα αντικείμενα της multimedia εφαρμογής Example1.

StopBtn και ολοκληρωθεί το video2 τότε εκκινήται το στάδιο 3. Σε αυτό η πρώτη πράξη αφορά τη διακοπή και του video1 ενώ αμέσως μετά εμφανίζεται το κουμπί εξόδου. Μόλις ο χρήστης πατήσει το κουμπί εξόδου η εκτέλεση της εφαρμογής διακόπτεται. Το σχήμα 3.8 περιέχει τα τέσσερα tuples του σεναρίου που μόλις περιγράψαμε.

EVENT _ExitBtnClick Subject = ExitBtn Action = Click	EVENT _StopBtnClick Subject = StopBtn Action = Click
EVENT _Video1Ends Subject = Video1 Action = Close	EVENT _Video2Ends Subject = Video2 Action = Close

Σχήμα 3.7: Τα γεγονότα της multimedia εφαρμογής Example1.

TUPLE Stage1 Start_Event = StartApp Stop_Event = Action_List = Video1> 0 Video2> 0 StopBtn> Start_Synch_Event = None Stop_Synch_Event = None
TUPLE Stage2 Start_Event = _StopBtnClick Stop_Event = Action_List = Video1! 0 Video2! 0 ExitBtn> Start_Synch_Event = None Stop_Synch_Event =
TUPLE Stage3 Start_Event = _Video2Ends Stop_Event = None Action_List = Video1! 0 ExitBtn> Start_Synch_Event = None Stop_Synch_Event = None
TUPLE ExitTuple Start_Event = _ExitBtnClick Stop_Event = None Action_List = ExitApplication Start_Synch_Event = None Stop_Synch_Event = None

Σχήμα 3.8: Τα tuples της multimedia εφαρμογής Example1.

Το παράδειγμα είναι τόσο απλό που αμέσως φαίνονται οι περιορισμοί που θέτει. Καταρχήν ας πάρουμε την περίπτωση στην οποία ο χρήστης πατάει το κουμπί

stop. Επειδή αυτό θα έχει σαν αποτέλεσμα το σταμάτημα των δύο video, το κουμπί πρέπει να πατηθεί ενώ τα δύο video είναι ακόμα ενεργά. Δηλαδή, πριν την ολοκλήρωση του μικρότερου σε διάρκεια Video1, που θα συμβεί στο δέκατο δευτερόλεπτο της εκτέλεσης της εφαρμογής. Στην περίπτωση που ο χρήστης δεν πατήσει το κουμπί το σενάριο είναι φανερά ασυνεπές. Το tuple Stage3 θα εκτελεστεί στο 15ο δευτερόλεπτο εκτέλεσης της εφαρμογής, δηλαδή μόλις ολοκληρωθεί το Video2. Ως τότε όμως θα έχει ολοκληρωθεί και το Video1. Επομένως η πράξη διακοπής του Video1 θα προκαλέσει πρόβλημα.

3.5 Εύρεση μονοπατιών εκτέλεσης

Οι εφαρμογές multimedia μπορούν να χωριστούν σε δύο κατηγορίες. Στην πρώτη κατηγορία μπορούμε να κατατάξουμε τις εφαρμογές εκείνες που δεν περιλαμβάνουν αλληλεπίδραση με το χρήστη. Τις εφαρμογές αυτές τις έχουμε ήδη ονομάσει προκαθορισμένες (preorchistrated) εξαιτίας του γεγονότος ότι η περιγραφή της εφαρμογής καθορίζει πλήρως της ροή εκτέλεσής της. Κατά συνέπεια στις προκαθορισμένες multimedia εφαρμογές υπάρχει ένα και μόνο μονοπάτι εκτέλεσης. Η δεύτερη κατηγορία εφαρμογών περιλαμβάνει τις εφαρμογές στις οποίες ο χρήστης μπορεί να επέμβει στη ροή εκτέλεσής τους. Στις εφαρμογές αυτές υπάρχουν περισσότερα από ένα μονοπάτια εκτέλεσης.

Όσον αφορά το IMD μοντέλο αυτό σημαίνει ότι η ακολουθία εκτέλεσης των tuples του σεναρίου δεν είναι προκαθορισμένη. Ανάλογα με τα γεγονότα που δημιουργούνται, άλλα tuples ενεργοποιούνται - και επομένως αρχίζει η εκτέλεση των ενεργειών που ορίζουν - ενώ άλλα διακόπτουν την εκτέλεσή τους. Κατά αντιστοιχία με τα αντικείμενα της εφαρμογής, τα tuples μπορούν να βρίσκονται είτε σε κατάσταση Active είτε σε κατάσταση Idle. Σε κάθε χρονική στιγμή της εκτέλεσης της εφαρμογής ένα υποσύνολο των tuples είναι ενεργό ενώ τα υπόλοιπα tuples βρίσκονται σε κατάσταση Idle. Τα δύο αυτά υποσύνολα ορίζουν την *συνολική κατάσταση* της εφαρμογής (*global state*).

Η αλλαγή της κατάστασης της εφαρμογής συνίσταται στην τροποποίηση της συνολικής κατάστασής της. Αυτό σημαίνει ότι είτε κάποια από τα tuples που ήταν στην προηγούμενη κατάσταση ενεργά θα πάψουν να εκτελούνται, είτε κάποια από τα tuples που βρίσκονταν σε κατάσταση Idle θα εκκινηθούν. Είναι πιθανόν να υπάρξουν ταυτόχρονα και οι δύο περιπτώσεις, για διαφορετικά φυσικά tuples. Αυτή η αλλαγή στην συνολική κατάσταση της εφαρμογής καλείται *μετάβαση* (*transition*). Είναι φανερό ότι οι μεταβάσεις προκαλούνται από τα γεγονότα που δέχεται η εφαρμογή και προκαλούν την εκκίνηση ή την διακοπή των tuples.

Σκοπός του πρώτου σταδίου της μεθοδολογίας είναι να προσδιορίσει τις καταστάσεις της εφαρμογής καθώς και τα γεγονότα που προκαλούν τις μεταβάσεις από την μία κατάσταση στην άλλη. Αποτέλεσμα της εφαρμογής του σταδίου, είναι η κατασκευή ενός διαγράμματος μετάβασης καταστάσεων (state transition diagram). Το διάγραμμα αυτό μπορεί να χρησιμοποιηθεί για να προσδιοριστούν στη συνέχεια τα μονοπάτια εκτέλεσης της εφαρμογής. Κατά την κατασκευή του διαγράμματος

μετάβασης καταστάσεων πραγματοποιούνται κάποιοι έλεγχοι για την συνέπεια των μονοπατιών εκτέλεσης. Στη συνέχεια θα παρουσιάσουμε τους βασικούς κανόνες κατασκευής του διαγράμματος αλλά και τους ελέγχους που πραγματοποιούνται.

3.5.1 Κανόνες κατασκευής διαγράμματος μετάβασης καταστάσεων

Βασικές μονάδες στο διάγραμμα μετάβασης καταστάσεων αποτελούν οι συνολικές καταστάσεις της εφαρμογής και οι μεταβάσεις που τις τροποποιούν. Η συνολική κατάσταση περιγράφεται από το σύνολο των tuples τα οποία βρίσκονται σε κατάσταση Active. Επομένως μια κατάσταση S μπορεί να περιγραφεί ως εξής:

$$S = \{T_1, T_2, \ldots, T_n\} \tag{3.13}$$

όπου T_i είναι τα tuples που είναι ενεργά κατά τη διάρκεια της κατάστασης. Πρέπει να σημειώσουμε ότι σε μία κατάσταση δεν επιτρέπεται να υπάρχουν δύο *στιγμιότυπα* (*instances*) ενός tuple. Δηλαδή δεν μπορούμε να εκκινήσουμε ένα tuple ενώ είναι ήδη ενεργό. Όσον αφορά τη σχέση 3.13 αυτό σημαίνει ότι τα tuples T_i είναι ανά δύο διαφορετικά μεταξύ τους.

Μια μετάβαση περιγράφεται από τα γεγονότα τα οποία την προκαλούν. Μια μετάβαση μπορεί να προκληθεί είτε από την σύζευξη γεγονότων είτε από την διάζευξη γεγονότων, όπου φυσικά τα γεγονότα μπορεί να είναι είτε απλά είτε σύνθετα. Αν υποθέσουμε ότι η μετάβαση Tr_c προκαλείται από τη σύζευξη των γεγονότων $e_1, e_2, \ldots, e_n$ τότε η μετάβαση θα αναπαρίσταται με το συμβολισμό:

$$Tr_c = e_1, e_2, \ldots, e_n \tag{3.14}$$

Αν η μετάβαση Tr_d προκαλείται από την διάζευξη των γεγονότων $e_1, e_2, \ldots, e_n$ τότε η μετάβαση θα αναπαρίσταται με το συμβολισμό:

$$Tr_d = e_1 / e_2 / \ldots / e_n \tag{3.15}$$

Κάθε γεγονός (απλό η συγχρονισμού) πρέπει να συνδέεται με μία τουλάχιστον μετάβαση. Η αναφορά του ονόματος ενός γεγονότος σε μία μετάβαση υπονοεί ότι το γεγονός συνέβη και προκάλεσε τη μετάβαση. Αν σε μία από τις επόμενες μεταβάσεις πρέπει να αναφερθούμε σε μια δεύτερη εμφάνιση του γεγονότος, θα πρέπει να χρησιμοποιήσουμε ένα διαφορετικό όνομα για να αναφερθούμε σε αυτό. Για κάθε εμφάνιση ενός γεγονότος στο διάγραμμα μετάβασης καταστάσεων, χρησιμοποιείται ένα μοναδικό όνομα.

Κάθε διάγραμμα μετάβασης καταστάσεων ξεκινά από την κατάσταση Idle. Είναι η κατάσταση στην οποία βρίσκεται η εφαρμογή πριν δεχθεί το γεγονός *StartApp* και εκκινηθεί. Επίσης είναι η κατάσταση στην οποία καταλήγει η εφαρμογή αφού δεχθεί το γεγονός ExitApplication, που διακόπτει την εκτέλεσή της. Το γεγονός StartApp προκαλεί την εκκίνηση ενός αριθμού tuples που σχηματίζουν την πρώτη συνολική κατάσταση της εφαρμογής. Ένα μονοπάτι εκτέλεσης ξεκινά από την

κατάσταση Idle και καταλήγει ξανά σε αυτήν αφού διανύσει μια διαδρομή καταστάσεων.

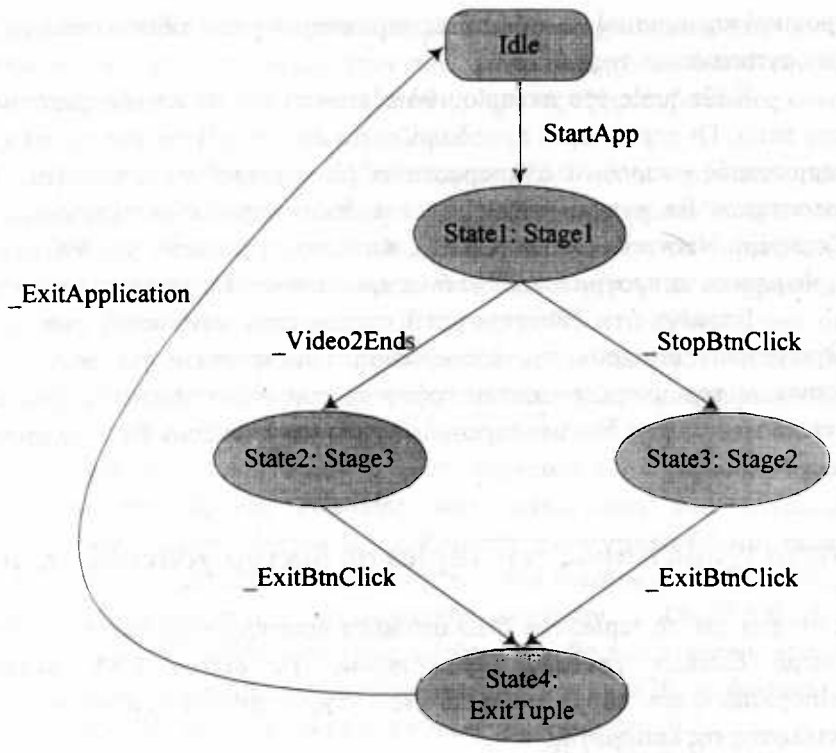
Η κατάσταση Idle αποτελεί μια ειδική κατάσταση αφού σε αυτήν κανένα από τα tuples του σεναρίου δεν είναι ενεργό. Υπάρχει μια ακόμα ειδική κατάσταση στην οποία επίσης δεν υπάρχει ενεργό tuple, ωστόσο η εφαρμογή παραμένει ενεργή. Η κατάσταση αυτή ονομάζεται *αόριστη κατάσταση* (*Undefined state)* εξαιτίας του γεγονότος ότι δεν είναι δυνατόν να προσδιοριστεί ο τρόπος με τον οποίο θα συνεχιστεί η ροή εκτέλεσης της εφαρμογής. Ένα μονοπάτι εκτέλεσης που ξεκινά από την κατάσταση Idle και καταλήγει στην κατάσταση Undefined θα καλείται στο εξής *αόριστο μονοπάτι* (*undefined path*).

Αυτοί είναι οι βασικοί κανόνες στους οποίους στηρίζεται η διαδικασία κατασκευής του διαγράμματος μετάβασης καταστάσεων. Από τον τρόπο με τον οποίο περιγράφηκε είναι φανερό ότι το διάγραμμα μπορεί να αναπαρασταθεί από έναν κατευθυνόμενο γράφο. Οι κόμβοι του γράφου θα αναπαριστούν τις συνολικές καταστάσεις της εφαρμογής και θα χαρακτηρίζονται από το σύνολο των tuples τα οποία συμμετέχουν σε αυτές. Οι συνδέσεις μεταξύ των κόμβων αναπαριστούν τις μεταβάσεις καταστάσεων και χαρακτηρίζονται από τα γεγονότα (ή για να είμαστε πιο ακριβείς την σύζευξη ή την διάζευξη γεγονότων) που τις προκαλούν. Η αναπαράσταση αυτή είναι ιδανική αφού μπορούμε να εφαρμόσουμε αλγόριθμους γράφων για να προσδιορίσουμε τα μονοπάτια εκτέλεσης της εφαρμογής, που αποτελούν το βασικό στόχο της κατασκευής του διαγράμματος. Περισσότερες λεπτομέρειες θα παρουσιαστούν στο επόμενο κεφάλαιο που ασχολείται με την υλοποίηση της μεθοδολογίας. Το διάγραμμα μετάβασης καταστάσεων για το απλό παράδειγμα της εφαρμογής Example1 απεικονίζεται στο σχήμα 3.9. Η κατάσταση Idle απεικονίζεται με ένα παραλληλόγραμμο, ακριβώς για να υποδειχθεί το γεγονός ότι πρόκειται για μια ειδική κατάσταση. Οι υπόλοιπες καταστάσεις απεικονίζονται με μία έλλειψη που εμπεριέχει τα tuples που τις αποτελούν. Όσον αφορά τις μεταβάσεις, χρησιμοποιούμε τους συμβολισμούς 3.14 και 3.15.

Από το διάγραμμα διαπιστώνουμε ότι υπάρχουν δύο μονοπάτια εκτέλεσης:

- *P1: StartApp - _StopBtnClick - _ExitBtnClick - ExitApplication.* Η πρώτη κατάσταση της εφαρμογής είναι η κατάσταση State1 στην οποία μόνο το tuple Stage1 είναι ενεργό. Το γεγονός _StopBtnClick ενεργοποιεί το tuple Stage2 που είναι το μοναδικό ενεργό tuple της κατάστασης State3. Τέλος η τελευταία κατάσταση είναι η State4 στην οποία οδηγούμαστε από το γεγονός _ExitBtnClick και στην οποία συμμετέχει το tuple ExitTuple. Το γεγονός ExitApplication οδηγεί ξανά στην κατάσταση Idle.

- *P1: StartApp - _Video2Ends - _ExitBtnClick - ExitApplication.* Η πρώτη κατάσταση της εφαρμογής είναι η κατάσταση State1 στην οποία μόνο το tuple Stage1 είναι ενεργό. Η ολοκλήρωση του Video2 ενεργοποιεί το tuple Stage3 που είναι το μοναδικό ενεργό tuple της κατάστασης State2. Τέλος η τελευταία κατάσταση είναι η State4 στην οποία οδηγούμαστε από το γεγονός _ExitBtnClick και στην οποία συμμετέχει το tuple ExitTuple.



Σχήμα 3.9: Το διάγραμμα μετάβασης καταστάσεων του παραδείγματος Example1.

Πρέπει να σημειώσουμε ότι ένα tuple θεωρείται ενεργό όσο υπάρχουν ενέργειες που ορίζει και δεν έχουν ακόμα εκτελεστεί. Για το λόγο αυτό ενώ κανένα από τα tuples δεν ορίζει ένα Stop Event κανένα δεν είναι παρών σε δύο συνεχόμενες καταστάσεις.

3.5.2 Έλεγχος ακεραιότητας μονοπατιών εκτέλεσης

Ο βασικός έλεγχος της ακεραιότητας των μονοπατιών εκτέλεσης της εφαρμογής αφορά τον εντοπισμό των μονοπατιών που οδηγούν στην αόριστη κατάσταση. Κάθε μονοπάτι που δεν οδηγεί στην κατάσταση Idle μπορεί να δημιουργήσει πρόβλημα στην εκτέλεση της εφαρμογής. Ωστόσο υπάρχουν περιπτώσεις που ο χρήστης επιθυμεί την μετάβαση στην αόριστη κατάσταση. Για το λόγο αυτό τα αόριστα μονοπάτια δεν σημειώνονται σαν *λάθη* (*errors*) αλλά σαν *προειδοποιήσεις* (*warnings*).

Στο παράδειγμά μας και τα δύο μονοπάτια οδηγούν ξανά σε κατάσταση Idle. Επομένως δεν προκαλείται κάποια προειδοποίηση.

3.6 Κατασκευή δικτύων χρονικών περιορισμών

Σκοπός αυτού του βήματος είναι η έκφραση του χρονικού προβλήματος που παρουσιάζει μια multimedia εφαρμογή με τη μορφή δικτύων χρονικών περιορισμών. Οι

χρονικοί περιορισμοί αφορούν τους περιορισμούς που τίθενται από τα χαρακτηριστικά των αντικειμένων της εφαρμογής.

Κάθε tuple του σεναρίου θα εξεταστεί για να προσδιοριστούν οι περιορισμοί που θέτει. Οι περιορισμοί προσδιορίζονται ακολουθώντας ένα σύνολο κανόνων. Κάθε περιορισμός μπορεί να αναπαρασταθεί με μία χρονική ανισότητα. Το σύνολο των ανισοτήτων θα αναπαρασταθεί με ένα δίκτυο χρονικών περιορισμών που καλείται Constraint Network Fragment (CNF). Κατά την κατασκευή των διαγραμμάτων για κάθε tuple μπορεί να πραγματοποιηθεί ένας πρωταρχικός έλεγχος ακεραιότητας.

Επομένως τα δύο σημαντικά σημεία στην κατασκευή των δικτύων χρονικών περιορισμών αφορούν την διαμόρφωση των κανόνων για τον προσδιορισμό των χρονικών περιορισμών και τον τρόπο με τον οποίο γίνεται ο πρωταρχικός έλεγχος ακεραιότητας. Στις δύο υποπαραγράφους που ακολουθούν θα αναλύσουμε τα δύο αυτά βασικά στοιχεία.

3.6.1 Μετασχηματισμός των tuples σε δίκτυα χρονικών περιορισμών

Κάθε ένα από τα tuples του IMD σεναρίου μετασχηματίζεται στο στάδιο αυτό σε έναν αριθμό δικτύων χρονικών περιορισμών. Τα δίκτυα CNF αναπαριστούν τους περιορισμούς που υπάρχουν μεταξύ των γεγονότων που συμβαίνουν κατά τη διάρκεια εκτέλεσης της εφαρμογής.

Ένα tuple μπορεί να δημιουργήσει περισσότερα από ένα δίκτυα CNF. Για παράδειγμα ένα tuple που έχει σαν γεγονός εκκίνησης ένα σύνθετο γεγονός που προκύπτει από τη διάζευξη δύο απλών γεγονότων, θα δημιουργήσει δύο τουλάχιστον (ανάλογα με το είδος του γεγονότος διακοπής) CNF για τα διαφορετικά σενάρια εκτέλεσης του tuple.

Ένα CNF μπορεί να αναπαρασταθεί με το σύνολο $NT = \{N, C, A\}$ στο οποίο:

- N είναι το σύνολο των κόμβων του δικτύου. Όπως έχουμε ήδη αναλύσει κατά την παρουσίαση των δικτύων χρονικών περιορισμών, κάθε κόμβος ενός δικτύου αναπαριστά ένα γεγονός της εφαρμογής. Τα γεγονότα μπορεί να αφορούν συμβάντα της εφαρμογής ή τις πράξεις που εκτελούνται κατά την εκτέλεση των tuples.

- A είναι το σύνολο των συνδέσεων που συνδέουν του κόμβους του δικτύου. Υπενθυμίζουμε ότι κάθε σύνδεση αναπαριστά ένα χρονικό περιορισμό μεταξύ των κόμβων-γεγονότων που διασυνδέει.

- Τέλος C είναι το σύνολο των χρονικών περιορισμών μεταξύ των κόμβων του δικτύου. Ο χρονικός περιορισμός είναι στην ουσία μια διπλή χρονική ανισότητα που είναι απαραίτητο να ισχύει μεταξύ των δύο γεγονότων για να είναι το tuple χρονικά συνεπές. Κάθε χρονικός περιορισμός αντιστοιχεί σε μία και μόνο σύνδεση του δικτύου.

Οι κανόνες που αφορούν την κατασκευή του δικτύου ασχολούνται στην ουσία με τον προσδιορισμό των χρονικών ανισοτήτων που πρέπει να ισχύουν. Κατά την κατασκευή του CNF ενός tuple πρέπει να λάβουμε υπόψη μας τους εξής έξι βασικούς κανόνες:

1. **Κόμβοι του δικτύου**. Κάθε ενέργεια (TAC action) που περιγράφεται στο αντίστοιχο πεδίο κάθε tuple ενός σεναρίου σχηματίζει έναν κόμβο του δικτύου CNF. Κόμβους επίσης σχηματίζουν τόσο τα γεγονότα εκκίνησης και διακοπής (start-stop events) όσο και τα γεγονότα συγχρονισμού (start-stop synchronization event). Τέλος για κάθε αντικείμενο *A* που συμμετέχει στην λίστα ενεργειών του tuple πρέπει να προστεθεί ένας κόμβος που αναπαριστά την ολοκλήρωση της παρουσίασης του αντικειμένου (*A*<).

2. **Διάρκεια των συνδέσεων μεταξύ ενεργειών**. Οι κόμβοι που σχηματίζονται από συνεχόμενες ενέργειες που εκτελούνται στα αντικείμενα της εφαρμογής, συνδέονται με τη διάρκεια που αντιστοιχεί στο διάστημα που μεσολαβεί μεταξύ της εκτέλεσης των δύο ενεργειών. Για παράδειγμα αν σε ένα tuple ορίζεται η εξής ακολουθία ενεργειών: ... *A> 10 B!0 C!* ... τότε το δίκτυο CNF θα περιέχει τρεις κόμβους που θα αντιστοιχούν στις τρεις TAC actions. Ο κόμβος *A>* θα συνδεθεί με τον κόμβο B! με διάρκεια 10 ενώ ο κόμβος B! θα συνδεθεί με τον κόμβο C! με διάρκεια 0.

3. **Σύνδεση ενεργειών στα ίδια αντικείμενα**. Αν σε ένα tuple εκτελούνται ενέργειες στο ίδιο αντικείμενο, οι οποίες δεν είναι συνεχόμενες, τότε πρέπει να προστεθούν συνδέσεις μεταξύ των δύο κόμβων που να αναπαριστούν τους περιορισμούς που ορίζονται στον πίνακα 3.2. Για παράδειγμα αν σε ένα tuple υπάρχει η εξής ακολουθία ενεργειών: *...A> B! A!...* και το αντικείμενο Α έχει διάρκεια d_A, τότε οι κόμβοι *A>* και *A!* πρέπει να συνδεθούν με διάρκεια 0 - d_A. Αν πρόκειται για αντικείμενα που δεν έχουν χρονικό περιορισμό (για παράδειγμα μια εικόνα ή ένα κείμενο) τότε η διάρκεια είναι 0 - ∞. Η τελευταία ενέργεια που εκτελείται σε ένα αντικείμενο πρέπει επίσης να συνδεθεί με τον κόμβο ολοκλήρωσης της παρουσίασης του αντικειμένου. Αν η τελευταία ενέργεια είναι η ενέργεια εκκίνησης τότε η σύνδεση θα έχει διάρκεια d_A - d_A και όχι *0* - d_A.

4. **Μετονομασία ενεργειών**. Για την ονομασία των κόμβων χρησιμοποιούμε το όνομα του γεγονότος που αναπαριστούν. Για παράδειγμα όταν ο κόμβος αναπαριστά το γεγονός *_StopBtnClick* τότε αυτό είναι το όνομα που χρησιμοποιούμε. Όταν ο κόμβος αναπαριστά την πράξη *A!* χρησιμοποιούμε αυτή την αναπαράσταση της ενέργειας σαν όνομα του κόμβου. Υπάρχει όμως περίπτωση σε ένα tuple να έχουμε το ίδιο γεγονός περισσότερες από μία φορές. Για παράδειγμα δύο ενέργειες pause στο ίδιο αντικείμενο (*A||*). Οι κανόνες κατασκευής του συνολικού δικτύου χρονικών περιορισμών δεν επιτρέπουν σε δύο κόμβους να έχουν το ίδιο όνομα, εκτός αν αναπαριστούν το ίδιο γεγονός. Επομένως κατά την κατασκευή των επιμέρους CNFs δεν

επιτρέπονται δύο κόμβοι με ίδιο όνομα. Για το λόγο αυτό πρέπει να ακολουθηθεί μια τακτική μετονομασίας. Η απλή λύση που ακολουθήθηκε στην υλοποίηση της μεθοδολογίας είναι η αρίθμηση των κόμβων με κοινό όνομα. Έτσι στο προηγούμενο παράδειγμα ο πρώτος κόμβος θα έχει το όνομα $A||$ ενώ ο δεύτερος θα ονομαστεί $A||2$.

5. **Σύνδεση γεγονότων διακοπής του tuple**. Τα γεγονότα διακοπής απαιτούν ειδικό χειρισμό εξαιτίας του γεγονότος ότι διακόπτουν την εκτέλεση των ενεργειών του tuple. Όταν σε ένα tuple ορίζεται ένα stop event πρέπει στο CNF να προστεθεί μία σύνδεση μεταξύ του κόμβου που αναπαριστά τον κόμβο διακοπής και των κόμβων ολοκλήρωσης όλων των αντικειμένων που συμμετέχουν στην Action list του tuple. Εδώ δημιουργείται ένα ζήτημα που αφορά τα χρονικά περιορισμένα αντικείμενα. Επειδή δεν γνωρίζουμε πότε θα συμβεί το stop event και επειδή δεν πρέπει να συνδέσουμε το γεγονός διακοπής με την ολοκλήρωση ενός αντικειμένου που ήδη έχει σταματήσει, προσθέτουμε έναν ειδικό κόμβο μεταξύ του κόμβου διακοπής του tuple και του κόμβου ολοκλήρωσης του αντικειμένου. Ο κόμβος αυτός συνδέεται με τον κόμβο διακοπής με διάρκεια 0 - 0 (αφού αναπαριστά ένα γεγονός που θα "συμβεί" ταυτόχρονα με το γεγονός διακοπής του tuple) ενώ η σύνδεση με τον κόμβο ολοκλήρωσης του αντικειμένου έχει διάρκεια 0 - ∞.

6. **Διάρκεια του tuple**. Το δίκτυο που αναπαριστά ένα tuple μπορεί να είναι διαφορετικό, ανάλογα με τη χρονική στιγμή στην οποία θα συμβεί το γεγονός διακοπής. Πράγματι υπάρχει η περίπτωση κάποιες από τις ενέργειες που περιλαμβάνονται στην Action list να μην συμβούν, αφού το tuple θα έχει διακόψει την εκτέλεσή του. Στην περίπτωση αυτή το CNF πρέπει να περιλαμβάνει μόνο τις ενέργειες που θα "προλάβουν" να εκτελεστούν. Αυτό σημαίνει ότι ανάλογα με την στιγμή στην οποία θα συμβεί ένα stop event σχηματίζονται πολλά διαφορετικά CNFs. Για το λόγο αυτό σε κάθε CNF προσθέτουμε μία επιπλέον σύνδεση μεταξύ του κόμβου εκκίνησης και του κόμβου διακοπής του tuple που αναπαριστά την διάρκεια εκτέλεσής του.

Ένα σημαντικό ζήτημα που δεν αναλύσαμε στους κανόνες που προηγήθηκαν αφορά την αναπαράσταση των start και stop events. Όταν πρόκειται για ένα απλό event απλά εισάγεται ένας κόμβος στο δίκτυο CNF που συνδέεται με διάρκεια 0 - 0 με τον κόμβο που αναπαριστά την πρώτη ενέργεια της λίστας ενεργειών. Τι γίνεται όμως στην περίπτωση που πρόκειται για ένα σύνθετο γεγονός;

Αναπαράσταση σύνθετων γεγονότων

Η αναπαράσταση των σύνθετων γεγονότων σε ένα CNF απαιτεί ειδική μεταχείριση αφού όπως θα δούμε οδηγούν στη δημιουργία πολλών διαφορετικών *επιμέρους δικτύων* (*network fragments*). Κάθε δίκτυο αναπαριστά ένα διαφορετικό σενάριο εκκίνησης διακοπής του tuple. Στη συνέχεια θα παρουσιάσουμε τους κανόνες για την αναπαράσταση των σύνθετων γεγονότων.

Πρέπει να σημειώσουμε ότι το μοντέλο SIC ορίζει τον τρόπο με τον οποίο αναπαρίστανται το σύνολο των τελεστών και των συναρτήσεων που παρουσιάσαμε στην υποπαράγραφο 2.3.3. Ωστόσο στην υλοποίησή μας χρησιμοποιήσαμε ένα υποσύνολο του μοντέλου. Το υποσύνολο αυτό είναι που θα μας απασχολήσει:

- *Τελεστής διάζευξης:* "|". Στην περίπτωση αυτή πρέπει να σχηματιστούν τόσα CNFs όσα είναι τα γεγονότα που συμμετέχουν στη διάζευξη. Για την αναπαράσταση της διάζευξης "e_1 | e_2 | ...| e_n" θα χρησιμοποιήσουμε n επιμέρους δίκτυα, ένα για κάθε ένα από τα απλά γεγονότα e_1, e_2, ... ,e_n.

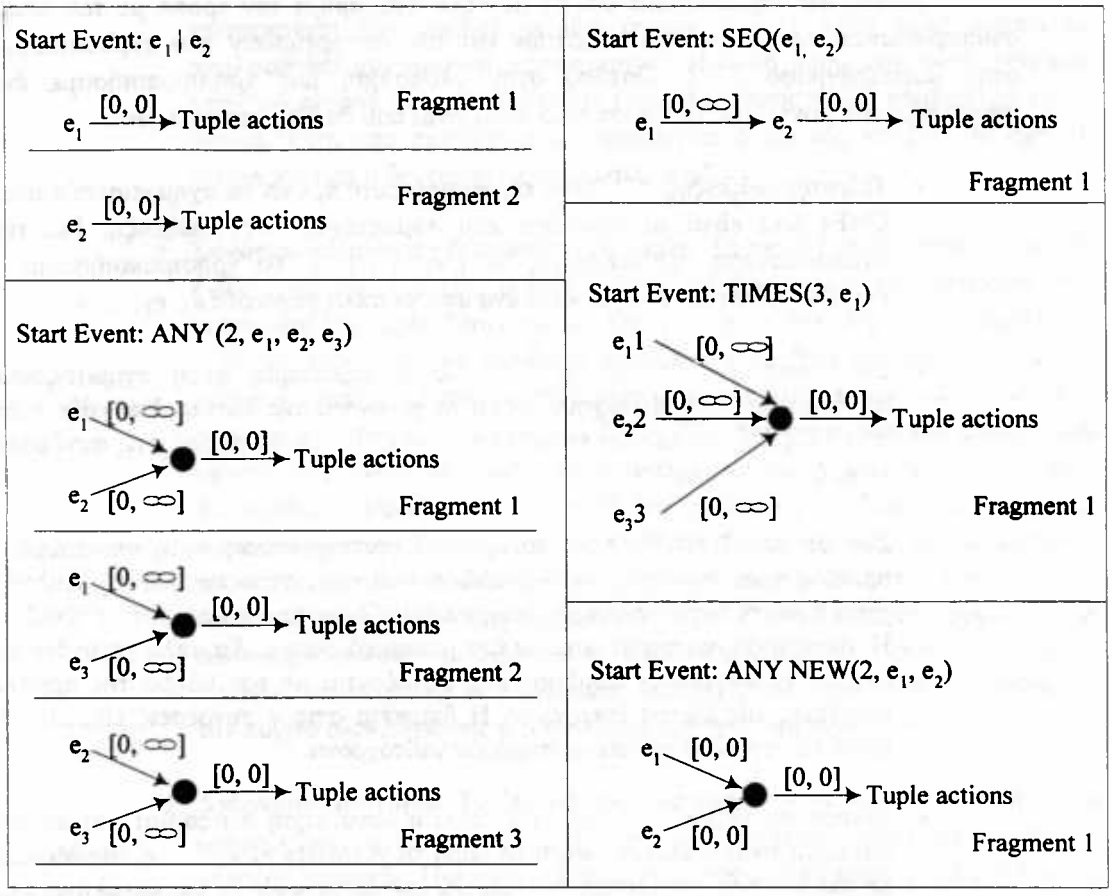
- *Συνάρτηση ANY(k; e_1, e_2,..., e_n).* Στην περίπτωση αυτή σχηματίζουμε συνδυασμούς που περιέχουν k από τα γεγονότα της λίστας. Για κάθε έναν από τους συνδυασμούς σχηματίζεται ένα CNF. Οι διάρκειες στις συνδέσεις είναι 0 - ∞.

- *Συνάρτηση ANYNEW(k; e_1, e_2,..., e_n).* Στην περίπτωση αυτή απαιτούμε ο αριθμός k να συμπίπτει με τον αριθμό των γεγονότων της λίστας. Δηλαδή χειριζόμαστε την συνάρτηση όπως θα χειριζόμασταν τον τελεστή σύζευξης. Η συνάρτηση αναπαρίσταται με ένα μοναδικό δίκτυο. Τα απλά γεγονότα e_1, e_2 , ... ,e_n αποτελούν κόμβους που συνδέονται με τον κόμβο της πρώτης ενέργειας της λίστας ενεργειών. Η διάρκεια στις n συνδέσεις είναι 0 - 0 αφού τα γεγονότα πρέπει να συμβούν ταυτόχρονα.

- *Συνάρτηση SEQ(e_1, e_2,..., e_n).* Και πάλι η συνάρτηση αναπαρίσταται με ένα και μόνο δίκτυο. Ωστόσο αυτή τη φορά τα γεγονότα e_1, e_2, ... ,e_n συνδέονται με μια αλυσίδα συνδέσεων που ξεκινά από το γεγονός e_1 και καταλήγει στο γεγονός e_n. Η διάρκεια στις n-1 συνδέσεις είναι 0 - ∞ αφού δεν μας ενδιαφέρει το διάστημα που μεσολαβεί μεταξύ δύο διαδοχικών γεγονότων της λίστας.

- *Συνάρτηση TIMES(k; e_1).* Σε αυτή την περίπτωση σχηματίζουμε ένα CNF που έχει σαν γεγονότα εκκίνησης τις k εμφανίσεις του γεγονότος e_1. Επειδή όπως προαναφέραμε δεν επιτρέπεται να υπάρχουν σε ένα CNF δύο κόμβοι με το ίδιο όνομα που αναπαριστούν διαφορετικά γεγονότα, στην περίπτωση της συνάρτησης *TIMES* είναι απαραίτητη η μετονομασία των διαφορετικών εμφανίσεων του γεγονότος e_1. Οι συνδέσεις που συνδέουν κάθε έναν από τους k κόμβους με τον κόμβο της πρώτης ενέργειας της λίστας έχουν διάρκεια 0 - ∞, αφού και πάλι δεν ενδιαφερόμαστε για το χρόνο που μεσολαβεί μεταξύ των διαφορετικών εμφανίσεων του γεγονότος.

Οι παραπάνω κανόνες απεικονίζονται στο σχήμα 3.10. Στο σχήμα φαίνονται τα διαφορετικά επιμέρους δίκτυα που σχηματίζονται για κάθε μία από τις πέντε περιπτώσεις που περιγράψαμε παραπάνω. Στην περίπτωση της διάζευξης των γεγονότων e1 και e2 κατασκευάζονται δύο επιμέρους δίκτυα (fragments) που το κάθε ένα εκκινήται με ένα από τα δύο γεγονότα.



Σχήμα 3.10: Αναπαράσταση σύνθετων γεγονότων.

Στην περίπτωση της εφαρμογής της συνάρτησης ANY έχουμε τρεις συνδυασμούς των γεγονότων e_1, e_2 και e_3: $F_1 = \{e_1, e_2\}$, $F_2 = \{e_1, e_3\}$ και $F_3 = \{e_2, e_3\}$. Για κάθε συνδυασμό σχηματίζουμε ένα διαφορετικό δίκτυο. Αυτό που πρέπει να σημειώσουμε είναι η χρήση ενός ειδικού κόμβου που αναπαριστά το γεγονός της εκκίνησης του tuple. Στον κόμβο αυτό κατευθύνονται οι συνδέσεις που ξεκινούν από τα γεγονότα, ενώ από την άλλη πλευρά ο κόμβος συνδέεται με τον κόμβο που αναπαριστά την πρώτη ενέργεια που ορίζει η λίστα ενεργειών. Ο κόμβος αυτός χρησιμοποιείται για να μην συνδέσουμε απευθείας τα γεγονότα με τους κόμβους ενεργειών. Όπως βλέπουμε στο σχήμα ο ειδικός αυτός κόμβος χρησιμοποιείται και στην περίπτωση των συναρτήσεων TIMES και ANYNEW.

Στην περίπτωση της συνάρτησης SEQ σχηματίζεται μια αλυσίδα που συνδέει το γεγονός e_1 με το γεγονός e_2 που με τη σειρά του συνδέεται με τον πρώτο κόμβο του δικτύου. Στην περίπτωση της συνάρτησης TIMES έχουμε τρεις κόμβους που αναπαριστούν ισάριθμες εμφανίσεις του γεγονότος e_1. Τέλος στην περίπτωση της συνάρτησης ANYNEW έχουμε ένα fragment στο οποίο κατασκευάζονται δύο κόμβοι για τα γεγονότα e_1 και e_2.

Κλείνουμε την αναφορά μας στην αναπαράσταση των γεγονότων με την περιγραφή του τρόπου που αναπαρίστανται τα "γεγονότα" θ και $\$$ (αν και δεν πρόκειται για πραγματικά γεγονότα). Το πρώτο αφορά το γεγονός εκκίνησης της εφαρμογής και

επομένως πρέπει να συνδεθεί με διάρκεια 0 με τον κόμβο που αναπαριστά το γεγονός StartApp. Το δεύτερο αφορά το γεγονός εκκίνησης ενός tuple και συνδέεται με τον ειδικό κόμβο που αναπαριστά την εκκίνηση του tuple.

Αυτό που πρέπει επίσης να σημειώσουμε είναι ότι οι παραπάνω κανόνες αφορούν τόσο τα start όσο και τα stop events ενός tuple. Τα δίκτυα που προκύπτουν από το γεγονός εκκίνησης πρέπει να συνδυαστούν με τα δίκτυα που έχουν προκύψει από την εφαρμογή των κανόνων στο γεγονός διακοπής. Αν για παράδειγμα ένα tuple έχει σαν γεγονός διακοπής τη διάζευξη των γεγονότων e_3 και e_4 και σαν γεγονός εκκίνησης την διάζευξη των γεγονότων e_1 και e_2 και τότε θα πρέπει να κατασκευαστούν συνολικά τέσσερα επιμέρους δίκτυα με τους εξής συνδυασμούς γεγονότων εκκίνησης και διακοπής:

- Εκκίνηση: e_1 - Διακοπή e_3
- Εκκίνηση: e_1 - Διακοπή e_4
- Εκκίνηση: e_2 - Διακοπή e_3
- Εκκίνηση: e_2 - Διακοπή e_4

Αλγόριθμος για την κατασκευή των δικτύων χρονικών περιορισμών

Οι κανόνες που παρουσιάσαμε στην συζήτηση που προηγήθηκε μπορούν να συνοψιστούν σε έναν γενικό αλγόριθμο. Ο αλγόριθμος παρουσιάζει φυσικά πολλές ελλείψεις αλλά στόχος του δεν είναι να αποτελέσει τον οδηγό για μια πραγματική υλοποίηση αλλά να παρουσιάσει τα βασικά στοιχεία της διαδικασίας κατασκευής που αναλύθηκε. Ο αλγόριθμος παρουσιάζεται στο σχήμα 3.11 εφαρμόζεται σε κάθε tuple του σεναρίου και χωρίζεται σε δύο φάσεις. Στην πρώτη φάση αναλύουμε τη λίστα των start events και κατασκευάζουμε τους κόμβους που αναπαριστούν την λίστα ενεργειών. Στη συνέχεια για κάθε CNF που προέκυψε από την πρώτη φάση εξετάζουμε το γεγονός διακοπής. Δημιουργούμε έναν κόμβο που να το αναπαριστά και στη συνέχεια το συνδέουμε με τους κόμβους ολοκλήρωσης των αντικειμένων που συμμετέχουν στη λίστα ενεργειών. Φυσικά αν δεν υπάρχει γεγονός ολοκλήρωσης η φάση 2 παραλείπεται.

3.6.2 Αναπαράσταση δικτύων χρονικών περιορισμών

Από τον τρόπο με τον οποίο περιγράψαμε ένα δίκτυο χρονικών περιορισμών είναι φανερό η καταλληλότερη αναπαράσταση είναι με τη χρήση ενός κατευθυνόμενου γράφου. Τα γεγονότα του tuple θα αποτελούν τους κόμβους του γράφου ενώ οι περιορισμοί μεταξύ τους θα αποτελούν τις συνδέσεις των κόμβων. Κάθε σύνδεση θα χαρακτηρίζεται από ένα διάστημα που αφορά τους χρονικούς περιορισμούς που συνδέονται με τα συνδεόμενα γεγονότα.

Ο τρόπος με τον οποίο χτίζουμε τα δίκτυα χρονικών περιορισμών ενός tuple θα γίνει περισσότερο κατανοητός παρουσιάζοντας τα δίκτυα χρονικών περιορισμών του παραδείγματος Example1. Ξεκινάμε από το δίκτυο που αφορά το tuple ExitTuple (σχήμα 3.12).

Φάση 1
Για κάθε event e που ανήκει στη λίστα των Start Events
Δημιούργησε ένα νέο κόμβο N_e;
Για κάθε πράξη που ανήκει στην Action list
Δημιούργησε έναν κόμβο M;
Δημιούργησε μια σύνδεση μεταξύ του κόμβου M-1 και του κόμβου M;
Καταχώρησε τη διάρκεια που μεσολαβεί μεταξύ M και M-1;

Φάση 2
Για κάθε event e που ανήκει στη λίστα των Stop Events των CNFs της Φάσης 1
Δημιούργησε έναν κόμβο N_e;
Για κάθε αντικείμενο O που συμμετέχει στην Action list
Αν δεν υπάρχει κόμβος που να σταματά την εκτέλεση του O
Δημιούργησε έναν κόμβο N_o;
Δημιούργησε μια σύνδεση μεταξύ του κόμβου N_e και του κόμβου N_o;
Καταχώρησε σαν διάρκεια της σύνδεσης το διάστημα 0-0;

Σχήμα 3.11: Γενικός αλγόριθμος για την κατασκευή των επιμέρους δικτύων χρονικών περιορισμών.

Όπως βλέπουμε κάθε κόμβος έχει ένα μοναδικό όνομα ενώ ο περιορισμός μεταξύ των κόμβων αναπαρίσταται με μια σύνδεση, πάνω στην οποία υπάρχει ένα διάστημα.

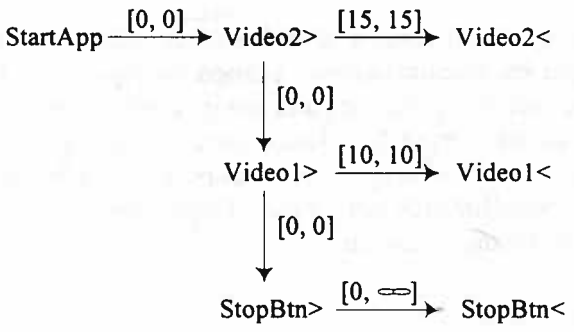
$$\text{\_ExitBtnClick} \xrightarrow{[0,\ 0]} \text{ExitApplication}$$

Σχήμα 3.12: Το δίκτυο χρονικών περιορισμών για το tuple ExitTuple του παραδείγματος Example1.

Το διάστημα περιέχει το πάνω και το κάτω όριο της ανισότητας που συνδέει τα δύο γεγονότα. Στο παράδειγμά μας:

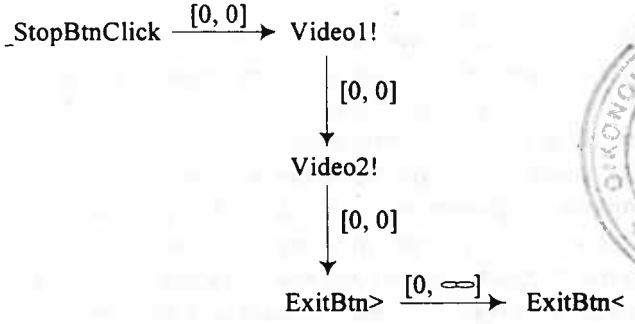
$$0 \leq \text{ExitApplication} - \text{\_ExitBtnClick} \leq 0$$

Ακολουθεί το CNF για το tuple Stage1 που παρουσιάζεται στο σχήμα 3.13. Σε αυτό το CNF βλέπουμε τη σύνδεση μεταξύ των ενεργειών στα αντικείμενα της εφαρμογής και του γεγονότος ολοκλήρωσης. Η σύνδεση αυτή έχει διάρκεια 15 - 15, και 10 - 10 για τα δύο video αφού το Video2 διαρκεί 15 δευτερόλεπτα και το Video1 10. Όσον αφορά το κουμπί StopBtn η σύνδεση έχει διάρκεια 0 - ∞ αφού η εμφάνιση ενός κουμπιού δεν περιορίζεται από κάποιο χρονικό περιορισμό.



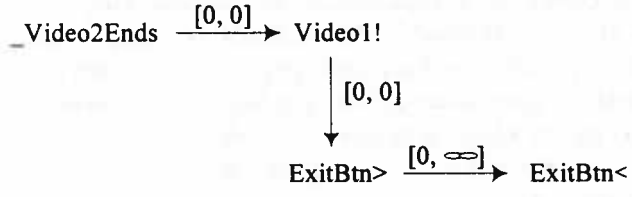
Σχήμα 3.13: Το δίκτυο χρονικών περιορισμών για το tuple Stage1 του παραδείγματος Example1.

Κλείνουμε την παρουσίαση των CNFs του παραδείγματος με την παρουσίαση των δικτύων για τα άλλα δύο tuples του παραδείγματός μας. Το σχήμα 3.14 παρουσιάζει το δίκτυο για το tuple Stage2 ενώ το σχήμα 3.15 το δίκτυο για το tuple Stage3.



Σχήμα 3.14: Το δίκτυο χρονικών περιορισμών για το tuple Stage2 του παραδείγματος Example1.

Τα δύο δίκτυα δεν παρουσιάζουν κάποια σημαντική δυσκολία στο σχηματισμό τους. Μοναδικό σημείο το οποίο πρέπει να προσεχτεί είναι ότι δεν υπάρχουν κόμβοι ολοκλήρωσης για τα δύο video. Αυτό συμβαίνει γιατί η τελευταία πράξη η οποία εκτελείται είναι η πράξη stop.



Σχήμα 3.15: Το δίκτυο χρονικών περιορισμών για το tuple Stage3 του παραδείγματος Example1.

Το παράδειγμα που χρησιμοποιήσαμε είναι αρκετά απλό ώστε να γίνουν κατανοητά τα βασικά στοιχεία της διαδικασίας κατασκευής των επιμέρους δικτύων χρονικών περιορισμών. Ωστόσο από τα CNFs του παραδείγματος απουσιάζουν βασικά

στοιχεία όπως είναι η ύπαρξη γεγονότων διακοπής και η ύπαρξη σύνθετων γεγονότων. Στη συνέχεια θα τροποποιήσουμε ελαφρά το tuple Stage1 του παραδείγματος για να επιδείξουμε κάποιους ακόμα από τους κανόνες που αναλύσαμε. Το νέο tuple παρουσιάζεται στο σχήμα 3.16. Όπως παρατηρούμε έχουν γίνει τρεις αλλαγές. Η πρώτη αφορά το γεγονός διακοπής. Αυτή τη φορά το tuple διακόπτεται από τη διάζευξη των γεγονότων _StopBtnClick και _Video2Ends. Επίσης η διακοπή του tuple προκαλεί το γεγονός συγχρονισμού _Synch1.

```
TUPLE Stage1
    Start_Event = StartApp
    Stop_Event = _StopBtnClick | _Video2Ends
    Action_List = $> 2 Video1> 0 Video2> 0 StopBtn> 2 Video1! 3 Video 1>
    Start_Synch_Event = None
    Stop_Synch_Event = _Synch1
```

Σχήμα 3.16: Το τροποποιημένο tuple Stage1 της εφαρμογής Example1.

Η τρίτη αλλαγή αφορά την λίστα ενεργειών. Δύο δευτερόλεπτα μετά την εκκίνηση του tuple εκκινούνται, όπως και πριν, τα δύο videos καθώς και το κουμπί StopBtn. Η προσθήκη αφορά τις επόμενες δύο πράξεις. Μετά από δύο δευτερόλεπτα το Video1 σταματά για να ξαναρχίσει από την αρχή μετά από τρία δευτερόλεπτα.

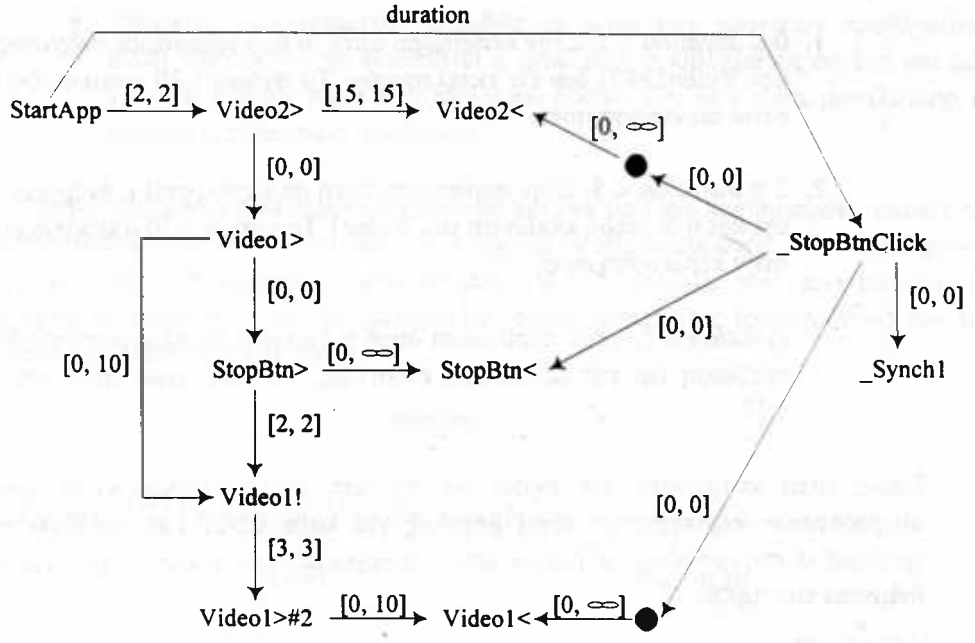
Το γεγονός ότι χρησιμοποιούμε μια διάζευξη γεγονότων σημαίνει ότι θα δημιουργηθούν περισσότερα από ένα δίκτυα για το tuple. Συγκεκριμένα θα δημιουργηθούν δύο δίκτυα που θα ξεκινούν με το γεγονός StartApp και θα διακόπτονται το πρώτο από το γεγονός _StopBtnClick ενώ το δεύτερο από το γεγονός _Video2Ends. Το σχήμα 3.17 παρουσιάζει το πρώτο δίκτυο για το tuple Stage1.

Όπως βλέπουμε το δίκτυο ξεκινά με το γεγονός StartApp και διακόπτεται από το γεγονός _StopBtnClick. Το τελευταίο συνδέεται με το γεγονός συγχρονισμού _Synch1. Επιπρόσθετα πρέπει να παρατηρήσουμε τη σύνδεση των κόμβων Video1> και Video1! με διάρκεια 0-10 καθώς και τη μετονομασία της δεύτερης πράξης start στο αντικείμενο Video1 (Video1>#2).

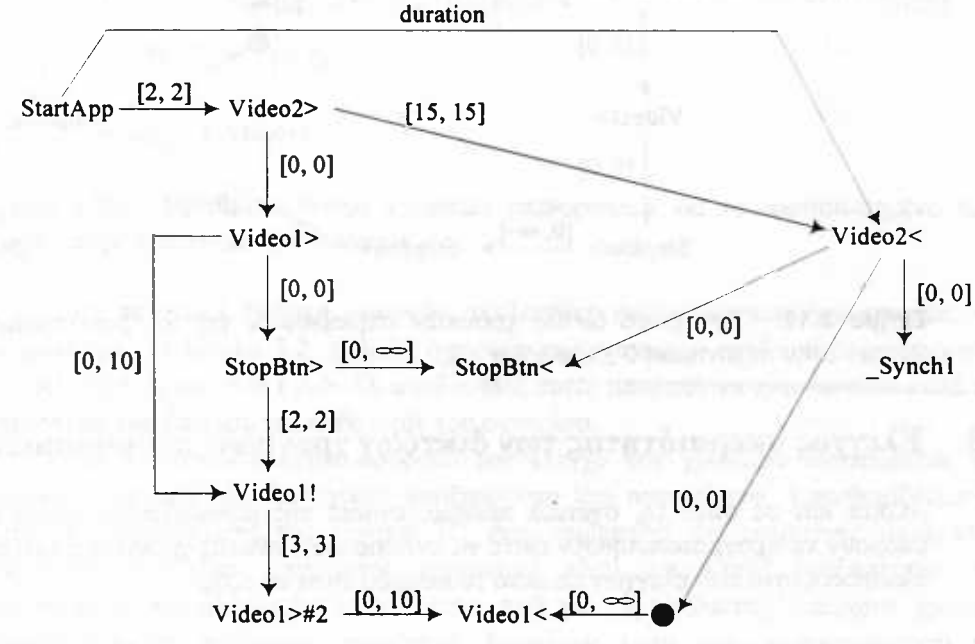
Τέλος το άλλο νέο στοιχείο που παρουσιάζει το δίκτυο αφορά τους ειδικούς κόμβους με τους οποίους συνδέονται τα γεγονότα ολοκλήρωσης των δύο χρονικά εξαρτώμενων αντικειμένων (Video1< και Video2<) με το γεγονός διακοπής του tuple.

Το σχήμα 3.18 παρουσιάζει το δεύτερο δίκτυο. Αυτή τη φορά το γεγονός διακοπής είναι το γεγονός _Video2Ends (κόμβος Video2<). Και πάλι χρειάζεται μετονομασία για ους κόμβους εκκίνησης του αντικειμένου Video1 ενώ αυτή τη φορά χρησιμοποιείται μόνο ένας ειδικός κόμβος, για να συνδέσει τον κόμβο ολοκλήρωσης του Video1 με τον κόμβο διακοπής του tuple.

Το σημείο στο οποίο δεν αναφερθήκαμε αφορά την σύνδεση των γεγονότων εκκίνησης και διακοπής. Όπως έχουμε ήδη αναφέρει, ανάλογα με την χρονική στιγμή στην οποία θα συμβεί το γεγονός διακοπής του tuple, τροποποιείται το CNF που το αναπαριστά. Για παράδειγμα ας πάρουμε το δίκτυο του σχήματος 3.17. Η χρονική στιγμή στην οποία θα πατηθεί το κουμπί από τον χρήστη και επομένως θα προκληθεί το



Σχήμα 3.17: Το πρώτο δίκτυο χρονικών περιορισμών για το τροποποιημένο tuple Stage1.



Σχήμα 3.18: Το δεύτερο δίκτυο χρονικών περιορισμών για το τροποποιημένο tuple Stage1.

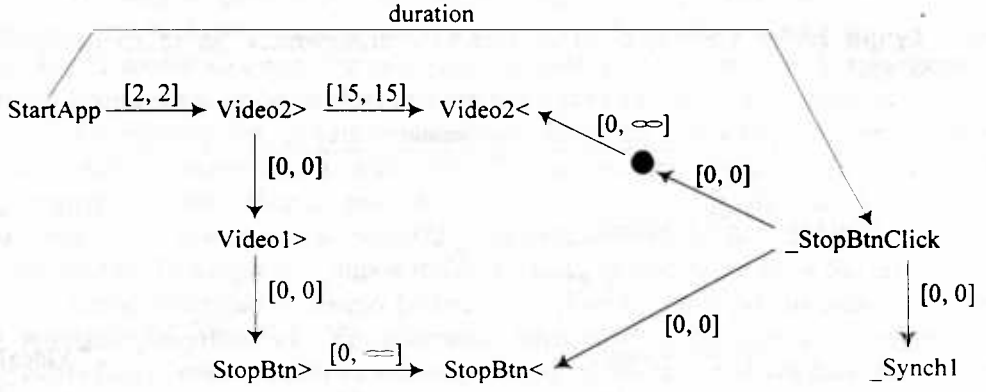
γεγονός _StopBtnClick, τροποποιεί το διάγραμμα αφού οι δύο τελευταίες ενέργειες μπορεί να μην εκτελεστούν. Μπορούμε να ξεχωρίσουμε τρεις περιπτώσεις για την τιμή της διάρκειας, *duration*, του tuple.

1. $0 \leq duration < 2$. Στην περίπτωση αυτή οι δύο τελευταίες ενέργειες (Video1! και Video1>#2) δεν θα εκτελεστούν. Το σχήμα 3.19 απεικονίζει το δίκτυο στην περίπτωση αυτή.

2. $2 \leq duration < 5$. Στην περίπτωση αυτή θα εκτελεστεί η ενέργεια stop αλλά όχι και η δεύτερη εκκίνηση του Video1 Το σχήμα 3.20 απεικονίζει το δίκτυο στην περίπτωση αυτή.

3. $duration \geq 5$. Στην περίπτωση αυτή η διακοπή θα πραγματοποιηθεί μετά το εκτέλεση και της τελευταίας ενέργειας. Το CNF είναι αυτό του σχήματος 3.17.

Όπως είναι κατανοητό για tuples με σύνθετη λίστα ενεργειών, ο αριθμός των διαφορετικών περιπτώσεων είναι μεγάλος για κάθε CNF. Για το λόγο αυτό είναι χρήσιμο ο συγγραφέας να βοηθά στον περιορισμό των περιπτώσεων, ορίζοντας τη διάρκεια του tuple.



Σχήμα 3.19: Το πρώτο δίκτυο χρονικών περιορισμών για το τροποποιημένο tuple Stage1 στην περίπτωση $0 \leq duration < 2$.

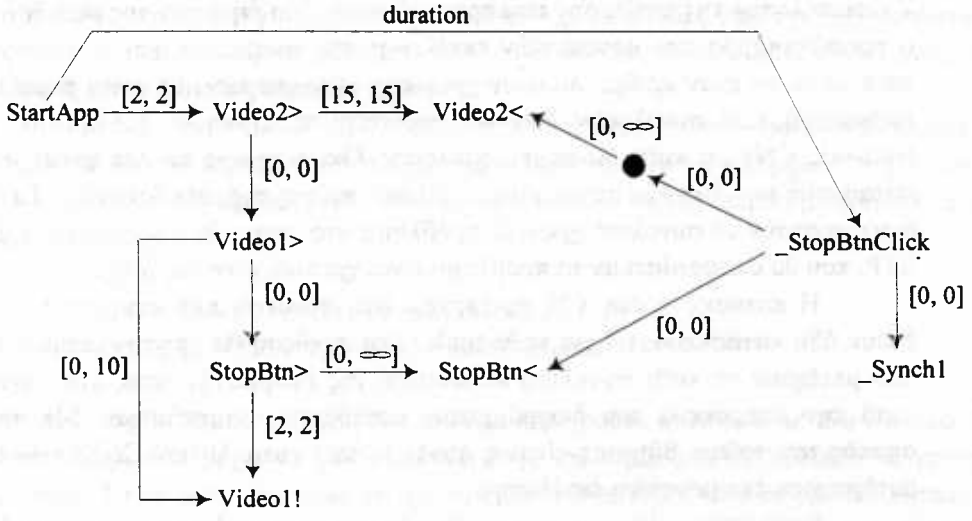
3.6.3 Έλεγχος ακεραιότητας των δικτύων χρονικών περιορισμών

Ακόμα και σε αυτό το, σχετικά πρώιμο, στάδιο της μεθοδολογίας κάποιοι έλεγχοι μπορούν να πραγματοποιηθούν ώστε να εντοπιστούν πιθανές χρονικές ασυνέπειες. Τα πλεονεκτήματα των ελέγχων σε αυτό το επίπεδο είναι τα εξής:

- *Μπορούν να εντοπιστούν χρονικά προβλήματα χωρίς να χρειαστεί να εκτελεστούν τα υπόλοιπα στάδια της μεθοδολογίας.* Αν ένα CNF παρουσιάζει χρονική ασυνέπεια, όλα τα συνολικά δίκτυα που το περιλαμβάνουν θα είναι ασυνεπή. Επομένως δεν είναι απαραίτητο να προχωρήσουμε στα άλλα δύο στάδια της μεθοδολογίας.

- *Μπορούν να εντοπιστούν ακριβώς τα αίτια των χρονικών προβλημάτων.* Είναι πιο εύκολο να εντοπιστεί η αιτία ενός προβλήματος σε ένα και μόνο χρονικό δίκτυο, παρά στο συνολικό δίκτυο CN που είναι μεγαλύτερο και φυσικά περισσότερο πολύπλοκο.

Οι έλεγχοι που πραγματοποιούνται έχουν να κάνουν με τους περιορισμούς μεταξύ των ενεργειών που πραγματοποιούνται στα αντικείμενα της εφαρμογής. Με άλλα λόγια με τους χρονικούς περιορισμούς του πίνακα 3.2. Οι έλεγχοι που πραγματοποιούμε μπορούν να χωριστούν σε δύο κατηγορίες, στους *ποσοτικούς* (*quantitative*) και τους *ποιοτικούς* (*qualitative*) ελέγχους.



Σχήμα 3.20: Το πρώτο δίκτυο χρονικών περιορισμών για το τροποποιημένο tuple Stage1 στην περίπτωση $2 \leq duration < 5$.

Οι ποιοτικοί έλεγχοι αφορούν ακολουθίες ενεργειών που δεν επιτρέπονται σύμφωνα με τον πίνακα 3.2. Δηλαδή αφορούν τον εντοπισμό ακολουθιών της μορφής: A! t A!, A|| t A|| και A\> t A\>. Οι ακολουθίες αυτές μπορούν να εντοπιστούν κατά την κατασκευή του δικτύου για καθε tuple του σεναρίου.

Οι ποσοτικοί έλεγχοι αφορούν τον έλεγχο του χρονικού διαστήματος που μεσολαβεί μεταξύ δύο διαδοχικών πράξεων στο ίδιο αντικείμενο. Υπενθυμίζουμε ότι κατά την παρουσίαση του πίνακα 3.2 είχε σημειωθεί ότι ορισμένες ακολουθίες ενεργειών δεν είναι επιτρεπτές, ορισμένες είναι επιτρεπτές ανεξαρτήτως του διαστήματος που μεσολαβεί μεταξύ τους, ενώ για τις υπόλοιπες υπάρχουν χρονικοί περιορισμοί που πρέπει να τηρούνται. Επομένως κατά την πραγματοποίηση των ποσοτικών ελέγχων εξετάζονται όλες οι ακολουθίες ενεργειών στα αντικείμενα που μετέχουν σε ένα tuple και επιβεβαιώνεται ότι το διάστημα που μεσολαβεί μεταξύ δύο διαδοχικών ενεργειών ικανοποιεί τον αντίστοιχο περιορισμό του πίνακα 3.2.

Αν όλα τα επιμέρους δίκτυα περάσουν επιτυχώς τον έλεγχο ακεραιότητας μπορούμε να προχωρήσουμε στη σύνθεσή τους, ανάλογα με τα μονοπάτια εκτέλεσης που εντοπίστηκαν στο προηγούμενο βήμα. Στην αντίθετη περίπτωση ο συγγραφέας της

εφαρμογής πρέπει να διορθώσει το λάθος που εντοπίστηκε, αφού δεν επιτρέπεται η εκτέλεση των επόμενων βημάτων της μεθοδολογίας.

Στην περίπτωση του "δικού μας" παραδείγματος τα τέσσερα δίκτυα που προκύπτουν δεν παρουσιάζουν κάποιο λάθος. Επομένως μπορούμε να προχωρήσουμε στη σύνθεσή τους, με βάση τα δύο διαφορετικά μονοπάτια εκτέλεσης που εντοπίστηκαν στο βήμα 1.

3.7 Σύνθεση επιμέρους δικτύων χρονικών περιορισμών

Το αποτέλεσμα της εκτέλεσης των προηγούμενων δύο βημάτων της μεθοδολογίας είναι ο προσδιορισμός των μονοπατιών εκτέλεσης της εφαρμογής και ο μετασχηματισμός κάθε tuple σε έναν αριθμό δικτύων χρονικών περιορισμών. Το τρίτο βήμα αφορά την κατασκευή των συνολικών δικτύων χρονικών περιορισμών (composite Constraint Network, CN) για κάθε μονοπάτι εκτέλεσης. Όπως έχουμε πολλές φορές αναφέρει, η κατασκευή των δικτύων αυτών είναι ο τελικός στόχος της μεθοδολογίας. Τα δίκτυα CN αναπαριστούν το συνολικό χρονικό πρόβλημα στο οποίο θα εφαρμοστεί ο αλγόριθμος STP, που θα αποφασίσει αν το πρόβλημα είναι χρονικά συνεπές ή όχι.

Η κατασκευή των CN συνίσταται στη σύνθεση των επιμέρους δικτύων που έχουν ήδη κατασκευαστεί για κάθε tuple. Στη σύνθεση θα χρησιμοποιηθούν τα tuples που μετέχουν σε κάθε συνολική κατάσταση της εφαρμογής όπως έχει προσδιοριστεί κατά την κατασκευή του διαγράμματος μετάβασης καταστάσεων. Με απλά λόγια, σκοπός του τρίτου βήματος είναι η σύνδεση των κατάλληλων CNFs των tuples που μετέχουν σε ένα μονοπάτι εκτέλεσης.

Στην υποπαράγραφο που ακολουθεί θα παρουσιάσουμε τους κανόνες στους οποίους βασίζεται η σύνθεση των επιμέρους δικτύων. Θα χρησιμοποιήσουμε και πάλι τα CNFs του παραδείγματος Example1 για να επιδείξουμε τον τρόπο με τον οποίο εφαρμόζονται στην πράξη οι κανόνες που θα παρουσιάσουμε. Όπως συνέβη και στα δύο προηγούμενα βήματα, μετά την κατασκευή των σύνθετων δικτύων μπορεί να πραγματοποιηθεί ένας έλεγχος της χρονικής ακεραιότητας των αποτελεσμάτων. Ο τρόπος με τον οποίο γίνεται ο έλεγχος θα αναλυθεί επίσης σε ξεχωριστή υποπαράγραφο.

3.7.1 Κατασκευή σύνθετων δικτύων χρονικών περιορισμών

Η σύνθεση των επιμέρους δικτύων χρονικών περιορισμών συνίσταται στην εκτέλεση δύο βασικών λειτουργιών:

1. Στην σύνδεση των κατάλληλων CNFs που έχουν ήδη κατασκευαστεί στο προηγούμενο βήμα, μέσω των γεγονότων εκκίνησης και διακοπής και
2. Στην προσθήκη των κατάλληλων συνδέσεων μεταξύ των κόμβων των CNFs

Τα δύο αυτά βήματα πρέπει να πραγματοποιηθούν για κάθε ένα από τα μονοπάτια εκτέλεσης που έχουν εντοπιστεί.

Σύνδεση επιμέρους δικτύων χρονικών περιορισμών

Η πρώτη βασική λειτουργία του τρίτου βήματος της μεθοδολογίας αφορά τη σύνδεση των κατάλληλων CNFs για τη δημιουργία ενός συνολικού δικτύου χρονικών περιορισμών. Η σύνδεση των CNFs αφορά την επιλογή των *κατάλληλων* CNFs για κάθε μονοπάτι εκτέλεσης και στη συνέχεια τη σύνδεση των γεγονότων εκκίνησης και διακοπής των διαφορετικών δικτύων που χρησιμοποιούνται.

Ένα από τα βασικά σημεία της διαδικασίας σύνθεσης "κρύβεται" πίσω από τη λέξη *κατάλληλων*. Κάθε tuple μετασχηματίζεται σε έναν αριθμό δικτύων χρονικών περιορισμών. Κάθε δίκτυο περιλαμβάνει έναν συνδυασμό γεγονότων εκκίνησης - διακοπής. Κατά την σύνθεση των CNFs πρέπει να επιλέξουμε εκείνα τα δίκτυα που ανταποκρίνονται στα γεγονότα που προκαλούν την μετάβαση από την μία κατάσταση στην άλλη (σύμφωνα με το μονοπάτι εκτέλεσης που ακολουθούμε) και επομένως την εκκίνηση ή την διακοπή κάποιων tuples.

Για να γίνει περισσότερο κατανοητός ο τρόπος με τον οποίο επιλέγονται τα κατάλληλα δίκτυα περιορισμών ας χρησιμοποιήσουμε ένα συγκεκριμένο παράδειγμα. Ας υποθέσουμε ότι έχουμε ένα σενάριο που μεταξύ άλλων περιλαμβάνει τα tuples Stage1 και Stage2:

Stage1 = {$event_1$ | $event_2$, $event_3$ | $event_4$, … ,- ,-}
Stage2 = {SEQ($event_1$, $event_3$), $event_5$ | $event_6$, … ,- ,-}

Ο άτυπος συμβολισμός που χρησιμοποιούμε περιγράφει ένα tuple με ένα σύνολο. Το πρώτο στοιχείο περιέχει το γεγονός εκκίνησης του tuple ενώ το δεύτερο το γεγονός διακοπής. Το επόμενο στοιχείο αφορά την λίστα ενεργειών ενώ τα δύο τελευταία τα δύο γεγονότα συγχρονισμού. Το σύμβολο "-" δηλώνει ότι ένα από τα στοιχεία του συνόλου είναι κενό ενώ το σύμβολο "…" δηλώνει ότι το περιεχόμενο δεν μας ενδιαφέρει. Δηλαδή ο συμβολισμός μπορεί να περιγραφεί ως εξής:

tuple = {start event, stop event, action list, start synch event, stop synch event}

Στο παράδειγμά μας το tuple Stage2 ξεκινά με το σύνθετο γεγονός SEQ($event_1$, $event_3$) και διακόπτεται με τη διάζευξη των γεγονότων $event_5$ και $event_6$.

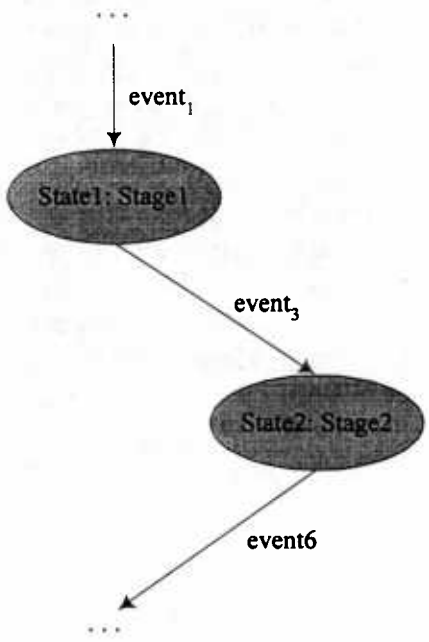
Επομένως για το tuple Stage1 σχηματίζονται τέσσερα CNFs ενώ για το Stage2 δύο. Συγκεκριμένα για το tuple Stage2 τα CNFs είναι τα εξής:

- CNF#1: Γεγονός εκκίνησης $event_1$ - Γεγονός διακοπής $event_3$
- CNF#2: Γεγονός εκκίνησης $event_1$ - Γεγονός διακοπής $event_4$
- CNF#3: Γεγονός εκκίνησης $event_2$ - Γεγονός διακοπής $event_3$
- CNF#4: Γεγονός εκκίνησης $event_2$ - Γεγονός διακοπής $event_4$

ενώ για το tuple Stage2:

- CNF#1: Γεγονός εκκίνησης SEQ($event_1$, $event_3$) - Γεγονός διακοπής $event_5$
- CNF#2: Γεγονός εκκίνησης SEQ($event_1$, $event_3$) - Γεγονός διακοπής $event_6$

Τέλος ας υποθέσουμε ότι εξετάζουμε ένα από τα μονοπάτια εκτέλεσης που περιλαμβάνει μεταξύ άλλων τις μεταβάσεις που απεικονίζονται στο σχήμα 3.21.



Σχήμα 3.21: Ένα τμήμα του διαγράμματος μετάβασης καταστάσεων για ένα υποτιθέμενο παράδειγμα εφαρμογής.

Είναι φανερό ότι πρέπει να συνδεθούν τα δίκτυα που αναπαριστούν τους περιορισμούς των tuples Stage1 και Stage2. Ποια δίκτυα όμως θα επιλεγούν; Όπως βλέπουμε το tuple Stage1 εκκινήται από το γεγονός event1 και διακόπτεται από το γεγονός event3. Επομένως πρέπει να χρησιμοποιηθεί το CNF#1. Το tuple Stage2 εκκινήται από την ακολουθία των γεγονότων event1 και event3 και διακόπτεται από το γεγονός event6. Επομένως το δίκτυο που θα χρησιμοποιηθεί στην σύνθεση είναι το CNF#2.

Επομένως η διαδικασία επιλογής των κατάλληλων δικτύων χρονικών περιορισμών χρησιμοποιεί τις πληροφορίες που παρέχει το διάγραμμα μετάβασης καταστάσεων για κάθε μονοπάτι εκτέλεσης.

Το δεύτερο σημαντικό ζήτημα της διαδικασίας σύνδεσης των επιμέρους δικτύων αφορά την *μετονομασία* (*renaming*) των κόμβων που συμμετέχουν σε ένα CN. Δεν επιτρέπεται να υπάρχουν δύο κόμβοι που να αναφέρονται σε διαφορετικά γεγονότα να έχουν την ίδια ονομασία. Αν πάλι υπάρχουν δύο κόμβοι με την ίδια ονομασία που όντως αναφέρονται στο ίδιο γεγονός τότε θα πρέπει να ενοποιηθούν σε έναν μοναδικό κόμβο. Η διαδικασία της μετονομασίας στηρίζεται σε δύο βασικούς κανόνες για την μετονομασία των κόμβων που αναπαριστούν TAC actions και την μετονομασία των κόμβων που αναπαριστούν γεγονότα εκκίνησης ή διακοπής:

- *Μετονομασία κόμβων TAC.* Ας πάρουμε μια συγκεκριμένη κατάσταση του διαγράμματος μετάβασης καταστάσεων. Αν συναντήσουμε κοινά ονόματα μεταξύ των CNFs που ανήκουν στην ίδια κατάσταση αυτό σημαίνει ότι

αναφέρονται στα ίδια αντικείμενα. Επομένως δεν χρειάζεται μετονομασία. Από την άλλη πλευρά αν συναντήσουμε κοινά ονόματα σε CNFs διαφορετικών καταστάσεων τότε οι ενέργειες δεν αναφέρονται στο ίδιο αντικείμενο. Άρα η μετονομασία των κόμβων είναι απαραίτητη.

- *Μετονομασία γεγονότων εκκίνησης και διακοπής*. Τα γεγονότα εκκίνησης και διακοπής αποτελούν όπως αναφέραμε τους συνδετικούς κρίκους μεταξύ των διαφορετικών CNFs. Οι κόμβοι οι οποίοι αναφέρονται σε μια κοινή εμφάνιση ενός γεγονότος πρέπει να ενοποιηθούν σε έναν μοναδικό κόμβο. Αντίθετα, κόμβοι οι οποίοι αναφέρονται σε διαφορετικές εμφανίσεις ενός γεγονότος πρέπει να μετονομαστούν αφού στην ουσία πρόκειται για δύο διαφορετικά γεγονότα.

Η διαδικασία μετονομασίας είναι αδύνατο να πραγματοποιηθεί με αυτόματο τρόπο για όλες τις περιπτώσεις. Για τη διαδικασία μετονομασίας των γεγονότων εκκίνησης και διακοπής μπορούν να διατυπωθούν ορισμένοι κανόνες του οποίους αναφέραμε παραπάνω. Ωστόσο, όσον αφορά τη διαδικασία μετονομασίας των κόμβων TAC δεν υπάρχουν σαφείς κανόνες που να προσδιορίζουν με αξιοπιστία αν δύο πράξεις εφαρμόζεται στο ίδιο στιγμιότυπο ενός αντικειμένου, ή αν έχει μεσολαβήσει επανεκκίνηση του αντικειμένου (ας μην ξεχνάμε ότι δεν επιτρέπεται να παρουσιάζονται ταυτόχρονα δύο στιγμιότυπα ενός αντικειμένου). Για το λόγο αυτό συχνά απαιτείται η συμβολή του συγγραφέα της εφαρμογής για να καθορίσει σε ποιες περιπτώσεις είναι απαραίτητη η μετονομασία ενός κόμβου.

Ο γενικός αλγόριθμος για τη σύνδεση και μετονομασία των κόμβων των CNFs ενός μονοπατιού απεικονίζεται στο σχήμα 3.22. Και πάλι πρέπει να σημειώσουμε ότι ο αλγόριθμος δεν είναι πλήρης απλά δίνει τις βασικές κατευθύνσεις για την υλοποίηση της διαδικασίας σύνδεσης των CNFs. Για παράδειγμα δεν περιέχει τους κανόνες για την επιλογή των κατάλληλων CNFs κάθε tuple.

Έστω δύο CNF: CNF1 και CNF2 που πρόκειται να συνδεθούν και $event_{start}$ και $event_{stop}$ τα γεγονότα εκκίνησης του δεύτερου και διακοπής του πρώτου αντίστοιχα:

Για κάθε CNF1 και CNF2
  Αν (τα CNF1 και CNF2 δεν ανήκουν στην ίδια συνολική κατάσταση) και
    (τα CNF1 και CNF2 έχουν κοινούς TAC κόμβους) τότε
      μετονόμασε τους κοινούς TAC κόμβους;
  Για κάθε $event_{sta}$, $event_{sto}$
    Αν ($event_{sta}$ = $event_{sto}$) και
      (το $event_{sto}$ δεν ανήκει σε κάποια μεταβάση του διαγράμματος)
        μετονόμασε το $event_{sta}$;

Σχήμα 3.22: Ο γενικός αλγόριθμος για τη σύνδεση των επιμέρους δικτύων χρονικών περιορισμών.

Προσθήκη συνδέσεων

Το αποτέλεσμα της σύνδεσης των επιμέρους δικτύων χρονικών περιορισμών είναι η κατασκευή ενός αριθμού συνολικών δικτύων για κάθε διαφορετικό μονοπάτι. Ωστόσο τα δίκτυα αυτά δεν είναι ακόμα πλήρη. Υπάρχουν κάποιες ακόμα συνδέσεις που πρέπει να γίνουν μεταξύ των κόμβων του δικτύου CN.

Η πρώτη ομάδα συνδέσεων αφορά τη σύνδεση των γεγονότων εκκίνησης με το υπόλοιπο δίκτυο. Αν πρόκειται για γεγονότα που προκαλούνται από ενέργειες που εκτελούνται κατά την εκτέλεση των tuples τότε δεν χρειάζεται να προστεθούν συνδέσεις, αφού υπάρχουν ήδη περιορισμοί που προέκυψαν κατά την κατασκευή των επιμέρους δικτύων. Το ίδιο συμβαίνει και με τα γεγονότα συγχρονισμού. Ωστόσο αν πρόκειται για γεγονότα που προέρχονται από τις ενέργειες του χρήστη, οι κόμβοι που τα αναπαριστούν δεν έχουν συνδεθεί με το υπόλοιπο δίκτυο. Πρέπει επομένως να προσθέσουμε μια σύνδεση μεταξύ των διαφορετικών start events που να υποδεικνύει την ακολουθία εμφάνισής τους αλλά και τους περιορισμούς που πρέπει να τηρούνται. Οι συνδέσεις στα γεγονότα που προέρχονται από ενέργειες του χρήστη θα έχουν άγνωστο άνω όριο αφού δεν περιορίζονται από κάποιο διάστημα.

Η δεύτερη ομάδα συνδέσεων αφορά τη σύνδεση των ενεργειών που εκτελούνται στα ίδια στιγμιότυπα αντικειμένων. Κατά την περιγραφή των κανόνων για την κατασκευή των επιμέρους δικτύων είχαμε αναφέρει ότι το δίκτυο πρέπει να περιλαμβάνει συνδέσεις μεταξύ των πράξεων που αναφέρονται στα ίδια αντικείμενα. Οι διάρκειες των συνδέσεων αυτών προσδιορίζονται από τον πίνακα 3.2. Ακριβώς η ίδια διαδικασία πρέπει να ακολουθηθεί και στην περίπτωση των συνολικών δικτύων.

Το βασικό πρόβλημα για την προσθήκη αυτών των συνδέσεων είναι ο εντοπισμός των ενεργειών που πραγματοποιούνται σε κοινά στιγμιότυπα των αντικειμένων. Είναι λάθος να συνδέσουμε ενέργειες που εκτελούνται σε δύο διαφορετικά στιγμιότυπα ενός αντικειμένου, αφού δεν υπάρχει κάποιος περιορισμός που να τα συνδέει.

Ας πάρουμε το εξής παράδειγμα: Σε ένα tuple Stage1 εκκινήται ένα αντικείμενο A (A>). Στο tuple Stage2 της ίδιας κατάστασης εκτελείται η ενέργεια Stop στο αντικείμενο A (A!). Πρόκειται για το ίδιο στιγμιότυπο του αντικειμένου. Σε μια άλλη κατάσταση το Stage2 επανεκτελείται και επομένως και πάλι έχουμε την ενέργεια Stop, αλλά αυτή τη φορά σε ένα άλλο στιγμιότυπο του αντικειμένου A: A!#2. Το προφανές ερώτημα είναι ποιες συνδέσεις πρέπει να γίνουν στο δίκτυο του παραπάνω παραδείγματος;

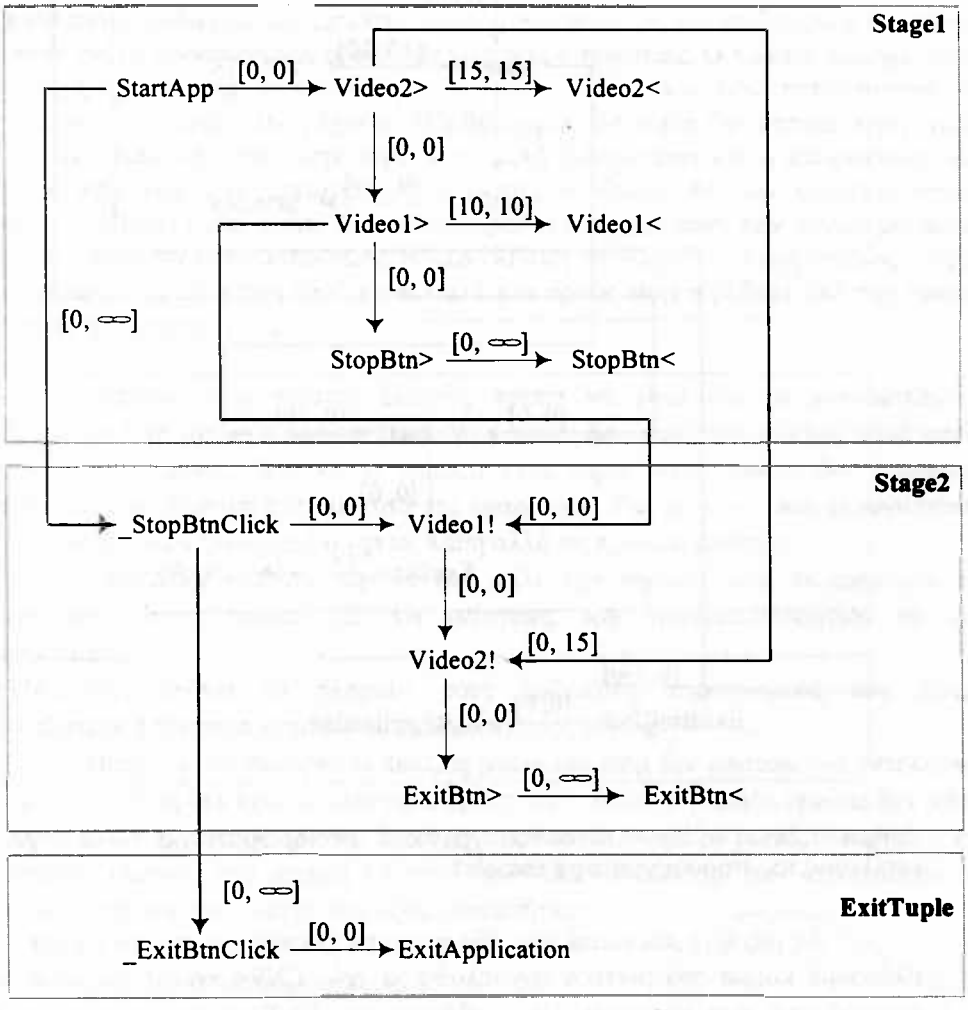
Ελέγχοντας τον πίνακα 3.2 βλέπουμε ότι οι ενέργειες A> και A! πρέπει να συνδεθούν με διάρκεια 0 - d_A, όπου d_A είναι η διάρκεια του αντικειμένου A. Αντίθετα είναι λάθος να συνδέσουμε την ενέργεια A!#2 με την ενέργεια A> αφού πρόκειται για ενέργειες που στην ουσία εκτελούνται σε δύο διαφορετικά αντικείμενα. Πρέπει να εντοπίσουμε το tuple το οποίο περιέχει τον κόμβο εκκίνησης του δεύτερου στιγμιότυπου του αντικειμένου A (A>#2) και να συνδέσουμε αυτόν τον κόμβο με τον κόμβο A!#2.

Είναι φανερό ότι ο εντοπισμός των ενεργειών που πραγματοποιούνται σε κοινά στιγμιότυπα βασίζεται στην μετονομασία που ήδη έχει γίνει. Επομένως για τη σωστή

σύνδεση των κόμβων σημαντική είναι η συνεισφορά του συγγραφέα της εφαρμογής κατά τη διαδικασία μετονομασίας.

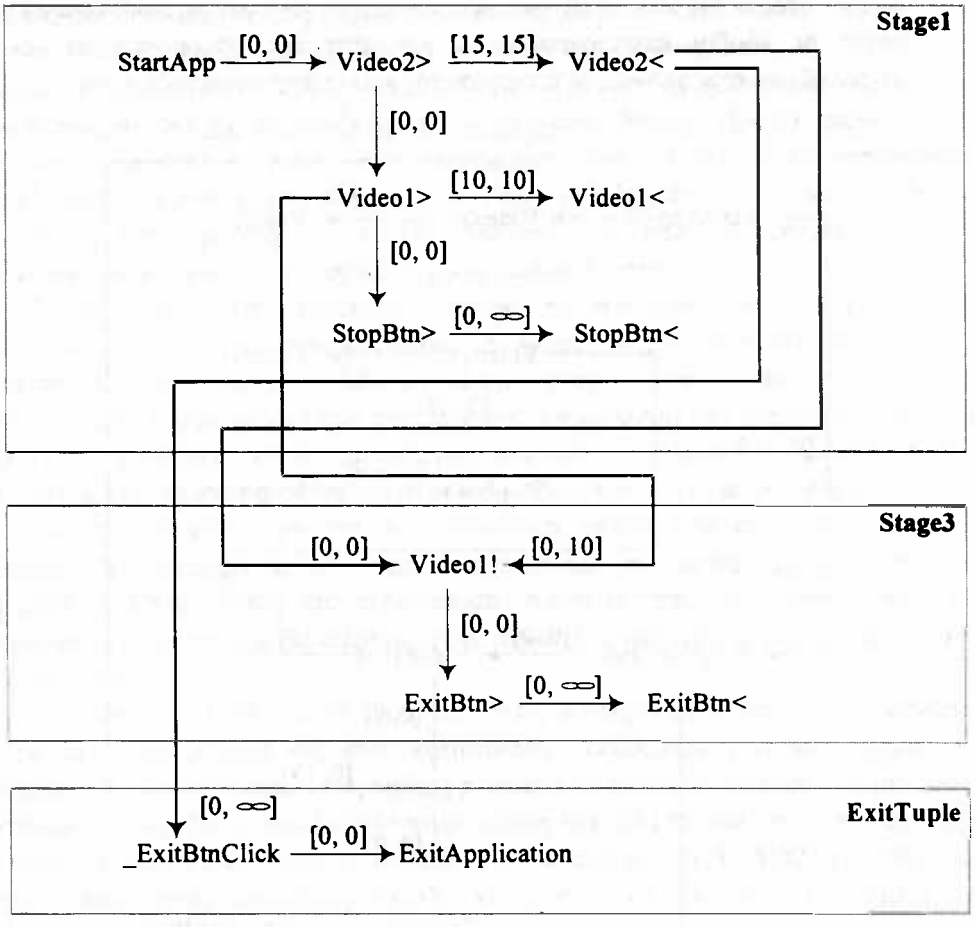
3.7.2 Αναπαράσταση σύνθετων δικτύων χρονικών περιορισμών

Η αναπαράσταση των σύνθετων δικτύων δεν διαφέρει σε τίποτα σε σχέση με την αναπαράσταση που χρησιμοποιούμε στα επιμέρους δίκτυα. Δεν θα μπορούσε άλλωστε να διαφέρει αφού ένα σύνθετο δίκτυο στην ουσία αποτελεί απλώς ένα μεγαλύτερο δίκτυο χρονικών περιορισμών. Φυσικά υπάρχουν διαφοροποιήσεις στον τρόπο με τον οποίο κατασκευάζεται, αλλά και πάλι πρόκειται για έναν κατευθυνόμενο γράφο στον οποίο οι κόμβοι αναπαριστούν τα γεγονότα της εφαρμογής και οι συνδέσεις συμβολίζουν τους χρονικούς περιορισμούς που υπάρχουν μεταξύ τους.



Σχήμα 3.23: Το σύνθετο δίκτυο χρονικών περιορισμών για το πρώτο μονοπάτι εκτέλεσης του παραδείγματος Example1.

Στη συνέχεια θα παρουσιάσουμε τα δίκτυα που κατασκευάζονται στην περίπτωση του παραδείγματος Example1. Το παράδειγμα είχε αναλυθεί σε δύο μονοπάτια εκτέλεσης. Κάθε tuple είχε αναπαρασταθεί με ένα CNF. Επομένως έχουμε ένα CN για κάθε μονοπάτι. Για το μονοπάτι P1 (StartApp - _StopBtnClick - _ExitBtnClick - ExitApplication) θα πρέπει να συνθέσουμε τα CNFs των tuples Stage1, Stage2 και ExitTuple. Για το μονοπάτι P2 (StartApp - _Video2Ends - _ExitBtnClick - ExitApplication) θα πρέπει να συνθέσουμε τα CNFs των tuples Stage1, Stage3 και ExitTuple. Τα δύο CNs αναπαριστώνται στα σχήματα 3.23 και 3.24.



Σχήμα 3.24: Το σύνθετο δίκτυο χρονικών περιορισμών για το δεύτερο μονοπάτι εκτέλεσης του παραδείγματος Example1.

Ας ρίξουμε μια ματιά στα δύο σχήματα ξεκινώντας από το σχήμα 3.23. Όπως βλέπουμε κορμό του δικτύου αποτελούν τα τρία CNFs για τα ισάριθμα tuples που συμμετέχουν στο μονοπάτι. Οι συνδέσεις των CNFs έχουν παραμείνει ως είχαν. Υπάρχουν ωστόσο τέσσερις νέες συνδέσεις, οι οποίες απεικονίζονται με πιο έντονες γραμμές. Οι δύο από τις συνδέσεις συνδέουν τις ενέργειες εκκίνησης με τις ενέργειες διακοπής των αντικειμένων Video1 και Video2. Οι συνδέσεις έχουν διάρκεια 0 - 10 για το Video1 και 0 - 15 για το Video2, όση είναι δηλαδή η διάρκεια των δύο αντικειμένων.

Οι άλλες δύο συνδέσεις αφορούν τη σύνδεση των γεγονότων εκκίνησης των τριών tuples. Το γεγονός StartApp συνδέεται με διάρκεια 0 - ∞ με το γεγονός _StopBtnClick, το οποίο με τη σειρά του συνδέεται με την ίδια διάρκεια με το γεγονός _ExitBtnClick. Οι δύο συνδέσεις έχουν διάρκεια 0 - ∞ αφού πρόκειται για γεγονότα που προκαλούνται από ενέργειες του χρήστη.

Στο δεύτερο CN έχουν προστεθεί τρεις συνδέσεις. Υπάρχει πάλι η σύνδεση των γεγονότων εκκίνησης και διακοπής του Video1. Αυτή τη φορά όμως το γεγονός εκκίνησης του tuple Stage3 είναι το γεγονός Video2<. Για το λόγο αυτό συνδέουμε τον κόμβο Video2< με τον πρώτο κόμβο της λίστας ενεργειών του tuple Stage3 (Video1!) όπως επίσης και με το γεγονός εκκίνησης του tuple ExitTuple (_ExitBtnClick).

3.7.3 Έλεγχος ακεραιότητας σύνθετων δικτύων χρονικών περιορισμών

Σε κάθε σύνθετο δίκτυο που έχει κατασκευαστεί μπορούμε να εκτελέσουμε ένα σύνολο ελέγχων για να προσδιορίσουμε πιθανές χρονικές ασυνέπειες. Ο πρώτος έλεγχος αφορά την ολοκλήρωση της παρουσίασης των tuples αλλά και των αντικειμένων που συμμετέχουν σε αυτά. Όπως έχουμε ήδη αναφέρει, ένα tuple διακόπτεται όταν συμβεί το γεγονός διακοπής του. Στην περίπτωση αυτή διακόπτεται και η παρουσίαση όλων των αντικειμένων που μετέχουν στην λίστα ενεργειών. Αν δεν ορίζεται γεγονός διακοπής το tuple διακόπτεται όταν ολοκληρωθεί η παρουσίαση των αντικειμένων που συμμετέχουν στην λίστα ενεργειών. Η παρουσίαση αντικειμένων όπως εικόνες, κείμενα και κουμπιά διαρκεί μέχρι να διακοπεί από μια πράξη stop η βέβαια από την διακοπή του tuple ή της εφαρμογής.

Επομένως ένας πρώτος έλεγχος πρέπει να γίνει για να εντοπιστούν τα αντικείμενα ή τα tuples η παρουσίαση των οποίων δεν διακόπτεται παρά μόνο από τη λήξη της εφαρμογής. Φυσικά η ύπαρξη ενός tuple χωρίς τέλος δεν δημιουργεί πρόβλημα στην χρονική ακεραιότητα της εφαρμογής. Για το λόγο αυτό οι περιπτώσεις που εντοπίζονται σημειώνονται όχι ως λάθη αλλά ως προειδοποιήσεις.

Ο δεύτερος έλεγχος αφορά και πάλι την τήρηση των περιορισμών που αναφέρονται στον πίνακα 3.2. Οι ενέργειες που πραγματοποιούνται σε κάθε στιγμιότυπο
αντικειμένου, πρέπει να πληρούν τους χρονικούς περιορισμούς του πίνακα. Οποιαδήποτε ασυνέπεια εντοπιστεί σημειώνεται ως λάθος.

Επίσης σε ένα σενάριο οι πράξεις pause και stop δεν μπορούν να εκτελεστούν αν προηγουμένως δεν έχει εκτελεστεί η πράξη start. Επίσης η πράξη resume δεν μπορεί να εκτελεστεί αν προηγουμένως δεν έχει εκτελεστεί η πράξη pause. Επομένως ένας πρόσθετος έλεγχος που μπορεί να γίνει πριν από τον έλεγχο των περιορισμών του πίνακα 3.2 αφορά των έλεγχο των εξής ανισοτήτων:

- $\exists\ x, y \geq 0$ τέτοια ώστε $x \leq A! - A> \leq y$ (3.16)
- $\exists\ x, y \geq 0$ τέτοια ώστε $x \leq A\| - A> \leq y$ (3.17)
- $\exists\ x, y \geq 0$ τέτοια ώστε $x \leq A\!> - A> \leq y$ (3.18)
- $\exists\ x, y \geq 0$ τέτοια ώστε $x \leq A\!> - A\| \leq y$ (3.19)

Πρέπει να σημειώσουμε ότι αυτοί οι έλεγχοι μπορούν να γίνουν μόνο μετά από την σύνθεση των επιμέρους δικτύων, αφού η πραγματοποίησή τους στο προηγούμενο βήμα μπορεί να οδηγήσει σε λάθος συμπεράσματα. Για παράδειγμα αναφέρουμε την περίπτωση στην οποία ένα αντικείμενο εκκινήται σε ένα tuple και σταματά σε ένα άλλο. Ο έλεγχος αυτός πρέπει ωστόσο να βασιστεί σε ενέργειες που πραγματοποιούνται σε κοινά στιγμιότυπα αντικειμένων, για τους λόγους που εξηγήθηκαν παραπάνω.

Μετά την ολοκλήρωση των ελέγχων της ακεραιότητας όλων των δικτύων χρονικών περιορισμών μπορούμε να προχωρήσουμε στον τελικό έλεγχο της ακεραιότητας, εφαρμόζοντας τον αλγόριθμο εύρεσης συντομότερων μονοπατιών σε κάθε ένα από τα διαφορετικά σύνθετα δίκτυα χρονικών περιορισμών που έχουν κατασκευαστεί. Αν σε κάποιο δίκτυο εντοπιστεί λάθος, δεν μπορούμε να προχωρήσουμε στο επόμενο βήμα πριν το λάθος διορθωθεί από τον συγγραφέα.

Στο δικό μας παράδειγμα δεν υπάρχει κάποιο λάθος στα δύο CNs που κατασκευάστηκαν ωστόσο εντοπίστηκαν δύο προειδοποιήσεις για κάθε CN. Οι προειδοποιήσεις αφορούν τα δύο κουμπιά StopBtn και ExitBtn, για τα οποία δεν υπάρχει ενέργεια Stop σε κανένα από τα μονοπάτια.

Ας υποθέσουμε ότι το tuple Stage1 είχε σαν γεγονός διακοπής τη διάζευξη των γεγονότων _StopBtnClick και _ExitBtnClick. Αυτό σημαίνει ότι και στα δύο μονοπάτια εκτέλεσης, τα αντικείμενα του Stage1 (και ειδικά τα Video1 και Video2) δεν θα είναι ενεργά κατά την εκτέλεση των πράξεων των tuples Stage2 και Stage1 που εκκινούνται στην επόμενη κατάσταση. Άρα οι πράξεις stop που εκτελούνται στα videos της εφαρμογής στα δύο tuples θα εντοπιστούν ως χρονικές ασυνέπειες.

3.8 Έλεγχος συνολικής χρονικής ακεραιότητας

Το τελευταίο βήμα της μεθοδολογίας είναι το περισσότερο "καθαρό". Έχουμε καταφέρει να εκφράσουμε το χρονικό πρόβλημα μιας multimedia εφαρμογής με τη χρήση ενός συνόλου δικτύων χρονικών περιορισμών. Σύμφωνα με την ανάλυση της διαδικασίας επίλυσης χρονικών προβλημάτων (§3.3), το μόνο που απομένει είναι ο μετασχηματισμός κάθε δικτύου σε έναν γράφο αποστάσεων στον οποίο θα εφαρμοστεί στη συνέχεια ο αλγόριθμος των Floyd - Warshall για την εύρεση των συντομότερων μονοπατιών του γράφου. Ο αλγόριθμος παράγει σαν αποτέλεσμα το ελάχιστο δίκτυο του προβλήματος. Το ελάχιστο δίκτυο υποδεικνύει αν ένα CN είναι χρονικά συνεπές ή περιέχει ασυνέπειες. Επίσης το ελάχιστο δίκτυο παρέχει πληροφορίες για τη διάρκεια των συνδέσεων που είχαν ακαθόριστο άνω όριο. Όπως θα δούμε από την εφαρμογή του αλγορίθμου στο παράδειγμά μας, το ελάχιστο δίκτυο προσδιορίζει το μέγιστο και ελάχιστο διάστημα που μπορεί να μεσολαβήσει μεταξύ δύο οποιονδήποτε γεγονότων για να είναι ένα μονοπάτι χρονικά συνεπές.

Ο τρόπος με τον οποίο γίνεται ο μετασχηματισμός ενός δικτύου χρονικών περιορισμών στον αντίστοιχο γράφο αποστάσεων έχει αναλυθεί πλήρως στην παράγραφο 3.3 και έχει παρουσιαστεί με τη χρήση ενός απλού παραδείγματος. Δεν κρίνεται απαραίτητο να παρουσιαστούν οι γράφοι αποστάσεων για τα δύο δίκτυα των σχημάτων 3.23 και 3.24.

Εφαρμόζοντας τον αλγόριθμο των Floyd - Warshall στους δύο κατευθυνόμενους γράφους που προκύπτουν από τα ισάριθμα CN λαμβάνουμε σαν αποτέλεσμα δύο ελάχιστα δίκτυα. Ένα τμήμα από το ελάχιστο δίκτυο που προέκυψε από την εφαρμογή του αλγορίθμου στο δεύτερο μονοπάτι παρουσιάζεται στον πίνακα 3.5. Βλέπουμε ότι το πρώτο στοιχείο της νοητής διαγωνίου (που σχηματίζεται από τα γεγονότα StartApp - StartApp) έχει αρνητική τιμή. Αυτό σημαίνει ότι υπάρχει αρνητικός κύκλος στον γράφο αποστάσεων και επομένως το πρόβλημα είναι ασυνεπές.

	StartApp	Video2>	Video1>	StopBtn>	Video2<	Video1<	StopBtn<
StartApp	[5, -5]	[0, 0]	[0, 0]	[0, 0]	[15, 15]	[10, 10]	[0, ∞]
Video2>	[0, 0]	[0, 0]	[0, 0]	[0, 0]	[15, 15]	[10, 10]	[0, ∞]
Video1>	[0, 0]	[0, 0]	[0, 0]	[0, 0]	[15, 15]	[10, 10]	[0, ∞]

Πίνακας 3.5: Τμήμα του ελάχιστου δικτύου για το μονοπάτι P1 του παραδείγματος Example1.

Το ελάχιστο δίκτυο για το δίκτυο χρονικών περιορισμών του πρώτου μονοπατιού εμφανίζεται στους πίνακες 3.6a και 3.6b. Όπως βλέπουμε όλα τα στοιχεία που βρίσκονται στη νοητή διαγώνιο έχουν τιμή [0, 0]. Αυτό είναι λογικό αφού κάθε στοιχείο περιέχει το διάστημα στο οποίο μπορεί να ανήκει η χρονική απόσταση μεταξύ δύο γεγονότων. Τα στοιχεία της διαγωνίου αναφέρονται στο ίδιο χρονικό γεγονός επομένως η απόσταση είναι 0.

	StartApp	Video2>	Video1>	StopBtn>	Video2<	Video1<	StopBtn<
StartApp	[0, 0]	[0, 0]	[0, 0]	[0, 0]	[15, 15]	[10, 10]	[0, ∞]
Video2>	[0, 0]	[0, 0]	[0, 0]	[0, 0]	[15, 15]	[10, 10]	[0, ∞]
Video1>	[0, 0]	[0, 0]	[0, 0]	[0, 0]	[15, 15]	[10, 10]	[0, ∞]
StopBtn>	[0, 0]	[0, 0]	[0, 0]	[0, 0]	[15, 15]	[10, 10]	[0, ∞]
Video2<	[-15, -15]	[-15, -15]	[-15, -15]	[-15 , -15]	[0, 0]	[-5, -5]	[-15, ∞]
Video1<	[-10, -10]	[-10, -10]	[-10, -10]	[-10, -10]	[5, 5]	[0, 0]	[-10, ∞]
StopBtn<	[-∞, 0]	[-∞, 0]	[-∞, 0]	[-∞, 0]	[-∞, 15]	[-∞, 10]	[0, 0]
_StopBtnClick	[-10, 0]	[-10, 0]	[-10, 0]	[-10, 0]	[5, 15]	[0, 10]	[-10, ∞]
Video1!	[-10, 0]	[-10, 0]	[-10, 0]	[-10, 0]	[5, 15]	[0, 10]	[-10, ∞]
Video2!	[-10, 0]	[-10, 0]	[-10, 0]	[-10, 0]	[5, 15]	[0, 10]	[-10, ∞]
ExitBtn>	[-10, 0]	[-10, 0]	[-10, 0]	[-10, 0]	[5, 15]	[0, 10]	[-10, ∞]
ExitBtn<	[-∞, 0]	[-∞, 0]	[-∞, 0]	[-∞, 0]	[-∞, 15]	[-∞, 10]	[-∞,∞]
_ExitBtnClick	[-∞, 0]	[-∞, 0]	[-∞, 0]	[-∞, 0]	[-∞, 15]	[-∞, 10]	[-∞,∞]
ExitApplic.	[-∞, 0]	[-∞, 0]	[-∞, 0]	[-∞, 0]	[-∞, 15]	[-∞, 10]	[-∞,∞]

Πίνακας 3.5a: Τμήμα του ελάχιστου δικτύου για το μονοπάτι P2 του παραδείγματος Example1.

Μελετώντας λίγο πιο προσεχτικά το ελάχιστο δίκτυο βλέπουμε να εμφανίζεται ο βασικός περιορισμός που υπάρχει για να είναι χρονικά συνεπές το μονοπάτι

εκτέλεσης. Είχαμε αναφέρει ότι το γεγονός _StopBtnClick πρέπει να συμβεί πριν από την ολοκλήρωση της παρουσίασης του αντικειμένου Video1, που διαρκεί 10 δευτερόλεπτα και φυσικά μετά την εκκίνηση του πρώτου tuple της εφαρμογής. Ο περιορισμός αυτός εμφανίζεται στο διάστημα μεταξύ των γεγονότων StartApp και _StopBtnClick ([0, 10]). Το ίδιο διάστημα συνδέει το γεγονός StartApp με όλες τις ενέργειες του tuple Stage2 που εκκινήται από το γεγονός _StopBtnClick. (Video1!, Video2!, ExitBtn>).

Πρέπει τέλος να σημειώσουμε την ύπαρξη αρνητικών χρονικών διαστημάτων καθώς και την ύπαρξη διαστημάτων χωρίς άνω ή κάτω όριο. Τα αρνητικά διαστήματα υπενθυμίζουμε ότι οφείλονται στη συμμετρία που χαρακτηρίζει το ελάχιστο δίκτυο. Για παράδειγμα ενώ το διάστημα StartApp-_StopBtnClick έχει διάρκεια όπως είδαμε [0,10], το "αντίστροφο" διάστημα (_StopBtnClick) έχει διάρκεια [-10, 0]. Όσον αφορά την ύπαρξη διαστημάτων χωρίς άνω όριο οφείλεται στο γεγονός ότι για κάποια γεγονότα δεν υπάρχει χρονικός περιορισμός. Για παράδειγμα το γεγονός _ExitBtnClick (που στην ουσία θα διακόψει την εκτέλεση της εφαρμογής) συνδέεται με την εκκίνηση της εφαρμογής (StartApp) με διάστημα [0, ∞]. Πράγματι, δεν υπάρχει χρονικός περιορισμός για την στιγμή που θα πατηθεί το κουμπί.

	_StopBtnCl.	Video1!·	Video2!	ExitBtn>	ExitBtn<	_ExitBtnCl.	ExitApp.
StartApp	[0, 10]	[0, 10]	[0, 10]	[0, 10]	[0, ∞]	[0, ∞]	[0, ∞]
Video2>	[0, 10]	[0, 10]	[0, 10]	[0, 10]	[0, ∞]	[0, ∞]	[0, ∞]
Video1>	[0, 10]	[0, 10]	[0, 10]	[0, 10]	[0, ∞]	[0, ∞]	[0, ∞]
StopBtn>	[0, 10]	[0, 10]	[0, 10]	[0, 10]	[0, ∞]	[0, ∞]	[0, ∞]
Video2<	[-15, -5]	[-15, -5]	[-15, -5]	[-15, -5]	[-15, ∞]	[-15, ∞]	[-15, ∞]
Video1<	[-10, 0]	[-10, 0]	[-10, 0]	[-10, 0]	[-10, ∞]	[-10, ∞]	[-10, ∞]
StopBtn<	[-∞, 10]	[-∞, 10]	[-∞, 10]	[-∞, 10]	[-∞, ∞]	[-∞, ∞]	[-∞, ∞]
_StopBtnCl.	[0, 0]	[0, 0]	[0, 0]	[0, 0]	[0, ∞]	[0, ∞]	[0, ∞]
Video1!	[0, 0]	[0, 0]	[0, 0]	[0, 0]	[0, ∞]	[0, ∞]	[0, ∞]
Video2!	[0, 0]	[0, 0]	[0, 0]	[0, 0]	[0, ∞]	[0, ∞]	[0, ∞]
ExitBtn>	[0, 0]	[0, 0]	[0, 0]	[0, 0]	[0, ∞]	[0, ∞]	[0, ∞]
ExitBtn<	[-∞, 0]	[-∞, 0]	[-∞, 0]	[-∞, 0]	[0, 0]	[-∞,∞]	[-∞,∞]
_ExitBtnCl.	[-∞, 0]	[-∞, 0]	[-∞, 0]	[-∞, 0]	[-∞,∞]	[0, 0]	[0, 0]
ExitApp.	[-∞, 0]	[-∞, 0]	[-∞, 0]	[-∞, 0]	[-∞,∞]	[0, 0]	[0, 0]

Πίνακας 3.5b: Τμήμα του ελάχιστου δικτύου για το μονοπάτι P2 του παραδείγματος Example1.

3.9 Άλλες μεθοδολογίες ελέγχου χρονικής ακεραιότητας

Το μοντέλο ελέγχου της χρονικής ακεραιότητας το οποίο περιγράψαμε στις προηγούμενες παραγράφους δεν αποτελεί την μοναδική προσπάθεια που έχει γίνει στο ζήτημα του ελέγχου της ακεραιότητας multimedia εφαρμογών. Έχουν κατατεθεί πολλές ενδιαφέρουσες προτάσεις τις οποίες πριν κλείσουμε το παρών κεφάλαιο οφείλουμε να εξετάσουμε και να συγκρίνουμε με την μεθοδολογία Scenario Integrity Checking.

Μια από τις πιο ενδιαφέρουσες προτάσεις είναι αυτή των Courtiat και De Oliveira [CO96]. Η περιγραφή της εφαρμογής μετασχηματίζεται αυτόματα σε RT-LOTOS μορφή, η οποία επιτρέπει τον έλεγχο της χρονικής ακεραιότητας εφαρμογής και τον εντοπισμό χρονικών ασυνεπειών. Οι εφαρμογές περιγράφονται με τη χρήση ενός ιεραρχικού μοντέλου το οποίο εκτός των άλλων επιτρέπει και τη χρήση αόριστων χρονικών διαστημάτων. Το μοντέλο υποστηρίζει αλληλεπίδραση με το χρήστη της εφαρμογής και παρέχει δυνατότητες ακριβούς περιγραφής των αντικειμένων που συμμετέχουν στην εφαρμογή. Τέλος το μοντέλο ορίζει μια μέθοδο για τον έλεγχο της ακεραιότητας των σεναρίων.

Οι βασικές πράξεις που μπορούν να εφαρμοστούν στα αντικείμενα του μοντέλου είναι η εκκίνηση και η διακοπή της παρουσίασής τους. Οι δύο αυτές ενέργειες αποτελούν τα βασικά γεγονότα που αναγνωρίζει η εφαρμογή. Ο συγχρονισμός των γεγονότων επιτυγχάνεται χρησιμοποιώντας τα ονόματα των γεγονότων ή μπορεί να καθοριστεί ακριβώς με τη χρήση μιας scripting γλώσσας. Τα αντικείμενα που υποστηρίζονται ορίζονται σε μια ειδική βιβλιοθήκη παρουσίασης.

Μια δεύτερη βιβλιοθήκη ορίζει τους περιορισμούς που ενυπάρχουν μεταξύ των γεγονότων του μοντέλου. Οι περιορισμοί αυτοί αφορούν την ταυτόχρονη εμφάνιση γεγονότων, τον ορισμό του διαστήματος που μεσολαβεί μεταξύ της εμφάνισης δύο γεγονότων και άλλες συνθήκες. Επίσης υποστηρίζονται περιορισμοί με χρήση χρονικών σχέσεων (before, while, overlaps,...) αλλά και περιορισμοί που βασίζονται στον τερματισμό της παρουσίασης ενός αντικειμένου wait latest, wait earliest, wait master, ...).

Ο έλεγχος της ακεραιότητας της εφαρμογής γίνεται μετασχηματίζοντας την RT-LOTOS μορφή του σεναρίου σε ένα *αυτόματο* (*automaton model*). Στην ουσία πρόκειται για έναν γράφο στον οποίο εκφράζονται οι χρονικοί περιορισμοί. Ο έλεγχος της ακεραιότητας συνίσταται στην πραγματοποίηση ενός *ελέγχου προσιτότητας* (*reachibility check*) στο αυτόματο που έχει κατασκευαστεί. Οι δύο ερευνητές χρησιμοποιούν επίσης την έννοια της ποσοτικής και ποιοτικής ακεραιότητας.

Συνολικά πρόκειται για μια σημαντική προσπάθεια. Συγκρίνοντάς την με την μεθοδολογία Scenario Integrity Checking σημειώνουμε τα εξής στοιχεία:

- Ο συγχρονισμός των πράξεων γίνεται με τη χρήση μιας βιβλιοθήκης συναρτήσεων, για τις οποίες δεν υπάρχει καμία ένδειξη ότι μπορούν να καλύψουν όλες τις βασικές χρονικές σχέσεις του Allen [All83]. Το μοντέλο IMD μοντέλο IMD που υιοθετεί η μεθοδολογία SIC αποδεδειγμένα καλύπτει τις 13 σχέσεις του Allen [VTS98].

- Το μοντέλο περιγραφής των multimedia σεναρίων υποστηρίζει μόνο τις δύο βασικές ενέργειες εκκίνησης και διακοπής της παρουσίασης των αντικειμένων. Οι ενέργειες παύσης (Pause) και επανεκκίνησης (Resume) δεν υποστηρίζονται.

- Δεν υποστηρίζεται σύνθεση γεγονότων. Όλα τα γεγονότα που χρησιμοποιεί η εφαρμογή αντιστοιχούν σε απλά συμβάντα.

- Το μοντέλο δεν εξάγει συμπεράσματα για τα διαστήματα που δεν έχουν προσδιοριστεί. Αντίθετα η μεθοδολογία SIC, παράγει το ελάχιστο δίκτυο για

κάθε μονοπάτι εκτέλεσης. Με τη χρήση του ελάχιστου δικτύου μπορούν να προσδιοριστούν οι άγνωστες διάρκειες στις συνδέσεις των γεγονότων.

Μια άλλη πρόταση κατατέθηκε από τους Buchanan και Zellweger [BZ95]. Κορμός της πρότασής τους αποτελεί ένας αλγόριθμος ικανοποίησης χρονικής ακεραιότητας των σεναρίων που χρησιμοποιούν. Ο συγγραφέας μιας εφαρμογής ορίζει τους περιορισμούς που πρέπει να πληρούν τα αντικείμενα της παρουσίασης. Ο αλγόριθμος παράγει συνεπή χρονικά σενάρια. Το μοντέλο που χρησιμοποιείται καλύπτει preorchistrated σενάρια αλλά και σενάρια που περιλαμβάνουν user interaction. Το βασικό μειονέκτημα της πρότασης των Buchanan και Zellweger αφορά την ασαφή διατύπωση του μοντέλου για τη χρονική σύνθεση γεγονότων. Η αλληλεπίδραση με το χρήστη περιορίζεται σε απλές ενέργειες και δεν υποστηρίζεται σύνθεση γεγονότων.

Μια ακόμα μεθοδολογία για τον έλεγχο της ακεραιότητας μιας χρονικής σύνθεσης αντικειμένων multimedia παρουσιάστηκε στο [Lay95]. Η χρονική σύνθεση ορίζεται με τη χρήση ενός κατευθυνόμενου, ακυκλικού γράφου στον οποίο οι κόμβοι αναπαριστούν τα αντικείμενα και οι συνδέσεις μεταξύ τους τις χρονικές σχέσεις που τα συνδέουν. Το μοντέλο χρησιμοποιεί τις έννοιες της ποσοτικής και ποιοτικής χρονικής ακεραιότητας. Η ποιοτική ακεραιότητα σχετίζεται με την ασυνέπεια των χρονικών σχέσεων μεταξύ των αντικειμένων ενώ η ποσοτική ακεραιότητα εξετάζει τη διάρκεια των χρονικών συνθέσεων. Οι βασικές διαφορές με την προσέγγιση που ακολουθούμε είναι οι εξής:

- Το μοντέλο δεν χειρίζεται επαρκώς user-interaction
- Το μοντέλο δεν παρέχει μηχανισμό για την σύνθεση γεγονότων
- Το μοντέλο χρονικής σύνθεσης που υιοθετείται καλύπτει μόνο τις χρονικές σχέσεις μεταξύ των αντικειμένων της εφαρμογής. Αντίθετα το μοντέλο IMD που υιοθετούμε χρησιμοποιεί την έννοια των γεγονότων που δημιουργούνται από τις ενέργειες που εκτελούνται στα αντικείμνα, παρέχοντας με τον τρόπο αυτό μεγάλες δυνατότητες σύνθεσης.

Η τελευταία μεθοδολογία που θα παρουσιάσουμε [PR94] στηρίζεται στο μοντέλο OCPN (Object Composition Petri Nets) που παρουσιάσαμε σύντομα στην παράγραφο 2.2. Το μοντέλο που προτείνεται από τους Prabhakaran και Raghavan υποστηρίζει αλληλεπίδραση με το χρήστη κατά τη διάρκεια της παρουσίασης. Οι συγγραφείς προτείνουν μια δομή Χρονικά Δυναμικών Petri Nets (Dynamic Timed Peti Nets) για να μοντελοποιήσουν την μεταβολή των χρονικών χαρακτηριστικών του δικτύου. Υποστηρίζουν ότι η δομή που υιοθετούν παρουσιάζει όλες τις ιδιότητες των Petri Nets και μπορεί να χρησιμοποιηθεί για να μοντελοποιήσει multimedia εφαρμογές με αλληλεπίδραση με το χρήστη. Τα βασικά χαρακτηριστικά της προσέγγισης των Prabhakaran και Raghavan μπορούν να συνοψιστούν στα ακόλουθα:

- Στηρίζεται σε ένα ισχυρότατο θεωρητικό υπόβαθρο που παρέχεται από την θεωρία των Petri Nets. Τα Petri Nets έχουν χρησιμοποιηθεί στο παρελθόν για να μοντελοποιήσουν τον χρονικό συγχρονισμό multimedia αντικειμένων.

- Ο όρος της αλληλεπίδρασης του χρήστη με την εφαρμογή αναφέρεται στην δυνατότητα που παρέχεται στο χρήστη να τροποποιήσει βασικά χαρακτηριστικά των αντικειμένων (όπως για παράδειγμα η διάρκειά τους), κατά τη διάρκεια της παρουσίασης. Επιπρόσθετα το μοντέλο υποστηρίζει τις βασικές TAC Actions που χρησιμοποιούνται και από το μοντέλο IMD.

- Η χρονική σειρά της παρουσίασης των αντικειμένω είναι προκαθορισμένη. Δεν παρέχεται η δυνατότητα αλληλεπίδρασης με τα αντικείμενα που μετέχουν στην εφαρμογή.

Συνολικά πρόκειται για μια ενδιαφέρουσα πρόταση που χρήζει ωστόσο αρκετών βελτιώσεων, ειδικά στον τομέα του ελέγχου της χρονικής ακεραιότητας.

ΚΕΦΑΛΑΙΟ 4

Υλοποίηση της Μεθοδολογίας Scenario Integrity Checking

Σκοπός των δύο προηγούμενων κεφαλαίων ήταν να αναλύσουν το θεωρητικό υπόβαθρο της διαδικασίας ελέγχου της χρονικής ακεραιότητας μιας multimedia εφαρμογής. Αρχικά παρουσιάσαμε το μοντέλο IMD που χρησιμοποιείται για να περιγράψει πλήρως μια multimedia εφαρμογή, που περιλαμβάνει αλληλεπίδραση με το χρήστη. Στη συνέχεια αφιερώσαμε ένα κεφάλαιο στην ανάλυση της μεθοδολογίας Scenario Integrity Checking, στην οποία θα βασιστούμε. Σκοπός του τέταρτου κεφαλαίου είναι να παρουσιάσει τον τρόπο με τον οποίο έχουν χρησιμοποιηθεί τα δύο παραπάνω μοντέλα για να υλοποιηθεί το εργαλείο ελέγχου της χρονικής ακεραιότητας μιας IMD multimedia εφαρμογής. Το όνομα της εφαρμογής είναι μάλλον αναμενόμενο: "*IMD Integrity Checking*".

Όπως είναι λογικό η ανάλυση της υλοποίησης θα είναι αρκετά περιγραφική προσπαθώντας να παρουσιάσει τα σημαντικότερα σημεία της υλοποίησης, την γενική αρχιτεκτονική και τους βασικούς αλγόριθμους που χρησιμοποιήθηκαν. Δεν πρόκειται για μια εξαντλητική ανάλυση, που θα απαιτούσε υπερβολικό χώρο και κατά τη γνώμη μας θα ήταν μάλλον κουραστική. Άλλωστε ο αναγνώστης ο οποίος ενδιαφέρεται για περισσότερες λεπτομέρειες, για συγκεκριμένα τμήματα της υλοποίησης, μπορεί να τις εντοπίσει στον κώδικα της εφαρμογής. Επίσης δεν πρόκειται για εγχειρίδιο χρήσης της εφαρμογής. Η εφαρμογή είναι αρκετά απλή και για να τη χρησιμοποιήσει κανείς χρειάζεται απλά να ξέρει τα βασικά στοιχεία της μεθοδολογίας.

Ο τρόπος με τον οποίο χωρίστηκε το παρόν κεφάλαιο σε παραγράφους συμβαδίζει με την έως τώρα δομή της εργασίας. Στη πρώτη παράγραφο θα παρουσιαστεί η γενική αρχιτεκτονική της υλοποίησης. Στην παράγραφο 4.2 θα ασχοληθούμε με την υλοποίηση του μοντέλου IMD. Οι τέσσερις παράγραφοι που ακολουθούν θα περιγράψουν τον τρόπο με τον οποίο υλοποιήθηκαν τα ισάριθμα

βήματα της μεθοδολογίας Scenario Integrity Checking. Σε κάθε παράγραφο η ανάλυση θα συνοδεύεται από την παρουσίαση των βασικών κλάσεων του κάθε τμήματος της υλοποίησης.

4.1 Γενική αρχιτεκτονική της υλοποίησης

Η υλοποίηση και η σχεδίαση του μοντέλου στο οποίο στηρίζεται έχουν στηριχθεί στον αντικειμενοστραφή προγραμματισμό (object-oriented programming). Το λειτουργικό σύστημα στο οποίο αποφασίστηκε να γίνει η υλοποίηση είναι τα Windows 95. Η επιλογή αυτή έγινε γιατί υπάρχει ήδη ένας αριθμός εφαρμογών για την συγγραφή (authoring) και την εκτέλεση (rendering) IMD σεναρίων που έχουν αναπτυχθεί σε Windows 95. Η εφαρμογή στηρίζεται στην πλατφόρμα Win32s της Microsoft και μπορεί να εκτελεστεί χωρίς πρόβλημα και σε περιβάλλον Windows NT. Όσον αφορά την επιλογή της γλώσσας οι επιλογές ήταν μάλλον περιορισμένες. Είχαμε να επιλέξουμε μεταξύ της γλώσσας Java και της C++ αφού είναι οι δύο αμιγώς αντικειμενοστραφείς γλώσσες προγραμματισμού σε Windows 95 με τις περισσότερες δυνατότητες. Τελικά επιλέχθηκε η C++ εξαιτίας του αποτελεσματικότερου κώδικα που παράγει αλλά και εξαιτίας μιας παλαιότερης υλοποίησης ενός τμήματος της μεθοδολογίας [BCTPL97] που είχε γίνει σε C. Η ανάπτυξη έγινε εξ' ολοκλήρου σε Visual C++ στο περιβάλλον Microsoft Developer Studio χρησιμοποιώντας την έκδοση 4.2 της βιβλιοθήκης MFC (Microsoft Foundation Classes).

Η παρουσίαση της υλοποίησης θα χωριστεί σε 5 τμήματα. Το πρώτο τμήμα αφορά την "μετάφραση" του μοντέλου IMD. Μία από τις προϋποθέσεις με τις οποίες χτίστηκε η εφαρμογή ήταν να μπορεί να συνεργαστεί με την υπάρχουσα εφαρμογή συγγραφής IMD σεναρίων [VTSH98, VTMHS98]. Η εφαρμογή Scenario Editor παρέχει ένα φιλικό user-interface που δίνει τη δυνατότητα στο χρήστη να αναπτύξει multimedia εφαρμογές. Μία από τις δυνατότητες της εφαρμογής αφορά την αποθήκευση σε ένα απλό αρχείο κειμένου των προδιαγραφών του σεναρίου. Στην ουσία πρόκειται για ένα αρχείο ASCII που περιέχει τα χαρακτηριστικά των actors, των events και των tuples του σεναρίου στη μορφή την οποία χρησιμοποιήσαμε στα παραδείγματα που παρουσιάσαμε ως τώρα. Αυτά τα αρχεία, ή αρχεία με την ίδια μορφή που μπορεί να γράψει ο ίδιος ο χρήστης, αποτελούν τα σενάρια που χρησιμοποιεί η εφαρμογή μας. Επομένως το πρώτο τμήμα της εφαρμογής αφορά την μετάφραση των αρχείων που περιέχουν την περιγραφή του σεναρίου και την αποθήκευσή τους στις κλάσεις της εφαρμογής. Στην ουσία οι κλάσεις που χρησιμοποιούνται για την αποθήκευση του σεναρίου είναι οι εξής:

- *CActor:* Πρόκειται για την κλάση στην οποία αποθηκεύονται τα χαρακτηριστικά κάθε αντικειμένου της εφαρμογής. Φυσικά για τα διαφορετικά είδη αντικειμένων υπάρχουν υποκλάσεις (*subclasses*) που περιέχουν τα ειδικά τους χαρακτηριστικά.
- *CScenarioEvent*: Στην κλάση αυτή αποθηκεύονται τα γεγονότα της multimedia εφαρμογής.
- *CSynchronizationEvent*: Πρόκειται για την κλάση στην οποία αποθηκεύονται τα γεγονότα συγχρονισμού του σεναρίου.
- *CEventFunction*. Αναπαριστά μια συνάρτηση γεγονότων.

- *CScenarioTuple*. Η κλάση η οποία αναπαριστά τα tuples του σεναρίου.
- *CScenario*: Η κεντρική κλάση του πρώτου αυτού τμήματος αλλά και ολόκληρης της εφαρμογής. Σε αυτή συγκεντρώνονται οι τέσσερις παραπάνω κλάσεις καθώς και όλες οι κλάσεις που θα αναφερθούν παρακάτω.

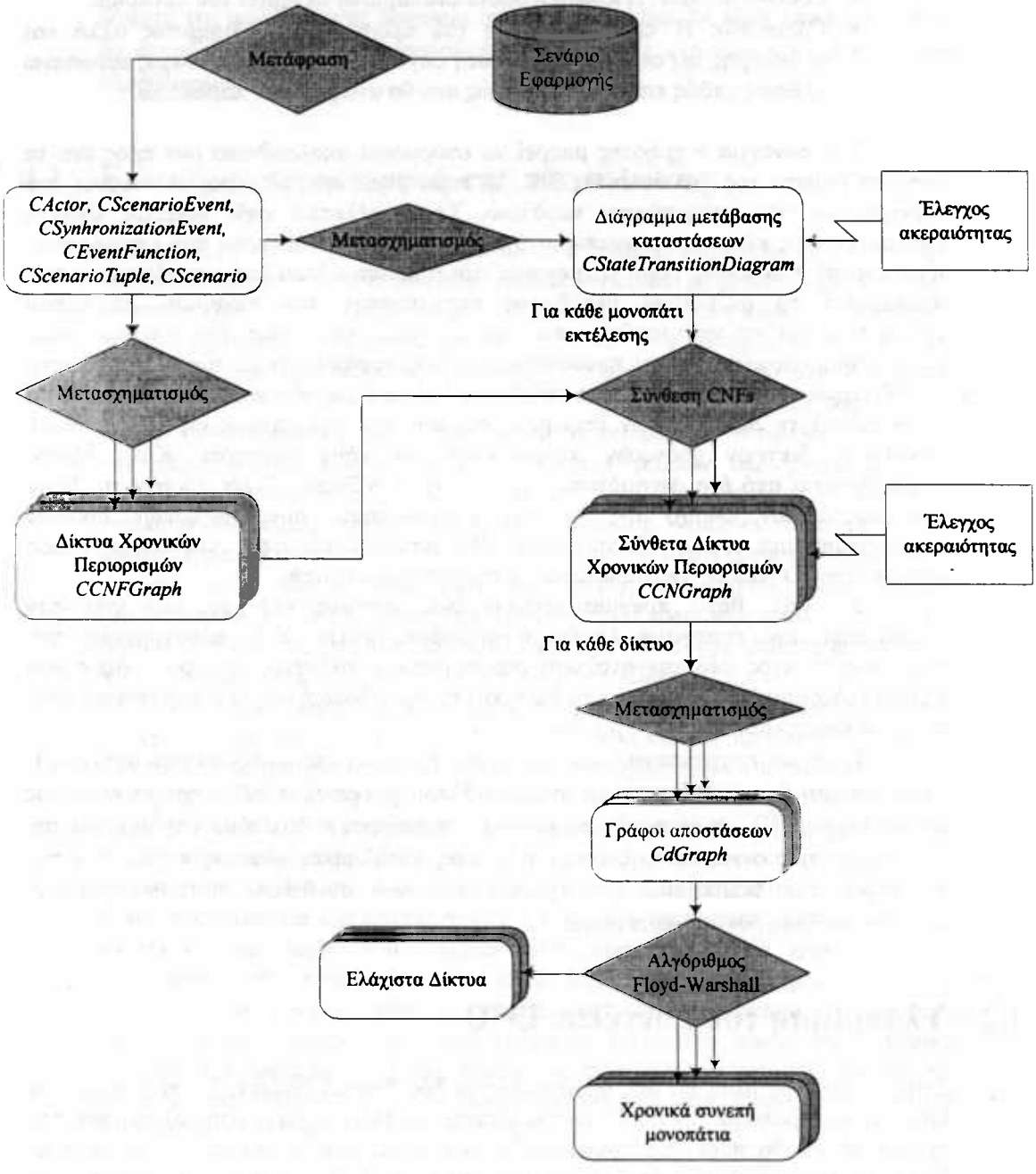
Στη συνέχεια ο χρήστης μπορεί να εφαρμόσει ακολουθιακά ένα προς ένα τα τέσσερα βήματα της μεθοδολογίας SIC. Σε κάθε βήμα εφαρμόζονται οι κανόνες που περιγράψαμε στο προηγούμενο κεφάλαιο. Το αποτέλεσμα κάθε βήματος είναι η κατασκευή μιας κλάσης που αναπαριστά την μορφή αναπαράστασης που χρησιμοποιεί η εφαρμογή. Στο πρώτο βήμα κατασκευάζεται η κλάση *CStateTransitionDiagram* που αναπαριστά το διάγραμμα μετάβασης καταστάσεων του σεναρίου. Η κλάση χρησιμοποιείται για να προσδιοριστούν τα μονοπάτια εκτέλεσης. Στη συνέχεια κάθε tuple χρησιμοποιείται για να κατασκευαστεί ένας αριθμός στιγμιοτύπων της κλάσης *CCNFGraph* που αναπαριστούν τα επιμέρους δίκτυα χρονικών περιορισμών. Το τρίτο βήμα αφορά τη σύνθεση των επιμέρους δικτύων για την κατασκευή ενός αριθμού συνολικών δικτύων χρονικών περιορισμών για κάθε μονοπάτι. Κάθε δίκτυο αναπαρίσταται από ένα στιγμιότυπο της κλάσης *CCNGraph*. Τέλος το τέταρτο βήμα που αφορά τον έλεγχο της χρονικής ακεραιότητας των συνολικών δικτύων χρησιμοποιεί μια ακόμα κλάση. Κάθε CN μεταφράζεται στον αντίστοιχο γράφο αποστάσεων. Ο γράφος αναπαρίσταται με την κλάση *CdGraph*.

Σε κάθε βήμα πραγματοποιείται ένας αριθμός ελέγχων που έχει σαν αποτέλεσμα τον εντοπισμό λαθών ή προειδοποιήσεων. Στις παραγράφους που ακολουθούν εκτός από την ανάλυση των παραπάνω κλάσεων και του τρόπου που κατασκευάζονται θα αναλύσουμε τα λάθη και τις προειδοποιήσεις που παράγονται κατά την διαδικασία κατασκευής.

Η γενική αρχιτεκτονική που μόλις περιγράψαμε απεικονίζεται στο σχήμα 4.1. Είναι φανερή ή ομοιότητα με το σχήμα 3.5 που απεικονίζει την αρχιτεκτονική της μεθοδολογίας SIC. Οι διαφορές αφορούν της προσθήκη ενός ακόμα σταδίου για την μετάφραση του σεναρίου της εφαρμογής στις κατάλληλες κλάσεις καθώς και την προσθήκη της διαδικασίας μετασχηματισμού των σύνθετων δικτύων χρονικών περιορισμών σε γράφους απόστασης.

4.2 Υλοποίηση του μοντέλου IMD

Στην παράγραφο αυτή θα μας απασχολήσουν δύο βασικά θέματα. Το πρώτο έχει να κάνει με τον τρόπο με τον οποίο αναπαρίσταται ένα IMD σενάριο στην υλοποίηση που έχουμε κάνει. Θα παρουσιαστούν δηλαδή οι κλάσεις που αναπαριστούν το μοντέλο IMD. Το δεύτερο θέμα αφορά τη μετάφραση ενός IMD σεναρίου στις κλάσεις της εφαρμογής.



Σχήμα 4.1: Η γενική αρχιτεκτονική της υλοποίησης της μεθοδολογίας Scenario Integrity Checking.

4.2.1 Οι κλάσεις αναπαράστασης του μοντέλου IMD

Μια από τις βασικές αρχές του αντικειμενοστραφούς σχεδιασμού είναι ότι σε κάθε πρόβλημα που πρέπει να αναπαρασταθεί με ένα object-oriented μοντέλο οι βασικές κλάσεις θα αναπαριστούν τα βασικά αντικείμενα του προβλήματος. Στηριζόμενοι σε

αυτή τη λογική προχωρήσαμε στην υιοθέτηση ενός απλού μοντέλου για τη αναπαράσταση ενός IMD σεναρίου.

Βασικά αντικείμενα του προβλήματος είναι τα αντικείμενα που παίρνουν μέρος στην παρουσίαση, τα γεγονότα που δέχεται η εφαρμογή (απλά ή γεγονότα συγχρονισμού) και τα tuples του σεναρίου. Κάθε ένα από τα αντικείμενα αυτά αναπαρίσταται με μία κλάση αντικειμένων. Η κλάση *CActor* αναπαριστά τα αντικείμενα της εφαρμογής, η κλάση *CScenarioEvent* τα απλά γεγονότα, η κλάση *CSynchronizationEvent* τα γεγονότα συγχρονισμού και τέλος η κλάση *CScenarioTuple* τα tuples του σεναρίου. Οι κλάσεις αυτές συγκεντρώνονται κάτω από την βασική κλάση του σεναρίου που είναι η κλάση *CScenario*. Τα βασικά μέλη της κλάσης είναι το όνομά της και ένα σύνολο από τέσσερις πίνακες που περιέχουν τα tuples, τα αντικείμενα και τα γεγονότα της εφαρμογής.

Ο ορισμός του κορμού της κλάσης παρουσιάζεται στο σχήμα 4.2. Οι κλάσεις *CArray* και *CTypedPtrArray* είναι κλάσεις που αναπαριστούν πίνακες κάθε στοιχείο των οποίων είναι με τη σειρά του αντικείμενο μιας άλλης κλάσης. Για παράδειγμα βλέπουμε ότι ο πίνακας *m_EventsList* περιέχει δείκτες σε αντικείμενα που ανήκουν στην κλάση *CScenarioEvents*. Ο πίνακας αυτός περιέχει δηλαδή το σύνολο των γεγονότων της κλάσης. Παρόμοιοι πίνακες υπάρχουν και για τα υπόλοιπες τέσσερις κλάσεις. Τέλος βλέπουμε και έναν πέμπτο πίνακα που περιέχει αντικείμενα της κλάσης *CCNGraph*. Περιέχει δηλαδή τα δίκτυα χρονικών περιορισμών για όλα τα μονοπάτια εκτέλεσης του σεναρίου. Η κλάση *CCNGraph* θα μας απασχολήσει αργότερα όπως και η κλάση *CStateTransitionDiagram* που αναπαριστά το διάγραμμα μετάβασης καταστάσεων.

```
class CScenario : public CObject
{
// Classs attributes
protected:
     // Array of CNs for the paths of the scenario
      CArray <CCNGraph *, CCNGraph *> m_CNGraphsArray;
      CStateTransitionDiagram m_Diagram;
     // Each list is a table which contain pointers to objects.
     // Each object (of any class) has a single name. The three
     // lists are sorted by this name.
     // Sorted list of scenario actors
     CTypedPtrArray<CObArray, CActor*> m_ActorsList;
     // Sorted list of scenario events
     CTypedPtrArray<CObArray, CScenarioEvent*> m_EventsList;
     // Sorted list of scenario tuples
     CTypedPtrArray<CObArray, CScenarioTuple*> m_TuplesList;
     // Sorted list of synchronization events
     CTypedPtrArray<CObArray,CSynchronizationEvent*>
          m_SynchEventsList;
.
.
.
};
```

Σχήμα 4.2: Τμήμα του ορισμού της κλάσης *CScenario*.

Φυσικά εκτός από τον ορισμό των μελών, η κλάση ορίζει και ένα σύνολο μεθόδων για την προσθήκη, διαγραφή και ανάκτηση των μελών των τεσσάρων πινάκων.

Ξεκινάμε την παρουσίαση των επιμέρους κλάσεων από την κλάση *CActor*. Η κλάση αυτή αναπαριστά τα αντικείμενα που παρουσιάζονται σε μία εφαρμογή. Όπως έχουμε αναφέρει το μοντέλο IMD μπορεί να χρησιμοποιήσει έναν μεγάλο αριθμό διαφορετικών τύπων αντικείμενων. Στην υλοποίησή μας έχουμε χρησιμοποιήσει 6 διαφορετικούς τύπους αντικειμένων. Κάθε ένας από τους τύπους έχει τα δικά του χαρακτηριστικά, ωστόσο όλοι χαρακτηρίζονται από ένα όνομα και φυσικά από το όνομα του τύπου στον οποίο ανήκουν. Επομένως μπορούμε να χρησιμοποιήσουμε μία υποκλάση (*subclass*) για να αναπαραστήσουμε κάθε τύπο αντικειμένων, ενώ η κεντρική κλάση (*base class*) θα είναι η κλάση *CActor*. Οι έξι υποκλάσεις που αναπαριστούν τους ισάριθμους τύπους αντικειμένων είναι οι ακόλουθες:

1. *CLabelButton*. Αναπαριστά τα κουμπιά που εμφανίζονται στην οθόνη (push buttons) ή τις απλές λεζάντες (label buttons).
2. *CImageBMP*. Αναπαριστά τις εικόνες που παρουσιάζονται στην εφαρμογή. Οι εικόνες πρέπει να έχουν μορφή bitmap (BMP format).
3. *CText*. Αναπαριστά τα κείμενα που εμφανίζονται στην εφαρμογή.
4. *CTimer*. Αναπαριστά τους χρονομετρητές που μπορούν να χρησιμοποιηθούν από μια εφαρμογή.
5. CVideoAVI. Αναπαριστά κινούμενη εικόνα. Τα videos που παρουσιάζονται πρέπει να έχουν τη μορφή αρχείων AVI (AVI format).
6. *CWave*. Αναπαριστά αρχεία ήχου. Οι ήχοι πρέπει να έχουν μορφή αρχείων WAV (WAV format).

Το σχήμα 4.3 παρουσιάζει τμήμα του ορισμού της βασικής κλάσης καθώς και του ορισμού της υποκλάσης *CImageBMP*. Όπως βλέπουμε τα μέλη των υποκλάσεων αντιστοιχούν στα χαρακτηριστικά κάθε τύπου αντικειμένου. Δύο ακόμα μέλη της κλάσης που φαίνονται στο σχήμα είναι τα *m_State* και *m_EventsArray*. Το πρώτο αναπαριστά την κατάσταση στην οποία βρίσκεται κάθε αντικείμενο, ενώ το δεύτερο είναι ένας πίνακας που περιέχει τα γεγονότα της εφαρμογής που έχουν σαν θέμα το παρόν αντικείμενο.

Οι δύο επόμενες κλάσεις που θα παρουσιάσουμε είναι οι κλάσεις που αναπαριστούν τα γεγονότα της εφαρμογής. Επιλέξαμε να δημιουργήσουμε δύο κλάσεις αντί για μία αφού τα γεγονότα που συνδέονται με ένα αντικείμενο παρουσιάζουν διαφορές με τα γεγονότα συγχρονισμού. Πράγματι τα γεγονότα συγχρονισμού δεν συνδέονται με κάποιο αντικείμενο αλλά με ένα από τα tuples του σεναρίου. Αυτή η διαφορά δημιουργεί και άλλες διαφορές στον τρόπο με τον οποίο γίνεται ο χειρισμός τους. Έτσι επιλέξαμε η κλάση *CScenarioEvent* να αναπαριστά τα απλά γεγονότα και η κλάση *CSynchronizationEvent* να αναπαριστά τα γεγονότα συγχρονισμού. Βασικά στοιχεία και των δύο κλάσεων αποτελούν το όνομα του γεγονότος και η λίστα με τις χρονικές στιγμές στις οποίες το γεγονός έχει εμφανιστεί. Τα απλά γεγονότα χαρακτηρίζονται από το αντικείμενο με το οποίο συνδέονται και με την πράξη η οποία τα προκαλεί. Τα γεγονότα συγχρονισμού χαρακτηρίζονται από το tuple στο οποίο ορίζονται καθώς και από μία σημαία (flag) που δηλώνει αν πρόκειται για γεγονός που εμφανίζεται με την εκκίνηση ή με την διακοπή του tuple (start synchronization event - stop synchronization event). Τα σχήματα 4.4a και 4.4b παρουσιάζουν τις δύο κλάσεις που αναπαριστούν τα γεγονότα της εφαρμογής.

```
////////////////////////////////////////////////////////////////
// CActor - Class inheritance product

class CActor : public CObject
{
public:
     typeOfActor m_Type;
     CString     m_Name;
protected:
     actorState m_State;
     CArray<CScenarioEvent *,CScenarioEvent *> m_EventsArray;

// Rest of declarations
.
.
.
};

////////////////////////////////////////////////////////////////
// CImageBMP - Class inheritance product

class CImageBMP : public CActor
{
// Attributes .
protected:
     CString     m_FileName;
     LONG        m_Height;
     LONG        m_Width;
     LONG        m_Xcoord;
     LONG        m_Ycoord;
     BYTE        m_Layer;
     BYTE        m_PreEffect;
     BYTE        m_PostEffect;

// Rest of declarations
.
.
.
};
```

Σχήμα 4.3: Τμήμα του ορισμού της κλάσης *CActor*.

Όσον αφορά τον ορισμό της κλάσης *CScenarioEvent* την προσοχή μας αξίζει ο τρόπο με τον οποίο γίνεται η σύνδεση με το αντικείμενο που προκαλεί το γεγονός. μέλη *m_SubjectIndex* και *m_SubjectPtr* είναι ένας δείκτης στον πίνακα των αντικειμένων της κλάσης *CScenario* και ένας δείκτης στο ίδιο το αντικείμενο αντίστοιχα. Όπως έχουμε αναφέρει είναι πιθανό η έννοια του αντικειμένου να αναφέρεται και σε αντικείμενα του συστήματος και όχι της εφαρμογής (πληκτρολόγιο, ποντίκι). Για το λόγο αυτό ο δείκτης *m_SubjectIndex* μπορεί να πάρει αρνητικές τιμές, υποδεικνύοντας τον τύπο του αντικειμένου. Επιπρόσθετα το είδος του γεγονότος μπορεί να διαπιστωθεί και από τις τρεις μεθόδους που παρέχονται από την κλάση: *TACEvent*(), *UserEvent*(), *TimerEvent*(). Τα γεγονότα που προκαλούνται από τους χρονομετρητές "ενεργοποιούν"

και ένα πρόσθετο μέλος της κλάσης (*m_TimeDuration*) που δείχνει το χρόνο στον οποίο θα συμβεί το γεγονός.

Στην κλάση *CSynchronizationEvent* η σύνδεση του γεγονότος με το tuple στο οποίο ορίζεται γίνεται με ένα δείκτη στο αντικείμενο *CScenarioTuple* που το αναπαριστά. Οι δύο κλάσεις αποθηκεύουν την ιστορία εμφανίσεων των γεγονότων στον πίνακα *m_EventsHistory*.

```
class CScenarioEvent : public CObject
{
// Attributes
public:
     CString m_Name;                // Strings
     CString m_Subject;
     CString m_Action;
     int m_SubjectIndex;            // Index of actor
     CActor * m_SubjectPtr;         // Pointer to the actor
     typeOfEventAction m_ActionID; // Identification of action
     int m_TimeDuration;            // Duration for timers
private:
// Keeps a timestamp history of the event occurences. The
// timestamps are relative to the application's start.
     CArray <int, int&> m_EventHistory;

// Operations
public:
     // Does the event happened right now?
     BOOL GetState(int time);
     // Is it an event caused by a TAC action?
     BOOL TACEvent();
     // Is it an user interaction event?
     BOOL UserEvent();
     // Is it an event caused by a timer?
     BOOL TimeEvent();
.
.
.
};
```

Σχήμα 4.4a: Τμήμα του ορισμού της κλάσης *CScenarioEvent*.

Η επόμενη κλάση που θα μας απασχολήσει έχει επίσης να κάνει με την αναπαράσταση γεγονότων. Πρόκειται για την κλάση *CEventFunction* που αναπαριστά τις συναρτήσεις γεγονότων. Το μοντέλο υποστηρίζει ένα σύνολο συναρτήσεων με διαφορετική σημασία και λειτουργικότητα. Η κλάση *CEventFunction* αποτελεί την βασική κλάση ενώ κάθε επιμέρους συνάρτηση αναπαρίσταται με μία υποκλάση. Συνολικά υποστηρίζονται 5 συναρτήσεις που αναπαρίστανται από ισάριθμες κλάσεις: *CFunctionAny*, *CFunctionAnyNew*, *CFunctionSequence*, *CFunctionTimes*, *CFunctionIn*. Μοναδικό μέλος της βασικής κλάσης είναι ο τύπος της συνάρτησης ενώ μοναδική μέθοδος είναι η συνάρτηση *GetState*(), που κατά αντιστοιχία με την ομώνυμη συνάρτηση των κλάσεων *CScenarioEvent* και *CSynchronizationEvent*, υποδεικνύει αν σε μία συγκεκριμένη χρονική στιγμή το σύνθετο γεγονός που ορίζεται από τη

συνάρτηση θα εμφανιστεί ή όχι. Φυσικά οι επιμέρους υποκλάσεις έχουν μέλη που αντιστοιχούν στα ιδιαίτερα χαρακτηριστικά τους. Το σχήμα 4.5 περιέχει τον ορισμό της κλάσης *CEventFunction* και μέρος του ορισμού μιας από τις υποκλάσεις της, της κλάσης *CFunctionAny*.

```
class CSynchronizationEvent : public CObject
{
// Attributes
public:
     CString m_Name;                // Strings
     BOOL m_StartEvent;             // Start or Stop event?
     CScenarioTuple * m_TuplePtr;   // Tuple which defines event

private:
// Keeps a timestamp history of the event occurences. The
// timestamps are relative to the application's start.
     CArray <int, int&> m_EventHistory;

public:
     // Does the event happened right now?
     BOOL GetState(int time);
     // Is it a start event?
     BOOL StartEvent();
.
.
.
};
```

Σχήμα 4.4b: Τμήμα του ορισμού της κλάσης *CSynchronizationEvent*.

Αφήσαμε τελευταία την κλάση *CScenarioTuple*. Ένα tuple αποτελείται από δύο σύνθετα γεγονότα (τα γεγονότα εκκίνησης και διακοπής), δύο γεγονότα συγχρονισμού και μία λίστα ενεργειών. Κάθε γεγονός συγχρονισμού αναπαρίσταται με ένα στιγμιότυπο της κλάσης *CSynchronizationEvent*. Όσον αφορά τα σύνθετα γεγονότα και την λίστα ενεργειών η κατάσταση είναι πιο σύνθετη.

Όπως έχουμε αναφέρει το γεγονός εκκίνησης και το γεγονός διακοπής ενός tuple αποτελεί μια αλγεβρική σύνθεση απλών γεγονότων, γεγονότων συγχρονισμού ή συναρτήσεων. Τα τρία είδη γεγονότων αναπαρίστανται με ισάριθμες κλάσεις. Επομένως το βασικό ζήτημα αφορά τον τρόπο με τον οποίο θα αναπαρασταθεί η σύνθεση των επιμέρους γεγονότων. Στην ουσία πρόκειται για μια αλγεβρική σύνθεση και επομένως η προφανής λύση είναι η χρήση *πολωνικού συμβολισμού* (*polish notation*). Η χρήση πολωνικού συμβολισμού πέρα από την ευκολία με την οποία μπορεί να υλοποιηθεί και την σαφήνεια που παρέχει, αποτελεί έναν οικονομικό -σε πόρους- τρόπο αναπαράστασης που επιπρόσθετα προσφέρεται για τον υπολογισμό παραστάσεων (*evaluation*).

Ας γίνουμε περισσότερο συγκεκριμένοι. Μια αλγεβρική παράσταση, για παράδειγμα η παράσταση (1+4) * 3 μπορεί να αναπαρασταθεί με τη χρήση πολωνικού συμβολισμού ως εξής: 14+*. Όπως βλέπουμε δε χρειάζονται παρενθέσεις αφού οι επιμέρους πράξεις εκτελούνται με τη σειρά που υποδεικνύεται από την παράσταση. Ας φανταστούμε τώρα την αντιστοίχηση με μια αλγεβρική σύνθεση γεγονότων. Ο

τελεστής σύζευξης (;) αντιστοιχεί στον πολλαπλασιασμό ενώ ο τελεστής διάζευξης (|) στην πρόσθεση. Έστω το σύνθετο γεγονός: (ANY(3; e_1, e_2, e_3, e_4) | e_4) ; e_5. Αν

```
////////////////////////////////////////////////////////////////
// CEventFunction : Base class
class CEventFunction : public CObject
{
// Common attributes
public:
     typeOfEventFunctions    m_Type;      // What function?

// Operations
public:
     // Does the event evaluate to TRUE?
     virtual BOOL GetState(int time, CScenario* scrPtr) const;
.
.
.
};



////////////////////////////////////////////////////////////////
// CFunctionAny : Subclass
class CFunctionAny : public CEventFunction
{
// Attributes
public:
     int m_FuncNum;
     int m_NumOfEvents;
     int * m_EventsList;
private:
     CArray <CString, CString&> m_Permutations;

// Operations
public:
     // Does the event evaluate to TRUE?
     virtual BOOL GetState(int time, CScenario* scPtr)  const;
.
.
.
};
```

Σχήμα 4.5: Τμήμα του ορισμού της κύριας κλάσης *CEventFunction και της υποκλάσης CFunctionAny.*

αναπαραστήσουμε με μία δομή κάθε επιμέρους γεγονός της παράστασης, η παράσταση παρίσταται ως εξής: (A | B) ; C όπου:

- με A αναπαριστούμε τη συνάρτηση ANY
- με B το απλό γεγονός e_4
- με C το γεγονός συγχρονισμού e_5

Ο πολωνικός συμβολισμός της παράστασης είναι: *ABC|;*. Όπως βλέπουμε η αναπαράσταση είναι απλή, οικονομική και σαφής. Επιπρόσθετα αν θελήσουμε να ελέγξουμε αν η παράσταση γίνεται αληθής αρκεί να ελέγξουμε τα επιμέρους γεγονότα και στη συνέχεια να ελέγξουμε την τιμή που λαμβάνει η παράσταση με βάση τους απλούς κανόνες της άλγεβρας BOOL για τους τελεστές σύζευξης και διάζευξης.

Η υλοποίησή μας ακολουθεί αυτή ακριβώς τη λογική. Τα επιμέρους γεγονότα αποθηκεύονται σε έναν πίνακα. Επομένως κάθε γεγονός η συνάρτηση αναπαρίσταται με τη θέση στην οποία έχει αποθηκευτεί. Στη συνέχεια ο πολωνικός συμβολισμός της παράστασης αποθηκεύεται σε μία συμβολοσειρά την οποία "μεταφράζουμε" όταν χρειαστεί να ανακτήσουμε το σύνθετο γεγονός. Στο παράδειγμα που παρουσιάσαμε, το γεγονός A αναπαρίσταται με ένα στιγμιότυπο της υποκλάσης *CFunctionAny*, το γεγονός B με ένα στιγμιότυπο της κλάσης *CScenarioEvent* και τέλος το γεγονός C με ένα αντικείμενο *CSynchronizationEvent*. Τα τρία γεγονότα αποθηκεύονται στις θέσεις 0, 1 και 2 αντίστοιχα του πίνακα γεγονότων. Η συμβολοσειρά που περιέχει τον πολωνικό συμβολισμό είναι η εξής: "0#1#2#|#;#", στην οποία το σύμβολο '#' χρησιμοποιείται για να χωρίζει τα επιμέρους σύμβολα.

Η αναπαράσταση της λίστας ενεργειών είναι πιο απλή. Υπενθυμίζουμε ότι κάθε μέλος της λίστας ενεργειών αποτελείται από:

- το αντικείμενο στο οποίο εκτελείται η ενέργεια
- την ίδια την ενέργεια
- το διάστημα που μεσολαβεί με την επόμενη ενέργεια (αν υπάρχει)

Επομένως αν αναπαραστήσουμε κάθε μέλος της λίστας με μια δομή που περιέχει τα τρία παραπάνω μέλη και αποθηκεύσουμε τα μέλη σε έναν πίνακα, έχουμε μια σαφής αναπαράσταση της λίστας ενεργειών.

Το σχήμα 4.6 περιέχει τμήμα του ορισμού της κλάσης *CScenarioTuple*. Τα μέλη *m_StartSynchEventPtr* και *m_StartSynchEventPtr* αποτελούν δείκτες προς τα αντίστοιχα γεγονότα συγχρονισμού. Τα μέλη *m_PostFixStartEvent* και *m_PostFixStopEvent* αποτελούν τις συμβολοσειρές αναπαράστασης των γεγονότων εκκίνησης και διακοπής. Τα επιμέρους γεγονότα αποθηκεύονται στους πίνακες *m_StartEventTermListRepr* και *m_StartEventTermListRepr*. Ο πίνακας *m_ActionListRepr* περιέχει τα μέλη της λίστας ενεργειών. Τέλος ορίζονται δύο ακόμα πίνακες: *m_CNFs* και *m_CNFInCN*. Ο πρώτος πίνακας αποθηκεύει το σύνολο των CNFs που αναπαριστούν τους περιορισμούς ενός tuple ενώ ο δεύτερος αποθηκεύει μόνο εκείνα τα δίκτυα που θα συμμετέχουν στο σχηματισμό ενός CN.

Το σχήμα 4.7 απεικονίζει το συνολικό αντικειμενοστραφές μοντέλο στο οποίο βασίστηκε η υλοποίηση της μεθοδολογίας. Στο μοντέλο περιλαμβάνονται οι κλάσεις που αναλύσαμε στην μέχρι τώρα συζήτησή μας. Επίσης έχουν περιληφθεί και οι κλάσεις στις οποίες θα αναφερθούμε στις επόμενες παραγράφους και συγκεκριμένα, η

```
class CScenarioTuple:public CObject
{
public:
     // Strings
     CString m_Name;
     CString m_StartEvent;
     CString m_StopEvent;
     CString m_ActionList;
     CString m_StartSynchEvent;
     CString m_StopSynchEvent;

     // Synchronization event for start-stop of tuple
     CSynchronizationEvent * m_StartSynchEventPtr;
     CSynchronizationEvent * m_StopSynchEventPtr;

     // Does the tuple starts with event StartApp?
     BOOL m_StartApp;
private:
     // String polish representation
     CString m_PostFixStartEvent;
     CString m_PostFixStopEvent;

     CArray <actionListRepresentation,
       actionListRepresentation &> m_ActionListRepr;
     CArray <eventTermListRepresentation,
       eventTermListRepresentation&> m_StartEventTermListRepr;
     CArray <eventTermListRepresentation,
       eventTermListRepresentation &> m_StopEventTermListRepr;

     // Constraint network fragments for the tuple
     CArray <CCNFGraph *, CCNFGraph *> m_CNFs;

     // Which of the CNFs should be used in a CN?
     CArray <int, int &> m_CNFinCN;
.
.
.
};
```

Σχήμα 4.6: Τμήμα του ορισμού της κλάσης *CScenarioTuple*.

κλάση *CStateTransitionDiagram* που αναπαριστά το διάγραμμα μετάβασης καταστάσεων, οι κλάσεις *CCNGraph* και *CCNFGraph* που αναπαριστούν τα σύνθετα και τα επιμέρους δίκτυα περιορισμών και η κλάση *CdGraph* που αναπαριστά το γράφο αποστάσεων.

4.2.2 Η γλώσσα του μοντέλου IMD

Στα παραδείγματα σεναρίων που παρουσιάσαμε ως τώρα, τα αντικείμενα, τα γεγονότα και τα tuples του σεναρίου παρουσιαζόταν ενδεικτικά σε απλούς πίνακες. Όπως αναφέραμε είσοδος της διαδικασίας ελέγχου της ακεραιότητας είναι ένα απλό αρχείο

κειμένου το οποίο περιέχει τους ορισμούς των τριών δομικών στοιχείων ενός σεναρίου.
Η εφαρμογή αναλαμβάνει να διαβάσει το αρχείο και να το μεταφράσει (*parse*) στις κλάσεις τις οποίες αναλύσαμε. Κατά τη διαδικασία αυτή πραγματοποιείται πλήθος ελέγχων που εξασφαλίζουν τη συμμόρφωση του αρχείου που μεταφράζεται με τους κανόνες της γλώσσας που έχει οριστεί. Μια αναλυτική παρουσίαση της γλώσσας ορισμού ενός σεναρίου και των ελέγχων που πραγματοποιούνται είναι εκτός των ορίων αυτής της εργασίας. Ο αναγνώστης μπορεί να προστρέξει για περισσότερες πληροφορίες στα [VTSH98, VTMHS98]. Ωστόσο είναι απαραίτητο να αναφερθούμε στους ελέγχους που αφορούν την ακεραιότητα του σεναρίου και που μπορούν ήδη να πραγματοποιηθούν κατά τη διαδικασία μετάφρασης του σεναρίου:

- *Δεν επιτρέπεται να υπάρχουν δύο γεγονότα που αφορούν την ίδια πράξη στο ίδιο αντικείμενο*. Κάθε γεγονός οφείλει να έχει διαφορετικό όνομα από όλα τα γεγονότα που έχουν οριστεί. Ωστόσο η γλώσσα δέχεται την ύπαρξη δύο γεγονότων που έχουν διαφορετικό όνομα αλλά αφορούν την ίδια πράξη που εφαρμόζεται στο ίδιο αντικείμενο. Για παράδειγμα μπορεί να οριστούν δύο γεγονότα *_BtnClick1* και *_BtnClick2* που εμφανίζονται όταν πατηθεί από το χρήστη το κουμπί *BTN*. Μια τέτοια περίπτωση δημιουργεί πρόβλημα στην εφαρμογή της μεθοδολογίας αφού δίνεται η δυνατότητα στο χρήστη να ορίσει διαφορετική συμπεριφορά για δύο γεγονότα που στην ουσία εμφανίζονται πάντα την ίδια χρονική στιγμή.

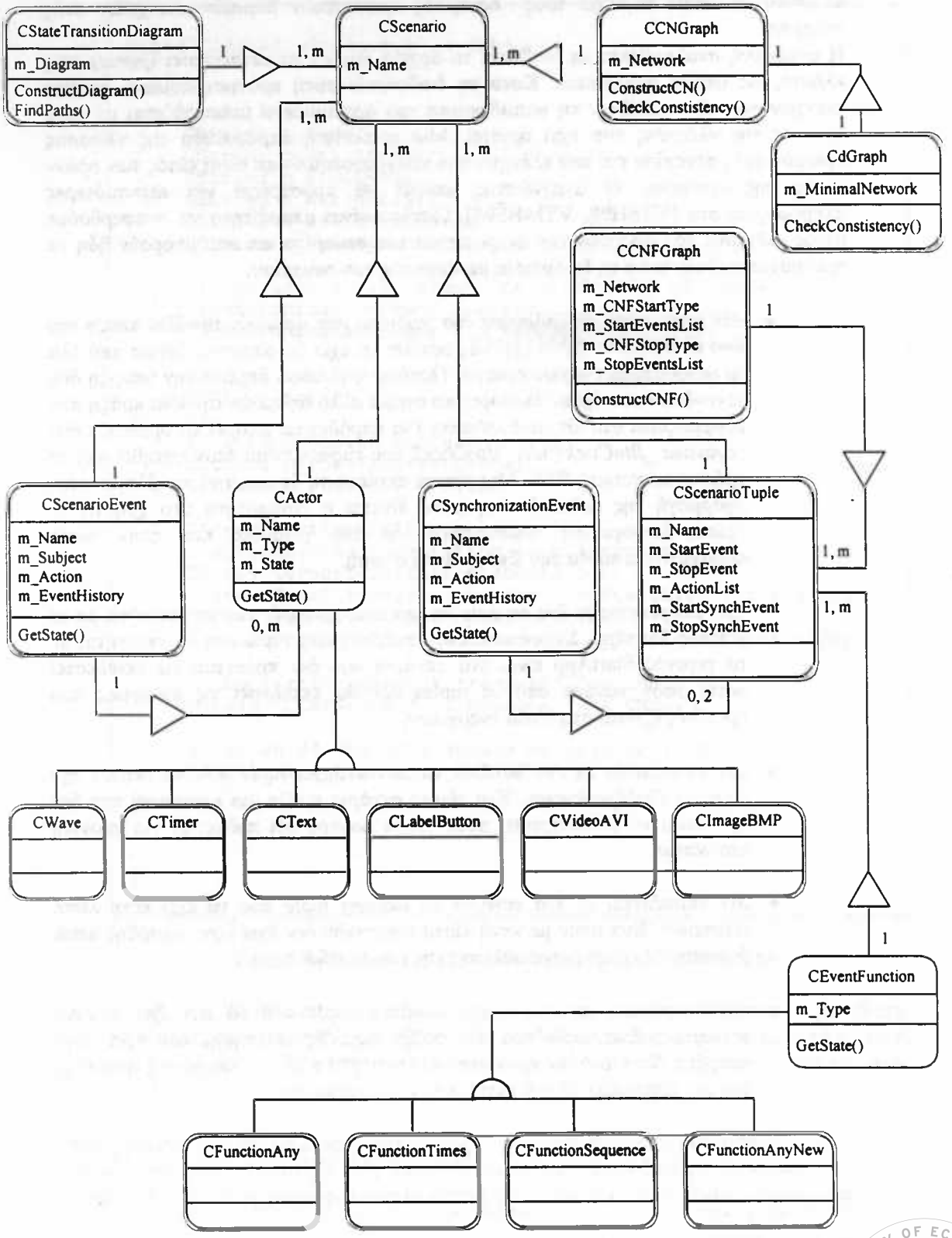
- *Δεν επιτρέπεται σε ένα σενάριο να μην υπάρχει tuple που να εκκινήται με το γεγονός StartApp*. Στην ουσία ένα σενάριο χωρίς tuple που να εκκινήται με το γεγονός StartApp είναι ένα σενάριο που δεν πρόκειται να εκτελεστεί ποτέ, αφού κανένα από τα tuples δεν θα εκτελέσει τις ενέργειες που προσδιορίζονται στη λίστα ενεργειών.

- *Δεν επιτρέπεται σε ένα σενάριο να μην υπάρχει tuple που να εκτελεί την ενέργεια ExitApplication*. Ένα τέτοιο σενάριο ορίζει μια εφαρμογή που δεν πρόκειται να ολοκληρωθεί ποτέ. Είναι φανερό ότι πρόκειται για χρονική ασυνέπεια.

- *Δεν επιτρέπεται σε ένα σενάριο να υπάρχει tuple που να έχει κενή λίστα ενεργειών*. Ένα tuple με κενή λίστα ενεργειών δεν έχει λόγο ύπαρξης αφού δεν επηρεάζει την ροή εκτέλεσης της εφαρμογής.

- *Δεν επιτρέπεται σε ένα σενάριο υπάρχει tuple που να μην έχει γεγονός εκκίνησης*. Ένα tuple που δεν ορίζει γεγονός εκκίνησης δεν έχει λόγο ύπαρξης. Το tuple δεν πρόκειται να εκκινηθεί ποτέ και επομένως η ύπαρξή του δεν επηρεάζει τη ροή εκτέλεσης της εφαρμογής.

Η παράβαση κάποιου από τους πέντε κανόνες που αναλύσαμε προκαλεί λάθος, που διακόπτει τη διαδικασία μετάφρασης. Είναι γεγονός ότι η παράβαση των δύο τελευταίων κανόνων θα μπορούσε να προκαλεί την εμφάνιση μιας προειδοποίησης



Σχήμα 4.7: Το συνολικό αντικειμενοστραφές μοντέλο της υλοποίησης.

προς το χρήστη και η διαδικασία μετάφρασης να συνεχίζεται κανονικά, αφού στην ουσία τέτοια λάθη μπορούν απλά να αγνοηθούν. Ωστόσο επιλέξαμε την διακοπή της διαδικασίας και στις δύο αυτές περιπτώσεις, κυρίως για λόγους ομοιομορφίας: *Οποιοδήποτε λάθος εντοπιστεί κατά την διαδικασία μετάφρασης του αρχείου σεναρίου, προκαλεί διακοπή της διαδικασίας.*

4.3 Εύρεση μονοπατιών εκτέλεσης

Το πρώτο στάδιο της μεθοδολογίας έχει σαν στόχο τον προσδιορισμό των διαφορετικών μονοπατιών που μπορούν να ακολουθηθούν κατά την εκτέλεση της εφαρμογής. Ο προσδιορισμός των μονοπατιών εκτέλεσης είναι απαραίτητος για να γίνει η σύνθεσητων δικτύων χρονικών περιορισμών στο βήμα 3. Όπως έχουμε αναφέρει κορμός της διαδικασίας προσδιορισμού των μονοπατιών εκτέλεσης είναι η αναπαράσταση της ροής εκτέλεσης της εφαρμογής με ένα διάγραμμα μετάβασης καταστάσεων.

Στην ουσία πρόκειται για έναν κατευθυνόμενο γράφο. Οι κόμβοι του γράφου αναπαριστούν τις συνολικές καταστάσεις της εφαρμογής και περιέχουν το σύνολο των tuples τα οποία συμμετέχουν σε αυτές. Οι συνδέσεις μεταξύ των κόμβων του γράφου αναπαριστούν τις μεταβάσεις καταστάσεων και περιέχουν τα γεγονότα (ή για να είμαστε πιο ακριβείς την σύζευξη ή την διάζευξη γεγονότων) που τις προκαλούν.

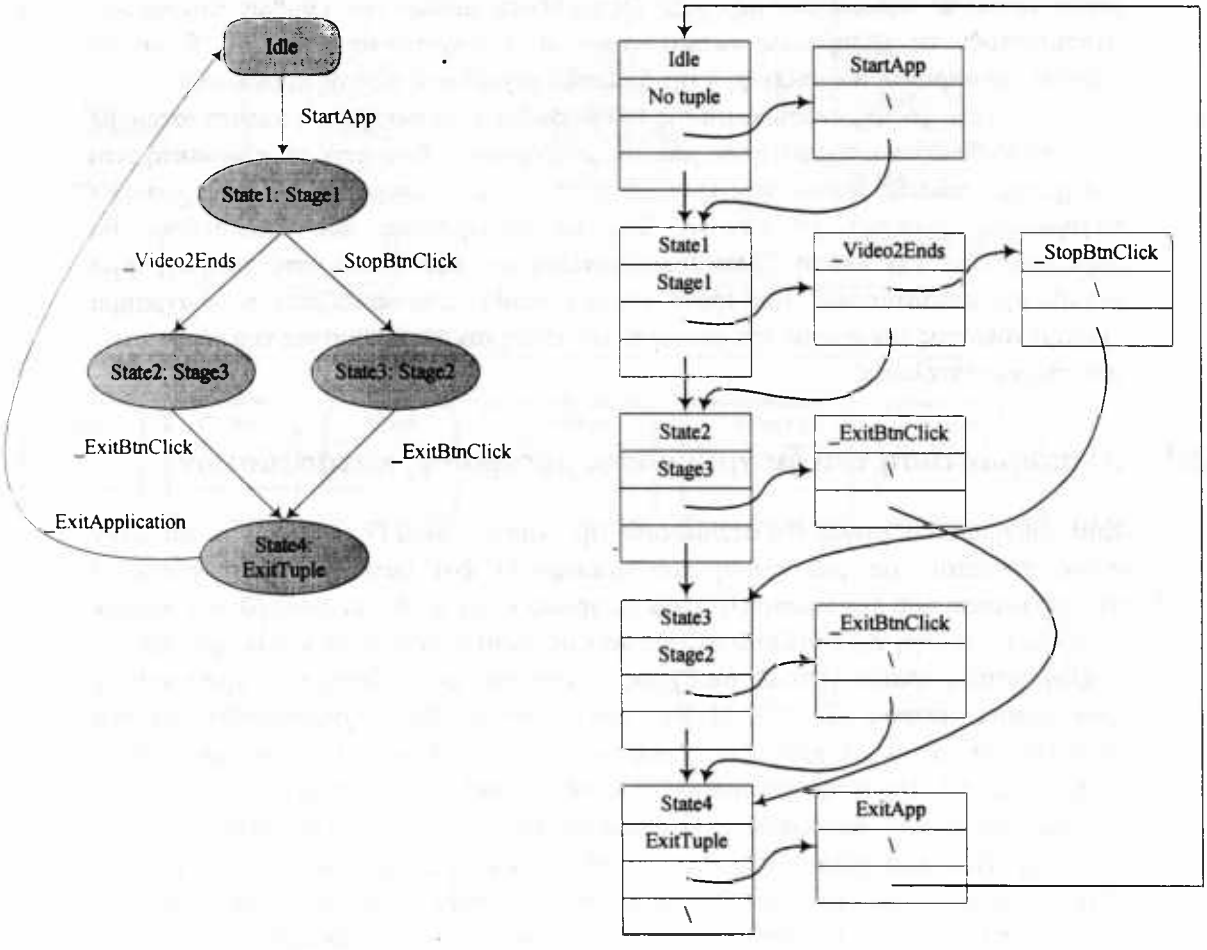
Η επιλογή της αναπαράστασης του διαγράμματος μετάβασης καταστάσεων με έναν κατευθυνόμενο γράφο ήταν μάλλον μονόδρομος. Ένα από τα πλεονεκτήματα αυτής της επιλογής αφορά την ευκολία με την οποία μπορούμε να εφαρμόσουμε αλγόριθμους εύρεσης μονοπατιών. Στις υποπαραγράφους που ακολουθούν θα παρουσιάσουμε την κλάση *CStateTransitionDiagram* που αναπαριστά το διάγραμμα μετάβασης καταστάσεων, τον τρόπο με τον οποίο κατασκευάζεται το διάγραμμα χρησιμοποιώντας τον ορισμό του σεναρίου και τέλος τον αλγόριθμο για την εύρεση των μονοπατιών εκτέλεσης.

4.3.1 Αναπαράσταση του διαγράμματος μετάβασης καταστάσεων

Κάθε διάγραμμα αποτελεί ένα στιγμιότυπο της κλάσης *CStateTransitionDiagram*. Στην ουσία πρόκειται για μία κλάση που αναπαριστά ένα κατευθυνόμενο γράφο. Η αναπαράσταση που χρησιμοποιείται για το γράφο είναι αυτή των ορθογώνιων λιστών [Dro95]. Η επιλογή της βασίζεται στο γεγονός ότι είναι περισσότερο κατάλληλη για την αναπαράσταση αραιών γράφων σε σχέση με την δεύτερη επιλογή, αναπαράσταση με ορθογώνιους πίνακες [Dro95]. Η ίδια αναπαράσταση θα χρησιμοποιηθεί και στη συνέχεια για τα δίκτυα χρονικών περιορισμών. Για το λόγο αυτό είναι χρήσιμο να εξετάσουμε τον τρόπο με τον οποίο υλοποιείται, καθώς και τα μειονεκτήματα και πλεονεκτήματα που παρουσιάζει. Η ανάλυση που θα ακολουθήσει στηρίζεται στα χαρακτηριστικά του γράφου που αναπαριστά το διάγραμμα μετάβασης καταστάσεων. Ωστόσο γνωρίζοντας τα χαρακτηριστικά των δικτύων χρονικών περιορισμών, ο αναγνώστης μπορεί να κατανοήσει τον τρόπο με τον οποίο γίνεται η προσαρμογή της υλοποίησης στις ανάγκες των CNFs και των CNs.

Κάθε κόμβος χαρακτηρίζεται από το όνομά του και μία δομή που περιέχει τις πληροφορίες της κατάστασης που αναπαριστά (δηλαδή τη λίστα των tuples που είναι ενεργά). Επίσης κάθε κόμβος συνδέεται με τον επόμενο κόμβο, σχηματίζοντας μια συνδεδεμένη λίστα. Πως αναπαρίστανται οι συνδέσεις μεταξύ κόμβων; Ας μην ξεχνάμε πως ο γράφος είναι κατευθυνόμενος. Επομένως κάθε σύνδεση ξεκινά από έναν κόμβο και καταλήγει σε έναν δεύτερο. Έτσι για κάθε κόμβο δημιουργούμε μία δεύτερη λίστα που περιέχει τις συνδέσεις που ξεκινούν από αυτόν. Η κάθε σύνδεση χαρακτηρίζεται από τον κόμβο στον οποίο καταλήγει, αφού ο κόμβος από τον οποίο ξεκινά είναι ο κόμβος που την περιέχει στη λίστα του. Οι συνδέσεις αναπαριστούν τις μεταβάσεις του διαγράμματος και επομένως η πληροφορίες που περιέχουν αφορούν τα γεγονότα που μετέχουν στην μετάβαση.

Το σχήμα 4.8 περιέχει ένα από τα διαγράμματα μετάβασης καταστάσεων που ήδη έχουμε παρουσιάσει και τον τρόπο με τον οποίο υλοποιείται με ορθογώνιες λίστες. Αριστερά φαίνονται οι κόμβοι του γράφου που σχηματίζουν μία συνδεδεμένη λίστα. Δεξιά βλέπουμε τη λίστα των συνδέσεων που σχηματίζεται για κάθε κόμβο. Το σύμβολο "/" χρησιμοποιείται για να δείξει το τέλος της λίστας.



Σχήμα 4.8: Υλοποίηση του κατευθυνόμενου γράφου ενός διαγράμματος μετάβασης καταστάσεων με ορθογώνιες λίστες.

Συγκρίνοντας την υλοποίηση με λίστες σε σχέση με την υλοποίησή με ορθογώνιο πίνακα, όσον αφορά τις απαιτήσεις σε αποθηκευτικό χώρο αναφέρουμε ότι η αναπαράσταση του ίδιου γράφου με χρήση πίνακα θα απαιτούσε τη χρήση ενός πίνακα 5×5, δηλαδή 25 θέσεις, κάθε μία από τις οποίες θα δήλωνε την ύπαρξη σύνδεσης μεταξύ των δύο κόμβων. Το μεγάλο πλεονέκτημα δεν έχει ωστόσο να κάνει με το χώρο που σπαταλάται όσο με την ευκολία με την οποία εκτελούνται οι αναζητήσεις στο γράφο. Στην περίπτωση των ορθογωνίων πινάκων η επεξεργασία των συνδέσεων του γράφου θα απαιτούσε αυξημένο χρόνο, αφού κάθε φορά πρέπει να εξετάζονται όλες οι θέσεις του πίνακα, μην γνωρίζοντας ποιες είναι κενές και ποιες δηλώνουν την ύπαρξη σύνδεσης. Αντίθετα στην υλοποίηση με ορθογώνιες λίστες, απλά διατρέχουμε τη λίστα των συνδέσεων κάθε κόμβου, που συχνά αποτελείται από ένα και μόνο μέλος. Το μειονέκτημα της υλοποίησης με ορθογώνιες λίστες αφορά το χρόνο που απαιτείται για τη δημιουργία και την εισαγωγή στο γράφο των κόμβων και των συνδέσεων. Στην περίπτωση όμως του διαγράμματος μετάβασης καταστάσεων αλλά και των δικτύων χρονικών περιορισμών, η εισαγωγή των κόμβων γίνεται μόνο μία φορά και όλες οι υπόλοιπες εργασίες έχουν να κάνουν με την επεξεργασία των κόμβων του γράφου. Επομένως η καταλληλότερη αναπαράσταση του γράφου είναι με τη χρήση ορθογώνιων λιστών.

Το σχήμα 4.9 παρουσιάζει τμήμα του ορισμού της κλάσης *CStateTransitionDiagram* που αναπαριστά το διάγραμμα μετάβασης καταστάσεων. Εκτός από την κλάση παρουσιάζονται, ενδεικτικά, και οι δομές που χρησιμοποιούνται για την αναπαράσταση των κόμβων και των συνδέσεων.

```
// Arc and node data structures
typedef struct nodeData_tag
{
    CString name;        // The name of the state
    int * stateTuples; // Which tuples are active in this state?
    int numOfTuples;    // How many are they?
} nodeData;

typedef struct arcData_tag
{
    CString data;          // Data on the arc
    int eventsList[100]; // List of events of the transition
} arcData;

// Pointers to structure
typedef struct stateNode_tag * stateNodePtr;
typedef struct stateArc_tag * stateArcPtr;

// Structure representing a graph node
```

Σχήμα 4.9: Τμήμα του ορισμού της κλάσης *CStateTransitionDiagram*. ***Συνεχίζεται***

```
typedef struct stateArc_tag
{
     nodeData data;     // Node data (see nodeData structure)
     stateNodePtr nextNode;     // Next node in graph
     stateArcPtr firstArc;      // First arc of current node
} stateNode;

// Structure representing a graph arc
typedef struct stateArc_tag
{
     arcData data; // Arc data (see arcData structure)
     stateNodePtr pointNode; // The node wich the arc points
     stateArcPtr nextArc;    // Next arc in arcs list
} stateArc;


//////////////////////////////////////////////////////////////
// CStateTransitionDiagram class definition

class CStateTransitionDiagram : public CObject
{
// Attributes
private:
     stateNode * m_Diagram;

// Operations
private:
     BOOL DetectSubGraph(stateNode * node);
public:
     // Find paths in scenario
     BOOL FindPaths(CArray<pathNode, pathNode&> & pathArray);
};
```

Σχήμα 4.9: Τμήμα του ορισμού της κλάσης *CStateTransitionDiagram*.

4.3.2 Αλγόριθμος εύρεσης μονοπατιών εκτέλεσης

Σκοπός της κατασκευής του διαγράμματος μετάβασης καταστάσεων είναι να χρησιμοποιηθεί για να εντοπιστούν τα μονοπάτια εκτέλεσης της εφαρμογής. Υπενθυμίζουμε ότι ένα μονοπάτι εκτέλεσης ξεκινά από τον κόμβο Idle, που είναι πάντα ο πρώτος κόμβος του διαγράμματος και καταλήγει ξανά σε αυτό αφού διατρέξει ένα πλήθος καταστάσεων. Επίσης σαν μονοπάτια εκτέλεσης θεωρούνται και τα μονοπάτια τα οποία καταλήγουν στην αόριστη κατάσταση

Στην υποπαράγραφο αυτή θα παρουσιάσουμε τον αλγόριθμο που χρησιμοποιούμε για τον εντοπισμό των μονοπατιών εκτέλεσης σε ένα διάγραμμα μετάβασης καταστάσεων. Ο σχεδιασμός αλγορίθμων για γράφους έχει αποτελέσει ένα από τα "αγαπημένα" θέματα έρευνας. Μια κατηγορία αλγορίθμων ασχολείται με την *διαδρομή των γράφων* (*graph traversals*) ενώ μια παρεμφερής κατηγορία ασχολείται με τον *εντοπισμό κύκλων* σε ένα γράφο (*cycle detection*). Το πρόβλημα του εντοπισμού των μονοπατιών εκτέλεσης είναι μια μίξη των δύο αυτών προβλημάτων. Πρέπει να διατρέξουμε τους κόμβους του κατευθυνόμενου γράφου και να εντοπίσουμε:

1. όλους τους κύκλους που ξεκινούν και καταλήγουν στον κόμβο Idle και
2. όλα τα μονοπάτια που ξεκινούν από τον κόμβο Idle και καταλήγουν στον κόμβο Undefined

Το πρόβλημα με τους συνηθισμένους αλγόριθμους διαδρομής είναι ότι σταματούν μόλις διατρεχτούν όλοι οι κόμβοι του γράφου ενώ όσον αφορά τους αλγόριθμους εντοπισμού κύκλων, σταματά μόλις εντοπίσει έναν κύκλο που ξεκινά από έναν κόμβο. Επομένως πρέπει να αναπτύξουμε έναν αλγόριθμο που θα μελετά εξαντλητικά όλα τα μονοπάτια που ξεκινούν από κάθε κόμβο του γράφου. Ο αλγόριθμος που χρησιμοποιούμε παρουσιάζεται στο σχήμα 4.10.

```
DetectSubGraph(u)
    Πρόσθεσε τον κόμβο u στο μονοπάτι που εξετάζεται;
    Για κάθε σύνδεση uX που ξεκινά από τον κόμβο u
        Αν ο κόμβος X στον οποίο καταλήγει η σύνδεση μετέχει στο μονοπάτι
            Αν ο κόμβος X είναι ο κόμβος Idle
                Προσέθεσε το μονοπάτι που σχηματίζεται στη λίστα των
                μονοπατιών;
            Αλλιώς αν ο κόμβος X δεν είναι ο κόμβος Idle
                Υπάρχει λάθος στο διάγραμμα;
        Αλλιώς αν ο κόμβος X δεν μετέχει στο μονοπάτι
            DetectSubGraph(X);
    Αν το μονοπάτι που δημιουργήθηκε καταλήγει στον κόμβο Undefined
        Προσέθεσε το μονοπάτι στη λίστα των μονοπατιών;
    Διέγραψε το μονοπάτι που ξεκινά από τον κόμβο u;
```

Σχήμα 4.10: Ο αλγόριθμος για τον εντοπισμό των μονοπατιών σε ένα διάγραμμα μετάβασης καταστάσεων.

Η λειτουργία του είναι απλή, βασίζεται στην *αναδρομή με οπισθοδρόμηση* (*backtracking recursion*) και χρησιμοποιεί δύο πίνακες. Έναν πίνακα που περιέχει τους κόμβους του μονοπατιού το οποίο εξετάζεται κάθε φορά και έναν πίνακα που περιέχει τα μονοπάτια που έχουν εντοπιστεί. Η αναδρομική συνάρτηση ονομάζεται *DetectSubGraph*(). Δέχεται σαν όρισμα ένα κόμβο του δικτύου και εντοπίζει όλα τα μονοπάτια (υπογράφους) που ξεκινούν από τον κόμβο αυτό. Κάθε φορά που ένας κόμβος εξετάζεται προστίθεται στη λίστα των κόμβων του μονοπατιού. Αν κάποιο από

τα μονοπάτια καταλήγει ξανά στην κατάσταση Idle προστίθεται στη λίστα μονοπατιών. Αν εντοπιστεί κύκλος που δεν καταλήγει στον κόμβο Idle προκαλείται λάθος αφού δεν επιτρέπεται να υπάρχουν κύκλοι στο διάγραμμα. Αν φθάσουμε σε ένα κόμβο που δεν έχει συνδέσεις, πρόκειται για κόμβο που "κλείνει" ένα μονοπάτι. Το μονοπάτι προστίθεται στη λίστα μονοπατιών και στη συνέχεια ο τελευταίος του κόμβος διαγράφεται ώστε να εξεταστούν τα επόμενα μονοπάτια.

4.4 Κατασκευή επιμέρους δικτύων χρονικών περιορισμών

Το δεύτερο στάδιο της μεθοδολογίας αφορά την κατασκευή των δικτύων χρονικών περιορισμών για κάθε ένα από τα tuples του σεναρίου. Ο τρόπος με τον οποίο αναπαρίσταται ένα δίκτυο χρονικών περιορισμών έχει ήδη αναλυθεί. Πρόκειται για έναν κατευθυνόμενο γράφο. Οι κόμβοι του γράφου αναπαριστούν τα γεγονότα που συνδέονται με την εκτέλεση του tuple. Συγκεκριμένα αναπαριστούν τα γεγονότα εκκίνησης και διακοπής, τις ενέργειες της λίστας ενεργειών και τα γεγονότα συγχρονισμού. Οι συνδέσεις μεταξύ των κόμβων αναπαριστούν τον περιορισμό που υπάρχει μεταξύ των δύο γεγονότων. Η κατασκευή του δικτύου στηρίζεται σε ένα σύνολο κανόνων που έχουν ήδη αναλυθεί. Επίσης έχουμε ήδη παρουσιάσει έναν γενικό αλγόριθμο στον οποίο στηρίζεται η κατασκευή των δικτύων για κάθε tuple. Στις δύο υποπαραγράφους που ακολουθούν θα παρουσιάσουμε τον τρόπο με τον οποίο αναπαρίσταται ένα δίκτυο χρονικών περιορισμών καθώς και τον πλήρη αλγόριθμο στον οποίο στηρίζεται η κατασκευή του.

4.4.1 Αναπαράσταση ενός δικτύου χρονικών περιορισμών

Ένα δίκτυο χρονικών περιορισμών αποτελεί ένα στιγμιότυπο της κλάσης *CCNFGraph*. Στην ουσία πρόκειται για μια κλάση που αναπαριστά και πάλι έναν κατευθυνόμενο γράφο, με τη διαφορά ότι οι κόμβοι και οι συνδέσεις είναι προσαρμοσμένες στις ανάγκες των δικτύων χρονικών περιορισμών. Η αναπαράσταση που χρησιμοποιείται για τον κατευθυνόμενο γράφο είναι η αναπαράσταση με συνδεδεμένες λίστες, όπως συνέβη και με την περίπτωση του διαγράμματος μετάβασης καταστάσεων.

Το σχήμα 4.11 περιέχει τμήμα του ορισμού της κλάσης *CCNFGraph*. Ο ορισμός της κλάσης παρουσιάζει μικρές διαφορές σε σχέση με τον ορισμό της κλάσης *CStateTransitionDiagam*. Η βασική διαφορά εντοπίζεται στην πληροφορία που περιέχεται σε κάθε κόμβο ή σύνδεση. Οι δύο δομές *CNFNodeData* και *CNFArcData* που έχουν περιληφθεί στο σχήμα παρουσιάζουν την πληροφορία που αποθηκεύεται στους κόμβους και τις συνδέσεις του γράφου αντίστοιχα. Επιπλέον πρέπει να σημειώσουμε την ύπαρξη τριών μελών που περιγράφουν τα γεγονότα που αποτελούν τον γεγονός εκκίνησης και ισάριθμων μελών που περιγράφουν το γεγονός διακοπής. Τα μέλη αυτά είναι απαραίτητα όχι μόνο για την κατασκευή του CNF αλλά όπως θα δούμε στη συνέχεια για τη σύνθεσή του με τα υπόλοιπα CNFs ενός μονοπατιού.

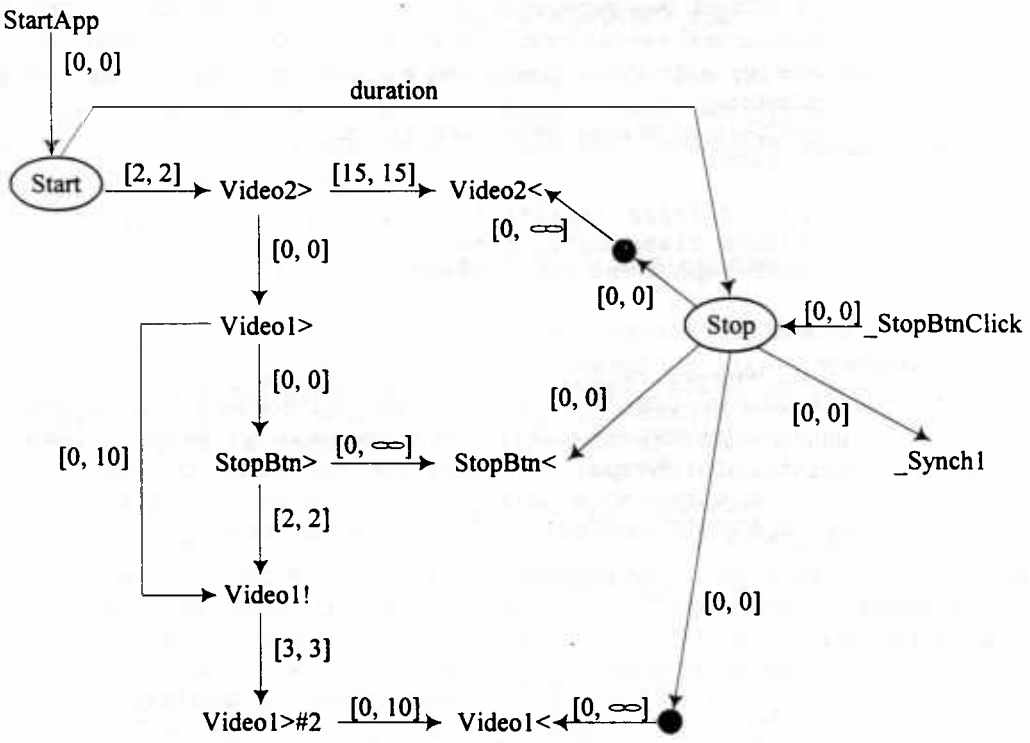
Ένα άλλο σημείο που πρέπει να σημειωθεί όσον αφορά την αναπαράσταση των δικτύων χονικών περιορισμών, αφορά την προσθήκη δύο "ειδικών" κόμβων σε κάθε δίκτυο. Κατά την ανάλυση των κανόνων για την κατασκευή των διαγραμμάτων είχαμε χρησιμοποιήσει έναν ειδικό κόμβο που αναπαριστά το γεγονός εκκίνησης κάθε tuple. Ο

```
// Data for each CNF node
typedef struct CNFNodeData_tag
{
    CString name;
    CNFNodeType type;          // What node is it?
    union
    {
        struct
        {
            CActor * actorPtr;     // Pointer to actor
            typeOfAction action;   // Action to perform on actor
            int timeStamp;         // Time of action
            CScenarioEvent * event;      // The event that is
                                   //connected with the action
        }actor;
        CScenarioEvent * event;
        CSynchronizationEvent * synchEvent;
    } data;
} CNFNodeData;

// Data for each CNF node
typedef struct CNFArcData_tag
{
    int min;      // minimum and maximum bounds of the constraint
    int max;
    typeOfArc type; // What arc is it?
} CNFArcData;

////////////////////////////////////////////////////////////////
// CCNFGraph class definition
class CCNFGraph : public CObject
{
// Attributes
public:
    int m_CNFStartType;          // Type of start event
    int * m_StartEventsList;     // List of start events
    int m_NumOfStartEvents;      // Number of start events
    int m_CNFStopType;
    int * m_StopEventsList;
    int m_NumOfStopEvents;
private:
     CNFNode * m_Network;
// Operations
public:
// CNF construction
    CString ConstructCNF(CScenarioTuple * tuplePtr,
  int CNFStartType, int * startEventsList, int numOfStartEvents,
  int CNFStopType, int * stopEventsList,
  int numOfStopEvents,int & numOfErrors,int & m_OfWarnings);
.
.
.
};
```

Σχήμα 4.11: Τμήμα του ορισμού της κλάσης *CCNFGraph*.

λόγος ήταν ότι στις περιπτώσεις που το γεγονός εκκίνησης ήταν ένα σύνθετο γεγονός, για παράδειγμα η συνάρτηση ANY, το γεγονός της εκκίνησης δεν μπορούσε να συνδεθεί άμεσα με έναν από τους κόμβους που αναπαριστούν τα γεγονότα που μετέχουν στη συνάρτηση. Αυτή η φιλοσοφία μεταφέρθηκε στην δική μας υλοποίηση. Τόσο το γεγονός εκκίνησης όσο και το γεγονός διακοπής αναπαρίστανται με ένα ειδικό κόμβο, ο οποίος με τη σειρά του συνδέεται με τον κόμβο που αναπαριστά την πρώτη πράξη της λίστας ενεργειών. Η προσθήκη των ειδικών κόμβων κάνει την αναπαράσταση πιο σαφή, ενώ διευκολύνει την διαδικασία σύνθεσης.

Το σχήμα 4.12 παρουσιάζει ένα δίκτυο χρονικών περιορισμών, χρησιμοποιώντας την αναπαράσταση που έχουμε υιοθετήσει. Χρησιμοποιούνται δηλαδή δύο ειδικοί κόμβοι για την αναπαράσταση των κόμβων εκκίνησης και διακοπής. Πρόκειται για το δίκτυο που παρουσιάσαμε στο σχήμα 3.17 με την προσθήκη των δύο ειδικών κόμβων. Όπως βλέπουμε όλες οι συνδέσεις που ξεκινούσαν από το γεγονός εκκίνησης (StartApp) ή το γεγονός διακοπής (_StopBtnClick) ξεκινούν τώρα από τους ειδικούς κόμβους εκκίνησης και διακοπής. Επίσης να σημειώσουμε ότι τα δύο γεγονότα έχουν συνδεθεί με τους ειδικούς κόμβους με διάρκεια 0, αφού πρόκειται για απλά γεγονότα και όχι για συναρτήσεις.



Σχήμα 4.12: Ένα δίκτυο χρονικών περιορισμών όπως αναπαρίσταται από την κλάση *CCNFGraph*.

4.4.2 Αλγόριθμος κατασκευής δικτύων χρονικών περιορισμών

Στην υποπαράγραφο 3.6.1 είχαμε παρουσιάσει τους κανόνες για τον μετασχηματισμό των tuples του σεναρίου σε δίκτυα χρονικών περιορισμών καθώς και τον γενικό

αλγόριθμο για την κατασκευή τους. Είχαμε σημειώσει ότι ο αλγόριθμος είναι αρκετά γενικός και απλά δίνει τις βασικές κατευθύνσεις που πρέπει να ακολουθήσει ένας πλήρης αλγόριθμος που θα κατασκευάζει δίκτυα χρονικών περιορισμών.

Σκοπός μας είναι να παρουσιάσουμε τον πλήρη αλγόριθμο τον οποίο χρησιμοποιούμε στην υλοποίησή μας. Ο αλγόριθμος στηρίζεται φυσικά στον αλγόριθμο του σχήματος 3.11 ωστόσο όπως μπορεί να παρατηρήσει κανείς έχουν προσδιοριστεί οι ακριβείς ενέργειες κάθε φάσης. Πρέπει να σημειώσουμε ότι και αυτός ο αλγόριθμος είναι αρκετά γενικός. Δίνει ωστόσο με ακρίβεια τα διαφορετικά στάδια που πρέπει να εκτελεστούν για τον μετασχηματισμό των tuples σε CNF.

Ο αλγόριθμος που παρουσιάζουμε στο σχήμα 4.13 αποτελείται από πέντε βήματα. Στο πρώτο βήμα σκοπός μας είναι να προσδιοριστούν τα διαφορετικά CNFs που θα αναπαραστήσουν το tuple. Για το λόγο αυτό αναλύονται τα γεγονότα εκκίνησης και διακοπής και σχηματίζονται οι κατάλληλοι συνδυασμοί τους που θα διαφοροποιήσουν τα δίκτυα του tuple. Στη συνέχεια εκτελούνται ακολουθιακά τα τέσσερα υπόλοιπα βήματα για κάθε συνδυασμό γεγονότων εκκίνησης-διακοπής.

Στο δεύτερο βήμα κατασκευάζεται ο ειδικός κόμβος εκκίνησης και συνδέεται τόσο με τα γεγονότα εκκίνησης όσο και με το γεγονός συγχρονισμού, αν φυσικά υπάρχει. Στο τρίτο βήμα αναλύεται η λίστα ενεργειών και κάθε επιμέρους ενέργεια σχηματίζει έναν κόμβο που συνδέεται με τον προηγούμενο κόμβο της λίστας αλλά και με τον κόμβο που αναπαριστά την προηγούμενη ενέργεια στο ίδιο αντικείμενο. Το τέταρτο βήμα πραγματοποιείται μόνο αν υπάρχει γεγονός διακοπής. Κατασκευάζεται ο αντίστοιχος κόμβος και συνδέεται με τα επιμέρους γεγονότα διακοπής και το γεγονός συγχρονισμού.

Το τελευταίο βήμα αφορά την προσθήκη κόμβων για την διακοπή εκτέλεσης των αντικειμένων. Για τα αντικείμενα που συνεχίζουν να παρουσιάζονται, προστίθεται ένας κόμβος διακοπής που συνδέεται κατάλληλα με το γεγονός διακοπής του tuple. Αν δεν υπάρχει γεγονός διακοπής προστίθεται ένας κόμβος ολοκλήρωσης.

Ο αλγόριθμος δεν αναλύει δύο από τα βασικά ζητήματα που προκύπτουν κατά την διαδικασία κατασκευής. Πρόκειται για την ανάγκη μετονομασίας των κόμβων και για τους ελέγχους που πραγματοποιούνται κατά την κατασκευή των δικτύων. Η διαδικασία μετονομασίας είναι απλή. Τα ονόματα των κόμβων που κατασκευάζονται αποτελούνται από το όνομα του αντικειμένου και το συμβολισμό της ενέργειας που εκτελείται σε αυτό (για τα γεγονότα δεν τίθεται θέμα μετονομασίας). Για παράδειγμα η ενέργεια "ξεκίνα το video A" αναπαρίσταται με τον κόμβο "A>". Κάθε φορά που χρειάζεται να προστεθεί ένας νέος κόμβος ελέγχεται το διάγραμμα ώστε να εντοπιστεί αν υπάρχουν κόμβοι με το ίδιο όνομα. Οι κόμβοι μετρούνται και ο αύξων αριθμός του νέου κόμβου προστίθεται στο όνομά του. Για παράδειγμα αν υπάρχουν 3 κόμβοι "A!" ο πρώτος θα παραμείνει ως έχει, ο δεύτερος θα μετονομαστεί σε "A!2" και ο τρίτος σε "A!3". Με τον τρόπο αυτό εξασφαλίζουμε τη μοναδικότητα των ονομάτων των κόμβων.

4.4.3 Έλεγχος ακεραιότητας

Σύμφωνα με την μεθοδολογία SIC μετά την κατασκευή των επιμέρους δικτύων χρονικών περιορισμών μπορούμε να εκτελέσουμε ποσοτικούς και ποιοτικούς ελέγχους. Συγκεκριμένα οι έλεγχοι πραγματοποιούνται στα διαφορετικά CNFs που έχουν προκύψει. Στην υλοποίησή μας διαφοροποιούμαστε ελαφρά αφού οι έλεγχοι που

Βήμα 1 (Εύρεση συνδυασμών)

Βρες όλους τους συνδυασμούς των γεγονότων που προκαλούν την εκκίνηση του tuple, δημιούργησε το σύνολο $A = \{\{e_{a1}, e_{a2}, \ldots\}, \{e_{a3}, e_{a4}, \ldots\}, \ldots\}$;
Βρες όλους τους συνδυασμούς των event που προκαλούν την διακοπή του tuple, δημιούργησε το σύνολο $B = \{\{e_{o1}, e_{o2}, \ldots\}, \{e_{o3}, e_{o4}, \ldots\}, \ldots\}$;
Πάρε όλους τους συνδυασμούς των συνόλων A και B, συνδυασέ τους και δημιούργησε το σύνολο $C = \{\{\{e_{a1}, e_{a2}, \ldots\}, \{e_{o1}, e_{o2}, \ldots\}\}, \{\{e_{a1}, e_{a2}, \ldots\}, \{e_{o3}, e_{o4}, \ldots\}, \ldots\}$;
Για κάθε έναν από τους συνδυασμούς που προέκυψαν εκτέλεσε τα βήματα 2, 3, 4 και 5 και δημιούργησε ένα CNF;

Βήμα 2 (Δημιουργία και σύνδεση κόμβου εκκίνησης)

Δημιούργησε έναν κόμβο εκκίνησης, N_{start}
Αν υπάρχει start synchronization event
    Δημιούργησε έναν κόμβο, $N_{startSyn}$;
    Δημιούργησε μια σύνδεση με τον κόμβο N_{start};
    Καταχώρησε σαν διάρκεια της σύνδεσης διάρκεια 0-0;
Για κάθε ένα από τα γεγονότα που σχηματίζουν το συνδυασμό $\{e_{a1}, e_{a2}, \ldots\}$
    Δημιούργησε έναν κόμβο, $N_{estartn}$;
    Ανάλογα με το είδος του συνδυασμού (Απλό γεγονός ή συνάρτηση) δημιούργησε μια σύνδεση με τον κόμβο N_{start};
    Ανάλογα με το είδος του συνδυασμού καταχώρησε την κατάλληλη διάρκεια στην σύνδεση;

Βήμα 3 (Δημιουργία και σύνδεση κόμβων από τη λίστα ενεργειών)

Για κάθε πράξη που ανήκει στην λίστα ενεργειών
    Δημιούργησε ένα νέο κόμβο, N_{ac};
    Δημιούργησε μια σύνδεση με τον προηγούμενο κόμβο στη λίστα N_{ac}-1
    Καταχώρησε σαν διάρκεια της σύνδεσης το χρόνο t που μεσολαβεί μεταξύ των δύο ενεργειών, t-t;
    Αν έχει προηγηθεί άλλη ενέργεια στο ίδιο αντικείμενο
        Συνέδεσε τον κόμβο N_{ac} με τον προηγούμενο κόμβο;
        Καταχώρησε σαν διάρκεια της σύνδεσης τη διάρκεια που προκύπτει από τον πίνακα των χρονικών περιορισμών (πίνακας 3.2);

Βήμα 4 (Δημιουργία και σύνδεση κόμβου διακοπής)

Αν υπάρχει γεγονός διακοπής

Σχήμα 4.13: Ο αλγόριθμος κατασκευής των δικτύων χρονικών περιορισμών. **συνεχίζ.**

Δημιούργησε έναν κόμβο διακοπής, N_{stop};
Δημιούργησε μια σύνδεση του κόμβου εκκίνησης N_{start} με τον κόμβο N_{stop};
Καταχώρησε σαν διάρκεια της σύνδεσης την υποτιθέμενη διάρκεια του tuple, d;
Αν υπάρχει stop synchronization event
    Δημιούργησε έναν κόμβο, $N_{stopSyn}$;
    Δημιούργησε μια σύνδεση με τον κόμβο N_{stop};
    Καταχώρησε σαν διάρκεια της σύνδεσης διάρκεια 0-0;
Για κάθε ένα από τα events που σχηματίζουν το συνδυασμό $\{e_{o1}, e_{o2}, \ldots\}$
    Δημιούργησε έναν κόμβο, N_{estopn};
    Ανάλογα με το είδος του συνδυασμού δημιούργησε μια σύνδεση με τον κόμβο N_{stop};
    Ανάλογα με το είδος του συνδυασμού καταχώρησε την κατάλληλη διάρκεια στην σύνδεση;

Βήμα 5 (Ολοκλήρωση ή διακοπή εκτέλεσης αντικειμένου)

Για κάθε αντικείμενο, O, που συμμετέχει στην λίστα ενεργειών
Αν το αντικείμενο δεν έχει ακόμα σταματήσει την εκτέλεσή του
    Δημιούργησε έναν κόμβο διακοπής για το αντικείμενο, N_{acstop};
    Δημιούργησε μια σύνδεση μεταξύ του κόμβου που αναπαριστά την τελευταία ενέργεια στο αντικείμενο, N_{aclast} , και του κόμβου N_{acstop};
    Καταχώρησε σαν διάρκεια της σύνδεσης την κατάλληλη διάρκεια μεταξύ των δύο ενεργειών (πίνακας 3.2);
    Αν υπάρχει κόμβος διακοπής για το tuple, N_{stop}
        Αν το αντικείμενο O εξαρτάται από το χρόνο
            Δημιούργησε έναν ειδικό κόμβο, N_{spec};
            Δημιούργησε μια σύνδεση του κόμβου N_{spec} με τον κόμβο N_{acstop}
            Καταχώρησε σαν διάρκεια της σύνδεσης 0-∞;
            Δημιούργησε μια σύνδεση του κόμβου N_o με τον κόμβο N_s;
            Καταχώρησε σαν διάρκεια της σύνδεσης 0-0;
        Αλλιώς, αν το αντικείμενο O δεν εξαρτάται από το χρόνο
            Δημιούργησε μια σύνδεση του κόμβου N_{stop} με τον κόμβο N_{acstop};
            Καταχώρησε σαν διάρκεια της σύνδεσης 0-0;
Αλλιώς, αν το αντικείμενο έχει σταματήσει την εκτέλεσή του
    Αν υπάρχει κόμβος παύσης για το tuple, N_{stop}
        Βρες τον κόμβο παύσης για το αντικείμενο O, N_{acstop};
        Δημιούργησε μια σύνδεση του κόμβου N_{stop} με τον κόμβο, N_{acstop}
        Καταχώρησε σαν διάρκεια της σύνδεσης 0-0;

Σχήμα 4.13: Ο αλγόριθμος για την κατασκευή των δικτύων χρονικών περιορισμών.

ορίζονται από τη μεθοδολογία πραγματοποιούνται κατά τη διάρκεια κατασκευής των δικτύων και όχι μετά την ολοκλήρωση της διαδικασίας.

Η φιλοσοφία που ακολουθούμε είναι απλή. Τόσο οι ποιοτικοί όσο και οι ποσοτικοί έλεγχοι αφορούν τις διαδοχικές πράξεις που εκτελούνται σε ένα αντικείμενο. Οι ποιοτικοί έλεγχοι εξετάζουν αν η ακολουθία των πράξεων είναι αποδεκτή, ενώ οι ποσοτικοί αν το διάστημα που μεσολαβεί μεταξύ δύο διαδοχικών πράξεων είναι το κατάλληλο, σύμφωνα πάντα με τους περιορισμούς του πίνακα 3.2. Αν λοιπόν κατά τη διαδικασία προσθήκης των κόμβων κρατάμε κάθε φορά την χρονική στιγμή στην οποία εκτελέστηκε η τελευταία πράξη σε ένα αντικείμενο, μπορούμε να ελέγξουμε αν η επόμενη πράξη παραβιάζει κάποιον από τους περιορισμούς.

Ένα ζήτημα που μας απασχόλησε αφορά το διαχωρισμό μεταξύ λαθών και προειδοποιήσεων. Αν εντοπιστούν κάποια χρονικά λάθη, τα επιμέρους δίκτυα που προκύπτουν είναι ασυνεπή. Ωστόσο υπάρχει η περίπτωση κατά τη σύνθεση των δικτύων να καλυφθούν αυτές οι ασυνέπειες. Για παράδειγμα έστω ότι σε ένα tuple υπάρχει η ακολουθία ενεργειών "A|| t A||". Κατά τους ελέγχους η παραπάνω ακολουθία θα προκαλέσει ένα ποιοτικό λάθος. Ωστόσο αν σε ένα άλλο tuple υπάρχει η ενέργεια "A\>" και τα δύο tuples εκτελεστούν έτσι ώστε μεταξύ των δύο ενεργειών pause να παρεμβληθεί η ενέργεια resume τότε δεν υπάρχει χρονικό λάθος. Το ερώτημα είναι προφανές. Πρόκειται για λάθος ή για μία προειδοποίηση;

Στην υλοποίησή μας ακολουθούμε αυστηρά την φιλοσοφία που ορίζεται από την μεθοδολογία SIC. Οι ασυνέπειες που προκύπτουν σε αυτό το επίπεδο αποτελούν χρονικά λάθη και επομένως η διαδικασία ελέγχου της ακεραιότητας πρέπει να διακοπεί.

Εκτός από τους ποσοτικούς και ποιοτικούς ελέγχους ένας ακόμα έλεγχος πραγματοποιείται κατά τη διαδικασία κατασκευής των δικτύων. Ας υποθέσουμε ένα tuple ,T1, που μεταξύ άλλων περιλαμβάνει την πράξη "A!" που ακολουθείται από άλλες ενέργειες. Ας υποθέσουμε επίσης ότι η πράξη "A!" προκαλεί τη διακοπή του tuple T1. Στην περίπτωση αυτή η εκτέλεση των ενεργειών του tuple θα σταματά αμέσως μετά την εκτέλεση της "A!", γεγονός που σημαίνει ότι οι υπόλοιπες πράξεις δεν έχουν λόγο ύπαρξης. Η περίπτωση που αναλύσαμε είναι φανερό ότι δεν διαταράσσει την χρονική ακεραιότητα του tuple ωστόσο μπορεί να οδηγήσει σε μη επιθυμητή συμπεριφορά της εφαρμογής. Για το λόγο αυτό αν εντοπιστεί παρόμοια περίπτωση προκαλείται μια ειδοποίηση προς τον χρήστη, που μπορεί να την αγνοήσει ή να διορθώσει το πρόβλημα.

Τέλος προειδοποίηση παράγεται σε μία ακόμα περίπτωση. Το μοντέλο IMD δεν επιτρέπει να υπάρχουν δύο στιγμιότυπα ενός αντικειμένου. Επομένως αν κατά τη διάρκεια της παρουσίασης ενός αντικειμένου (δηλαδή όσο βρίσκεται σε κατάσταση Idle η Suspended) εκτελεστεί μια πράξη start, αυτή δεν θα προκαλέσει καμία μεταβολή στην κατάσταση του αντικειμένου. Κατά τη διαδικασία κατασκευής των CNFs, παρόμοιες ακολουθίες ενεργειών εντοπίζονται και ο χρήστης προειδοποιείται ότι η ενέργεια start θα αγνοηθεί.

4.5 Σύνθεση επιμέρους δικτύων χρονικών περιορισμών

Το τρίτο αυτό στάδιο της μεθοδολογίας είναι το περισσότερο πολύπλοκο και απαιτητικό. Αιτία για την αυξημένη πολυπλοκότητα είναι η πολυπλοκότητα των κανόνων που χρησιμοποιούνται για να αποφασιστεί ο τρόπος με τον οποίο θα γίνει η σύνθεση.

Η διαδικασία της σύνθεσης μπορεί να χωριστεί σε τρία στάδια. Το πρώτο αφορά την επιλογή των CNFs που θα συμμετέχουν σε κάθε συνολικό δίκτυο. Η επιλογή

γίνεται ανάλογα με τα γεγονότα που ορίζουν το μονοπάτι εκτέλεσης που εξετάζουμε. Στο δεύτερο στάδιο γίνεται η σύνδεση των επιμέρους δικτύων που γίνεται με τη σύνδεση των κόμβων που αναπαριστούν τα γεγονότα εκκίνησης και διακοπής κάθε tuple. Κατά το στάδιο αυτό γίνεται επίσης, όπου κριθεί απαραίτητο και η μετονομασία των κόμβων του συνολικού δικτύου. Το τελευταίο στάδιο της διαδικασίας σύνθεσης είναι και το περισσότερο πολύπλοκο. Αφορά την προσθήκη των κατάλληλων συνδέσεων μεταξύ των κόμβων των επιμέρους δικτύων.

Εξαιτίας της πολυπλοκότητας που παρουσιάζουν οι αλγόριθμοι που χρησιμοποιούνται για το κάθε στάδιο κρίθηκε απαραίτητο τα τρία στάδια να παρουσιαστούν σε ξεχωριστές υποπαραγράφους. Παράλληλα με την ανάλυση των τριών σταδίων θα γίνεται και η παρουσίαση των ελέγχων που πραγματοποιούνται για να εντοπίσουν πιθανές χρονικές ασυνέπειες. Πρώτα όμως θα παρουσιάσουμε την κλάση *CCNGraph* που αναπαριστά ένα συνολικό δίκτυο χρονικών περιορισμών.

4.5.1 Αναπαράσταση συνολικών δικτύων χρονικών περιορισμών

Ένα συνολικό δίκτυο χρονικών περιορισμών προκύπτει από τη σύνθεση ενός αριθμού επιμέρους δικτύων. Είναι αναμενόμενο οι διαφορές στην αναπαράσταση να είναι ελάχιστες. Πράγματι ένα συνολικό δίκτυο αναπαρίσταται με ένα ακόμα κατευθυνόμενο γράφο. Μάλιστα τόσο οι κόμβοι του γράφου όσο και οι συνδέσεις μεταξύ τους, περιέχουν τις ίδιες ακριβώς πληροφορίες με τους κόμβους και τις συνδέσεις ενός δικτύου χρονικών περιορισμών. Οι διαφορές στις κλάσεις *CCNFGraph* και *CCNGraph* εντοπίζονται φυσικά στον τρόπο με τον οποίο κατασκευάζονται αλλά και στα μέλη που περιέχουν. Για παράδειγμα κάθε στιγμιότυπο της κλάσης *CCNGraph* συνδέεται με ένα στιγμιότυπο της κλάσης *CdGraph* που αναπαριστά τον γράφο αποστάσεων που αναπαριστά το δίκτυο χρονικών περιορισμών. Για λόγους πληρότητας παρουσιάζουμε στο σχήμα 4.14 τμήμα του ορισμού της κλάσης *CCNGraph*.

4.5.2 Επιλέγοντας τα κατάλληλα δίκτυα χρονικών περιορισμών

Εξετάζοντας επιφανειακά το θέμα της επιλογής των CNFs που θα μετέχουν σε ένα συνολικό δίκτυο μπορούμε να υποστηρίξουμε ότι αρκεί να πάρουμε τα CNFs κάθε tuple που μετέχει σε μία συνολική κατάσταση. Στη συνέχεια πρέπει να υπολογίσουμε όλους τους συνδυασμούς αυτών των CNFs, Κάθε συνδυασμός θα χρησιμοποιηθεί για να κατασκευαστούν τα διαφορετικά συνολικά δίκτυα.

Δυστυχώς η διαδικασία επιλογής δεν είναι τόσο απλή. Στην ανάλυση που είχαμε πραγματοποιήσει στην παράγραφο 3.7.1, είχαμε αποδείξει ότι δεν πρέπει να συμπεριλαμβάνονται όλα τα CNFs κάθε tuple σε ένα μονοπάτι εκτέλεσης αλλά μόνο εκείνα τα CNFs που αντιστοιχούν στα γεγονότα που προκαλούν τις μεταβάσεις των καταστάσεων. Επομένως η διαδικασία επιλογής των καταλλήλων CNFs απαιτεί την εξέταση των γεγονότων που σχηματίζουν τις μεταβάσεις κάθε μονοπατιού και τον προσδιορισμό των CNFs τα οποία περιέχουν αυτά τα γεγονότα. Στη συνέχεια

```
class CCNGraph : public CObject
{
// Attributes
private:
CNNode * m_Network;           // First node of graph
CScenario * m_Scenario;       // Scenario instance pointer
public:
int m_PathIndex;        // Index of CN's path
int m_CNIndex;          // Index of CN
BOOL m_Consistent;            // Is the CN consistent?
CdGraph * m_dGraph;           // Pointer to the distance graph

// Operations
public:
CString ConstructCN(CnData * cnData, int & errors,
 int & warnings, CString pathName,
    CArray<pathNode, pathNode &> & path);
BOOL CheckConsistency(int & pathIndex);
.
.
.
};
```

Σχήμα 4.14: Τμήμα του ορισμού της κλάσης CCNGraph.

σχηματίζουμε όλους τους πιθανούς συνδυασμούς των CNFs και για κάθε συνδυασμό κατασκευάζουμε ένα συνολικό δίκτυο χρονικών περιορισμών. Ο πλήρης αλγόριθμος παρουσιάζεται στο σχήμα 4.15 και εφαρμόζεται για κάθε μονοπάτι εκτέλεσης.

Ο αλγόριθμος είναι αρκετά σαφής. Ο τρόπος με τον οποίο αποφασίζεται αν ένα CNF πρέπει να μετέχει στο συνολικό δίκτυο ενός μονοπατιού έχει αναλυθεί στην υποπαράγραφο 3.7.1, όπως και ο τρόπος με τον οποίο σχηματίζονται οι συνδυασμοί των διαφορετικών CNFs. Αυτό που πρέπει μόνο να σημειώσουμε είναι η ύπαρξη δύο λιστών στις οποίες αποθηκεύονται τα CNFs ενός tuple. Η πρώτη λίστα (*m_CNFs*) περιέχει όλα τα δίκτυα που έχουν κατασκευαστεί για ένα tuple. Η δεύτερη λίστα (*m_CNFInCN*) χτίζεται κατά την εκτέλεση του αλγορίθμου που μόλις παρουσιάσαμε και περιέχει μόνο εκείνα τα CNFs που πρέπει να μετέχουν στο μονοπάτι που εξετάζεται.

Έλεγχος ακεραιότητας

Ο μοναδικός έλεγχος που γίνεται σε αυτό το στάδιο αφορά την ύπαρξη των καταλλήλων CNFs. Υπάρχει η περίπτωση να μην υπάρχουν δίκτυα που να αντιστοιχούν στα γεγονόα που εκκινούν ή που σταματούν ένα tuple. Αυτό σημαίνει ότι δεν υπάρχει CNF που να μετέχει στην κατασκευή του συνολικού δικτύου.

Ο έλεγχος για την ύπαρξη των κατάλληλων δικτύων γίνεται σε δύο σημεία του αλγορίθμου μας. Συγκεκριμένα ο πρώτος έλεγχος γίνεται όταν συναντήσουμε για πρώτη φορά το tuple σε μία από τις καταστάσεις του μονοπατιού. Με τον έλεγχο εξασφαλίζουμε ότι υπάρχει CNF που περιέχει το γεγονός εκκίνησης του tuple, σύμφωνα με την μετάβαση που εξετάζουμε. Ο δεύτερος έλεγχος γίνεται όταν

συναντήσουμε την τελευταία κατάσταση στην οποία μετέχει το tuple. Η μετάβαση στην επόμενη κατάσταση πρέπει να περιέχει ένα από τα γεγονότα διακοπής του tuple. Σε διαφορετική περίπτωση προκαλείται λάθος. Οι δύο περιπτώσεις προκαλούν λάθη και όχι προειδοποιήσεις αφού μας εμποδίζουν να συνεχίσουμε τη διαδικασία σύνθεσης των επιμέρους δικτύων.

```
Για κάθε συνολική κατάσταση S_i του μονοπατιού
    Για κάθε tuple T_j που μετέχει στην κατάσταση S_i
        Αν το tuple T_j δεν μετέχει στην προηγούμενη κατάσταση S_{i-1}
            Για κάθε CNF N_k του tuple T_j
                Αν το CNF N_k πρέπει να μετέχει στο συνολικό δίκτυο
                    Προσέθεσε το N_k στη λίστα των CNFs του tuple T_j;
            Αν δεν προσθέθηκε κανένα CNF στη λίστα
                Λάθος ακεραιότητας;
        Αλλιώς αν το tuple T_j μετείχε στην προηγούμενη κατάσταση S_{i-1}
            Για κάθε CNF N_k του tuple T_j που έχει ήδη προστεθεί στη λίστα
                Αν το N_k δεν μετέχει στην επόμενη κατάσταση S_{i-1}
                    Αν το N_k δεν πρέπει να μετέχει στο συνολικό δίκτυο
                        Διέγραψε το N_k από τη λίστα των CNFs του
                        tuple T_j;
            Αν η λίστα των CNFs είναι κενή
                Λάθος ακεραιότητας;

    Εξέτασε το σύνολο των CNFs που καθορίστηκε ότι θα μετέχουν στο μονοπάτι
    εκτέλεσης και σχημάτισε όλους τους δυνατούς συνδυασμούς;
    Για κάθε συνδυασμό CNFs κατασκεύασε ένα συνολικό δίκτυο;
```

Σχήμα 4.15: Ο αλγόριθμος επιλογής των καταλλήλων CNFs για την κατασκευή των συνολικών δικτύων χρονικών περιορισμών.

4.5.3 Συνδέοντας τα επιμέρους δίκτυα χρονικών περιορισμών

Μετά την ολοκλήρωση του πρώτου σταδίου έχει σχηματιστεί ένας αριθμός συνδυασμών από διαφορετικά επιμέρους δίκτυα. Στόχος του δεύτερου σταδίου είναι να πάρει έναν προς έναν τους συνδυασμούς και να συνδέσει τα CNFs μεταξύ τους. Τι περιλαμβάνει αυτή η σύνδεση; Στην πραγματικότητα η σύνδεση των CNFs που μετέχουν σε ένα συνολικό δίκτυο συνίσταται στην ένωση των κόμβων που αναπαριστούν τα γεγονότα εκκίνησης και διακοπής κάθε επιμέρους δικτύου.

Μέχρι τώρα οι κόμβοι που αναπαριστούν τα γεγονότα εκκίνησης και διακοπής είναι ανεξάρτητοι μεταξύ τους και δεν συνδέονται με κάποιον περιορισμό. Ωστόσο κάθε γεγονός που μετέχει σε ένα επιμέρους δίκτυο μετέχει σε μία τουλάχιστον από τις μεταβάσεις του μονοπατιού που εξετάζουμε. Το γεγονός είτε προκαλεί την εκκίνηση ενός ή περισσοτέρων tuples, είτε προκαλεί τη διακοπή ενός η περισσοτέρων tuples ή μπορεί να συμμετέχει τόσο στην εκκίνηση όσο και στη διακοπή, διαφορετικών φυσικά tuples. Αν εξετάσουμε ένα προς ένα τα επιμέρους δίκτυα θα συναντήσουμε πολλούς

κόμβους που αναπαριστούν εμφανίσεις γεγονότων που αποτελούν τα γεγονότα εκκίνησης και διακοπής των tuples. Στην ουσία πολλοί από αυτούς τους κόμβους αναπαριστούν τις ίδιες εμφανίσεις γεγονότων. Στόχος του σταδίου αυτού είναι να εντοπίσει ποιοι κόμβοι αναφέρονται στα ίδια γεγονότα και να τους ενώσει και ποιοι κόμβοι αναφέρονται σε διαφορετικές εμφανίσεις γεγονότων, οπότε απαιτείται μετονομασία.

Η γενική αρχιτεκτονική της διαδικασίας σύνδεσης των επιμέρους δικτύων είναι σχετικά απλή. Λαμβάνονται ένα προς ένα όλα τα επιμέρους δίκτυα του συνδυασμού που θα σχηματίσει το συνολικό δίκτυο. Στη συνέχεια λαμβάνονται ένας προς ένα οι κόμβοι που σχηματίζουν το επιμέρους δίκτυο. Οι κόμβοι που αναπαριστούν ενέργειες απλά εισάγονται στο συνολικό δίκτυο και αν είναι απαραίτητο μετονομάζονται. Ο τρόπος με τον οποίο γίνεται η μετονομασία είναι ο ίδιος που περιγράψαμε κατά την ανάλυση της διαδικασίας κατασκευής των επιμέρους δικτύων. Οι κόμβοι που αναπαριστούν γεγονότα εκκίνησης ή διακοπής ελέγχονται με ένα σύνολο κανόνων. Αν αναπαριστούν μια νέα εμφάνιση ενός γεγονότος ο κόμβος τους προστίθεται στο συνολικό δίκτυο. Αν αναπαριστούν ένα γεγονός που ήδη αναπαρίσταται με έναν κόμβο του δικτύου τότε ο κόμβος αυτός αντικαθιστά τον κόμβο του επιμέρους δικτύου. Ο τρόπος με τον οποίο γίνεται αυτή η αντικατάσταση είναι απλός: όλες οι συνδέσεις από και προς τον κόμβου του επιμέρους δικτύου προστίθενται στις συνδέσεις του κόμβου που ήδη έχει δημιουργηθεί.

Είναι φανερό ότι ο κορμός του αλγορίθμου σύνδεσης των επιμέρους δικτύων χρονικών περιορισμών είναι η διατύπωση των κανόνων που καθορίζουν αν υπάρχει ήδη κόμβος του δικτύου που αναπαριστά ένα γεγονός ή αν πρέπει να δημιουργηθεί νέος. Στην ουσία πρόκειται για δύο σύνολα κανόνων. Το πρώτο αφορά τα γεγονότα εκκίνησης και το δεύτερο τα γεγονότα διακοπής. Η ανάγκη για τον διαχωρισμό αυτό θα φανεί κατά την παρουσίαση των κανόνων. Επιπρόσθετα ο διαχωρισμός των κανόνων γίνεται ανάλογα με το είδος του γεγονότος που εξετάζεται. Υπενθυμίζουμε ότι κατά την παρουσίαση του μοντέλου IMD τα γεγονότα είχαν χωριστεί σε τέσσερις κατηγορίες:

1. Γεγονότα που προκαλούνται από το χρήστη (user events).
2. Γεγονότα που προκαλούνται από την αλληλεπίδραση των αντικειμένων της εφαρμογής (intra-object events).
3. Γεγονότα που σχετίζονται με την ίδια την εφαρμογή (application events).
4. Γεγονότα συγχρονισμού (synchronization events).

Ο διαχωρισμός που γίνεται για την διατύπωση των κανόνων είναι ελαφρά διαφορετικός. Η πρώτη κατηγορία περιλαμβάνει τα εσωτερικά γεγονότα της εφαρμογής (γεγονότα που προκαλούνται από αλλαγή της κατάστασης των αντικειμένων), η δεύτερη τα γεγονότα συγχρονισμού και η τρίτη τα γεγονότα που προκαλούνται από τις ενέργειες του χρήστη αλλά και τα γεγονότα που σχετίζονται με χρονομετρητές. Υπάρχει και μια τέταρτη κατηγορία στην οποία περιλαμβάνεται μόνο το γεγονός εκκίνησης της εφαρμογής (StartApp). Η ιδιορρυθμία αυτού του γεγονότος είναι ότι είναι μοναδικό σε κάθε συνολικό δίκτυο.

Ξεκινάμε την παρουσίαση των κανόνων από τους κανόνες που αφορούν τα γεγονότα εκκίνησης.

- *Γεγονός StartApp*. Εδώ ο κανόνας είναι πολύ απλός. Υπάρχει ένα μόνο γεγονός σε κάθε δίκτυο. Επομένως αν συναντήσουμε κόμβο που αναπαριστά

το γεγονός StartApp θα πρέπει απλά να ψάξουμε το δίκτυο που έχει ήδη σχηματιστεί. Αν δεν υπάρχει κόμβος StartApp δημιουργούμε έναν νέο. Αν υπάρχει ήδη απλά τον συνδέουμε με τον CNF που εξετάζουμε.

- *Γεγονότα συγχρονισμού*. Και στην περίπτωση αυτή ο κανόνας είναι απλός. Όταν ένα γεγονός συγχρονισμού μετέχει σε ένα γεγονός εκκίνησης, πρέπει να υπάρχει ήδη κόμβος στο συνολικό δίκτυο που το αναπαριστά αφού πρέπει το γεγονός να έχει ήδη δημιουργηθεί. Αν δεν υπάρχει κόμβος πρόκειται για λάθος.

- *Εσωτερικά γεγονότα*. Όταν ένα γεγονός εκκίνησης προκαλείται από ένα εσωτερικό γεγονός, τότε το γεγονός θα πρέπει να έχει ήδη δημιουργηθεί πριν από την εκκίνηση του tuple. Επομένως και σε αυτή την περίπτωση θα πρέπει να υπάρχει ήδη κόμβος στο δίκτυο που αναπαριστά το απλό γεγονός. Αν δεν υπάρχει προκαλείται λάθος.

- *Γεγονότα αλληλεπίδρασης με το χρήστη και γεγονότα που προκαλούνται από χρονομετρητές*. Η ιδιαιτερότητα που παρουσιάζουν αυτά τα γεγονότα είναι ότι υπάρχει η πιθανότητα να μην έχουν δημιουργηθεί ακόμα κόμβοι που τα αναπαριστούν. Στις τρεις παραπάνω κατηγορίες γεγονότων αυτό θα προκαλούσε λάθος αφού είναι απαραίτητο να έχει ήδη δημιουργηθεί κόμβος σε κάποιο από το προηγούμενα CNFs. Στην περίπτωση των γεγονότων αλληλεπίδρασης-χρόνου υπάρχει και πάλι η πιθανότητα να έχει δημιουργηθεί ήδη κόμβος (συγκεκριμένα αν πρόκειται για το γεγονός εκκίνησης ενός tuple το CNF του οποίου έχει ήδη προστεθεί στο δίκτυο ή αν πρόκειται για το γεγονός διακοπής ενός προηγούμενου CNF), αλλά υπάρχει και η πιθανότητα ο κόμβος να πρέπει να δημιουργηθεί κατά την εισαγωγή του νέου CNF. Ο διαχωρισμός των δύο περιπτώσεων γίνεται με τη βοήθεια των γεγονότων που μετέχουν στην μετάβαση που οδηγεί στην εκκίνηση του tuple. Αν το γεγονός που εξετάζουμε μετέχει στην μετάβαση, αυτό σημαίνει ότι πρόκειται για ένα νέο γεγονός. Επομένως αν δεν έχει δημιουργηθεί κόμβος που το αναπαριστά τον δημιουργούμε ενώ αν έχει ήδη δημιουργηθεί κόμβος, τον συνδέουμε με το νέο CNF. Η κατάσταση είναι πιο σύνθετη στην περίπτωση που το γεγονός δεν μετέχει στην μετάβαση. Αυτό σημαίνει ότι πρόκειται για ένα γεγονός που έχει δημιουργηθεί σε προηγούμενη μετάβαση και μετέχει σε μία από τις συναρτήσεις ANY ή SEQ. Επομένως πρέπει να ελέγξουμε πρώτα ότι όντως πρόκειται για μια από τις δύο συναρτήσεις και δεύτερον ότι το γεγονός έχει αναπαρασταθεί με έναν από τους κόμβους που ήδη υπάρχουν στο δίκτυο. Γιατί θα πρέπει να υπάρχει ήδη το γεγονός; Το γεγονός θα πρέπει να μετέχει οπωσδήποτε σε μία από τις μεταβάσεις του μονοπατιού εκτέλεσης. Εφόσον το γεγονός δεν μετέχει στην τελευταία μετάβαση τότε πρέπει να έχει δημιουργηθεί σε προηγούμενη κατάσταση και επομένως πρέπει να υπάρχει ήδη κόμβος που να το αναπαριστά. Αν το γεγονός δεν υπάρχει σε καμία από τις μεταβάσεις τότε δεν έχει ακόμα δημιουργηθεί, σύμφωνα πάντα με το μονοπάτι που εξετάζουμε, και επομένως πρόκειται για λάθος.

Το σχήμα 4.16a παρουσιάζει τον αλγόριθμο που χρησιμοποιείται για τη σύνδεση των γεγονότων εκκίνησης των δικτύων που αναπαριστούν τα tuples που μετέχουν σε ένα μονοπάτι εκτέλεσης. Είναι φανερή η διαφοροποίηση του χειρισμού των γεγονότων ανάλογα με το είδος τους.

```
Για κάθε γεγονός e που μετέχει στα γεγονότα εκκίνησης του CNF N
    Αν το γεγονός e είναι το StartApp
        Αν υπάρχει ήδη κόμβος E_node που αναπαριστά το γεγονός StartApp
            Συνέδεσε το CNF N με τον κόμβο E_node;
        Αλλιώς αν δεν υπάρχει κόμβος που αναπαριστά το γεγονός StartApp
            Δημιούργησε τον κόμβο E_node και εισήγαγέ τον στο δίκτυο;
    Αλλιώς αν το γεγονός e είναι ένα γεγονός συγχρονισμού
        Αν υπάρχει ήδη κόμβος E_node που αναπαριστά το γεγονός e
            Συνέδεσε το CNF N με τον κόμβο E_node;
        Αλλιώς αν δεν υπάρχει κόμβος που αναπαριστά το γεγονός e
            Λάθος ακεραιότητας;
    Αλλιώς αν το γεγονός e είναι ένα απλό γεγονός
        Αν το γεγονός e είναι ένα εσωτερικό γεγονός
            Αν υπάρχει ήδη κόμβος E_node που αναπαριστά το γεγονός e
                Συνέδεσε το CNF N με τον κόμβο E_node;
            Αλλιώς αν δεν υπάρχει κόμβος που αναπαριστά το γεγονός e
                Λάθος ακεραιότητας;
        Αλλιώς αν το γεγονός e είναι ένα γεγονός αλληλεπίδρασης με το χρήστη
        ή αν το γεγονός e είναι ένα γεγονός χρόνου
            Αν το γεγονός e μετέχει στην μετάβαση που οδηγεί στην εκκίνηση
            του tuple
                Αν υπάρχει ήδη κόμβος E_node που αναπαριστά το γεγονός e
                    Συνέδεσε το CNF N με τον κόμβο E_node;
                Αλλιώς αν δεν υπάρχει κόμβος που αναπαριστά το γεγονός e
                    Δημιούργησε τον κόμβο E_node;
            Αλλιώς αν το γεγονός e δεν μετέχει στην μετάβαση
                Αν υπάρχει ήδη κόμβος E_node που αναπαριστά το γεγονός e
                    Συνέδεσε το CNF N με τον κόμβο E_node;
                Αλλιώς αν δεν υπάρχει κόμβος που αναπαριστά το γεγονός e
                    Λάθος ακεραιότητας;
```

Σχήμα 4.16a: Ο αλγόριθμος σύνδεσης των γεγονότων εκκίνησης.

Όσον αφορά την σύνδεση των γεγονότων διακοπής των tuples οι κανόνες διαφοροποιούνται αρκετά. Αιτία για αυτή τη διαφοροποίηση είναι το γεγονός ότι ένα γεγονός διακοπής δεν είναι ανάγκη να έχει δημιουργηθεί σε κάποιο από τα προηγούμενα CNFs. Μπορεί να δημιουργηθεί σε ένα από τα επόμενα CNFs. Ο περιορισμός που τίθεται είναι ο κόμβος να μετέχει είτε στην τελευταία κατάσταση στην οποία είναι ενεργό το tuple είτε στην κατάσταση που την ακολουθεί. Η λογική μας είναι απλή: Ένα γεγονός διακοπής πρέπει να μετέχει σε μία μετάβαση. Αν το γεγονός μετέχει σε μία από τις συναρτήσεις ANY και SEQ τότε η μετάβαση μπορεί να έχει προηγηθεί

της διακοπής του tuple. Αν το γεγονός δεν μετέχει σε μία από τις δύο συναρτήσεις πρέπει να μετέχει στη μετάβαση που διακόπτει την εκτέλεση του tuple. Σε οποιαδήποτε περίπτωση που δεν συμμορφώνεται στους παραπάνω γενικούς κανόνες, προκαλείται λάθος. Ας δούμε όμως και πάλι μία προς μία τις κατηγορίες των γεγονότων και ας εξετάσουμε τους κανόνες σύνδεσης που χρησιμοποιούνται:

- *Γεγονός StartApp*. Το γεγονός αυτό δεν μπορεί να μετέχει σε ένα γεγονός διακοπής ενός tuple.

- *Γεγονότα συγχρονισμού*. Διαχωρίζουμε δύο περιπτώσεις. Στην πρώτη περίπτωση δεν έχει ακόμα δημιουργηθεί κόμβος που να αναπαριστά το γεγονός. Σύμφωνα με τον γενικό κανόνα που περιγράψαμε, στην περίπτωση αυτή θα πρέπει το γεγονός συγχρονισμού να μετέχει στην μετάβαση που προκαλεί την διακοπή του tuple οπότε και δημιουργούμε έναν νέο κόμβο. Αν το γεγονός δεν μετέχει στην τελευταία μετάβαση πρόκειται για λάθος. Στην δεύτερη περίπτωση υπάρχει ήδη κόμβος που αναπαριστά το γεγονός συγχρονισμού που μας ενδιαφέρει. Ωστόσο προκύπτει το ερώτημα για το αν πρόκειται για την ίδια εμφάνιση του γεγονότος ή για μία από τις προηγούμενες εμφανίσεις του. Ο έλεγχος βασίζεται στην κατάσταση στην οποία ανήκουν οι κόμβοι που αναπαριστούν τα δύο γεγονότα. Αν η κατάσταση είναι η ίδια πρόκειται για την ίδια εμφάνιση του γεγονότος οπότε και ενώνουμε τον κόμβο που βρέθηκε με το CNF που εξετάζουμε. Αν η κατάσταση είναι διαφορετική πρέπει να δημιουργήσουμε έναν νέο κόμβο. Ακόμα όμως και σε αυτή την περίπτωση υπάρχει μία εξαίρεση που αφορά τις συναρτήσεις ANY και SEQ. Αν το γεγονός μετέχει σε μία από αυτές τις συναρτήσεις δεν είναι ανάγκη να δημιουργήσουμε έναν νέο κόμβο αλλά πρέπει να χρησιμοποιήσουμε τον κόμβο που εντοπίσαμε.

- *Εσωτερικά γεγονότα*. Ο χειρισμός είναι ίδιος με τον χειρισμό των γεγονότων συγχρονισμού.

- *Γεγονότα αλληλεπίδρασης με το χρήστη και γεγονότα που προκαλούνται από χρονομετρητές*. Οι κανόνες που χρησιμοποιούμε στην περίπτωση αυτή είναι παρόμοιοι με τους κανόνες που περιγράψαμε στην περίπτωση των γεγονότων εκκίνησης. Αυτή τη φορά όμως εξετάζουμε τα γεγονότα της μετάβασης που διακόπτει το tuple και όχι τα γεγονότα της μετάβασης που το εκκινούν. Αν το γεγονός διακοπής που εξετάζουμε μετέχει στη λίστα των γεγονότων που σχηματίζουν την μετάβαση που διακόπτει το tuple τότε αυτό είναι το γεγονός που μας ενδιαφέρει. Αν υπάρχει ήδη κόμβος για αυτό το γεγονός, συνδέεται με το CNF ενώ αν δεν υπάρχει τον δημιουργούμε. Στην περίπτωση που το γεγονός που εξετάζουμε δεν μετέχει στην μετάβαση που διακόπτει το tuple τότε πρέπει να έχει ήδη δημιουργηθεί κόμβος που να το αναπαριστά τον οποίο και συνδέουμε με το CNF.

Ο αλγόριθμος που χρησιμοποιούμε για τη σύνδεση των γεγονότων διακοπής και περιγράφει τους κανόνες που αναλύσαμε παρουσιάζεται στο σχήμα 4.16b.

Έλεγχος ακεραιότητας

Οι έλεγχοι που πραγματοποιούνται στο δεύτερο αυτό στάδιο έχουν ήδη αναλυθεί. Εξετάζουμε αν τα γεγονότα που εκκινούν ή που σταματούν τα tuples που μετέχουν σε ένα μονοπάτι εκτέλεσης συμφωνούν με τα γεγονότα που περιγράφονται στο μονοπάτι που εξετάζεται. Σε κάθε περίπτωση που διαπιστωθεί ότι υπάρχει πρόβλημα, η διαδικασία σύνδεσης διακόπτεται και το δίκτυο σημειώνεται ως χρονικά ασυνεπές. Ο τρόπος με τον οποίο εκτελούνται οι έλεγχοι παρουσιάζεται στους δύο αλγόριθμους των σχημάτων 4.16a και 4.16b.

Υπάρχει ωστόσο ένας ακόμα έλεγχος που δεν εμφανίζεται στους αλγόριθμους που αναλύσαμε. Μας ενδιαφέρει να εντοπίσουμε τα tuples τα οποία δεν διακόπτονται από τα γεγονότα διακοπής τους αλλά από το γεγονός ExitApplication. Τα tuples αυτά συνεχίζουν να είναι ενεργά τη στιγμή που διακόπτεται η εκτέλεση της εφαρμογής. Ωστόσο εφόσον η κατάσταση αυτή δεν δημιουργεί πρόβλημα αναφέρεται ως προειδοποίηση προς το χρήστη και όχι ως χρονικό λάθος.

4.5.4 Προσθέτοντας συνδέσεις

Η μορφή στην οποία βρίσκεται το συνολικό δίκτυο μετά την επιτυχή ολοκλήρωση του δεύτερου σταδίου της διαδικασίας σύνθεσης είναι ατελής. Έχουν συνδεθεί τα γεγονότα εκκίνησης και διακοπής, έχουν πραγματοποιηθεί οι απαραίτητες μετονομασίες, αλλά υπάρχουν χρονικοί περιορισμοί που δεν έχουν ακόμα αναπαρασταθεί στο δίκτυο.

Η πρώτη κατηγορία περιορισμών αφορά τα γεγονότα εκκίνησης. Πράγματι, τα γεγονότα εκκίνησης των διαφορετικών tuples δεν έχουν ακόμα συνδεθεί μεταξύ τους με αποτέλεσμα να υπάρχουν χρονικοί περιορισμοί που δεν αναπαρίστανται στο δίκτυο. Ας πάρουμε για παράδειγμα ένα γεγονός, *_BtnClick*, που συνδέεται με το πάτημα ενός κουμπιού, *Btn*. Στο δίκτυο υπάρχει ήδη ο κόμβος που αναπαριστά το γεγονός *_BtnClick*. Ωστόσο ο κόμβος δεν έχει ακόμα συνδεθεί με τους υπόλοιπους κόμβους του δικτύου. Δεν υπάρχει δηλαδή περιορισμός που να "εμποδίζει" το γεγονός να συμβεί πριν από το γεγονός εμφάνισης του κουμπιού στην οθόνη (Btn>). Είναι επομένως φανερό ότι πρέπει να προστεθούν στο δίκτυο συνδέσεις που να αναπαριστούν τους χρονικούς περιορισμούς που αφορούν τα γεγονότα εκκίνησης.

Η δεύτερη ομάδα περιορισμών αφορά την προσθήκη συνδέσεων μεταξύ των ενεργειών που εκτελούνται στο ίδιο στιγμιότυπο ενός αντικειμένου. Αν οι ενέργειες εκτελούνται στο ίδιο tuple οι συνδέσεις έχουν δημιουργηθεί κατά την διαδικασία κατασκευής του δικτύου χρονικών περιορισμών. Αν όμως οι ενέργειες εκτελούνται σε διαφορετικά tuples δεν υπάρχει περιορισμός που να εμποδίζει την πράξη stop να προηγείται χρονικά της πράξης start, στο ίδιο στιγμιότυπο ενός αντικειμένου.

Σύνδεση των γεγονότων εκκίνησης

Οι περιορισμοί που αφορούν τα γεγονότα εκκίνησης προέρχονται από τους περιορισμούς του αντικειμένου με το οποίο συνδέεται η εμφάνιση του γεγονότος.

Για κάθε γεγονός *e* που μετέχει στα γεγονότα διακοπής του CNF *N* του tuple *T*
Αν το γεγονός *e* είναι ένα γεγονός συγχρονισμού ή αν το γεγονός *e* είναι ένα εσωτερικό γεγονός
Αν δεν υπάρχει κόμβος που αναπαριστά το γεγονός *e*
Αν το γεγονός δεν μετέχει στην μετάβαση που διακόπτει την εκτέλεση του tuple *T*
Λάθος ακεραιότητας;
Αλλιώς αν το γεγονός μετέχει στη μετάβαση που διακόπτει την εκτέλεση του tuple *T*
Δημιούργησε τον κόμβο E_{node} που αναπαριστά το γεγονός *e*;
Αλλιώς αν υπάρχει ήδη κόμβος E_{node} που αναπαριστά το γεγονός *e*
Αν η κατάσταση στην οποία εμφανίζεται το γεγονός *e* είναι η τελευταία κατάσταση στην οποία εμφανίζεται το tuple *T*
Συνέδεσε το CNF *N* με τον κόμβο E_{node};
Αλλιώς αν το γεγονός δεν μετέχει στη μετάβαση που διακόπτει την εκτέλεση του tuple *T*
Αν το γεγονός μετέχει στη συνάρτηση ANY ή στη συνάρτηση SEQ
Συνέδεσε το CNF *N* με τον κόμβο E_{node};
Αλλιώς
Λάθος ακεραιότητας;
Αλλιώς αν το γεγονός μετέχει στη μετάβαση που διακόπτει την εκτέλεση του tuple *T*
Δημιούργησε τον κόμβο E_{node} που αναπαριστά το γεγονός *e*;
Αλλιώς αν το γεγονός *e* είναι ένα γεγονός αλληλεπίδρασης με το χρήστη ή αν το γεγονός *e* είναι ένα γεγονός χρόνου
Αν το γεγονός *e* μετέχει στην μετάβαση που οδηγεί στην διακοπή του tuple
Αν υπάρχει ήδη κόμβος E_{node} που αναπαριστά το γεγονός *e*
Αν η κατάσταση στην οποία εμφανίζεται το γεγονός *e* είναι η τελευταία κατάσταση στην οποία εμφανίζεται το tuple *T*
Συνέδεσε το CNF *N* με τον κόμβο E_{node};
Αλλιώς
Δημιούργησε τον κόμβο E_{node} που αναπαριστά το γεγονός *e*;
Αλλιώς αν δεν υπάρχει κόμβος που αναπαριστά το γεγονός *e*
Δημιούργησε τον κόμβο E_{node} που αναπαριστά το γεγονός *e*;
Αλλιώς αν το γεγονός *e* δεν μετέχει στην μετάβαση
Αν υπάρχει ήδη κόμβος E_{node} που αναπαριστά το γεγονός *e*
Αν το γεγονός μετέχει στη συνάρτηση ANY ή στη συνάρτηση SEQ
Συνέδεσε το CNF *N* με τον κόμβο E_{node};
Αλλιώς
Λάθος ακεραιότητας;
Αλλιώς αν δεν υπάρχει κόμβος που αναπαριστά το γεγονός *e*

Σχήμα 4.16b: Ο αλγόριθμος σύνδεσης των γεγονότων διακοπής.

Επομένως αφού προσδιοριστεί ο περιορισμός θα ήταν λογικό η σύνδεση να γίνει με τον κόμβο που αναπαριστά την κατάλληλη ενέργεια που εκτελείται σε ένα αντικείμενο. Ωστόσο είναι περισσότερο χρήσιμο, για την πραγματοποίηση του τελικού ελέγχου της ακεραιότητας, η σύνδεση να γίνει με τον ειδικό κόμβο που αναπαριστά την εκκίνηση του tuple το οποίο περιλαμβάνει την ενέργεια.

Οι κανόνες που χρησιμοποιούμε για τη σύνδεση διαφοροποιούνται με βάση το είδος των γεγονότων εκκίνησης. Συγκεκριμένα διακρίνουμε τις εξής περιπτώσεις:

- *Εσωτερικά γεγονότα*. Στην περίπτωση που το γεγονός προέρχεται από μια ενέργεια που εκτελείται σε ένα αντικείμενο, δεν υπάρχει ανάγκη να προστεθούν συνδέσεις. Η σύνδεση του κόμβου με το υπόλοιπο δίκτυο έχει ήδη γίνει κατά την κατασκευή του CNF που περιλαμβάνει την ενέργεια.

- *Γεγονότα συγχρονισμού*. Τα γεγονότα συγχρονισμού συνδέονται άμεσα είτε με τον κόμβο εκκίνησης (start synch event) είτε με το γεγονός διακοπής (stop synch event) του tuple. Επομένως δεν υπάρχει ούτε σε αυτή την περίπτωση ανάγκη προσθήκης συνδέσεων.

- *Γεγονότα χρόνου*. Τα γεγονότα που συνδέονται με χρονομετρητές αποτελούν ιδιάζουσα περίπτωση. Στην περίπτωση που ο χρονομετρητής είναι ο μετρητής της εφαρμογής η σύνδεση είναι αυτονόητη: Συνδέουμε το γεγονός με τον κόμβο που αναπαριστά το γεγονός StartApp. Η διάρκεια της σύνδεσης είναι ίση με το χρόνο στον οποίο θα συμβεί το γεγονός. Στην περίπτωση όμως που ο χρονομετρητής είναι ένα απλό αντικείμενο, υπάρχει η περίπτωση να παρεμβληθεί μια πράξη pause οπότε είναι λάθος να συνδέσουμε το γεγονός με την ενέργεια έναρξης (start) του χρονομετρητή. Στην περίπτωση αυτή η σύνδεση γίνεται με τον κόμβο εκκίνησης του tuple που περιλαμβάνει την τελευταία πράξη επανεκκίνησης του χρονομετρητή (resume). Η διάρκεια της σύνδεσης είναι και πάλι ίση με το χρόνο στον οποίο θα συμβεί το γεγονός στην οποία προστίθεται η χρονική στιγμή στην οποία εκτελείται η πράξη. Αν δεν βρεθεί κόμβος που να αναπαριστά την εκκίνηση του χρονομετρητή, τότε προκαλείται λάθος.

- *Γεγονότα αλληλεπίδρασης με το χρήστη*. Στην περίπτωση αυτή η κατάσταση είναι λιγότερο πολύπλοκη αφού η σύνδεση δεν έχει άνω χρονικό όριο. Συνεπώς πρέπει απλά να εντοπίσουμε το tuple στο οποίο εκτελείται η τελευταία ενέργεια που θέτει το κουμπί σε κατάσταση Active, δηλαδή την τελευταία ενέργεια start ή resume. Στη συνέχεια συνδέουμε το γεγονός εκκίνησης του tuple με το γεγονός αλληλεπίδρασης με διάρκεια 0-∞. Αν δεν βρεθεί κόμβος που να αναπαριστά την εκκίνηση του αντικειμένου, τότε προκαλείται λάθος.

Κατά τη διαδικασία προσθήκης των συνδέσεων μεταξύ των γεγονότων εκκίνησης ο μοναδικός έλεγχος που πραγματοποιούνται αφορά την ύπαρξη της πράξης start για το αντικείμενο που εξετάζεται. Αν το αντικείμενο δεν έχει εμφανιστεί δεν είναι δυνατόν να προκληθεί το γεγονός που συνδέεται με αυτό και επομένως πρόκειται για χρονική ασυνέπεια. Το σχήμα 4.17 παρουσιάζει τον αλγόριθμο προσθήκης των συνδέσεων μεταξύ των γεγονότων εκκίνησης.

Σύνδεση των κόμβων TAC

Στο ζήτημα της σύνδεσης των κόμβων που αναπαριστούν τις ενέργειες που εκτελούνται στα αντικείμενα της εφαρμογής (TAC nodes) έχουμε ήδη αναφερθεί. Υπενθυμίζουμε ότι στόχος των συνδέσεων που θα προστεθούν είναι η αναπαράσταση των περιορισμών που υπάρχουν μεταξύ των πράξεων που εκτελούνται σε ένα στιγμιότυπο ενός αντικειμένου. Το πρόβλημα που πρέπει να αντιμετωπιστεί αφορά τον εντοπισμό των στιγμιοτύπων αφού είναι λάθος να συνδέσουμε διαφορετικά στιγμιότυπα του ιδίου αντικειμένου. Το συμπέρασμα ήταν ότι είναι αδύνατο να διατυπωθούν γενικοί κανόνες οι οποίοι θα εξασφαλίζουν τον εντοπισμό των στιγμιοτύπων και για το λόγο αυτό είναι απαραίτητη η επέμβαση του χρήστη.

Το γεγονός ότι δεν υπάρχουν γενικοί κανόνες δεν σημαίνει ότι δεν μπορούμε να χειριστούμε τη σύνδεση των κόμβων. Υπάρχει ένας αριθμός κανόνων τους οποίους μπορούμε να εφαρμόσουμε για να χειριστούμε την πλειοψηφία των περιπτώσεων σύνδεσης. Στόχος μας είναι να προσδιορίσουμε και να συνδέσουμε τους κόμβους TAC που αναπαριστούν ενέργειες που εκτελούνται στο ίδιο στιγμιότυπο ενός αντικειμένου. Η διάρκεια της σύνδεσης καθορίζεται από τους κανόνες που ορίσαμε στον πίνακα 3.2.

```
Για κάθε κόμβο του δικτύου που έχει κατασκευαστεί
    Αν ο κόμβος, E_node, αναπαριστά ένα γεγονός εκκίνησης e
        Αν το γεγονός e είναι ένα user interaction event
            Προσδιόρισε τον τελευταίο κόμβο N_last του δικτύου στον οποίο το
            αντικείμενο του γεγονότος τίθεται σε κατάσταση Active;
            Προσδιόρισε το tuple T στο οποίο ανήκει ο κόμβος N_last;
            Συνέδεσε τον κόμβο εκκίνησης του tuple T με τον κόμβο E_node;
        Αλλιώς αν το γεγονός e είναι ένα γεγονός χρόνου
            Αν ο χρονομετρητής είναι ο χρονομετρητής της εφαρμογής
                Συνέδεσε τον κόμβο StartApp με τον κόμβο E_node;
            Αλλιώς
                Προσδιόρισε τον τελευταίο κόμβο N_last του δικτύου στον
                οποίο ο χρονομετρητής τίθεται σε κατάσταση Active;
                Προσδιόρισε το tuple T στο οποίο ανήκει ο κόμβος N_last;
                Συνέδεσε τον κόμβο εκκίνησης του tuple T με τον κόμβο
                E_node;
```

Σχήμα 4.17: Ο αλγόριθμος προσθήκης συνδέσεων μεταξύ των γεγονότων εκκίνησης.

Η λογική, στην οποία στηριζόμαστε για να εντοπίσουμε τις ενέργειες που εκτελούνται στο ίδιο στιγμιότυπο ενός αντικειμένου, είναι απλή. Έχουμε ήδη θέσει τον περιορισμό ότι δεν μπορούν να υπάρχουν ταυτόχρονα δύο στιγμιότυπα του ιδίου αντικειμένου. Επίσης το μοντέλο IMD ορίζει ότι η παρουσίαση ενός αντικειμένου ξεκινά με την ενέργεια Start και διακόπτεται είτε όταν εκτελεστεί η ενέργεια Stop, είτε όταν ολοκληρωθεί ο χρόνος παρουσίασης ενός αντικειμένου, είτε όταν ένα από τα tuples στο οποίο συμμετέχει διακόψει την εκτέλεσή του. Επομένως έχοντας δύο

πράξεις[1], για να διαπιστώσουμε αν εκτελούνται στο ίδιο στιγμιότυπο αντικειμένου, πρέπει να ελέγξουμε τις εξής τρεις συνθήκες:

1. Πρέπει να έχει προηγηθεί η πράξη Start που ξεκινά την εκτέλεση του αντικειμένου. Σε διαφορετική περίπτωση το αντικείμενο δεν έχει ξεκινήσει την παρουσίασή του.
2. Οι δύο πράξεις πρέπει να ανήκουν σε tuples τα οποία να είναι ταυτοχρόνως ενεργά. Αν κάποιο από τα tuples έχει διακοπεί έχει διακοπεί και η παρουσίαση του αντικειμένου.
3. Μεταξύ των δύο πράξεων δεν πρέπει να έχει παρεμβληθεί πράξη Stop.

Αν οι τρεις συνθήκες ικανοποιούνται η εκτέλεση του αντικειμένου δεν έχει διακοπεί και οι δύο πράξεις εκτελούνται στο ίδιο στιγμιότυπο. Επομένως θα πρέπει να συνδεθούν. Αντίθετα αν κάποια από τις συνθήκες 2 και 3 δεν ικανοποιείται, η εκτέλεση έχει διακοπεί και είναι λάθος να συνδέσουμε τις δύο πράξεις.

Ο αλγόριθμος που χρησιμοποιούμε για την προσθήκη των συνδέσεων μεταξύ των αντικειμένων ουσιαστικά υλοποιεί τους κανόνες που περιγράψαμε. Για κάθε αντικείμενο της εφαρμογής, διατρέχουμε το δίκτυο περιορισμών και εντοπίζουμε τους κόμβους TAC που αναπαριστούν τις ενέργειες που εκτελούνται σε αυτό. Αν ο κόμβος αναπαριστά μία πράξη Start δεν χρειάζεται κάποια σύνδεση με τους κόμβους που έχουν προηγηθεί, αφού η εκκίνηση ενός αντικειμένου δεν προϋποθέτει κάποιον χρονικό περιορισμό. Αν ο κόμβος αναπαριστά μία από τις υπόλοιπες τρεις ενέργειες (Stop, Pause Resume) πρέπει να εξασφαλιστεί ότι έχει προηγηθεί ενέργεια Start (κανόνας 1). Επιπρόσθετα πρέπει να ελεγχθεί αν ισχύουν οι υπόλοιποι δύο κανόνες (κανόνες 2 και 3). Αν οι κανόνες ισχύουν, οι δύο κόμβοι συνδέονται με την κατάλληλη διάρκεια (πίνακας 3.2). Αν κάποιος από τους κανόνες δεν ισχύει δεν προστίθεται κάποια σύνδεση μεταξύ των δύο κόμβων.

Ο αλγόριθμος που περιγράψαμε -και παρουσιάζεται συνοπτικά στο σχήμα 4.18- έχει ένα αρκετά λεπτό σημείο. Η κατεύθυνση των συνδέσεων που προσθέτουμε καθορίζεται από τη σειρά με την οποία θα εκτελεστούν οι ενέργειες. Όπως σε όλες τις συνδέσεις του δικτύου, η σύνδεση πρέπει να ξεκινά από τον κόμβο που προηγείται χρονικά και να καταλήγει στον κόμβο που έπεται. Στις συνδέσεις που έχουμε προσθέσει ως τώρα η χρονική σειρά ήταν ήδη γνωστή αφού επρόκειτο για συνδέσεις που αφορούσαν ένα και μόνο tuple. Στη σύνδεση όμως των κόμβων TAC, οι κόμβοι ανήκουν σε διαφορετικά tuples και η χρονική τους σειρά δεν είναι πάντα γνωστή. Επομένως δεν είναι γνωστή και η κατεύθυνση της σύνδεσης, ούτε φυσικά ο περιορισμός που θα αναπαριστά. Ας γίνουμε περισσότερο κατανοητοί με ένα συγκεκριμένο παράδειγμα που θα αναδεικνύει το πρόβλημα.

Έστω δύο tuples T_1 και T_2. Στο πρώτο εκτελείται η ενέργεια "A||" ενώ στο δεύτερο η ενέργεια "A\>". Γνωρίζουμε από το διάγραμμα μετάβασης καταστάσεων ότι πρώτο θα εκκινηθεί το tuple T_1 και στη συνέχεια θα εκκινηθεί το tuple T_2. Ωστόσο δεν γνωρίζουμε το διάστημα που μεσολαβεί μεταξύ των δύο tuples και επομένως δεν γνωρίζουμε και τη σειρά με την οποία θα εκτελεστούν οι ενέργειες. Αν εκτελεστεί πρώτα η ενέργεια pause και μετά η ενέργεια resume, η σύνδεση πρέπει να έχει διάρκεια [0, ∞] (πίνακας 3.2). Αν οι ενέργειες εκτελεστούν με την αντίθετη σειρά η σύνδεση θα

[1] Οι πράξεις που εξετάζουμε ανήκουν σε διαφορετικά tuples. Οι συνδέσεις μεταξύ των ενεργειών του ίδιου tuple έχουν ήδη κατασκευαστεί.

έχει αντίστροφη κατεύθυνση και διάρκεια [0, r] (πίνακας 3.2). Για μία ακόμα φορά υπεύθυνος να καθορίσει τη σειρά των ενεργειών είναι ο χρήστης. Η εφαρμογή χρησιμοποιεί τη σειρά με την οποία έχουν τοποθετηθεί στο δίκτυο οι δύο κόμβοι για να αποφασίσει αυθαίρετα την κατεύθυνση της σύνδεσης. Ωστόσο ο χρήστης ενημερώνεται για αυτή την απόφαση με την κατάλληλη προειδοποίηση. Στο παράδειγμά μας, η σύνδεση θα έχει κατεύθυνση από τον κόμβο "A||" προς τον κόμβο "A\>" και διάρκεια [0, ∞].

Υπάρχει όμως μία συγκεκριμένη περίπτωση στην οποία γνωρίζουμε με σιγουριά τη σειρά με την οποία θα εκτελεστούν οι δύο πράξεις. Είναι η περίπτωση στην οποία γνωρίζουμε το χρονικό διάστημα που μεσολαβεί μεταξύ της εκκίνησης δύο tuples. Για παράδειγμα αναφέρουμε την περίπτωση στην οποία τα tuples εκκινούνται ταυτόχρονα, ή την περίπτωση στην οποία ένα γεγονός που παράγεται από την εκτέλεση των ενεργειών του πρώτου tuple είναι υπεύθυνο για την εκκίνηση του δεύτερου. Στις περιπτώσεις αυτές γνωρίζουμε το σχετικό χρόνο στον οποίο θα εκτελεστούν οι δύο πράξεις που μας ενδιαφέρουν. Επομένως γνωρίζουμε τη σειρά με την οποία θα εκτελεστούν και τον χρονικό περιορισμό που τις συνδέει.

Στον αλγόριθμο του σχήματος 4.18 παρουσιάζονται και οι έλεγχοι που πραγματοποιούνται για τον εντοπισμό χρονικών ασυνεπειών κατά την προσθήκη των συνδέσεων. Ο βασικοί έλεγχοι αφορούν την ικανοποίηση των ανισοτήτων 3.16-3.19 (§3.7.3). Αν κάποια από τις ανισότητες δεν ικανοποιείται προκαλείται λάθος και η διαδικασία διακόπτεται.

```
Για κάθε αντικείμενο A της εφαρμογής
    Για κάθε κόμβο N_TAC του δικτύου
        Αν ο κόμβος N_TAC αναπαριστά μια ενέργεια A_ACT που εκτελείται στο
        αντικείμενο A
            Αν η ενέργεια A_ACT είναι η ενέργεια Start
                Αν έχει προηγηθεί ενέργεια Start στο ίδιο στιγμιότυπο του
                αντικειμένου A
                    Προειδοποίηση;
            Αλλιώς αν η ενέργεια A_ACT είναι μία από τις Pause, Resume, Stop
                Αν δεν έχει προηγηθεί ενέργεια Start
                    Λάθος ακεραιότητας;
                Εντόπισε τον κόμβο N_TAC-1 που αναπαριστά την
                προηγούμενη ενέργεια που εκτελέστηκε στο αντικείμενο A;
                Αν οι δύο κόμβοι αναπαριστούν ενέργειες που
                πραγματοποιούνται στο ίδιο στιγμιότυπο του αντικειμένου A
                    Συνέδεσε τον κόμβο N_TAC-1 με τον κόμβο N_TAC;
```

Σχήμα 4.18: Ο αλγόριθμος προσθήκης συνδέσεων μεταξύ των γεγονότων εκκίνησης.

Το σχήμα 4.18 παρουσιάζει έναν επιπρόσθετο έλεγχο. Έχουμε αναφέρει ότι σε μία χρονική στιγμή μπορεί να υπάρχει ένα και μόνο στιγμιότυπο ενός αντικειμένου. Επομένως όσο συνεχίζεται η παρουσίαση ενός αντικειμένου οι πράξεις Start που εκτελούνται σε αυτό αγνοούνται. Ο αλγόριθμος ελέγχει το δίκτυο και αν εντοπίσει μια

πράξη Start που εκτελείται σε ένα αντικείμενο, η παρουσίαση του οποίου δεν έχει ακόμα διακοπεί, ενημερώνει τον χρήστη με μία προειδοποίηση.

4.6 Έλεγχος συνολικής χρονικής ακεραιότητας

Ο έλεγχος της συνολικής ακεραιότητας θα εκτελεστεί σε κάθε ένα από τα συνολικά δίκτυα χρονικών περιορισμών που προέκυψαν από τη σύνθεση των επιμέρους CNFs. Υπενθυμίζουμε ότι πρόκειται για μία απλή εφαρμογή του αλγορίθμου εύρεσης συντομότερων μονοπατιών σε ένα κατευθυνόμενο γράφο. Ο αλγόριθμος και η περιγραφή της εφαρμογής του έχουν παρουσιαστεί στην υποπαράγραφο 3.3.3. Για λόγους πληρότητας παρουσιάζουμε και πάλι τον πλήρη αλγόριθμο στο σχήμα 4.19.

```
for i = 1 to n do d[i, i] = 0;
    for k = 1 to n do
        for i, j = 1 to n do
            d[i, j] = min(d[i, j], d[i,k] + d[k, j]);
for i = 1 to n do
    if (d[i, i] < 0)
        Error;
```

Σχήμα 4.19: Ο αλγόριθμος των Floyd-Warshall για την εύρεση των συντομότερων μονοπατιών σε ένα γράφο και τον έλεγχο της ακεραιότητάς τους.

Είναι φανερό ότι η εφαρμογή του αλγορίθμου δεν μπορεί να γίνει άμεσα στο δίκτυο που χρησιμοποιούμε. Ο αλγόριθμος απαιτεί την "μεταφορά" του γράφου σε έναν πίνακα ο οποίος αναπαριστά τους περιορισμούς που συνδέουν τους κόμβους του δικτύου. Η μεταφορά γίνεται ως εξής. Διατρέχουμε όλους τους κόμβους του συνολικού δικτύου και αριθμούμε τον κάθε κόμβο. Ο αύξων αριθμός του κόμβου καταχωρείται μαζί με τα δεδομένα του ίδιου του κόμβου. Επίσης σε έναν μονοδιάστατο πίνακα κρατάμε την αντιστοίχηση των αριθμών με τους κόμβους.

Μπορούμε τώρα να αναπαραστήσουμε το δίκτυο χρονικών περιορισμών με ένα διδιάστατο πίνακα. Κάθε διάστασή του έχει μέγεθος που καθορίζεται από τον αριθμό των κόμβων του δικτύου. Ας πάρουμε δύο τυχαίους κόμβους του δικτύου, *A* και *B*, που έχουν αύξοντες αριθμούς *i* και *j* αντίστοιχα. Ας υποθέσουμε ότι οι δύο κόμβοι συνδέονται με μία σύνδεση που ξεκινά από τον κόμβο *A* και έχει διάρκεια [*min*, *max*]. Για την μεταφορά αυτής της σύνδεσης στον διδιάστατο πίνακα απαιτούνται δύο θέσεις. Στην θέση [*i*, *j*] θα αποθηκευτεί η μέγιστη διάρκεια του διαστήματος (άρα η τιμή *max*) ενώ στην [*j*, *i*] η ελάχιστη (*min*). Η διαδικασία αυτή πραγματοποιείται για όλους τους κόμβους και όλες τις συνδέσεις του δικτύου. Αν μεταξύ δύο κόμβων δεν υπάρχει σύνδεση, καταχωρούμε στην αντίστοιχη θέση μία ειδική τιμή που να υποδηλώνει την έλλειψη σύνδεσης.

Στη συνέχεια μπορούμε να εφαρμόσουμε στον πίνακα που προέκυψε τον αλγόριθμο των Floyd-Warshall. Ο πίνακας που προκύπτει αποτελεί το ελάχιστο δίκτυο του δικτύου χρονικών περιορισμών. Ο έλεγχος της ακεραιότητας του δικτύου είναι τώρα πολύ απλός. Απλά ελέγχουμε τα διαγώνια στοιχεία του πίνακα. Αν κάποιο από

αυτά είναι αρνητικό, αυτό σημαίνει ότι υπάρχει αρνητικός κύκλος στο δίκτυο και επομένως το δίκτυο είναι χρονικά ασυνεπές.

Ουσιαστικά το τελευταίο βήμα της μεθοδολογίας συνίσταται στον σχηματισμό του πίνακα που αναπαριστά κάθε δίκτυο χρονικών περιορισμών και στη συνέχεια στην εφαρμογή του αλγορίθμου του σχήματος 4.19. Θα μπορούσαμε να ενσωματώσουμε αυτές τις δύο λειτουργίες στην κλάση *CCNGraph* που αναπαριστά το δίκτυο. Ωστόσο προτιμήθηκε να δημιουργηθεί μία νέα κλάση, κάθε στιγμιότυπο της οποίας θα συνδέεται με ένα στιγμιότυπο της *CCNGraph*. Η κλάση ονομάστηκε *CdGraph* αφού αναπαριστά τον γράφο αποστάσεων του δικτύου χρονικών περιορισμών. Το βασικό τμήμα του ορισμού της κλάσης παρουσιάζεται στο σχήμα 4.20.

```
class CdGraph : public CObject
{
// Attributes
private:
    CCNGraph * m_CNGraph;      // Which constraint network?
    int m_NumOfNodes;          // How many nodes?
    CNNode* * m_NodesArray;    // These are the nodes
    int * m_MinimalNetwork;    // This is the minimal network

// Operations
public:
    int Create(CCNGraph * cnGraph=NULL);
    BOOL CheckConsistency(int * element=NULL);
    CNNode ** GetNodesArray(int & numOfNodes) const;
    int * GetMinimalNetwork(int & numOfNodes) const;
    int GetMinimalNetworkElement(int i, int j) const;
.
.
.
};
```

Σχήμα 4.20: Τμήμα του ορισμού της κλάσης *CdGraph*.

Οι δύο πίνακες *m_NodesArray* και *m_MinimalNetwork* έχουν τη λειτουργικότητα που αναλύσαμε παραπάνω. Όσον αφορά τις μεθόδους που παρέχει η κλάση, πρέπει να σημειώσουμε τον χωρισμό του αλγορίθμου που χρησιμοποιείται σε δύο φάσεις. Η ρουτίνα *Create*() υλοποιεί τη μεταφορά του δικτύου στους δύο εσωτερικούς πίνακες της κλάσης, ενώ η ρουτίνα *CheckConsistency*() κατασκευάζει και ελέγχει το ελάχιστο δίκτυο.

Το βασικό μειονέκτημα του τελικού ελέγχου αφορά τον ελλιπή προσδιορισμό των λαθών. Από την μία πλευρά ο αλγόριθμος υπολογίζει την ελάχιστη και μέγιστη τιμή του χρονικού διαστήματος που πρέπει να μεσολαβήσει μεταξύ δύο οποιονδήποτε γεγονότων, παρέχοντας ένα σημαντικό βοήθημα στο συγγραφέα της εφαρμογής. Από την άλλη πλευρά, από τη στιγμή που εντοπιστεί χρονική ασυνέπεια, το ελάχιστο δίκτυο δεν παρέχει καμία απολύτως πληροφορία για το πια ακριβώς είναι τα δύο γεγονότα που προκαλούν το λάθος. Η ευθύνη για τον εντοπισμό της αιτίας του λάθους μεταφέρεται στον συγγραφέα της εφαρμογής. Αντίθετα τα λάθη που εντοπίζονται σε κάποιο από τα προηγούμενα στάδια, προσδιορίζονται επακριβώς από την ίδια την εφαρμογή, και το μόνο που έχει να κάνει ο συγγραφέας είναι να τα διορθώσει.

ΚΕΦΑΛΑΙΟ 5

Συμπεράσματα

Η επιθυμία για κατασκευή όλο και περισσότερο σύνθετων εφαρμογών multimedia έχει δημιουργήσει την ανάγκη για την δημιουργία ενός νέου μοντέλου αναπαράστασης multimedia εφαρμογών. Η ανάγκη αυτή έγινε γρήγορα κατανοητή και είχε σαν αποτέλεσμα την παρουσίαση ενός πλήθους μοντέλων αναπαράστασης. Πολλά από τα μοντέλα αυτά παρουσιάζουν ενδιαφέροντα χαρακτηριστικά, ενώ άλλα παρουσιάζουν σημαντικά μειονεκτήματα. Το βασικά μειονεκτήματα των περισσότερων μοντέλων αφορούν την ελλιπή αναπαράσταση των χρονικών σχέσεων μεταξύ των αντικειμένων μιας εφαρμογής και την αδυναμία χειρισμού γεγονότων που παράγονται από το χρήστη.

Ένα από τα πιο ενδιαφέροντα μοντέλα για την αναπαράσταση interactive multimedia εφαρμογών είναι το μοντέλο Interactive Multimedia Document. Το μοντέλο στηρίζεται σε ένα ισχυρό θεωρητικό υπόβαθρο και δεν παρουσιάζει τις ελλείψεις που παρουσιάζουν τα υπόλοιπα μοντέλα. Το μοντέλο χρησιμοποιεί την έννοια των γεγονότων. Με τον τρόπο αυτό μπορεί να υποστηρίζει τόσο preorchistrated εφαρμογές όσο και εφαρμογές που περιλαμβάνουν αλληλεπίδραση με το χρήστη. Τα βασικά πλεονεκτήματα του μοντέλου IMD είναι τα ακόλουθα:

- Χρησιμοποιεί ένα πλήρες σύνολο ενεργειών και χρονικών τελεστών, δίνοντας τη δυνατότητα στο συγγραφέα μιας εφαρμογής να ορίσει οποιαδήποτε χρονική σχέση μεταξύ των αντικειμένων μιας multimedia εφαρμογής.

- Χρησιμοποιεί ένα σύνολο τελεστών και συναρτήσεων για τη σύνθεση των γεγονότων, δίνοντας τη δυνατότητα στο συγγραφέα να περιγράψει σύνθετες καταστάσεις εκτέλεσης. Επιπρόσθετα το μοντέλο δίνει μεγάλη έμφαση στον χειρισμό γεγονότων που προέρχονται από ενέργειες του χρήστη.

- Χρησιμοποιεί την έννοια των tuples για να ορίσει την ροή εκτέλεσης μιας multimedia εφαρμογής. Κάθε tuple ορίζει τα γεγονότα που θα προκαλέσουν την εκκίνηση και την διακοπή του καθώς και τις ενέργειες οι οποίες θα εκτελεστούν στα αντικείμενα της εφαρμογής. Τα tuples μπορούν να εκφραστούν με τη χρήση μιας scripting γλώσσας, γεγονός που δίνει μεγάλη ευελιξία στις εφαρμογές που στηρίζονται στο μοντέλο (εφαρμογές authoring, rendering, ελέγχου χρονικής ακεραιότητας). Επιπρόσθετα δίνει τη δυνατότητα αποθήκευσης των σεναρίων σε μία βάση δεδομένων, με όλα τα πλεονεκτήματα που παρέχονται από το γεγονός αυτό.

Ένα από τα βασικά ζητήματα που προκύπτουν κατά τη διαδικασία συγγραφής μιας multimedia εφαρμογής αφορά το πρόβλημα του ελέγχου της χρονικής ακεραιότητας. Ο συγγραφέας της εφαρμογής επικεντρώνει την προσοχή του σε συγκεκριμένα τμήματα της παρουσίασης, χάνοντας την γενική αντίληψη της εφαρμογής. Το γεγονός αυτό έχει συχνά σαν αποτέλεσμα την ύπαρξη χρονικών ανακολουθιών, που πιθανώς να δημιουργούν προβλήματα κατά την παρουσίαση της εφαρμογής. Η κατασκευή ενός εργαλείου ελέγχου της χρονικής ακεραιότητας ενός σεναρίου θα έδινε τη δυνατότητα στο συγγραφέα να εντοπίζει τις χρονικές ασυνέπειες, να κρίνει την σοβαρότητα των προβλημάτων που δημιουργούν και να τις διορθώνει αν κρίνεται απαραίτητο.

Έχουν προταθεί αρκετές μεθοδολογίες για τον έλεγχο της χρονικής ακεραιότητας μιας multimedia εφαρμογής. Οι μεθοδολογίες στηρίζονται στα μοντέλα αναπαράστασης εφαρμογών και επομένως παρουσιάζουν κάποια από τα μειονεκτήματα που συναντούμε σε αυτά. Για την υλοποίηση της δικής μας εφαρμογής ελέγχου της χρονικής ακεραιότητας, επιλέξαμε τη μεθοδολογία Scenario Integrity Checking.

Η μεθοδολογία στηρίζεται στο μοντέλο αναπαράστασης IMD. Επομένως μπορεί να χειριστεί preorchistrated αλλά και interactive multimedia εφαρμογές. Για τον έλεγχο της χρονικής ακεραιότητας ενός σεναρίου η μεθοδολογία χρησιμοποιεί το μοντέλο των δικτύων χρονικών περιορισμών. Σύμφωνα με το μοντέλο αυτό, κάθε πρόβλημα που περιλαμβάνει χρονικές ανισότητες μπορεί να εκφραστεί με ένα δίκτυο χρονικών περιορισμών. Στη συνέχεια το δίκτυο αυτό μπορεί να εξεταστεί για να εντοπιστούν χρονικές ασυνέπειες. Η μεθοδολογία προτείνει ένα πλήρες σύνολο κανόνων για την μετάφραση ενός IMD σεναρίου σε ένα πλήθος δικτύων χρονικών περιορισμών. Αρχικά κάθε tuple του σεναρίου μεταφράζεται σε έναν αριθμό επιμέρους δίκτυο χρονικών περιορισμών. Στη συνέχεια, χρησιμοποιώντας τα διαφορετικά μονοπάτια εκτέλεσης, τα επιμέρους δίκτυα συνθέτονται για να σχηματίσουν τα συνολικά δίκτυα χρονικών περιορισμών της εφαρμογής.

Η μεθοδολογία χωρίζεται σε τέσσερα διακριτά στάδια. Τα τρία πρώτα στάδια της μεθοδολογίας ασχολούνται με την κατασκευή των συνολικών δικτύων χρονικών περιορισμών. Κατά τη διαδικασία κατασκευής εφαρμόζονται έλεγχοι για τον εντοπισμό χρονικών λαθών που σχετίζονται με μεμονωμένα αντικείμενα της εφαρμογής.

Το τελικό στάδιο της μεθοδολογίας έχει δύο στόχους. Χρησιμοποιώντας την θεωρία των δικτύων χρονικών περιορισμών μπορεί να εντοπίσει χρονικά λάθη που αφορούν τη συνολική ακεραιότητα της εφαρμογής. Επιπρόσθετα, κατασκευάζει μια νέα

χρονική αναπαράσταση της εφαρμογής που καλείται ελάχιστο δίκτυο. Το ελάχιστο δίκτυο προσδιορίζει το ελάχιστο και το μέγιστο χρονικό διάστημα που μπορεί να μεσολαβήσει μεταξύ δύο οποιονδήποτε γεγονότων ώστε η εφαρμογή να μην παρουσιάζει χρονικές ασυνέπειες.

Τα βασικά πλεονεκτήματα της μεθοδολογίας Scenario Integrity Checking είναι τα ακόλουθα:

- Η μεθοδολογία στηρίζεται στο μοντέλο των δικτύων χρονικών περιορισμών. Το μοντέλο παρουσιάστηκε το 1991 από τους Dechter, Meiri και Pearl, στηρίζεται σε ένα ισχυρό θεωρητικό υπόβαθρο και έχει γίνει αποδεκτό από την επιστημονική κοινότητα.

- Ο έλεγχος για τον εντοπισμό των χρονικών ασυνεπειών πραγματοποιείται σε τρία διαφορετικά επίπεδα: ακεραιότητα αντικειμένων, ακεραιότητα μεμονωμένων tuples, συνολική ακεραιότητα σεναρίου. Τα λάθη που εντοπίζονται στα δύο πρώτα επίπεδα, συνοδεύονται από μια πλήρης επεξήγηση του αιτίου που τα προκαλεί, δίνοντας τη δυνατότητα στο συγγραφέα να τα κατανοήσει και να τα διορθώσει.

- Η μεθοδολογία στηρίζεται στο μοντέλο IMD που παρέχει πλούσιες δυνατότητες χειρισμού user interaction.

Η υλοποίηση του μοντέλου Scenario Integrity Checking, που αποτελεί και τον κορμό αυτής της εργασίας, παρουσιάζει τα εξής πλεονεκτήματα:

- Στηρίζεται σε ένα object-oriented μοντέλο, γεγονός που κάνει την εφαρμογή εύκολα επεκτάσιμη.

- Παρέχει έναν μεταγλωττιστή της scripting γλώσσας του μοντέλου IMD. Ο μεταγλωττιστής παρέχει τη δυνατότητα σε κάθε χρήστη να χρησιμοποιήσει την εφαρμογή, έχοντας στα χέρια του μόνο ένα απλό αρχείο κειμένου, που περιγράφει το σενάριο της multimedia εφαρμογής.

- Υιοθετεί ένα Windows 95 user interface.

- Παρέχει έναν πλήρη μηχανισμό αναφοράς των λαθών που εντοπίζονται κατά την εκτέλεση των τεσσάρων σταδίων της μεθοδολογίας Scenario Integrity Checking.

Η εφαρμογή ωστόσο δεν μπορεί να θεωρηθεί ακόμα πλήρης. Παρουσιάζει κάποιους περιορισμούς, όσον αφορά την υλοποίηση του μοντέλου IMD. Επίσης είναι απαραίτητο να ελεγχθεί η λειτουργία της, χρησιμοποιώντας όσο το δυνατόν πιο πολύπλοκα σενάρια ώστε να εξασφαλιστεί η αξιόπιστη λειτουργία της σε όλες τις περιπτώσεις.

Οι μελλοντικές επεκτάσεις της εφαρμογής αφορούν δύο βασικές κατευθύνσεις. Την αύξηση της πολυπλοκότητας των περιορισμών και την υλοποίηση ενός μηχανισμού ερωτήσεων που αφορούν τα αντικείμενα της εφαρμογής.

Όσον αφορά την πολυπλοκότητα των περιορισμών οι προτάσεις είναι πολυάριθμες. Τα γεγονότα μπορούν να παρουσιάζουν διαφορετική σημασία, ανάλογα με την κατάσταση στην οποία βρίσκεται η εφαρμογή αλλά και την χωρική τους υπογραφή. Για παράδειγμα, το γεγονός *mouse-click* έχει διαφορετική σημασία αν συνδεθεί με ένα κουμπί και διαφορετική σημασία αν συνδεθεί με μία εικόνα. Θα ήταν χρήσιμο να εισάγουμε στα επιμέρους δίκτυα χρονικών περιορισμών τη σημασιολογία που μπορούν να έχουν τα γεγονότα. Επίσης θα ήταν χρήσιμο να μπορεί ο ίδιος ο χρήστης να εισάγει πρόσθετους περιορισμούς που να αφορούν τα γεγονότα της εφαρμογής. Πρόκειται για περιορισμούς που δεν προκύπτουν από τους βασικούς κανόνες της μεθοδολογίας και επομένως δεν εμφανίζονται στα δίκτυα χρονικών περιορισμών.

Το τέταρτο στάδιο της μεθοδολογίας Scenario Integrity Checking καταλήγει στην κατασκευή του ελάχιστου δικτύου για κάθε μονοπάτι εκτέλεσης. Το ελάχιστο δίκτυο μπορεί να χρησιμοποιηθεί σε συνδυασμό με τα δίκτυα χρονικών περιορισμών για να τεθούν από το συγγραφέα ερωτήσεις που αφορούν τα γεγονότα της εφαρμογής. Σκοπός των ερωτήσεων αυτών είναι να προσδιοριστούν οι χρονικοί περιορισμοί που συνδέουν τα γεγονότα. Στη συνέχεια ο συγγραφέας μπορεί να μεταβάλει κάποια από τα χαρακτηριστικά του σεναρίου για να βελτιώσει την χρονική συνέπειά του. Φυσικά μετά από κάθε διόρθωση είναι απαραίτητο να επαναλαμβάνεται ο έλεγχος της χρονικής ακεραιότητας ώστε να εξασφαλιστεί ότι δεν δημιουργούνται πρόσθετα προβλήματα από τις τροποποιήσεις που έγιναν.

ΠΑΡΑΡΤΗΜΑ Α

Αποδείξεις Θεωρημάτων

Στο παράρτημα αυτό περιέχονται οι αποδείξεις των θεωρημάτων 3.3 και του πορίσματος 3.5 που κρίθηκαν αρκετά πολύπλοκες για να περιληφθούν στο κυρίως κεφάλαιο.

Α.1 Απόδειξη θεωρήματος 3.3

Θεώρημα 3.3 (Αποσύνθεση STP προβλήματος) *Ένα συνεπές χρονικό πρόβλημα, T, μπορεί να αποσυντεθεί χρησιμοποιώντας τις ανισότητες που εκφράζονται στο διάγραμμα αποστάσεων που το αναπαριστά.*

Απόδειξη 3.3 Αρκεί να αποδείξουμε ότι οποιοδήποτε υποσύνολο *S* *k* μεταβλητών ($1 \leq k \leq n$) που ικανοποιεί όλους τους περιορισμούς που τίθενται από τα συντομότερα μονοπάτια μεταξύ των μεταβλητών του συνόλου, μπορεί να επεκταθεί σε οποιαδήποτε άλλη μεταβλητή του δικτύου. Θα χρησιμοποιήσουμε την μέθοδο της επαγωγής.

Αρχικά υποθέτουμε ότι $k = 1$. Το σύνολο *S* περιέχει μία και μόνο μεταβλητή X_i που έστω ότι έχει τιμή x_i. Θα αποδείξουμε ότι για οποιαδήποτε άλλη μεταβλητή X_j μπορούμε να βρούμε μια τιμή *u* που να ικανοποιεί τον περιορισμό που τίθεται από το συντομότερο μονοπάτι που συνδέει τις δύο μεταβλητές. Η τιμή *u* πρέπει να ικανοποιεί τη σχέση:

$$-d_{ji} \leq u - x_i \leq d_{ij} \qquad \text{(A.1)}$$

Εφόσον ο γράφος απόστασης είναι συνεπής δεν περιέχει αρνητικούς κύκλους. Επομένως ισχύει $d_{ji} + d_{ij} \geq 0$ και επομένως υπάρχει μια τιμή *u* που να ικανοποιεί τη σχέση Α.1 αρκεί $u \geq x_i$. Άρα για $k = 1$ η υπόθεσή μας ισχύει.

Υποθέτουμε ότι ισχύει για k -1 μεταβλητές. Θα αποδείξουμε ότι ισχύει και για k μεταβλητές. Υποθέτουμε ότι οι k μεταβλητές του συνόλου S είναι οι μεταβλητές X_1 έως X_k και έστω ότι $X_i = x_i$ είναι μια ανάθεση που ικανοποιεί τον περιορισμό του συντομότερου μονοπατιού μεταξύ των μεταβλητών του συνόλου S. Έστω μια μεταβλητή X_{k+1} που δεν ανήκει στο σύνολο S. Πρέπει να βρούμε μια τιμή u που να ικανοποιεί τους περιορισμούς του συντομότερου μονοπατιού μεταξύ της μεταβλητής και όλων των μεταβλητών του συνόλου S. Με άλλα λόγια η τιμή u πρέπει να ικανοποιεί τις σχέσεις:

$$u - x_i \le d_{i,\,k+1} \tag{A.2}$$
$$x_i - u \le d_{i,\,k+1,\,i} \tag{A.3}$$

για $i = 1, 2, \ldots, k$ ή αν το εκφράσουμε με διαφορετικό τρόπο:

$$u \le \min\{x_i + di_{,\,k+1} |\ 1 \le i \le k\} \tag{A.4}$$
$$u \ge \max\{x_i - di_{,\,k+1} |\ 1 \le i \le k\} \tag{A.5}$$

Ας υποθέσουμε ότι η ελάχιστη τιμή λαμβάνεται για την τιμή n ενώ η μέγιστη για την τιμή m. Επομένως το u πρέπει να ικανοποιεί τη σχέση:

$$x_m - d_{k+1,\,m} \le u \le x_n + d_{n,\,k+1} \tag{A.6}$$

Εφόσον οι τιμές m και n ικανοποιούν τον περιορισμό μεταξύ τους, έχουμε:

$$x_m - x_n \le d_{n,m} \tag{A.7}$$

Η σχέση αυτή μαζί με τη σχέση $d_{n,m} \le d_{n,k+1} + d_{k+1,\,m}$ οδηγούν στη σχέση:

$$x_m - d_{k+1,\,m} \le x_n - d_{n,k+1} \tag{A.8}$$

Επομένως υπάρχει μια τιμή u που να ικανοποιεί τη σχέση Α.6. □

Α.2 Απόδειξη πορίσματος 3.5

Πόρισμα 3.5 *Έστω ένα συνεπές STP πρόβλημα, Τ. Ο γράφος αποστάσεων οι αποστάσεις του οποίου ορίζονται από τη σχέση:*

$$\forall\ i,j,\ M_{ij} = \{[-d_{ij},\ d_{ij}]\} \tag{A.9}$$

αποτελεί το ελάχιστο δίκτυο του προβλήματος Τ.

Απόδειξη 3.5 Θα αποδείξουμε ότι το σύνολο Μ είναι το ελάχιστο δίκτυο αποδεικνύοντας ότι δεν υπάρχει υποσύνολο του Μ που να ικανοποιεί τα

χαρακτηριστικά του. Με άλλα λόγια ξεκινώντας από την ανάθεση $X_0 = 0$, για κάθε $d \in [-d_{ji}, d_{ij}]$ υπάρχει μια λύση $X = \{x_0, x_1, ..., x_n\}$στην οποία ισχύει $x_j - x_i = d$. Διακρίνουμε δύο περιπτώσεις:

Περίπτωση 1: $d \leq d_{0j} - d_{0i}$ (Α.10)

Σύμφωνα με το πόρισμα 3.4, $X_i = d_{0i}$ είναι μια πιθανή τιμή. Επομένως είναι φανερό ότι ισχύει:

$$d_{oi} + d \geq d_{oi} - d_{ij} \quad \text{(Α.11)}$$

και εφόσον

$$d_{ji} \leq d_{j0} + d_{0i} \quad \text{(Α.12)}$$

έχουμε

$$d_{0i} + d \geq -d_{j0} \quad \text{(Α.13)}$$

Σε συνδυασμό με τη σχέση Α.10 έχουμε

$$-d_{j0} \leq d_{0i} + d \leq d_{0j} \quad \text{(Α.14)}$$

Επομένως η ανάθεση $X_j = d_{0i} + d$ ικανοποιεί τους περιορισμούς που ισχύουν για την μεταβλητή X_j, ενώ το σύνολο

$$\{X_0 = 0, X_i = d_{0i}, X_j = d_{0i} + d\} \quad \text{(Α.15)}$$

ικανοποιεί τους περιορισμούς που ισχύουν για τις μεταβλητές $\{X_0, X_i, X_j\}$. Από το θεώρημα 3.3 συμπεραίνουμε ότι η σχέση Α.15 μπορεί να επεκταθεί σε μια λύση του προβλήματος.

Περίπτωση 2: $d \geq d_{0j} - d_{0i}$ (Α.16)

Σύμφωνα με το πόρισμα 3.4, η τιμή $X_j = d_{0j}$ αποτελεί μια πιθανή λύση. Επομένως ισχύει

$$d_{0j} - d \geq d_{0j} - d_{ij} \quad \text{(Α.17)}$$

και εφόσον

$$d_{ij} \leq d_{i0} + d_{0j} \quad \text{(Α.18)}$$

έχουμε

$$d_{0j} - d \leq -d_{i0} \quad \text{(A.19)}$$

Σε συνδυασμό με τη σχέση Α.16 έχουμε

$$-d_{i0} \leq d_{0j} - d \leq d_{0i} \quad \text{(A.20)}$$

Επομένως, η ανάθεση $Xi = d_{0j} - d$ ικανοποιεί τους περιορισμούς που ισχύουν για την μεταβλητή X_i. Επιπρόσθετα το σύνολο

$$\{X_0 = 0, X_i = d_{0j} - d, X_j = d_{0j}\} \quad \text{(A.21)}$$

ικανοποιεί τους περιορισμούς που ισχύουν για τις μεταβλητές $\{X_0, X_i, X_j\}$. Από το θεώρημα 3.3 συμπεραίνουμε ότι η σχέση Α.21 μπορεί να επεκταθεί σε μια λύση του προβλήματος. □

Βιβλιογραφία

[All83] Allen J. F. *Maintaining Knowledge about Temporal Intervals*. Communications of the ACM, Volume 26 (11), November 1983.

[AS80] Aspvall B. and Shiloach Y. *A Polynomial Time Algorithm for Solving Systems of Linear Inequalities With Two Variables per Inequality*. SIAM Journal of Computation, Vulume 9 (4), pages 827 - 845, 1980.

[ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey Ullman. *Compilers Principles, Techniques, and Tools*. Computer Science, Addison Wesley, second edition, 1986.

[BCTP94] Brusoni V., Console L., Terenziani P., and Pernici B.*LaTeR: a general purpose manager of temporal information*. In Methodologies for Intelligent Systems 8, pages 255-264Springler Verlag, 1994.

[BCTP97] Brusoni V., Console L., Terenziani P., and Pernici B. *Qualitative and Quantitative Temporal Constraints and Relational Databases: Theory, Architecture, and Applications*.

[BCTPl97] Brusoni V., Console L., Terenzianni P., and Pernici B. *Later: Managing Temporal Information Efficiently*. IEEE Expert, Volume 12, Number 4, July/August 1997, pages 56-64.

[Blak96] Blakowski G. *A Media Synchronization Survey: Reference Model, Specification, and Case Studies*. IEEE Journal on Selected Areas in Communications, Volume 14 (1), pages 5-35, January 1996.

[Buf96] Buford J. Evaluating HyTime: *An Examination and Implementation Experience*. In proceedings of ACM Hypertext '96 Conference.

[BZ95] Buchanan M.C, Zellweger P. T. *Automatically Generating Consistent Schedules for Multimedia Documents*. ACM Multimedia Systems Journal, Volume 1 (2), pages 55.67.

[CM93] Chakravarthy S., Mishra D. *Snoop: An Expressive Event Specification Language for Active Databases*. Technical report, UF-CIS-TR-93-00, University of Florida, 1993.

[CO96] Courtiat J. P., De Oliveira R. C. *Proving Temporal Consistency in a new Multimedia Synchronization Model*. In the proceedings of ACM Multimedia 1996 Conference.

[CT95] Chomicky J. and Toman D. *Implementing Temporal Integrity Constraints using an Active DBMS*. IEEE Transactions on Knowledge and Data Engineering, Volume 7 (4), pages 566-581, 1995.

[CY90] Coad P. and Yourdon E.*Object Oriented Analysis*. Prentice Hall International, Second Edition, 1991.

[Dan62] Dantzig G. B. *Linear Programming and Extensions*. Princeton University Press, 1962.

[DK95] Duda A. and Keramana C. *Structured Temporal Composition of Multimedia Data*. In proceedings of 1st IEEE International Workshop for MM-DBMS, 1995.

[DMP91] Dechter R. Meiri I., and Pearl J. *Temporal Constraint Networks*. Artificial Inteligence, Volume 49, 1991, pages 61-95.

[DP87] Dechter R. and Pearl J. *Nework-Based Heuristics for Constraint Satisfaction Problems*. Artificial Intelligence, Volume 34 (1), pages 1-38, 1987.

[Dro95] Adam Drodzec. *Data Structures and Algorithms in C++*. Computer Science, PWS Publishing, 1995.

[Ev79] Even S. *Graph Algorithms*. Computer Science Press, Rockville, MD, 1979.

[Freu78] Freuder E. C. *Synthesizing Constraint Expressions*. Communications of the ACM, Volume 21, Number 11, pages 958-965, 1978.

[GJS92] Gehan N. H., Jagadish H. V., and Shmueli O. *Composite Event Specification in an Active Database: Model and Implementation*. Proceedings of VLDB conference, 1992, pages 327-338.

[Ham72] Hamblin C. *Instants and Intervals, the Study of Time*.Springler-Verlag, 1972.

[Han96] Handl M. *A New Multimedia Synchronization Model*. IEEE Journal on Selected Areas in Communications, Volume 14 (1), pages 73-83, January 1996.

[HFK95] Hirzalla N., Falchuk B., and Karmouch A. *A Temporal Model for Interactive Multimedia Scenarios*. IEEE Multimedia, Fall 1995, pages 24-31.

[ISO92] International Standard Organization. *HyperMEdia Time-Based Document Structuring Language (HyTime)*. ISO/IEC IS 10744, April 1992.

[ISO93] International Standard Organization. *Coded Representation of Multimedia and HyperMedia Information Objects (MHEG)*. ISO/IEC IS 10031, June 1993.

[Kha79] Khachiyan L. G. *A Polynomial algorithm in Linear Programming*. Soviet Math. Dokl. 20, pages 191-194, 1979.

[Kin94] P. R. King. Towards a Temporal Logic Based Formalism for Expressing Temporal Constraints in Multimedia Documents. Technical Report 942, LRI, Universite de Paris-Sud, Orsay, France, December 1995.

[Lay95] Layaida N. and C. *Maintening Temporal Consistency of Multimedia Documents*. In proceedings of the ACM Workshop on Effective Abstractions in Multimedia, San Fransisco, November 1995.

[LG93] Litlle T. D. C., Ghafoor A. *Interval-Based Conseptual Models for Time-Dependent Multimedia Data*. IEEE Transactions on Data and Knowledge Engineering, Volume 5, Number 4, August 1993, pages 551-563.

[LP98] Harry R. Lewis, Christos H. Papadimitriou. *Elements of the theory of computation*. Prentice Hall International, 1998.

[LS83] Leiserson C. E. Saxe J. B. *A Mixed-Integer Linear Programming Problem which is Efficiently Solvable*. In prodeedings of the 21st Annual Allerton Conference on Communications, Control, and Computing, pages 204-213, 1983.

[LW83] Liao Y. Z. and C. K. Wong. *An Algorithm to Compact a VLSI Symbolic Layout With Mixed Constraints*. Computer Aided Design of Integrated Circuits and Systems. Volume 2 (2), pages 62-69, 1983.

[Mac77] Mackworth A .K. *Consistency in Networks of Relations.* Artificial Intelligence Journal, Volume 8 (1), pages 99-118, 1977.

[Mei91] Meiri I. *Combining Qualitative and Quantitative Constraints in Temporal Reasoning*. In proceedings of AAAI'91, pages 260-267, 1991.

[Mon74] Montanari U. *Networks of Constraints:Fundamental properties and applications to picture processing*.Information and Science, Volume 7, pages 95-132, 1974.

[Mor97] Bernard M. Moret. *The Theory of Computation*. Computer Science. Addison Wesley, 1997.

[MPSV98] Mirbel I., Pernici B., Sellis T., and Vazirgiannis M. *Integrity Constraints for Interactive Multimedia Scenarios*.

[PR94] Prabhakaran B. and S. V. Raghavan. *Synchronization Models for Multimedia Presentation with User Participation*. ASM/Springler-Verlag Journal of Multimedia Systems, Anaheim, California, pages 157-166, August 1993.

[SD96] Schnepf J. and Du C. *Doing FLIPS: Flexible Interactive Presentation Synchronization*. IEEE Journal on Selected Areas in Communications, Volume 14 (1), pages 114-125, January 1996.

[Sho81] Shostak R. *Deciding Linear Inequalities by Computing Loop Residues*. ACM Journal, Volume 28, Number 4, pages 769 - 779, 1981.

[SW95] Schloss G. and Wynblatt M. *Providing Definition and Temporal Structure from Mulimedia Data*. Multimedia Systems Journal, Volume 3, pages 264 - 277, 1995.

[Tar81] R. E. Tarjan. *Fast algorithms for solving path problems*. ACM Journal Volume 28, Number 3, pages 566-579, 1984.

[Vaz96] Vazirgiannis M. *Multimedia Data Base Object and Application Modeling Issues and an Object Oriented Model* in *Multimedia Database Systems: Design and Impelmentation Strategies (editors Kingsley C. Nwosu, Bhavani Thuraisingham and P. Bruce Berra)*. Kluwer Academic Publishers, 1996, Pages 208-250.

[VB97] Vazirgiannis M. and Boll Suzanne. Events in Interactive Multimedia Applications: Modeling and Implementation Design. In the proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS'97), Ottawa, Canada, June 1997.

[VH93] Vazirgiannis M. and Hatzopoulos M. *A Script Based Approach for*

Interactive Multimedia Applications. In proceedings of the International Conference on Multimedia Modeling, Singapore, November 1993.

[VH95] Vazirgiannis M. and Hatzopoulos M. *Integrated Multimedia Object and Application Modeling Based on Events and Scenarios*. In proceedings of 1st IEEE International Workshop for MMDBMSs, Blue Mountain Lake, August 1995.

[VKV89] Vilain M., Kautz H., and VanBeek P. *Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report*. In D.S Weld and J. de Kleer, editors, *Reading in Qualitative Reasoning about Physical Systems*, pages 373-381. Morgan Kaufmann, 1989.

[VK86] Vilain M. and Kautz H. *Constraint Propagation Algorithms for Temporal Reasoning*. In proceedings AAAI 86, pages 377-382, 1986.

[VM93] Vazirgiannis M. and Mourlas C. *An Object Oriented Model for Interactive Multimedia Applications*. The Computer Journal, British Computer Society, Volume 36 (1), 1993.

[VS96] Vazirgiannis M. and Sellis T. *Event and Action Representation and Composition for Multimedia Application Scenario Modeling*, 1996.

[VTMS98] Vazirgiannis M., Tsirikos D. Markousis T., Trafalis M., Stamati Y., Hatzopoulos M., Sellis T. *On Interactive Multimedia Dosument Scenario Modeling Authoring and Rendering*.

[VTS98] Vazirgiannis M., Theodoridis Y., and Sellis T. *Spatio-Temporal Composition and Indexing for Large Multimedia Applications*. To appear in ACM/Springer-Verlag Multimedia Systems Journal, 1998.

[VTSH98] Vazirgiannis M., Trafalis M., Stamati Y., Hatzopolulos M. *Interactive Multimedia Scenario: Modeling and Rendering*. To appear in the proceedings of the Multimedia track of ACM-SAC'98 conference.

Περίληψη

Ο χώρος των εφαρμογών multimedia γνωρίζει πολύ μεγάλη άνθηση την τελευταία δεκαετία. Έχουν παρουσιαστεί πολλά εμπορικά πακέτα για την ανάπτυξη εφαρμογών multimedia και ακόμα περισσότερες εφαρμογές. Κοινό χαρακτηριστικό των εφαρμογών που κατασκευάζονται είναι η πλήρης εκμετάλλευση των διαφορετικών τύπων multimedia αντικειμένων (ήχου, εικόνας, κινούμενης εικόνας, video, κειμένου κ.ο.κ) αλλά και η ελλιπής υποστήριξη αλληλεπίδρασης μεταξύ των αντικειμένων. Επιπρόσθετα η ροή εκτέλεσης της εφαρμογής είναι αυστηρά καθορισμένη και οι δυνατότητες που δίνονται στο χρήστη να επέμβει σε αυτή είναι περιορισμένες. Συνήθως παρέχεται στον χρήστη ένα σύνολο κουμπιών μέσω των οποίων μπορεί απλά να παρέμβει στη σειρά με την οποία εμφανίζονται οι οθόνες της εφαρμογής.

Βασική αιτία για αυτές τις αδυναμίες των συνηθισμένων εφαρμογών multimedia είναι η έλλειψη ενός μοντέλου που θα υποστηρίζει την αλληλεπίδραση των αντικειμένων που συμμετέχουν στην εφαρμογή, αλλά και θα περιγράφει τον τρόπο με τον οποίο επιδρούν οι ενέργειες του χρήστη ή άλλα εξωτερικά γεγονότα στην ροή εκτέλεσής της. Τα τελευταία χρόνια έχουν παρουσιαστεί αρκετά μοντέλα που προσπαθούν να καλύψουν αυτές τις αδυναμίες, με στόχο να αυξήσουν την λειτουργικότητα των εφαρμογών multimedia και να δώσουν τη δυνατότητα στους χρήστες να επεμβαίνουν στη ροή εκτέλεσής τους. Ένα μοντέλο που περιγράφει μια εφαρμογή multimedia, που περιλαμβάνει αλληλεπίδραση με το χρήστη, θα πρέπει να ικανοποιεί απαραίτητα τα τρία παρακάτω χαρακτηριστικά:

- ακριβής περιγραφή των αντικειμένων που συμμετέχουν στην εφαρμογή των μετασχηματισμών τους καθώς και των χρονικών και χωρικών σχέσεων που τα συνδέουν
- ακριβής περιγραφή της λειτουργικότητας της εφαρμογής, δηλαδή της ροής εκτέλεσής της και της επίδρασης των γεγονότων σε αυτή
- υποστήριξη σύνθετων γεγονότων

Τα μοντέλα που έχουν αναπτυχθεί ικανοποιούν κάποια μόνο από τα παραπάνω χαρακτηριστικά. Το κοινό τους χαρακτηριστικό είναι ότι περιγράφουν με ακρίβεια τα αντικείμενα που συμμετέχουν σε μια εφαρμογή αλλά και τη λειτουργικότητα της εφαρμογής. Ωστόσο τα περισσότερα μοντέλα δεν υποστηρίζουν σύνθετα γεγονότα ούτε

καθορίζουν την επίδραση των ενεργειών του χρήστη στη λειτουργικότητα της εφαρμογής.

Το μοντέλο στο οποίο στηρίζεται η εργασία ικανοποιεί σε μεγάλο βαθμό και τις τρεις βασικές προϋποθέσεις που προαναφέρθηκαν. Πρόκειται για το μοντέλο Interactive Multimedia Document (IMD scenario model). Το μοντέλο υιοθετεί μια scripting γλώσσα για να ορίσει τα χαρακτηριστικά των αντικειμένων που συμμετέχουν σε μία εφαρμογή, τα γεγονότα τα οποία αναγνωρίζει και τον τρόπο με τον οποίο επιδρούν στη ροή εκτέλεσής της. Επιπρόσθετα παρέχει ένα σύνολο τελεστών που ορίζουν τον τρόπο με τον οποίο γίνεται η χωροχρονική σύνθεση των αντικειμένων και των γεγονότων της εφαρμογής. Το βασικό χαρακτηριστικό όμως του μοντέλου είναι η πλήρης υποστήριξη αλληλεπίδρασης με το χρήστη, που βασίζεται στην χρήση των γεγονότων (events).

Τα γεγονότα προκαλούνται από εσωτερικές ή εξωτερικές ενέργειες και χρησιμοποιούνται για να καθορίσουν την ροή εκτέλεσης εξέλιξη της εφαρμογής. Με τον όρο εξωτερικές ενέργειες, δεν εννοούμε μόνο το συνηθισμένο πάτημα ενός πλήκτρου του πληκτρολογίου ή ενός από τα κουμπιά που εμφανίζονται στην οθόνη της εφαρμογής. Ο όρος καλύπτει ενέργειες όπως scrolling, drag and drop καθώς και ενέργειες που επιδρούν άμεσα στην κατάσταση ενός αντικειμένου της εφαρμογής. Από την άλλη πλευρά οι εσωτερικές ενέργειες παρέχουν τη δυνατότητα ορισμού της αλληλεπίδρασης μεταξύ των αντικειμένων της εφαρμογής. Κάθε γεγονός συνδέεται με ένα από τα αντικείμενα που παρουσιάζονται κατά την εκτέλεση της εφαρμογής και ονομάζονται στην ορολογία του μοντέλου IMD, *actors*.

Ένα αντικείμενο μιας εφαρμογής μπορεί να βρίσκεται σε μία από τρεις διαφορετικές καταστάσεις εκτέλεσης: *Active*, *Idle*, *Paused-Suspended*. Η μετάβαση των αντικειμένων από τη μία κατάσταση στην άλλη γίνεται μέσω των ενεργειών (*actions*). Το μοντέλο ορίζει ένα σύνολο ενεργειών καθώς και τις μεταβάσεις που αυτές προκαλούν.

Η σύνδεση της λειτουργικότητας των τριών βασικών εννοιών του IMD σεναρίου (γεγονότα, αντικείμενα, ενέργειες) γίνεται μέσω των tuples του σεναρίου. Μια IMD εφαρμογή περιγράφεται από ένα σενάριο. Το σενάριο καθορίζει τον τρόπο με τον οποίο μεταβάλλεται η ροή εκτέλεσης της εφαρμογής όταν δέχεται τα γεγονότα που προκαλούνται κατά την παρουσίαση των αντικειμένων. Με τον τρόπο αυτό το σενάριο ορίζει τη ροή εκτέλεσης της εφαρμογής.

Κάθε σενάριο αποτελείται από μια ομάδα αυτόνομων τμημάτων που ονομάζονται scenario tuples. Ένα tuple μπορεί να θεωρηθεί σαν ένα σύνθετο αντικείμενο που εκκινήται και σταματά με την εμφάνιση γεγονότων. Με την εκκίνηση ενός tuple εκτελείται μια λίστα ενεργειών που ορίζονται σε αυτό. Η ίδια η εκκίνηση και η διακοπή της εκτέλεσης ενός tuple προκαλούν γεγονότα που καλούνται γεγονότα συγχρονισμού.

Το μοντέλο IMD χρησιμοποιείται για την περιγραφή μιας multimedia εφαρμογής. Η διαδικασία κατά την οποία ο συγγραφέας μιας εφαρμογής χρησιμοποιεί το μοντέλο για να ορίσει τον τρόπο με τον οποίο εκτελείται η εφαρμογή ονομάζεται συγγραφή και αποτελεί το αρχικό στάδιο της διαδικασίας ανάπτυξης.

Ένα από τα προβλήματα που παρουσιάζονται κατά την διαδικασία συγγραφής αφορά την χρονική ακεραιότητα της εφαρμογής που περιγράφεται. Ειδικά όταν

πρόκειται για εφαρμογές οι οποίες περιλαμβάνουν αλληλεπίδραση με το χρήστη το πρόβλημα γίνεται περισσότερο πολύπλοκο αφού δεν είναι δυνατόν να προσδιοριστούν οι χρονικές στιγμές στις οποίες θα πραγματοποιηθούν οι ενέργειες του χρήστη, ούτε να εξεταστούν όλα τα πιθανά μονοπάτια εκτέλεσης της εφαρμογής. Το αποτέλεσμα είναι ότι υπάρχει η πιθανότητα, κατά την εκτέλεση της εφαρμογής, να παρουσιαστεί ανεπιθύμητη συμπεριφορά από ορισμένα από τα αντικείμενα της εφαρμογής, ειδικά αν πρόκειται για αντικείμενα που συνδέονται με χρονικούς περιορισμούς.

Το πρόβλημα της χρονικής ακεραιότητας των multimedia εφαρμογών σχετίζεται εν' μέρει με τη φιλοσοφία με την οποία αντιμετωπίζουν οι συγγραφείς τη διαδικασία ανάπτυξης. Ο συγγραφέας συγκεντρώνει κάθε φορά την προσοχή του σε ξεχωριστά τμήματα της παρουσίασης. Το αποτέλεσμα είναι ότι δεν έχει μια συνολική εικόνα της παρουσίασης. Έτσι ενώ τα επιμέρους τμήματα είναι χρονικά συνεπή είναι πιθανόν η εφαρμογή στο σύνολό της να παρουσιάζει χρονικές ασυνέπειες.

Οι ασυνέπειες αυτές δεν γίνονται αντιληπτές παρά μόνο κατά την εκτέλεση της εφαρμογής: Κάποια τμήματα της παρουσίασης δεν εμφανίζονται στην οθόνη, κάποια από τα αντικείμενα έχουν λανθασμένη συμπεριφορά ή οι ενέργειες του χρήστη δεν έχουν καμία επίδραση στην ροή εκτέλεσης της εφαρμογής.

Σκοπός της παρούσας εργασίας είναι η υλοποίηση ενός εργαλείου που θα αναλαμβάνει τον έλεγχο της χρονικής ακεραιότητας ενός σεναρίου. Το εργαλείο χρησιμοποιείται κατά τη διαδικασία συγγραφής, δέχεται ένα IMD σενάριο, το αναλύει και εντοπίζει πιθανές χρονικές ασυνέπειες. Η χρησιμότητα ενός εργαλείου ελέγχου της χρονικής ακεραιότητας δεν γίνεται τόσο φανερή αν πρόκειται για απλές εφαρμογές, με μικρό αριθμό αντικειμένων και περιορισμένη αλληλεπίδραση με το χρήστη. Ο συγγραφέας μπορεί να εντοπίσει τα διαφορετικά μονοπάτια εκτέλεσης της εφαρμογής και να ελέγξει την χρονική τους συνέπεια. Σε πολύπλοκες όμως εφαρμογές είναι δύσκολο για τον συγγραφέα να εντοπίσει όλα τα πιθανά μονοπάτια εκτέλεσης και ακόμα πιο δύσκολο να ελέγξει με αξιοπιστία την συνέπειά τους. Είναι επομένως απαραίτητο να παρέχεται στον συγγραφέα ένα αξιόπιστο εργαλείο με το οποίο θα μπορεί να αποφασίσει την χρονική συνέπεια της εφαρμογής που σχεδιάζει ή των εφαρμογών που ήδη έχει αναπτύξει.

Ο χώρος των χρονικών προβλημάτων αποτελεί έναν από τους πιο ενδιαφέροντες τομείς της τεχνητής νοημοσύνης. Τα ζητήματα που ανήκουν στο χώρο αυτό χαρακτηρίζονται από αυξημένη πολυπλοκότητα. Ο έλεγχος της χρονικής ακεραιότητας, τον οποίο υλοποιεί η εφαρμογή που κατασκευάσαμε, στηρίζεται σε μία μεθοδολογία Scenario Integrity Checking (SIC) που έχει προταθεί από τους Mirbel, Pernici, Selli και Vazirgianni και χρησιμοποιεί το μοντέλο IMD για να μοντελοποιήσει interactive multimedia εφαρμογές.

Η μεθοδολογία Scenario Integrity Checking έχει σαν σκοπό τον εντοπισμό χρονικών ασυνεπειών, κατά τη διαδικασία της συγγραφής του σεναρίου. Χρησιμοποιώντας τη μεθοδολογία ο συγγραφέας μπορεί να αποφασίσει αν η σειρά των ενεργειών που προσδιορίζουν τα tuples του σεναρίου είναι συνεπής, έχοντας σαν δεδομένο ένα σύνολο χρονικών περιορισμών που αφορούν τα αντικείμενα της παρουσίασης.

Αυτοί οι περιορισμοί μπορούν είτε να έχουν γενικό χαρακτήρα (για παράδειγμα "... ένα αντικείμενο που βρίσκεται σε κατάσταση Idle δεν μπορεί να δεχτεί ενέργεια Stop") είτε να αφορούν τη συγκεκριμένη παρουσίαση (για παράδειγμα "... η παρουσίαση του video A δεν θα πρέπει να συμπίπτει χρονικά με την παρουσίαση του ήχου B " ή "... δεν επιτρέπεται να υπάρχουν δύο ήχοι που να βρίσκονται ταυτόχρονα σε κατάσταση Idle").

Η μεθοδολογία SIC έχει τα εξής βασικά χαρακτηριστικά:

- Βασίζεται στον μετασχηματισμό του σεναρίου σε ένα δίκτυο το οποίο αναπαριστά τους χρονικούς περιορισμούς μεταξύ των αντικειμένων της εφαρμογής. Η διαδικασία στηρίζεται σε ένα σύνολο κανόνων για την κατασκευή του δικτύου χρονικών περιορισμών (*Temporal Constraint Network*). Κατά το μετασχηματισμό λαμβάνονται επίσης υπόψη οι επιδράσεις των ενεργειών του χρήστη στη ροή εκτέλεσης της εφαρμογής.

- Από τον μετασχηματισμό προκύπτει ένας αριθμός δικτύων που αναπαριστούν τα διαφορετικά μονοπάτια εκτέλεσης της εφαρμογής. Σε κάθε μονοπάτι εφαρμόζονται τεχνικές ελέγχου της χρονικής ακεραιότητας που αποφασίζουν ποια μονοπάτια είναι συνεπή και προσδιορίζουν τα προβλήματα που παρουσιάζονται στα ασυνεπή μονοπάτια.

- Κατά την εφαρμογή των τεχνικών επαλήθευσης κατασκευάζεται μια μορφή αναπαράστασης ενός συνεπούς μονοπατιού που ονομάζεται *ελάχιστο δίκτυο* (*minimal network*). Το δίκτυο αυτό μπορεί να χρησιμοποιηθεί για να απαντηθούν ερωτήσεις που αφορούν τα χρονικά χαρακτηριστικά της παρουσίασης των αντικειμένων της εφαρμογής.

Η μεθοδολογία Scenario Integrity Checking στηρίζεται στην έννοια των δικτύων χρονικών περιορισμών για να εκφράσει το πρόβλημα της χρονικής ακεραιότητας μιας multimedia εφαρμογής. Η έκφραση ενός χρονικού προβλήματος με τη βοήθεια ενός δικτύου χρονικών περιορισμών προτάθηκε για πρώτη φορά από τους Dechter, Meiri και Pearl και χρησιμοποιήθηκε για την επίλυση απλών χρονικών προβλημάτων. Σύμφωνα με τη μεθοδολογία επίλυσης χρονικών προβλημάτων, κάθε χρονικό πρόβλημα -απλό ή σύνθετο- μπορεί να εκφραστεί με τη μορφή χρονικών ανισοτήτων. Οι ανισότητες αναπαριστούν τους χρονικούς περιορισμούς που συνδέουν τα αντικείμενα ενός προβλήματος. Επομένως η επίλυση του χρονικού προβλήματος ανάγεται στην επίλυση ενός συστήματος χρονικών ανισοτήτων.

Η επίλυση ενός συνόλου ανισοτήτων είναι ένα πρόβλημα με αυξημένη πολυπλοκότητα. Οι ιδιαιτερότητες των χρονικών περιορισμών επιτρέπουν την αναπαράσταση των χρονικών ανισοτήτων που τους εκφράζουν με ένα δίκτυο χρονικών περιορισμών. Ένα δίκτυο χρονικών περιορισμών αποτελεί στην ουσία έναν κατευθυνόμενο γράφο με συγκεκριμένη δομή και ιδιότητες. Η αναπαράσταση αυτή των χρονικών περιορισμών επιτρέπει την απλοποίηση των αλγορίθμων για την επίλυση των χρονικών προβλημάτων.

Στόχος της μεθοδολογίας Scenario Integrity Checking είναι ο προσδιορισμός των χρονικών περιορισμών που υπάρχουν σε ένα σενάριο και η μετάφρασή τους σε έναν αριθμό δικτύων χρονικών περιορισμών. Η μεθοδολογία ορίζει ένα σύνολο κανόνων που προδιαγράφει τον τρόπο με τον οποίο θα γίνει η κατασκευή των δικτύων χρονικών περιορισμών για κάθε μονοπάτι εκτέλεσης της εφαρμογής. Στη συνέχεια σε κάθε δίκτυο εφαρμόζεται ο αλγόριθμος επίλυσης απλών χρονικών προβλημάτων, που εντοπίζει χρονικές ασυνέπειες και αποφασίζει αν ένα σενάριο είναι χρονικά συνεπές.

Η μεθοδολογία αποτελείται από τέσσερα διακριτά βήματα. Το πρώτο βήμα αφορά την εύρεση των διαφορετικών μονοπατιών της εφαρμογής. Το αποτέλεσμα του πρώτου βήματος είναι ο σχηματισμός ενός διαγράμματος μετάβασης καταστάσεων που περιγράφει τον τρόπο με τον οποίο αντιδρά η εφαρμογή στα διάφορα γεγονότα που δέχεται. Το διάγραμμα χρησιμοποιείται για να εντοπιστούν τα διαφορετικά μονοπάτια εκτέλεσης της εφαρμογής. Φυσικά σε μία προκαθορισμένη εφαρμογή που δεν περιλαμβάνει αλληλεπίδραση με το χρήστη, θα υπάρχει ένα και μόνο μονοπάτι εκτέλεσης.

Στο δεύτερο βήμα τα tuples του σεναρίου εξετάζονται και μετασχηματίζονται σε επιμέρους δίκτυα χρονικών περιορισμών (constraint network fragments). Τα δίκτυα αυτά εκφράζουν τους τοπικούς περιορισμούς που υπάρχουν μεταξύ των αντικειμένων που μετέχουν σε κάθε tuple του σεναρίου. Δεν εκφράζουν όμως τους συνολικούς χρονικούς περιορισμούς της εφαρμογής.

Το τρίτο βήμα έχει στόχο την σύνθεση των επιμέρους δικτύων. Η σύνθεση βασίζεται στα διαφορετικά μονοπάτια εκτέλεσης που εντοπίστηκαν κατά το πρώτο βήμα. Τα επιμέρους δίκτυα ενώνονται, χρησιμοποιώντας ένα σύνολο κανόνων, ενώ όπου κρίνεται απαραίτητο εισάγονται πρόσθετοι χρονικοί περιορισμοί. Το αποτέλεσμα του τρίτου βήματος είναι η κατασκευή ενός αριθμού δικτύων χρονικών περιορισμών για κάθε μονοπάτι εκτέλεσης.

Τέλος στο τέταρτο βήμα της μεθοδολογίας, σε κάθε ένα από τα δίκτυα περιορισμών που έχουν προκύψει εφαρμόζεται η μεθοδολογία επίλυσης απλών χρονικών προβλημάτων των Dechter, Meiri και Pearl. Το αποτέλεσμα είναι ο διαχωρισμός των μονοπατιών σε χρονικά συνεπή και χρονικά ασυνεπή μονοπάτια. Για τα χρονικά συνεπή μονοπάτια, ο αλγόριθμος κατασκευάζει το ελάχιστο δίκτυο του μονοπατιού. Το ελάχιστο δίκτυο προσδιορίζει για κάθε ζευγάρι γεγονότων το ελάχιστο και το μέγιστο χρονικό διάστημα που πρέπει να μεσολαβήσει έτσι ώστε το μονοπάτι να παραμείνει συνεπές. Το ελάχιστο δίκτυο μπορεί να χρησιμοποιηθεί για να διορθωθούν τα αδύναμα σημεία ενός σεναρίου.

Όσον αφορά το ίδιο το εργαλείο που κατασκευάστηκε, η υλοποίηση και η σχεδίασή του έχουν στηριχθεί στον αντικειμενοστραφή προγραμματισμό (object-oriented programming). Το λειτουργικό σύστημα στο οποίο αποφασίστηκε να γίνει η υλοποίηση είναι τα Windows 95. Η επιλογή αυτή έγινε γιατί υπάρχει ήδη ένας αριθμός εφαρμογών για την συγγραφή και την εκτέλεση IMD σεναρίων που έχουν αναπτυχθεί σε Windows 95. Η εφαρμογή στηρίζεται στην πλατφόρμα Win32s της Microsoft και μπορεί να εκτελεστεί, με κάποιες μικρές τροποποιήσεις, και σε περιβάλλον Windows NT. Όσον αφορά την επιλογή της γλώσσας επιλέχθηκε η γλώσσα Visual C++ εξαιτίας του αποτελεσματικού κώδικα που παράγει αλλά και εξαιτίας μιας παλαιότερης υλοποίησης ενός τμήματος της μεθοδολογίας που είχε γίνει σε γλώσσα C.