Ατομική Διπλωματική Εργασία

ΣΥΝΤΑΚΤΙΚΟΣ ΑΝΑΛΥΤΗΣ ΓΙΑ ΤΗ ΓΛΩΣΣΑ T-CDL

Χριστίνα Καραγιάννη

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σεπτέμβριος 2012

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Συντακτικός Αναλυτής Για Τη Γλώσσα T-CDL

Χριστίνα Καραγιάννη

Επιβλέπουσα Καθηγήτρια

Άννα Φιλίππου

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

Σεπτέμβριος2012

Ευχαριστίες

Η διπλωματική εργασία αποτελεί μέρος ολοκλήρωσης του προπτυχιακού μου προγράμματος Γενικής Κατεύθυνσης Πληροφορικής. Κατ'αρχάς, θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτριά μου, την κυρία Άννα Φιλίππου που μου έδωσε την ευκαιρία να εργαστώ πάνω σε αυτή την εργασία. Επίσης, θέλω να την ευχαριστήσω για την όντως πολύτιμη βοήθεια, καθοδήγηση και συμβουλές που μου πρόσφερε κατά την εκπόνηση αυτής της Ατομικής Διπλωματικής Εργασίας, αλλά και για την εμπιστοσύνη που μο υ έδειξε ώστε να φέρω σε πέραςαυτή την εργασία.Ως επιβλέπουσα καθηγήτρια, μου έδωσε την κατάλληλη επιστημονική καθοδήγηση και με τις πάμπολλες εύστοχεςπαρατηρήσεις, υποδείξεις και συμβουλές της, κατάφερα να ολοκληρώσω τηνεργασία αυτή.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου Μαρία και Παναγιώτη καθώς και το θείο μου Ανδρέα για την ολόψυχή τους αγάπη, αλλά και για την ανυπολόγιστη ηθική υποστήριξη, τη συμπαράσταση και την κατανόηση που μου πρόσφεραν όλα αυτά τα χρόνια.

Περίληψη

HWS-CDL (WebServicesChoreographyDescriptionLanguage) είναι μία γλώσσα χορογραφίας την οποία έχει προτείνει ο οργανισμός W3C και βασίζεται στην XML που περιγράφει peer-to-peer συνεργασίες των διαφόρων συμμετέχοντωνμε τον καθορισμό των κοινών και συμπληρωματικών παρατηρήσιμων συμπεριφορών τους, όπου διατεταγμένες ανταλλαγές μηνυμάτων έχουν ως αποτέλεσμα την ολοκλήρωση ενός κοινού στόχου των επιχειρήσεων.

Σκοπός αυτής της διπλωματικήςείναι η συντακτική ανάλυση μιας χορογραφίας, η οποία θα δίνεται στη μορφή της T-CDL (timed CDL), μιας άλγεβρας διεργασιών που είχε προταθεί σε προηγούμενη διατριβή [1]. Συγκεκριμένα, η T-CDL είναι μια επέκταση της CDL, μιας γλώσσας που έχει προταθεί ως μαθηματικό πρότυπο ενός υποσυνόλου της WS-CDL και περιλαμβάνει έννοιες όπως είναι οι ρόλοι, οι μεταβλητές, οι βασικές δραστηριότητες, οι δραστηριότητες ελέγχου ροής και οι μονάδες εργασίας. Η επέκταση της CDL γίνεται με τα χρονικά στοιχεία: την ενέργεια καθυστέρησης και τον τελεστή χρονικής προθεσμίας.

Η συντακτική ανάλυση είναι απαραίτητη, έτσι ώστε να γνωρίζουμε εκ των προτέρων κατά πόσο το αρχείο εισόδου πο υ θα δίνεται με τη T-CDLχορογραφία, ακολουθεί τη σύνταξη της γλώσσας. Ακολούθως, αφού βεβαιωθούμε ότι το αρχείο αυτό είναι ορθό συντακτικά, τότε ακολουθώντας τον αλγόριθμο μετάφρασης που έχει προταθεί σε προηγούμενη διατριβή, θα μεταφράσουμε τη T-CDLχορογραφία σε κώδικα Promela. Το μεταφρασμένο αρχείο, θα χρησιμοποιηθεί ως είσοδος στο εργαλείο μοντελο-ελέγχου Spinγια σκοπούς προσομοίωσης αλλά και για την επαλήθευση των ιδιοτήτων της χορογραφίας.

Για τη συντακτική ανάλυσημιαςT-CDL χορογραφίας, θα χρησιμοποιήσουμε τομεταγλωττιστή ανοιχτού κώδικα JavaCC (JavaCompiler’sCompiler).

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Υποκίνηση Έρευνας

ΗΑρχιτεκτονική Προσανατολισμένη σε Υπηρεσίες- ΑΠΥ (Service-OrientedArchitecture - SOA),είναιείναι ένα σύνολο αρχών και μεθοδολογιών, για τον σχεδιασμό και την ανάπτυξη λογισμικού με τη μορφή των αλληλεξαρτώμενων υπηρεσιών, που έχει προταθεί από το Services Research Roadmap [2] για να δημιουργήσετε ένα δραστικό και προσαρμοστικό πληροφοριακό περιβάλλον. Οι υπηρεσίες αυτές είναι καθορισμένες λειτουργίες των επιχειρήσεων που έχουν κατασκευαστεί ως συστατικά στοιχεία του λογισμικού (π.χ. διακριτά κομμάτια κώδικα ή/καιδομές δεδομένων) που μπορούν να επαναχρησιμοποιηθούν για διαφορετικούς σκοπούς.

Σύμφωνα με τον ορισμό πο υ έδωσε το W3C, μια ΥπηρεσίαΔιαδικτύου (WebService) [3]είναι ένα σύστημα λογισμικού που έχει σχεδιαστεί για τη στήριξη διαλειτουργικών αλληλεπίδρασεων μηχανής-προς-μηχανή μέσω του διαδικτύου. Οι υπηρεσίες διαδικτύου,βασίζονται στην ιδέα της ΑΠΥ και παρέχουν ένα πρότυπο διαλειτουργικών μέσων ανάμεσα σε διαφορετικές εφαρμογές λογισμικού, που τρέχει σε μια ποικιλία από πλατφόρμες ή/και πλαισίων (frameworks). Επίσης, χαρακτηρίζονται από τη μεγάλη διαλειτουργικότητα και επεκτασιμότητά τους, καθώς και από τις μηχανικά-επεξεργάσιμες περιγραφές τους χάρη στη χρήση της XML. Μπορο ύν να συνδυαστούν με ένα χαλαρά συνδεδεμένο τρόπο, προκειμένου να επιτευχθούν πολύπλοκες λειτουργίες. Επιπλέον, καλύπτουν ένα ευρύ φάσμα

συστημάτων τα οποία σε πολλές περιπτώσεις παρουσιάζουν ισχυρούς χρονικούς περιορισμούς.

Η σύνθεση των Υπηρεσιών Διαδικτύου, έχει ως αποτέλεσμα την εκτέλεση πιο δύσκολων καθηκόντων από αυτά που εκτελεί κάθε υπηρεσία από μόνη της και αποτελείται από δύο βασικές προσεγγίσεις: την ενορχήστρωση και τη χορογραφία.

ΣτηνΕνορχήστρωση (WebServicesOrchestration) [4], υπάρχει ένας κεντρικός συντονιστής, ο οποίος είναι υπεύθυνος για το συγχρονισμό των υπηρεσιών. Έστω δύο υπηρεσίες οι οποίες προκειμένουν να επικοινωνήσουν η μια με την άλλη, θα πρέπει η μία εκ των δύο να στείλει ένα μήνυμα συγχρονισμού στον κεντρικό συντονιστή, και τότε αυτός με τη σειρά του θα μεταφέρει το μήνυμα στην άλλη υπηρεσία.

ΣτηΧορογραφία (WebServicesChoreography) [4], περιγράφεται ο τρόπος με τον οποίο ένα σύνολο υπηρεσιών συνεργάζονται μεταξύ τους για την επίτευξη ενός κοινού στόχου. Στην προσέγγιση αυτή, απουσιάζει η έννοια του κεντρικού συντονιστή ο οποίος είναι υπεύθυνος για τον συντονισμό των υπηρεσιών και η επικοινωνία ανάμεσα στις υπηρεσίες, επιτυγχάνεται μέσω δυαδικών αλληλεπιδράσεων μεταξύ των συνεργαζομένων υπηρεσιών.

Η WS-CDL (Web Services Choreography Description Language) [5]είναι μια γλώσσα χορογραφίας την οποία έχει προτείνει ο οργανισμός W3C και περιγράφει τη στατική διεπαφή μιας υπηρεσίας διαδικτύου. Ορίζει το σύνολο των μηνυμάτων και τα χαρακτηριστικά των τελικών σημείων των μηνυμάτων. Οι τύποι δεδομένων, ορίζονται από τις προδιαγραφές του XML Schema, το οποίο υποστηρίζει πλούσους ορισμούς τύπων και επιτρέπει την έκφραση κάθε είδους απαίτησης τύπου XML για τα στοιχεία της εφαρμογής.

Εξαιτίας της φύσης των υπηρεσιών διαδικτύου, με την ανταλλαγή μηνυμάτων, μπορούν να προκύψουν πολλά λάθη όπως τη μη παραλαβή των μηνυμάτων, αδιέξοδα, ασυμβίβαστησυμπεριφορά κλπ. Η WS-CDLαπό μόνη της δεν μπορεί να αντιμετωπίσει τέτοιου είδους λάθη και γι’αυτό κρίθηκε η ανάγκη για μια άλλη γλώσσα η οποία θα μπορεί να διαχειρίζεται τα λάθη αυτά. Στο άρθρο [5] οι ερευνητές προτείνουν τη CDL,

μια απλή γλώσσα που επικεντρώνεται στη μοντελοποίηση βασικών χαρακτηριστικών της WS-CDL. Μέσω αυτής της γλώσσας αφού εφαρμοστούν οι προτεινόμενοι κανόνες σε μια χορογραφία, ακολούθως μπορεί να γίνει η επαλήθευση της ορθότητας και των ιδιοτήτων της χορογραφίας, χρησιμοποιώντας το μοντελο-ελεγκτή SPIN,αφού πρώτα γίνει η μετάφραση της χορογραφίας σε κώδικα Promela (PROcessMetaLAnguage) [6].

Τα πιο πάνω είχαν μελετηθεί εκτενέστερα στα πλαίσια μιας προηγούμενηςδιατριβής [1], στην οποία και βασίζεται η διπλωματική αυτή εργασία. Όπως έχει προαναφερθεί, οι υπηρεσίες διαδικτύου, καλύπτουν ένα ευρύ φάσμα συστημάτων τα οποία στις πλείστες περιπτώσεις έχουν ισχυρούς χρονικούς περιορισμούς. Επομένως, ο χρόνος ήταν ο βασικός λόγος που οδήγησε στην επέκταση της CDLμε την έννοια του χρόνου. Ως εκ τούτου, στη διατριβή [1] παρουσιάζεται η T-CDL, η οποία εισάγει στη CDL την ενέργεια της καθυστέρησης, χρ(d), καθώς και τον τελεστή λήξης A (E,F), ο οποίος επιβάλει μία χρονική προθεσμία στη διεργασία A. Ακολούθως, στη διατριβή [1] προτείνεται ένας αλγόριθμος ο οποίος μεταφράζει μια T-CDLχορογραφία σε κώδικα Promela. Η μετάφραση αυτή επιτυγχάνεται με δύο βήματα: Αρχικά γίνεται η εξαγωγή των όψεων ενορχήστρωσης μέσω της συνάρτησης Pr, η οποία με παραμέτρους μια δραστηριότητα και ένα ρόλο, εξάγει τις δραστηριότητες που αφορούν το ρόλο αυτό. Στη συνέχεια, γίνεται η μετάφραση των όψεων ενορχήστρωσης σε κώδικα Promela αφού προηγηθεί η μελέτη της συμπεριφοράς του κάθε ρόλου μέσα σε κάθε αλληλεπίδραση της χορογραφίας. Με την ολοκλήρωση της μετάφρασης, ολοκληρώνεται και η δημιουργία ενός αντιπροσωπευτικού μοντέλου το οποίο θα μπορούμε να τρέξουμε στο εργαλείο μοντελο-ελέγχου Spin. Έτσι θα είμαστε σε θέση να επιβεβαιώσουμε κάποιες ιδιότητες της χορογραφίας όπως το κατά πόσο επιστρέφει ορθό αποτέλεσμα, εάν περιέχει αδιέξοδα, εάν ικανοποιεί τις αρχικές προδιαγραφές κλπ. Στο Υποκεφάλαιο 2.2.1 παρουσιάζεται η σύνταξη της T-CDL.

1.2 Σκοπός της Διπλωματικής

Σκοπός αυτής της διπλωματικής είναι η συντακτική ανάλυση μιας T-CDLχορογραφίας. Μέσω της συντακτικής ανάλυσηςεξετάζουμε κατά πόσο το αρχείο εισόδου που θα δίνεται με τη T-CDLχορογραφία, ακολουθεί τη σύνταξη της γλώσσας αυτής. Η συντακτική ανάλυση επιτυγχάνεται με το μεταγλωττιστή ανοιχτού κώδικα JavaCC (JavaCompiler'sCompiler) ο οποίος βάση της σύνταξης της T-CDL που είχε προταθεί στην προηγούμενη διατριβή [1] και που φαίνεται στους πίνακες 4.1, 4.2 και 4.3.

1.3 Δομή της Εργασίας

Αρχικά, στο Κεφάλαιο 2, παρουσιάζεται η προηγούμενη δουλειά που έχει γίνειόσο αφορά την περιοχή στην οποία στηρίχτηκε η έρευνά μας.Στο Υποκεφάλαιο 2.1, γίνεται αναφορά στις Άλγεβρες Διεργασιών και στο πώς συνδέονται με τον αλγόριθμο μετάφρασης που προτάθηκε στην προηγούμενη διατριβή [1]. Στο Υποκεφάλαιο 2.2, παρουσιάζουμε τη γλώσσα T-CDL και συγκεκριμένα στο Υποκεφάλαιο 2.2.1 παρουσιάζουμε τη σύνταξη της T-CDL. Στο Υποκεφάλαιο 2.3, γίνεται αναφορά στο εργαλείο JavaCC που θα χρησιμοποιήσουμε για τη συντακτική ανάλυση. Συγκεκριμένα στο Υποκεφάλαιο 2.3.1περιγράφουμε τι είναι ο JavaCCκαι στο Υποκεφάλαιο 2.3.2 δίνεται μια σύντομη επεξήγηση για το τι πρέπει να κάνει κάποιος για να εγκαταστήσει το plugin του JavaCC στο Eclipse, καθώς και ποιες είναι οι απαιτήσεις. Ακολούθως στο Υποκεφάλαιο 2.3.3 δίνονται ένα παράδειγμα λεκτικού αναλυτή με χρήση του εργαλείου JavaCC, ενώ στο Υποκεφάλαιο 2.3.4 δίνεται ένα παράδειγμα ελέγχου σύνταξης με χρήση του εργαλείου JavaCC. Στο Υποκεφάλαιο 2.3.5 γίνεται επεξήγηση για το πώς θα εξαγάγουμε τα αρχεία του parser καθώς και του λεκτικού αναλυτή. Στο Υποκεφάλαιο 2.3.6 εξηγούμε τι είναι η αριστερή αναδρομή και τρόπους απαλειφής της, ενώ στο Υποκεφάλαιο 2.3.7 εξηγούμε τι είναι ο Lookahead και πότε χρησιμοποιείται.Έχοντας κατανοήσει τα πιο πάνω, στο Υποκεφάλαιο 2.3.8 παρουσιάζεται ένα παράδειγμα διερμηνέα.

Στο Κεφάλαιο 3, παρουσιάζεται ο Κύκλος Ζωής και Ανάπτυξης Λογισμικού, για τη Συντακτική Ανάλυση Χορογραφιών γραμμένες σε T-CDL. Στο Υποκεφάλαιο

3.1, γίνεται μια εισαγωγή στον Κύκλο Ζωής και Ανάπτυξης Λογισμικού. Ακολούθως στο Υποκεφάλαιο 3.2 γίνεται αναφορά στη Φάση των Απαιτήσεων, στο Υποκεφάλαιο 3.3 γίνεται αναφορά στη Φάση των Προδιαγραφών, ενώ στο Υποκεφάλαιο 3.4 γίνεται αναφορά στη Φάση του Σχεδιασμού. Στη συνέχεια στο Υποκεφάλαιο 3.4.1 γίνεται αναφορά στην Αρχιτεκτονική Σχεδίαση, ενώ στο Υποκεφάλαιο 3.4.2 γίνεται αναφορά στη Λεπτομερή Σχεδίαση. Στο Υποκεφάλαιο 3.5 παρουσιάζεται η Φάση της Υλοποίησης. Συγκεκριμένα στο Υποκεφάλαιο 3.5.1 δίνεται η γλώσσα προγραμματισμού που επιλέχθηκε να χρησιμοποιηθεί για την υλοποίηση του προγράμματος καθώς και οι λόγοι που μας οδήγησαν στην επιλογή αυτή. Ακολούθως στο Υποκεφάλαιο 3.5.2 παρουσιάζεται η γραμματική της T-CDL και συγκεκριμένα στο Υποκεφάλαιο 3.5.3 γίνεται επεξήγηση της γραμματικής αυτής.Έπειτα, στο Υποκεφάλαιο 3.5.4δίνεται ο αλγόριθμος για τη συντακτική ανάλυση μιας T-CDL χορογραφίας και συγκεκριμένα στα Υποκεφάλαια 3.5.5 και 3.5.6 παρουσιάζονται οι δομές και οι μεθόδοι που χρησιμοποιήθηκαν αντίστοιχα.Επιπλέον, στο Υποκεφάλαιο 3.5.7 παρουσιάζονται ορισμένα παραδείγματα εκτέλεσης του συντακτικού αναλυτή που υλοποιήθηκε.

Στο Κεφάλαιο 4,παρουσιάζονται τα συμπεράσματα και οι γενικές εντυπώσεις μας από αυτή τη διπλωματική. Συγκεκριμένα στο Υποκεφάλαιο 4.1 παρουσιάζουμε τα τελικά συμπεράσματα από την υλοποίηση του συντακτικού αναλυτή για χορογραφίες της γλώσσας T-CDL, ενώ στο Υποκεφάλαιο 4.2 προτείνονται ιδέες για επέκταση της υλοποίησης αυτής, μέσα από άλλες μελλοντικές εργασίες. Στο Υποκεφάλαιο 4.3 παρουσιάζεται ό,τι θετικό αποκομίσαμε από τη διπλωματική εργασία και τέλος στο Υποκεφάλαιο 4.4 παρουσιάζονται διάφορα προβλήματα που αντιμετωπίσαμε κατά την υλοποίηση του συντακτικού αναλυτή.

Κεφάλαιο 2

Ανασκόπηση Προηγούμενης Ερευνητικής Εργασίας

2.1 Άλγεβρες Διεργασιών

Όπως έχει αναφερθεί και στην εισαγωγή, η WS-CDLδεν παρέχει κάποιο εργαλείο το οποίο να είναι σε θέση να επικυρώσει τις προδιαγραφές των χορογραφιών που είναι διατυπωμένες στη γλώσσα αυτή και να πιστοποιεί τη διαλειτουργικότητα μεταξύ των υπηρεσιών διαδικτύου. Εξαιτίας όμως της φύσης των υπηρεσιών διαδικτύου, με την ανταλλαγή μηνυμάτων, είναι πολύ πιθανό να προκύψουν πολλά λάθη όπως τη μη παραλαβή των μηνυμάτων, αδιέξοδα, ασυμβίβαστησυμπεριφορά κλπ. Εφόσον ηWS-CDLαπό μόνη της δεν μπορεί να αντιμετωπίσει αυτά τα λάθη, απαιτείται ο συσχετισμός της WS-CDL με μια τυπική μέθοδο που θα παρέχει αυτές τις δυνατότητες επικύρωσης.

Σύμφωνα με την προηγούμενη διατριβή [1], θεωρήθηκε πως ο συσχετισμός της WS-CDLμε μια τυπική μέθοδο θα μπορούσε να γίνει με μια νέα γλώσσα διεργασιών η οποία θα είναι επέκταση της CDL [1,4]με κάποιους χρονικούς τελεστές και φέρει την

ονομασία T-CDL (Timed - CDL). Η επιλογή της CDLέγινε εξαιτίας του γεγονότος ότι ο έλεγχος των ιδιοτήτων μιας χορογραφίας μπορεί να γίνει μέσω του μοντελο-ελεγκτή Spin.

2.2Χρονική CDL (T-CDL)

Όπως έχουμεαναφέρει και στο Υποκεφάλαιο 1.1, οι συγγραφείς του άρθρου[4] όρισαν ένα τυπικό μοντέλο, τη CDL, το οποίο επικεντρώνεται στα βασικά χαρακτηριστικά της WS-CDL, όπως είναι οι μεταβλητές, οι δραστηριότητες ελέγχου ροής, οι αλληλεπιδράσεις, οι δραστηριότητες ανάθεσης, οι συμμετέχοντες ρόλοι σε μια χορογραφία, καθώς και οι μεταξύ τους συνεργασίες.

Οι υπηρεσίες διαδικτύου, καλύπτουν ένα ευρύ φάσμα συστημάτων τα οποία στις πλείστες περιπτώσεις έχουν ισχυρούς χρονικούς περιορισμούς. Επομένως, λαμβάνοντας υπόψη πόσο χρήσιμος είναι ο χρόνος,στο [1] έγινε επέκταση της CDL με την έννοια του χρόνου. Ηεπίτευξη της εισαγωγής της έννοιας του χρόνου που υπάρχει στην T-CDL, έγινε με την εισαγωγή στη CDL της ενέργειας της καθυστέρησης, χρ(d), καθώς και του τελεστή λήξης $A \triangleright^{d} (E,F)$, ο οποίος επιβάλει μία χρονική προθεσμία στη διεργασία A. Στο Υποκεφάλαιο 2.2.1 παρουσιάζεται η σύνταξη της T-CDL.

2.2.1 Σύνταξη της T-CDL

Όπως έχουμε αναφέρει, η T-CDL, είναι μια επέκταση τηςCDL με την εισαγωγή της έννοιας του χρόνου. Η σύνταξη τηςCDL [5] αποτελείται από ένα ευρύ φάσμα όρων: Ο όρος R, παρουσιάζει τη δήλωση κάποιου ρόλου που συμμετέχει στη χορογραφία και οι όροι A και B δηλώνουν τις διεργασίες που εκτελούνται στη χορογραφία. Οι όροι r, f, t αναφέρονται στα ονόματα ρόλων και οι όροι x, y, u, v δηλώνουν μεταβλητές. Οι όροι e_1, e_2, e_3 δηλώνουν XPath εκφράσεις και οι όροι g, g_1, g_2 δηλώνουν λογικές εκφράσεις. Ο όρος c που εμφανίζεται στα τόξα των δραστηριοτήτων αλληλεπίδρασης, δηλώνει το κανάλι επικοινωνίας. Επίσης, η μεταβλητή op αναφέρεται στη λειτουργία που εκτελείται από τον παραλήπτη της αλληλεπίδρασης όταν αυτός παραλάβει το μήνυμα. Ο όρος $\overline{R}$, χρησιμοποιείται ως η συντομογραφία που εκφράσει το σύνολο των

ρόλων $R_1,\ldots,R_n$, για κάποιο n. Αυτό γίνεται παρόμοια και για τις μεταβλητές $\bar{x}, \overline{op}, \bar{e}$ κτλ. Επίσης, ο όρος r.x αναφέρεται στην μεταβλητή x του ρόλου r και ο όρος r.x:=e προσδιορίζει ότι η μεταβλητή x που ανήκει στο ρόλο r θα πάρει το αποτέλεσμα της έκφρασης e, που επίσης ανήκει στο ρόλο r. Ομοίως, σύμφωνα με τα πιο πάνω, ο όρος $r.\bar{x} := \bar{e}$ είναι συντομογραφία των αναθέσεων $r.x_1:=e_1,\ldots,r.x_n:=e_n$.

Η δήλωση μιας χορογραφίας στη CDL, αποτελείται από το όνομα της χορογραφίαςC, το σύνολο των ρόλων $\bar{R}$ που συμμετέχουν σε αυτή και την αρχική διεργασία A με την οποία θα ξεκινήσει η εκτέλεση της χορογραφίας:

$$C[\bar{R}, A]$$

Κάθε συμμετέχοντας ρόλος στη χορογραφία, αποτελείται από το όνομα του ρόλου r, το σύνολο των μεταβλητών του $\bar{x}$ και το σύνολο των συμπεριφορών που θα παρουσιάζει $\overline{op}$:

$$R ::= r[\bar{x}, \overline{op}]$$

Μιαδραστηριότητα στην CDL μπορεί είναι μια βασική δραστηριότητα, είτε μια δραστηριότητα ελέγχου ροής, είτε μια μονάδα εργασίας, είτε μια ενέργεια καθυστέρησης, είτεμια δραστηριότητα προθεσμίας.

Βασικές δραστηριότητες:

- Η δραστηριότητα skip,εκφράζει ότι ένας ρόλος δεν εκτελεί καμία ενέργεια.
- Η δραστηριότητας της ανάθεσης, $r.\bar{x} := \bar{e}$, εκφράζει ότι οι μεταβλητές $\bar{x}$ του ρόλου r θα πάρουν το αποτέλεσμα των εκφράσεων $\bar{e}$. Ο όρος $r.\bar{x} := \bar{e}$, είναι συντομογραφία των αναθέσεων $r.x_1:=e_1,\ldots,r.x_n:=e_n$.
- Ένα αίτημα, εκφράζεται με τον όρο comm(f.x $\xrightarrow{c}$ t.y,rec,time), ο οποίος εκφράζει την αποστολή της τιμής της μεταβλητής x του ρόλου f στην μεταβλητή y του ρόλου t, μέσω του καναλιού c.
- Η απάντηση, μοντελοποιείται από τον όρο comm(f.x $\xleftarrow{c}$ t.y,rec,time), ο οποίος εκφράζει την αποστολή της τιμής της μεταβλητής y του ρόλου t στην μεταβλητή x του ρόλου f, μέσω του καναλιού c.
- Μια αλληλεπίδραση αιτήματος-απάντησης εκφράζεται με τον όρο

comm(f.x $\xrightarrow{c}$ t.y,f.u $\xleftarrow{c}$ t.v,rec,time). Σε αυτή την αλληλεπίδραση, στέλνεται αρχικά ένα αίτημα από τον ρόλο f στο ρόλο t, μέσω του καναλιού c_t, και ακολούθως στέλνεται η απάντηση από τον ρόλο t στο ρόλο f, μέσα από το κανάλι c_f.

Ο όρος rec, προσδιορίζει τις αναθέσεις που θα γίνουν αντίστοιχα στους δύο ρόλους για ενημέρωση της κατάστασης τους μετά την αποστολή και την παραλαβή των μηνυμάτων. Ο όρος αυτός είναι η συντομογραφία των αναθέσεων $f.\bar{x} := \bar{e}_f$ και $t.\bar{y} := \bar{e}_t$ όπου $\bar{e}_f$ και $\bar{e}_t$ είναι οι λίστες των μεταβλητών που ανήκουν στους ρόλους f και t αντίστοιχα και όπου και το σύνολο των XPath εκφράσεων που υπάρχουν στους ρόλους f και t αντίστοιχα.

Η εισαγωγή της ενέργειας της καθυστέρησης στη CDL γίνεται μέσα από την δραστηριότητα op, η οποία μπορεί να προκαλέσει καθυστέρηση προκαθορισμένου χρόνου d, όπου dReal, ή καθυστέρηση απροσδιορίστου χρόνου δ. Έτσι, η δραστηριότητα op ορίζεται ως op::= ε | χρ(d) | δ. Ως εκ τούτου, οι δραστηριότητες της αλληλεπίδρασης στην T-CDL(αίτημα, απάντηση, αίτημα – απάντηση) διαφοροποιούνται σε σύγκριση με τις αντίστοιχες δραστηριότητες αλληλεπίδρασης στη CDL,για να συμπεριλάβουν την καθυστέρηση που μπορεί να παρουσιαστεί κατά την ανταλλαγή πληροφοριών.

Η σύνταξη των βασικών δραστηριοτήτων στην T-CDL δίνεται ως ακολούθως:

$BA ::= skip$	(Καμία Ενέργεια)
$\mid r.\bar{x} := \bar{e}$	(Ανάθεση)
$\mid comm(f.x \xrightarrow{c} t.y, rec, op)$	(Αίτημα)
$\mid comm(f.x \xleftarrow{c} t.y, rec, op)$	(Απάντηση)
$\mid comm(f.x \xrightarrow{c_t} t.y,\ f.u \xleftarrow{c_f} t.v, rec, op)$	(Αίτημα - Απάντηση)

Πίνακας2.1 –Σύνταξη των βασικών δραστηριοτήτων στην T-CDL [1]

Η σύνταξη των υπόλοιπων δραστηριοτήτων της T-CDLδίνεται ως ακολούθως:

$A, B ::= BA$	(Βασικές Δραστηριότητες)
$\mid p ? A$	(Συνθήκη)
$\mid p * A$	(Επανάληψη)
$\mid g : A : p$	(Μονάδα Εργασίας)
$\mid A; B$	(Δραστηριότητα Διαδοχής)
$\mid A + B$	(Επιλογή)
$\mid g_1 \Rightarrow A + g_2 \Rightarrow B$	(Γενική Επιλογή)
$\mid A \parallel B$	(Παραλληλισμός)
$\mid \chi\rho(d) A$	(Τελεστής Καθυστέρησης)
$\mid A \triangleright^d (E, F)$	(Δραστηριότητα Προθεσμίας)

Πίνακας 2.2–Σύνταξη των υπόλοιπων δραστηριοτήτων της T-CDL[1]

Μια δραστηριότητα στη CDL μπορεί να ανήκει είτε στις βασικές δραστηριότητες, ή να είναι είτε μία συνθήκη ή μία δραστηριότητα επανάληψης, ή μία Μονάδα Εργασίας, ή να ανήκει στις δραστηριότητες ελέγχου ροής.

Η δραστηριότητα της συνθήκης, p ?A, θα ξεκινήσει την εκτέλεση της με την αξιολόγηση του φρουρού p . Εάν ο φρουρό ς είναι ορθός, τότε η διεργασία A θα εκτελεστεί διαφορετικά η δραστηριότητα θα τερματιστεί.

Η δραστηριότητα της επανάληψης, p*A, εκφράζει την επαναλαμβανόμενη εκτέλεση μίας διεργασίας. Η δραστηριότητα αυτή θα ξεκινήσει με την αξιολόγηση του φρουρού p. Εάν ο φρουρό ς είναι ορθός, τότε η διεργασία A θα εκτελεστεί για πρώτη φορά διαφορετικά η δραστηριότητα επανάληψης θα τερματιστεί. Όταν ολοκληρωθεί η εκτέλεση της διεργασίας A τότε θα επανεξεταστεί ο φρουρός p εάν ο φρουρός είναι ορθός τότε θα ξαναεκτελεστεί η διεργασία A. Έτσι η διεργασία A θα εκτελείται κάθε φορά που ο φρουρός p είναι ορθός.

Μια Μονάδα Εργασίας εκφράζεται με τον όρο g:A:p. Η μονάδα εργασίας όπως έχει προαναφερθεί, περιγράφει την ελεγχόμενη και επαναλαμβανόμενη εκτέλεση μιας διεργασίας. Μια μονάδα εργασίας αποτελείται από την διεργασία Α, τον φρουρό g και το φρουρό επανάληψης p. Ο φρουρός g, και ο φρουρός επανάληψης p, είναι λογικές εκφράσεις και καθορίσουν εάν η εσωκλειόμενη διεργασία, Α θα εκτελεστεί μια ή περισσότερες φορές. Η εκτέλεση της μονάδας εργασίας θα ξεκινήσει με την αξιολόγηση του φρουρού g. Εάν ο φρουρός είναι ορθός, τότε η διεργασία Α θα εκτελεστεί για πρώτη φορά διαφορετικά η μονάδα εργασίας θα τερματιστεί. Μετά την εκτέλεση της διεργασίας Α, θα αξιολογηθεί ο φρουρός επανάληψης p. Εάν είναι ορθός, τότε η διεργασία Α θα ξαναεκτελεστεί διαφορετικά θα τερματιστεί η εκτέλεση της μονάδας εργασίας. Κάθε φορά που η διεργασία εκτελείται επιτυχώς θα αξιολογείται ο φρουρός επανάληψης p και ανάλογα με το αποτέλεσμα, είτε θα ξαναεκτελεστεί η διεργασία είτε θα τερματιστεί η εκτέλεση της μονάδας εργασίας.

Μια δραστηριότητα ελέγχου ροής μπορεί να είναι είτε μια διαδοχική δραστηριότητα A;B, είτε μια δραστηριότητα επιλογής A+B, είτε μια δραστηριότητα γενικής επιλογής $g_1 \Rightarrow A + g1 \Rightarrow B$ ή μία παράλληλη επιλογή A||B.

Ο τελεστής χρ(d)A, όπου d Real, δηλώνει ότι η διεργασία Α θα είναι διαθέσιμη και θα μπορέσει να εκτελεστεί μετά από καθυστέρηση διάρκειας d μονάδων χρόνου. Επιπλέον, στην T-CDL εμφανίζεται και η δραστηριότητα της προθεσμίας $A \triangleright^{d} (E,F)$ η οποία επιβάλει στη διεργασία Α χρονική προθεσμία d μονάδων χρόνου, όπου d Real. Εάν η διεργασία Α δεν εκτελεστεί εντός αυτού του χρονικού πλαισίου τότε θα εκτελεστεί η μονάδα εξαίρεσης Ε. Σε περίπτωση που η διεργασία Α εκτελεστεί εντός αυτού του χρονικού πλαισίου, τότε θα εκτελεστεί η μονάδα τεκμηρίωσης του αποτελέσματος, F.

2.3 JavaCC

2.3.1 Τι είναι ο JavaCC

ΤοεργαλείοJavaCC [8],είναιέναςπαραγωγός λεκτικού και συντακτικού αναλυτή. Οιλεκτικοί και οι συντακτικοί αναλυτές,είναιστοιχείαλογισμικούπου ασχολούνται μεεισόδουςαπόακολουθίεςχαρακτήρων. Οιμεταγλωττιστέςκαιοιδιερμηνείς ενσωματώνουν λεκτικούς και συντακτικούς αναλυτές για να διαβάζουν αρχεία που περιέχουν προγράμματα.Όμως, μπορούν να χρησιμοποιηθούν σε μια μεγάλη ποικιλία άλλων εφαρμογών όπως θα δούμε και με κάποια παραδείγματα πιο κάτω. Οι λεκτικοί αναλυτές μπορούν να σπάσουν μια ακολουθία χαρακτήρων σε υποακολουθίες που λέγονται ***tokens***.

2.3.2 Εγκατάσταση JavaCCplugin στο Eclipse

Για την εγκατάσταση του JavaCCpluginστο Eclipse, χρειαζόμαστε το πρόγραμμα Eclipseμε έκδοση 3.6 ή νεότερη, καθώς και το JDKμε έκδοση 1.7 ή νεότερη. Τα βήματα για την εγκατάσταση δίνονται στο [7].

Αφού ολοκληρωθεί η εγκατάσταση, ακολουθώντας τα πιο πάνω βήματα, θα παρατηρήσουμε πως με τη δημιουργία ενός JavaCC αρχείου, πάντοτε εμφανίζεται το ίδιο παράδειγμα. Για την απενεργοποίηση της επιλογής αυτής και για να μπορέσουμε να γράψουμε κατ' ευθείαν τη δική μας γραμματική, πηγαίνεουμε στο Run και φεύγουμε την επιλογή από το BuildAutomatically.

2.3.3 Παράδειγμα χρήσης του λεκτικού αναλυτή

Έστω το ακόλουθο πρόγραμμα σε γλώσσα C:

```
intmain()
{
	return 0;
}
```

Ο λεκτικός αναλυτής ενός μεταγλωττιστή C, θα έσπαζε το πιο πάνω πρόγραμμα στην ακόλουθη ακολουθία από tokens:

“int”, “ ”, “main”, “(”, “)”,
“ ”, “{”, “\n”, “\t”, “return”
“ ”, “0”, “ ”, “;”, “\n”,
“}”, “\n”, “”

Ο λεκτικός αναλυτήςθα αναγνώριζε και το είδος (***kind***)για το κάθε token. Για το παράδειγμά μας, η ακολουθία του είδους των tokens μπορεί να είναι η ακόλουθη:

KWINT, SPACE, ID, OPAR, CPAR,
SPACE, OBRACE, SPACE, SPACE, KWRETURN,
SPACE, OCTALCONST, SPACE, SEMICOLON,
SPACE,
CBRACE, SPACE, EOF

Totokenπου έχει ως kindto EOF, αναπαριστά το τέλος του αρχικού αρχείου. Στη συνέχεια, η ακολουθία των tokensπερνά στον parser. ΣτηνπερίπτωσητηςC, oparserδε χρειάζεται όλα τα tokens. Στο παράδειγμά μας, όσα αντιστοιχούν στο “SPACE”, δεν περνούν στο parser. Ακολούθωςoparserαναλύει την ακολουθία των tokensγια να καθορίσει τη δομή του προγράμματος. Συχνά στους μεταγλωττιστές, oparserπαράγει ένα δέντρο που αναπαριστά τη δομή του προγράμματος. Αυτό το δέντρο, μπορεί να αποτελεί είσοδο σε στοιχείατου μεταγλωττιστή που είναι υπεύθυνα για την ανάλυση και για την παραγωγή κώδικα.

Ο JavaCCαπό μόνος του δεν είναι ούτε ένας parserούτε ένας λεκτικός αναλυτής, αλλά ένας παραγωγός (generator).Αυτό σημαίνει ότι παράγει σαν έξοδο λεκτικούς

αναλυτές και parser, σύμφωνα με τις προδιαγραφές που διαβάζει από ένα αρχείο. Οι λεκτικοί αναλυτές και οι parserseίναι γραμμένοι σε Java.

2.3.4Παράδειγμα Ελέγχου Σύνταξης

Έστω η εξής γραμματική:

SKIP → [“ ” | “\t” | “\n” | “\r”]
NUM → [0-9]+
TOKEN → [+*()] | NUM
E → T ("+" T)*
T → F ("*" F)*
F → NUM | "(" E ")"

Δημιουργούμε ένα αρχείο Exp1.jj, όπως περιγράψαμε στο Υποκεφάλαιο 2.3.2 με τον ακόλουθο κώδικα:

Exp1.jj

```
PARSER_BEGIN(SyntaxChecker)

public class SyntaxChecker {
    public static void main(String[] args) {
        try {
            new SyntaxChecker(new java.io.StringReader(args[0])).S();
            System.out.println("Syntax is okay");
        } catch (Throwable e) {
            // Catching Throwable is ugly but JavaCC throws Error objects!
            System.out.println("Syntax check failed: " + e.getMessage());
        }
    }
}

PARSER_END(SyntaxChecker)

SKIP:  { " " | "\t" | "\n" | "\r"                    }
TOKEN: { "(" | ")" | "+" | "*" | <NUM: (["0"-"9"])+> }

void S(): {} { E() <EOF>           }
void E(): {} { T() ("+" T())*      }
void T(): {} { F() ("*" F())*      }
void F(): {} { <NUM> | "(" E() ")" }
```

Σχήμα 2.1: Παράδειγμα γραμματικής με χρήση του εργαλείου JavaCC

Όπως βλέπουμε στο Σχήμα 2 πρέπει να ορίσουμε μια κλάση για τον parserανάμεσα στο PARSER_BEGINκαι PARSER_END και ακολούθως για να αγνοήσουμε κάποιους χαρακτήρες, απλά τους δηλώνουμε στο πεδίο SKIP, όπως φαίνεται πιο πάνω, ενώ για να δηλώσουμε tokensτα δηλώνουμε στο πεδίο TOKEN.Πρέπει να οριστούν οι μεθόδοι για το parsingγια κάθε ένα μη-τερματικό στη γραμματική και μοιάζουν με την EBNFστη γραμματική. Σημειώστε ότι τα μη-τερματικά σύμβολα μοιάζουν με κλήσεις συναρτήσεων, τα tokenseίναι όπως τα ξέρουμε και έχουμε τα συνηθισμένα μετασύμβολα όπως είναι το "|" (επιλογή), "*" (Kleenestar) και "+" (Kleeneplus).

Σημείωση: Εάν μπροστά από ένα token προσθέσουμε το "#", τότε αυτόματα το συγκεκριμένο tokenδηλώνεται μόνο τοπικά. Αυτό το κάνουμε στις περιπτώσεις όπου το συγκεκριμένο tokenδε θέλουμε να αναγνωρίζεται κατά τη διάρκεια του parsing, αλλά θα χρησιμοποιείται μόνο για το σχηματισμό πιο πολύπλοκων tokens.

2.3.5Παραγωγή συντακτικού και λεκτικού αναλυτή

Αφού έχουμε τελειώσει με το αρχείο Exp1.jj, βρίσκουμε από τον Package Explorer το αρχείο αυτό, δεξί κλικ και ***CompilewithJavaCC.***Με αυτή την εντολή παράγονται αυτόματα 7 κλάσεις .java, η κάθε μια στο δικό της αρχείο:

- **TokenMgrError**:Είναι μια απλή κλάση για διαχείριση των λαθών. Χρησιμοποιείται για λάθη που εντοπίστηκαν από το λεξικό αναλυτή και είναι υποκλάση της Throwable.
- **ParseException:**Είναι κι αυτή μια κλάση για διαχείριση των λαθών. Χρησιμοποιείται για λάθη που εντοπίστηκαν από τον parser και είναι υποκλάση της Exceptionεπομένως και της Throwable.
- **Token:**Είναι μια κλάση που αναπαριστά τα tokens. Κάθε αντικείμενο Token έχει για είδος (kind) ένα ακέραιο πεδίο που αναπαριστά το είδος του Token(για το παράδειγμά μας η αναπαράσταση θα γίνει μόνο για το είδος NUM, μιας και στα υπόλοιπα tokens "(" | ")" | "+" | "*"δεν έχει δωθεί κάποια ονομασία για το είδος τους) καθώς και ένα πεδίο Stringγια την εικόνα (image) του token, που αναπαριστά την ακολουθία χαρακτήρων από το αρχείο εισόδου που αναπαριστά το συγκεκριμένο token.

- **SimpleCharStream:** Είναιμιακλάσηadapter (adapterclass) ηοποίαπαραδίδειχαρακτήρες στο λεκτικό αναλυτή
- **SyntaxCheckerConstants:** Είναιμιαδιεπαφή (interface) η οποία καθρίζει τον αριθμό των κλάσεων που χρησιμοποιούνται τόσο στο λεκτικό αναλυτή όσο και στο parser.
- **SyntaxCheckerTokenManager:**Είναι ο λεκτικός αναλυτής.
- **SyntaxChecker:**Είναι ο parser

Τώρα πλέον είμαστε σε θέση να κάνουμε compiletον parserSyntaxChecker.javaμε τον Javacompiler.

2.3.6Αριστερή Αναδρομή

Μια γραμματική ονομάζεται αριστερά αναδρομική (left-recursive), εάν έχει τουλάχιστον ένα μη-τερματικό σύμβολο Αγια το οποίο ισχύει:

$$A \rightarrow Aa \mid b$$

όπουa,bακολουθίες τερματικών ή μη-τερματικών και bδεν ξεκινά με Α.

Μια γραμματική ονομάζεται έμμεσα αριστερά αναδρομική (indirectleft-recursive), εάν για τα μη-τερματικά σύμβολα A_0, A_1,… ,A_nισχύει η μορφή:

$$A_0 \rightarrow A_1 a_1 |...$$

$$A_1 \rightarrow A_2 a_2 |...$$

...

$$A_n \rightarrow A_0 a_{n+1} |...$$

όπουa_0,$a_{1, ...}$, a_nακολουθίες τερματικών ή μη-τερματικών.

Ο JavaCC δεν μπορεί να κάνει parseγραμματικές που περιλαμβάνουν αριστερή αναδρομή. Ας υποθέσουμε ότι έχουμε την εξής γραμματική:

$$E \rightarrow T \mid E \text{ "+" } T$$

$$T \rightarrow F \mid T \text{ "*" } F$$

$$F \rightarrow NUM \mid \text{"(" } E \text{ ")"}$$

Αλλάζοντας τη γραμματική στο αρχείο .jj με την πιο πάνω, στην κονσόλα του JavaCCθα δούμε το ακόλουθο μήνυμα:

```
>java -classpath C:/eclipse/plugins/sf.eclipse.javacc_1.5.26/javacc.jar javaccbadLeftRecursion.jj
Java Compiler Compiler Version 5.0 (Parser Generator)
(type "javacc" with no arguments for help)
Reading from file badLeftRecursion.jj . . .
Error: Line 22, Column 1: Left recursion detected: "T... --> T..."
Error: Line 21, Column 1: Left recursion detected: "E... -->E..."
Detected 2 errorsand 0 warnings.
```

Επομένως πρέπει να απαλείψουμε την αριστερή αναδρομή πριν γράψουμε τους κανόνες της γραμματικής μας.

Αφαίρεση της άμεσης αριστερής αναδρομής:

Για κάθε κανόνα της μορφής:

$$A \rightarrow Aa_1 |...| Aa_n | b_1 |...| b_m$$

όπου Α είναι αριστερά αναδρομικό μη-τερματικό, αείναι μια ακολουθία από μη-τερματικά και τερματικά τα οποία δεν είναι κενά ($a \neq \varepsilon$)και βείναι μια ακολουθία από μη-τερματικά και τερματικά τα οποία δεν ξεκινούν με Α, αντικαθιστούμε τον κανόνα Αμε τον κανόνα:

$$A \rightarrow b_1A' |...| b_mA'$$

και δημιουργούμε το νέο μη-τερματικό κανόνα:

$$A' \rightarrow \varepsilon | \alpha_1A' |...| \alpha_vA'$$

Αφαίρεση της έμμεσης αριστερής αναδρομής:

Εάν η γραμματική δεν έχει ε-κανόνες, κανόνες δηλαδή της μορφής $A \rightarrow \ldots|\varepsilon|\ldots$ ούτε κύκλους, δηλαδή παραγωγές της μορφής $A \Rightarrow \ldots \Rightarrow A$για κάποιο μη-τερματικό Α, εφαρμόζουμε τον ακόλουθο αλγόριθμο για την αφαίρεση της έμμεσης αριστερής αναδρομής:

```
for  i=1 to n
{
        for j=1 to i-1
        {
                Έστω ο κανόνας Aj, Aj→δ1|...|δk
                Αντικαθιστούμε κάθε κανόνα Aι→Ajγ με τον κανόνα
                Aι→δ1γ|...|δkγ
        }
        Αφαιρούμε την άμεση αριστερή αναδρομή για τον κανόνα Aj
}
```

2.3.7Lookahead

Ας προσθέσουμε κάποιους identifiers (id) και εκφράσεις αναθέσεων στη γλώσσα μας:

E -> id ":=" E | T ("+" T)*
T -> F ("*" F)*
F -> NUM | ID | "(" E ")"

Εάν προσπαθήσουμε να τρέξουμε αυτή τη γραμματική με τον JavaCC, θα δούμε στην κονσόλα του JavaCCτο ακόλουθο μήνυμα:

```
>java -classpath C:/eclipse/plugins/sf.eclipse.javacc_1.5.26/javacc.jar javaccnotEnoughLookahead.jj
Java Compiler Compiler Version 5.0 (Parser Generator)
(type "javacc" with no arguments for help)
Reading from file notEnoughLookahead.jj . . .
Warning:Choice conflict involving two expansions at
        line 24, column 16 and line 24, column 32 respectively.
        A common prefix is: <ID>
        Consider using a lookahead of 2 for earlier expansion.
Parser generated with 0 errors and 1 warnings.
```

Αυτό σημαίνει ότι όταν επεκτείνεταιτο Εκαι ψάχνει για ένα id, δεν ξέρουμε κατά πόσο αυτό το idξεκινά μια ανάθεση ή είναι μόνο μια μεταβλητή, εκτός κι αν εξετάσουμε όχι μόνο το id, αλλά και το επόμενό του token. Επομένως ο parserθα πρέπει να κοιτάξει τα 2 επόμενα σύμβολα. (Γι'αυτό στην κονσόλα του JavaCCπροτείνει να χρησιμοποιήσουμε ένα lookaheadτων 2).

Αυτά τα "σημεία επιλογής" μπορούν να εμφανιστούν σε διάφορα σημεία μέσα στη γραμματική. Σίγουρα εμφανίζονται στους διαχωριστές "|", αλλά και σε σημεία που αντιστοιχούν με $()^*$, ()+ και ()?.

Η γραμματική αλλάζει ως εξής:

Exp3.jj

```
PARSER_BEGIN(SyntaxChecker)

public class SyntaxChecker {
    public static void main(String[] args) {
        try {
            new SyntaxChecker(new java.io.StringReader(args[0])).S();
            System.out.println("Syntax is okay");
        } catch (Throwable e) {
            // Catching Throwable is ugly but JavaCC throws Error objects!
            System.out.println("Syntax check failed: " + e.getMessage());
        }
    }
}

PARSER_END(SyntaxChecker)

SKIP: { " " | "\t" | "\n" | "\r" }
TOKEN: {
    "(" | ")" | "+" | "*" | ":="
    | <NUM: (["0"-"9"])+> | <ID: (["a"-"z"])+>
}

void S(): {} { E() <EOF>                                    }
void E(): {} { LOOKAHEAD(2) <ID> ":=" E() | T() ("+" T())* }
void T(): {} { F() ("*" F())*                               }
void F(): {} { <NUM> | <ID> | "(" E() ")"                   }
```

Σχήμα 2.2: Παράδειγμα γραμματικής με χρήση του Lookahead

2.3.8Γράφοντας ένα Διερμηνέα (Interpreter)

Η δουλειά ενός parser είναι να κάνει διάφορα πράγματα καθώς κάνει το parsing. Οι συναρτήσεις για το parsingμπορούν να παίρνουν παραμέτρους, να επιστρέφουν αποτελέσματα και να παραθέτουν κομμάτια κώδικα Java. Πιο κάτω μπορούμε να δούμε πώς μπορούμε να παραθέσουμε τις εκφράσεις, γράφοντας δηλαδή ένα διερμηνέα.

Exp2.jj

```
PARSER_BEGIN(Evaluator)

public class Evaluator {

    public static void main(String[] args) throws Exception {
        int result = new Evaluator(new java.io.StringReader(args[0])).S();
        System.out.println(result);
    }
}

PARSER_END(Evaluator)

SKIP:  { " " | "\t" | "\n" | "\r"                    }
TOKEN: { "(" | ")" | "+" | "*" | <NUM: (["0"-"9"])+> }

int S(): {int sum;}
{
  sum=E() <EOF> {return sum;}
}

int E(): {int sum, x;}
{
  sum=T() ("+" x=T() {sum += x;} )* {return sum;}
}

int T(): {int sum, x;}
{
  sum=F() ("*" x=F() {sum *= x;} )* {return sum;}
}

int F(): {int x; Token n;}
{
  n=<NUM> {return Integer.parseInt(n.image);}
|
  "(" x=E() ")" {return x;}
}
```

Σχήμα 2.3:Παράδειγμα γραμματικής με διερμηνέα

Μια πιθανή εκτέλεση για την πιο πάνω γραμματική φαίνεται πιο κάτω:

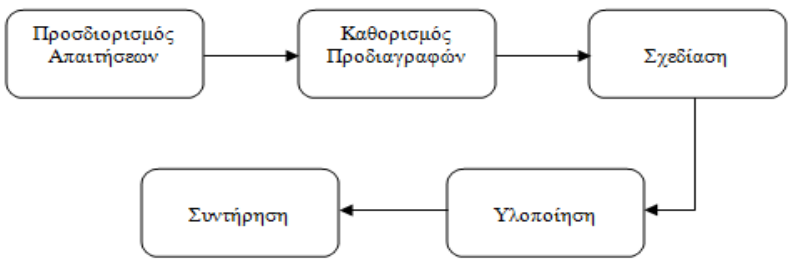
```
>java -classpath C:/eclipse/plugins/sf.eclipse.javacc_1.5.26/javacc.jar javaccExp2.jj
Java Compiler Compiler Version 5.0 (Parser Generator)
(type "javacc" with no arguments for help)
Reading from file Exp2.jj . . .
Parser generated successfully.
>java *.java
Note: Evaluator.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
> java Evaluator "34"
34
>javaEvaluator "34 * 2 + ((3 + 11))"
82
```

Κεφάλαιο 3

Κύκλος Ζωής και Ανάπτυξης Λογισμικού για τη Συντακτική Ανάλυση Χορογραφιών γραμμένες σε T-CDL

3.1Εισαγωγή

Όπως έχουμε προαναφαίρει και στο Υποκεφάλαιο 1.2,σκοπός αυτής της διπλωματικής είναι η συντακτική ανάλυση μιας T-CDL χορογραφίας, χρησιμοποιώντας το εργαλείο JavaCC.Για την επίτευξη του στόχου αυτού, θα ακολουθήσω τον Κύκλο Ζωής και Ανάπτυξης ενός πληροφοριακού συστήματος, που ορίζεται ως η όλη διαδικασία μετάβασης από φάση σε φάση.



Σχήμα3.1: *Κύκλος Ζωής και Ανάπτυξης του Πληροφοριακού Συστήματος*

Όπως φαίνεται από το σχήμα 3.1, ο Κύκλος Ζωής και Ανάπτυξης του Πληροφοριακού Συστήματος αποτελείται από 5 φάσεις:

1. **Προσδιορισμός Απαιτήσεων:**Η φάση του Προσδιορισμού των Απαιτήσεων, είναι μέγιστης σημασίας για την ανάπτυξη ενός πληροφοριακού συστήματος. Αποτελεί τον ακρογωνιαίο λίθο πάνω στον οποίο πρόκειται να δομηθούν όλες οι επόμενες φάσεις του Κύκλου Ζωής και Ανάπτυξης. Κάθε λογισμικό φτιάχνεται για να ικανοποιήσει τις ανάγκες κάποιου/ων ατόμων. Βασικός στόχος της φάσης αυτής, είναι ο καθορισμός και η καταγραφή των αναγκών αυτών. Μέσα από αυτές τις ανάγκες, θα εξαχθούν και οι απαιτούμενες λειτουργίες (απαιτήσεις) που πρέπει να διαθέτει ένα σύστημα, προκειμένου να τις ικανοποιεί.

2. **Καθορισμός Προδιαγραφών:**Η φάση Καθορισμού Προδιαγραφών, είναι το σημείο κατά το οποίο θα καταγραφούν με ακρίβεια και με τυποποιημένο τρόπο οι λειτουργίες του συστήματος προς ανάπτυξη οι οποίες πρέπει να φέρει το προϊόν (**τι** θα κάνει), ώστε να ικανοποιεί τις ανάγκες που διαπιστώθηκαν στην προηγούμενη φάση – όχι όμως ο τρόπος με τον οποίο θα διεξάγονται (**πώς** θα το κάνει). Το αποτέλεσμα της φάσης αυτής, είναι ένα έγγραφο προδιαγραφών, το οποίο χρησιμοποιείται ως συμβόλαιο μεταξύ του πελάτη και της εταιρείας ανάπτυξης του συστήματος. Επομένως θα πρέπει αφενός να είναι αρκετά λεπτομερές, ώστε να αποτελέσει τη βάση για τη μετάβαση στην επόμενη φάση του κύκλου ζωής (τη φάση Σχεδίασης), και αφετέρου πρέπει να είναι κατανοητό από τον πελάτη.

3. **Σχεδίαση:**Εφόσον έχουν καθοριστεί επακριβώς οι λειτουργίες τις οποίες θα διεξάγει πληροφοριακό σύστημα (το τι θα κάνει) και εφόσον έχουν επιλυθεί προβλήματα στο έγγραφο προδιαγραφών, όπως ασάφειες και παραλείψεις, μπορούμε να προχωρήσουμε στον καθορισμό του πώς θα το κάνει. Αυτός είναι και ο στόχος της φάσης Σχεδίασης. Κατά τη φάση αυτή, προσδιορίζονται τα μέρη από τα οποία αποτελείται το σύστημα, ο τρόπος λειτουργίας τους και ο τρόπος σύνδεσης μεταξύ τους. Επίσης, προσδιορίζεται η μορφή των απαιτούμενων βάσεων δεδομένων και των οθονών εισαγωγής και εξαγωγής δεδομένων (φόρμες και αναφορές).

Η φάση του Σχεδιασμού αποτελείται από 4 μέρη:

i. *Αρχιτεκτονική Σχεδίαση:*Κατά την Αρχιτεκτονική Σχεδίαση, καθορίζονται τα τμήματα από τα οποία απαρτίζεται το προϊόν και ο τρόπος σύνδεσής τους. Βασικό χαρακτηριστικό της φάσης αυτής είναι ότι γίνεται η υπόθεση ότι όλα τα τμήματα αυτά υπάρχουν.

ii. *Λεπτομερής Σχεδίαση:*Κατά τη Λεπτομερή Σχεδίαση, κάθε τμήμα σχεδιάζεται με λεπτομέρεια. Στο σημείο αυτό επιλέγονται οι συγκεκριμένοι αλγόριθμοι και οι δομές δεδομένων οι οποίες θα χρησιμοποιηθούν. Η σύνδεση μεταξύ των τμημάτων εδώ θεωρείται δεδομένη.

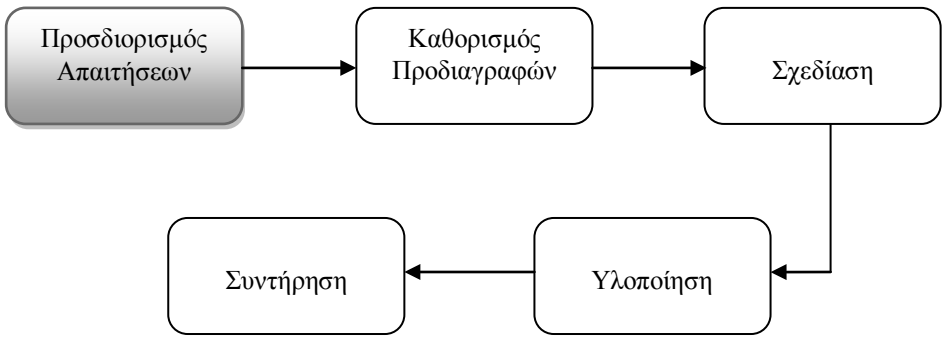
iii. *Σχεδίαση Βάσεων Δεδομένων:*Η Σχεδίαση Βάσεων Δεδομένων περιλαμβάνει την εκλέπτυνση των Διαγραμμάτων Σχέσεων Οντοτήτων, τα οποία φτιάχτηκαν κατά τη φάση των Προδιαγραφών.**(Σημ.το εργαλείο που θα υλοποιήσουμε δεν τίθεται η ανάγκη για την ύπαρξη Β.Δ., επομένως αγνοούμε το βήμα αυτό.)**

iv. *Σχεδίαση Φορμών και Αναφορών:*Αφού έχει σχεδιαστεί η απαραίτητη βάση δεδομένων, στη συνέχεια σχεδιάζονται οποιεσδήποτε φόρμες ή αναφορές χρειάζονται για τη λειτουργία του συστήματος. Οι φόρμες σύνηθες έχουν τη μορφή οθονών εισόδου δεδομένων, ενώ οι αναφορές έχουν τη μορφή οθονών εξόδου δεδομένων οι οποίες συχνά προσδιορίζονται για εκτύπωση.**(Σημ. το εργαλείο που θα χρησιμοποιήσουμε για την ανάπτυξη του συστήματός μας, είναι το περιβάλλον Eclipse και επομένως όλα τα απαραίτητα δεδομένα που χρειάζονται για εισαγωγή – στην περίπτωσή μας, μονάχα το όνομα του αρχείου που θα περιέχει τη χορογραφία γραμμένη σε T-CDL – δεν απαιτούν τη δημιουργία κάποιων φορμών και αναφορών)**

4. **Υλοποίηση:**Μετά τον Προσδιορισμό των Απαιτήσεων, τον Καθορισμό των Προδιαγραφών και τη Σχεδίαση, ακολουθεί η φάση κατά την οποία τα διάφορα μέρη του συστήματος υλοποιούνται και συνδέονται μεταξύ τους. Κατά τη φάση Υλοποίησης λοιπόν, η λεπτομερής σχεδίαση του προϊόντος υλοποιείται με κώδικα, αφού χρησιμοποιηθεί κάποια ή κάποιες γλώσσες προγραμματισμού.

5. **Συντήρηση:**Η φάση της Συντήρησης έρχεται όταν το προϊόν έχει περάσει τον έλεγχο αποδοχής και έχει παραδοθεί στον πελάτη. Οποιαδήποτε μετέπειτα αλλαγή του προϊόντος αποτελεί συντήρηση. Τα προβλήματα σε ένα λογισμικό δημιουργούνται συνήθως απ' αυτούς που το ανέπτυξαν και όχι από αυτούς που το συντηρούν. Τα λάθη όμως τα επιδιορθώνουν οι υπεύθυνοι της συντήρησης. Ο πελάτης συχνά δεν κατανοεί ότι η συντήρηση μπορεί να είναι ιδιαίτερα επίπονη ή πολλές φορές αδύνατη.

3.2Προσδιορισμός Απαιτήσεων

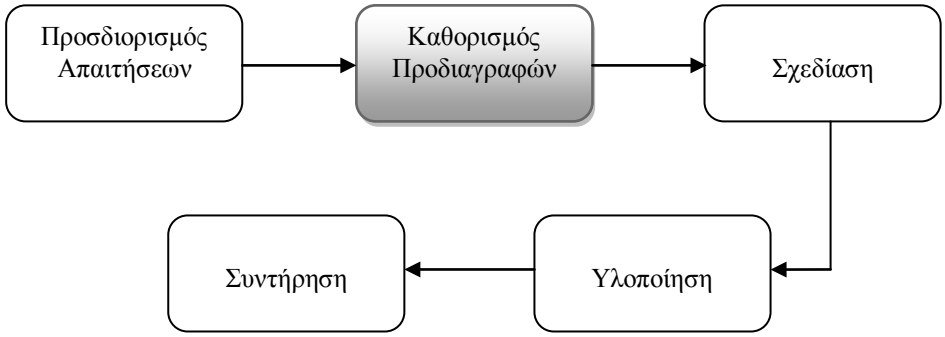


Σχήμα 3.2: *Κύκλος Ζωής και Ανάπτυξης του Πληροφοριακού Συστήματος- Φάση Προσδιορισμού Απαιτήσεων*

Στο παρόν στάδιο, χρειαζόμαστε ένα εργαλείο το οποίο θα ελέγχει κατά πόσο μια χορογραφία είναι ορθά και συντακτικά γραμμένη στη γλώσσα T-CDL. Αυτό μπορεί να επιτευχθεί μέσω ενός συντακτικούαναλυτή για τη T-CDL, όπου ο χρήστης θα

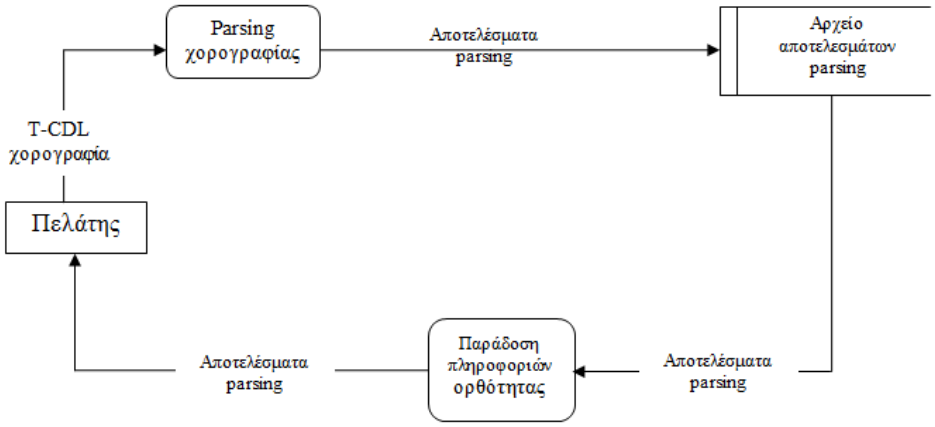
καλείται να δώσει ως είσοδο στο πρόγραμμα ένα αρχείο το οποίο θα περιλαμβάνει μια χορογραφία γραμμένη στη γλώσσα T-CDL.

3.3Καθορισμός Προδιαγραφών



Σχήμα 3.3: *Κύκλος Ζωής και Ανάπτυξης του Πληροφοριακού Συστήματος- Φάση Καθορισμού Προδιαγραφών*

Διάγραμμα Ροής Δεδομένων για τον έλεγχο μιας χορογραφίας:

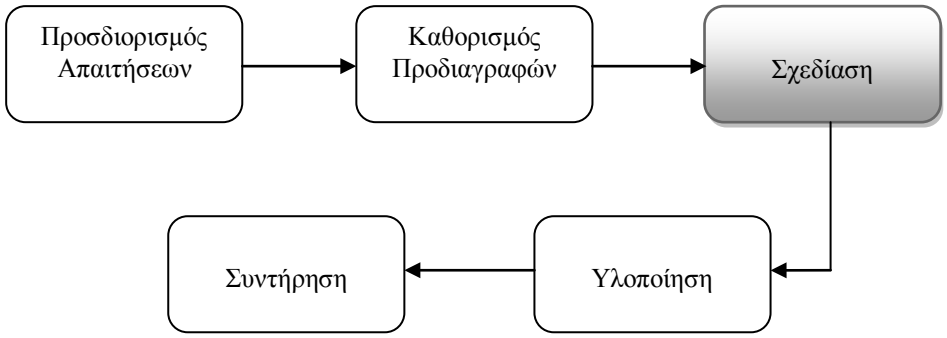


Σχήμα3.4: *Διάγραμμα Ροής Δεδομένων (Δ.Ρ.Δ) για τη συντακτική ανάλυσημιας χορογραφίας γραμμένη σε T-CDL*

Το πιο πάνω Διάγραμμα Ροής Δεδομένων, παρουσιάζει την υλοποίηση του συντακτικού αναλυτή που θα κατασκευάσουμε, ο οποίος θα δέχεται ως είσοδο μια χορογραφία γραμμένη σε T-CDL - σύμφωνα με τη σύνταξη που έχει προταθεί στο Υποκεφάλαιο 2.2.1. Ο συντακτικός αναλυτής θα εντοπίζει τυχόν λάθη, όσο αφορά τη σύνταξη της εν λόγω χορογραφίας που θα δοθεί ως είσοδος στο εργαλείο αυτό και θα ενημερώνει το χρήστη με τα αντίστοιχα μηνύματα για το τι ακριβώς έκανε λάθος. Αν η

χορογραφία που θα δώσει ως είσοδο είναι ορθή συντακτικά βάσει της γραμματικής της T-CDL, τότε απλά θα του εμφανίζεται ένα μήνυμα το υ τύπο υ “Parsinghassuccessfullycompletedwithnoerrors.”

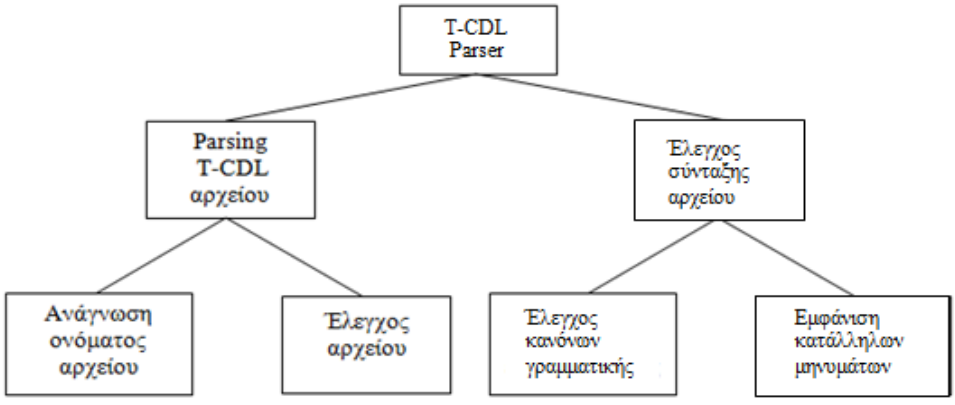
3.4Σχεδίαση



Σχήμα 3.5: *Κύκλος Ζωής και Ανάπτυξης του Πληροφοριακού Συστήματος- Φάση Σχεδίασης*

3.4.1Αρχιτεκτονική Σχεδίαση

Τμήματα από τα οποία αποτελείται το πρόγραμμα parsing T-CDL αρχείου



Σχήμα 3.6: *Τμήματα από τα οποία αποτελείται το πρόγραμμα parsingT-CDLαρχείου*

Το πιο πάνω διάγραμμα, δείχνει (από πάνω προς τα κάτω) την κλήση ενός τμήματος από κάποιο άλλο. Για παράδειγμα, το τμήμα «ParsingT-CDLαρχείου» καλεί

τα τμήματα «Ανάγνωση ονόματος αρχείου» και «Έλεγχος αρχείου». Κάθε τμήμα στην ουσία αποτελεί και μια συνάρτηση.

3.4.2 Λεπτομερής Σχεδίαση

Όνομα Τμήματος:	T-CDL Parser
Τύπος Επιστροφής:	Κανένα
Παράμετροι Εισόδου:	Καμία
Παράμετροι Εξόδου:	Καμία
Αρχεία τα οποία χρησιμοποιεί:	Κανένα
Αρχεία τα οποία τροποποιεί:	Κανένα
Καλούμενα τμήματα:	• **Parsing T-CDL Αρχείου** Είσοδος: *Καμία* Έξοδος:<*όνομα αρχείου.txt*> • **Έλεγχος Σύνταξης Αρχείου** Είσοδος:<*όνομα αρχείου.txt*> Έξοδος:*Καμία*
Περιγραφή: Το τμήμα αυτό χρησιμοποιεί το τμήμα **Parsing T-CDL Αρχείου** για να εξάγει το όνομα του αρχείου που δίνει ο χρήστης και για να κάνει το parsing. Ακολούθως καλεί το τμήμα **Έλεγχος Σύνταξης Αρχείου** για ναελέγξει κατά πόσο το αρχείο που έδωσε ο χρήστης, ακολουθεί τη σύνταξη της T-CDL.	

Πίνακας 3.1

Όνομα Τμήματος:	Parsing T-CDL Αρχείου
Τύπος Επιστροφής:	Κανένα
Παράμετροι Εισόδου:	Καμία
Παράμετροι Εξόδου:	Καμία
Αρχεία τα οποία χρησιμοποιεί:	Κανένα
Αρχεία τα οποία τροποποιεί:	Κανένα
Καλούμενα τμήματα:	• **Ανάγνωση Ονόματος Αρχείου** Είσοδος:*Καμία* Έξοδος:<*Όνομα αρχείου*>*.txt* • **Έλεγχος Αρχείου** Είσοδος:<*Όνομα αρχείου*>*.txt* Έξοδος:*Μήνυμα υποβολής ύπαρξης ή μη ύπαρξης του αρχείου*
Περιγραφή: Το τμήμα αυτό χρησιμοποιεί το τμήμα **Ανάγνωση Ονόματος Αρχείου** για να εξάγει το όνομα του αρχείου που δίνει ο χρήστης και ακολούθως καλεί το τμήμα **Έλεγχος Αρχείου** για ναδιαπιστωθεί κατά πόσο υπάρχει το αρχείο ή όχι, εμφανίζοντας το ανάλογο μήνυμα.	

Πίνακας3.2

Όνομα Τμήματος:	Ανάγνωση Ονόματος Αρχείου
Τύπος Επιστροφής:	String
Παράμετροι Εισόδου:	Καμία
Παράμετροι Εξόδου:	Καμία
Αρχεία τα οποία χρησιμοποιεί:	Κανένα
Αρχεία τα οποία τροποποιεί:	Κανένα
Καλούμενα τμήματα:	Κανένα
Περιγραφή: Το πρόγραμμα καλείται από το χρήστη, μέσω του προγράμματος Eclipse και δίνει ως παράμετρο το όνομα του αρχείου. Χρησιμοποιώντας κατάλληλες εντολές της γλώσσας προγραμματισμού, το τμήμα αυτό διαβάζει τις παραμέτρους που περάστηκαν από το χρήστη, παίρνει το <*όνομα αρχείου*> και το επιστρέφει ως τιμή επιστροφής.	

Πίνακας 3.3

Όνομα Τμήματος:	Έλεγχος Αρχείου
Τύπος Επιστροφής:	String
Παράμετροι Εισόδου:	<*όνομα αρχείου.txt*>
Παράμετροι Εξόδου:	Καμία
Αρχεία τα οποία χρησιμοποιεί:	<*όνομα αρχείου.txt*>
Αρχεία τα οποία τροποποιεί:	Κανένα
Καλούμενα τμήματα:	Κανένα
Περιγραφή: Το τμήμα αυτό καλείται από το χρήστη, και δίνει ως παράμετρο το <*όνομα αρχείου*>. Χρησιμοποιώντας κατάλληλες εντολές της γλώσσας προγραμματισμού, το τμήμα αυτό παίρνει το <*όνομα αρχείου*>, ελέγχει κατά πόσο αυτό το αρχείο υπάρχει και επιστρέφει το ανάλογο μήνυμα στην περίπτωση που δεν υπάρχει.	

Πίνακας 3.4

Όνομα Τμήματος:	Έλεγχος Σύνταξης Αρχείου
Τύπος Επιστροφής:	Κανένα
Παράμετροι Εισόδου:	*<όνομα αρχείου.txt>*
Παράμετροι Εξόδου:	Καμία
Αρχεία τα οποία χρησιμοποιεί:	Κανένα
Αρχεία τα οποία τροποποιεί:	Κανένα
Καλούμενα τμήματα:	• **Έλεγχος Κανόνων Γραμματικής** Είσοδος:*<όνομα αρχείου.txt>* Έξοδος:*Εντοπισμός Λαθών* • **Εμφάνιση Κατάλληλων Μηνυμάτων** Είσοδος:*Εντοπισμός Λαθών* Έξοδος:*Εμφάνιση Μηνύματος Λάθους αν υπάρχουν,διαφορετικά ParsingOk*
Περιγραφή: Το τμήμα αυτό χρησιμοποιεί το τμήμα **Έλεγχος Κανόνων Γραμματικής** παίρνοντας ως είσοδο*<όνομα αρχείου.txt>*, για να εξάγει πιθανά λάθη – εάν υπάρχουν - και ακολούθως καλεί το τμήμα**Εμφάνιση Κατάλληλων Μηνυμάτων** για να εμφανίσει το κατάλληλο μήνυμα εξηγώντας στο χρήστη ποιο είναι το λάθος στο αρχείο εισόδου, διαφορετικά εάν δεν υπάρχουν λάθη στο αρχείο εισόδου τότε απλά του εμφανίζει ένα μήνυμα “Parsinghassuccessfullycompletedwithnoerrors!”	

Πίνακας3.5

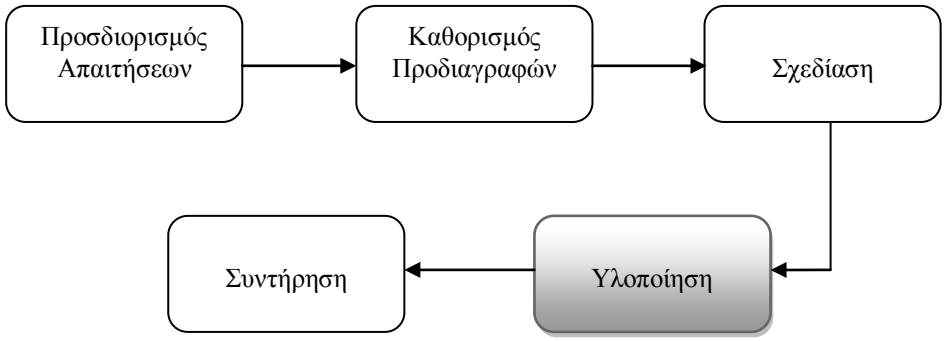
Όνομα Τμήματος:	Έλεγχος Κανόνων Γραμματικής
Τύπος Επιστροφής:	String
Παράμετροι Εισόδου:	*<όνομα αρχείου.txt>*
Παράμετροι Εξόδου:	Καμία
Αρχεία τα οποία χρησιμοποιεί:	*<όνομα αρχείου.txt>*
Αρχεία τα οποία τροποποιεί:	Κανένα
Καλούμενα τμήματα:	Κανένα
Περιγραφή: Το τμήμα αυτό καλείται με παράμετρο το*<όνομα αρχείου.txt>* καιπερνά από τους κανόνες της γραμματικής, βρίσκοντας ποιος ταιριάζει στο αρχείο εισόδου. Σε περίπτωση που δε βρεθεί κάποιος κανόνας που να αντιστοιχεί με κάποιο κομμάτι του αρχείου εισόδου, τότε επιστρέφει το σημείο που εντοπίστηκε το λάθος.	

Πίνακας 3.6

Όνομα Τμήματος:	Εμφάνιση Κατάλληλων Μηνυμάτων
Τύπος Επιστροφής:	Κανένα
Παράμετροι Εισόδου:	*Εντοπισμός Λαθών*
Παράμετροι Εξόδου:	Καμία
Αρχεία τα οποία χρησιμοποιεί:	*<όνομα αρχείου.txt>*
Αρχεία τα οποία τροποποιεί:	Κανένα
Καλούμενα τμήματα:	Κανένα
Περιγραφή: Το τμήμα αυτό καλείται με παραμέτρους, το σημείο/α όπου υπάρχει κάποιο λάθος (εάν υπάρχει) όσο αφορά τους κανόνες της γραμματικής και επιστρέφει στο χρήστη το κατάλληλο μήνυμα εξηγώντας του ποιο είναι το λάθος στο αρχείο εισόδου, διαφορετικά εάν δεν υπάρχουν λάθη στο αρχείο εισόδου τότε απλά του εμφανίζει ένα μήνυμα "Parsinghassuccessfullycompletedwithnoerrors!"	

Πίνακας 3.7

3.5Υλοποίηση



Σχήμα 3.7:*Κύκλος Ζωής και Ανάπτυξης του Πληροφοριακού Συστήματος- Φάση Υλοποίησης*

3.5.1 Επιλογή Γλώσσας Προγραμματισμού

Η γλώσσα προγραμματισμού που επιλέξαμε για να υλοποιήσουμε το parser μιας T-CDLχορογραφίας, είναι η αντικειμενοστραφής γλώσσαJava.Η επιλογή μας για τη γλώσσα αυτή δεν είναι τυχαία, καθ' ότι το εργαλείο στο οποίο στηρίζεται η υλοποίησή μας, είναι το JavaCC (JavaCompilerCompiler).

OJavaCC, είναι ο πιο δημοφιλής παραγωγός συντακτικού αναλυτή (parsergenerator)που χρησιμοποιείται σε Javaεφαρμογές. Ένας parsergenerator, είναι ένα εργαλείο το οποίο διαβάζει μια προδιαγραφή γραμματικής και τη μετατρέπει σε ένα πρόγραμμα το οποίο αναγνωρίζει κομμάτια που αντιστοιχούν στη εν λόγω γραμματική.Επιπλέον, ο JavaCC παρέχει και άλλες δυνατότητες που σχετίζονται με την

παραγωγή μεταφραστή, όπως είναι το κτίσιμο δέντρων (treebuilding), μέσω ενός εργαλείου που λέγεται JJTreeκαι εμπερικλείεται στο JavaCC, καθώς επίσης και διάφορες ενέργειες, αποσφαλμάτωση κτλ.Περισσότερες λεπτομέρειες για τον JavaCCαναφέραμε στο Υποκεφάλαιο 2.3,καθώς και στοάρθρο [8]

3.5.2 ΓραμματικήT-CDL

Η γραμματική της T-CDL, στηρίζεται στη σύνταξη που προτάθηκε στην προηγούμενη διατριβή [1] και που έχουμε αναφέρει στο Υποκεφάλαιο 2.2.1.στους Πίνακες 1 και 2.

Για την υλοποίηση του parser, έγιναν κάποιες προσαρμογές οι οποίες περιγράφοναι πιο κάτω.

Συγκεκριμένα οι αλλαγές έγιναν στο Αίτημα, στην Απάντηση, στο Αίτημα – Απάντηση, στη Δραστηριότητα Διαδοχής, στην Επιλογή, στη Γενική Επιλογή στον Παραλληλισμό και στη Δραστηριότητα Προθεσμίας. Οι αλλαγές φαίνονται πιο κάτω:

Αίτημα:

$$comm(request(f.x,t.y,c),rec,op)$$

Απάντηση:

$$comm(answer(f.x,t.y,c),rec,op)$$

Αίτημα–Απάντηση:

$$comm(requestAnswer(request(f.x,t.y,c_i),answer(f.x,t.y,c_j),rec,op))$$

Συνθήκη:

$$condition(g?A)$$

Επανάληψη:

$$repetition(g*A)$$

Μονάδα Ελέγχου:

$$workUnit(g:A:g)$$

ΔραστηριότηταΔιαδοχής:

$$sequence(A;B)$$

Επιλογή:

$$choice(A+B)$$

Γενική Επιλογή:

$$case(g_1:A,g_2:B)$$

Παραλληλισμός:

$$parallelism(A \| B)$$

Δραστηριότητα Προθεσμίας:

$$deadline(A, d, E, F)$$

Εισήχθηκαν νέοι κανόνες για να διαβάζεται αρχικά η χορογραφία και ακολούθως να γίνεται η δήλωση των ρόλων καθώς και της κύριας συνάρτησής της. Θα μιλήσουμε αναλυτικότερα πιο κάτω για το τι ακριβώς κάνει η γραμματική που φαίνεται στους πίνακες 4.1 και 4.2.

Η γραμματική της T-CDL δίνεται ως ακολούθως:

```
LETTER ::= [a-z] | [A-Z]
DIGIT ::= [0-9]
ID ::= LETTER( LETTER | DIGIT | "_" )*
PRIMARY ::= DIGIT+
EQOP ::= "==" | "!="
        TCDL_FILE :: = CODE EOF
        CODE :: = CHOREOGRAPHY DECLARATION
        CHOREOGRAPHY :: = ID "[" CHOR_ROLES "," MAIN_FUN "]"
        CHOR_ROLES :: = "(" ( ID "," )+ ID ")"
                      | ID
        MAIN_FUN :: = ID

DECLARATION :: = ID " = " ID "[" ROLE_X "," ROLE_OP "]" DECLARATION
                 | "~" D
ROLE_X :: = "(" ( ID ",")+ ID ")"
ROLE_OP :: = ( ID ",")+ ID
D :: = ID " = " ACTIVITY (D)*
```

```
ACTIVITY :: = BA
           | CONDITION
           | REPETITION
           | WORK _UNIT
           | SEQUENCE _ ACTIVITY
           | CHOICE
           | GENERAL _CHOICE
           | PARALLELISM
           | DELAY ACTIVITY
           | DEADLINE
```

Πίνακας 4.1

```
BA ::="skip"
    | "comm" "(" (REQUEST | ANSWER | REQUEST _ ANSWER) ")"
REQUEST ::= "request" "("ID "."ID ","ID "."ID ","ID ")" ","
                        ( "(" ID "." ID ":=" EXP
                            ( "," ID "." ID ":=" EXP)* ")"
                          | ID "." ID ":=" EXP) ","
                        ( "(" ID "." ID ":=" EXP
                            ( "," ID "." ID ":=" EXP)* ")"
                          | ID "." ID ":=" EXP) "," OP ")"
ANSWER ::= "answer" "("ID "."ID ","ID "."ID ","ID ")" ","
                        ( "(" ID "." ID ":=" EXP
                            ( "," ID "." ID ":=" EXP)* ")"
                          | ID "." ID ":=" EXP) ","
                        ( "(" ID "." ID ":=" EXP
                            ( "," ID "." ID ":=" EXP)* ")"
                          | ID "." ID ":=" EXP ) "," OP ")"
REQUEST _ ANSWER ::= "requestAnswer" "("
                          "request" "("ID "."ID ","ID "."ID ","ID ")" ","
                          "answer" "("ID "."ID ","ID "."ID ","ID ")" ","
                          ( "(" ID "." ID ":=" EXP
                            ( "," ID "." ID ":=" EXP)* ")"
                          | ID "." ID ":=" EXP) ","
                        ( "(" ID "." ID ":=" EXP
                            ( "," ID "." ID ":=" EXP)* ")"
                          | ID "." ID ":=" EXP) ","
                          OP ")"
```

```
OP ::= "e" | DELAY | "delta"
DELAY ::= "xr" "(" DIGIT ")"
EXP ::= TERM ("+" TERM | "−" TERM)*
TERM ::= FACTOR ("*" FACTOR | "/" FACTOR)
FACTOR ::= ID ( "." ID)?
        | "\"" ID "\""
        | PRIMARY
        | "true"
        | "false"
        | "(" EXP ")"
```

Πίνακας 4.2

```
CONDITION ::= GUARD "?" ACTIVITY
REPETITION ::= GUARD "*" ACTIVITY
WORK_UNIT ::= GUARD ":" ACTIVITY ":" GUARD
SEQUENCE ::= "sequence" "(" (ACTIVITY ";")+ ACTIVITY ")"
CHOICE ::= "choice" "(" (ACTIVITY "+")+ ACTIVITY ")"
GENERAL_CHOICE ::= "case" "(" GUARD ":" ACTIVITY "," GUARD ":" ACTIVITY ")"
PARALLELISM ::= "parallelism" "(" (ACTIVITY "||")+ ACTIVITY ")"
DEADLINE ::= "deadline" "(" ACTIVITY "," DIGIT "," ACTIVITY "," ACTIVITY ")"
GUARD ::= BOOL_OR
BOOL_OR ::= BOOL_AND ("|" BOOL_AND)*
BOOL_AND ::= COMP_EXP ("&" COMP_EXP)*
COMP_EXP ::= SIMPLE_EXP (("==" | "!=") SIMPLE_EXP)*
SIMPLE_EXP ::= EXP
        | "(" GUARD ")"
```

Πίνακας 4.3

3.5.3Επεξήγηση Γραμματικής:

- **LETTER**:Αποτελείται από μικρά ή κεφαλαία γράμματα.
- **DIGIT:**Αποτελείται από ένα ψηφίο από το το 0-9.
- **ID**: Θα ξεκινά πάντοτε με LETTER και στη συνέχεια θα ακολουθείται από LETTER ή DIGITή underscore.
- **PRIMARY**: Θα έχει ένα DIGIT μία ή περισσότερες φορές.
- **EQOP**: Αποτελείται από το λογικό ίσον "==" και το λογικό άνισο "! =".
- **TCDL_FILE**: Είναι η αρχή της γραμματικής μας και αποτελείται από το CODE μέχρι να βρειEOF.
- **CODE**: Είναι όλος ο κώδικας που θα υπάρχει στο αρχείο εισόδου και θα ξεκινάει με μια χορογραφίαCHOREOGRAPHYκαι ακολούθως με τη δήλωση των ρόλων και της κύριας συνάρτησης της χορογραφίας που γίνεται με το DECLARATION.
- **CHOREOGRAPHY**: Θα ξεκινάει με ένα ID, μετά θα βρίσκει "[" ακολούθως θα γίνεται η αναφορά των CHOR_ROLES, μέχρι να συναντήσει ","όπου ακολουθεί η αναφορά της MAIN_FUNκαι αμέσως μετά θα βρίσκει "]".

  PurchaseOrder[(buyer, seller, credit, inv), purchaseOrder]

  CHOR_ROLES: Εάν δηλώνονται πολλά ID's, τότε αυτά θα εμπερικλείονται σε παρενθέσεις και θα διαχωρίζονται με ",":

  *PurchaseOrder [**(buyer, seller, credit, inv)**, purchaseOrder],*

  ενώ σε αντίθετη περίπτωση θα δηλώνεται ένα ID:

  *PurchaseOrder[**buyer**, purchaseOrder]*
- **MAIN_FUN**: Αποτελείται από ένα ID:

  *PurchaseOrder[(buyer, seller, credit, inv), **purchaseOrder**]*
- **DECLARATION**: Θα ξεκινάει με ένα ID, στη συνέχεια θα βρίσκει "=" και αιμέσως πάλι ID. Μετά θα βρίσκει "[" ακολούθως θα γίνεται η αναφορά των ROLE_X, μέχρι να συναντήσει ","όπου ακολουθεί η αναφορά της ROLE_OPκαι αμέσως μετά θα βρίσκει "]". Μετά ακολουθεί και πάλι το DECLARATIONκαι αυτό θα συνεχίζεται μέχρι να βρεθεί το σύμβολο "~", το οποίο θα δηλώνει τον τερματισμό των δηλώσεων των ρόλων της χορογραφίας επομένως ακολουθεί ο ορισμός της κύριας συνάρτησης της χορογραφίας με το D:

buyer = Buyer[(po, poAck, poResp, poRej, poState),(PoReqOP, DisplayOP)]

...

credit = Credit[(ccReq, ccResp, ccState), CreditCheckOP] ~

- **ROLE_X:**Εάν δηλώνονται πολλά ID's, τότε αυτά θα εμπερικλείονται σε παρενθέσεις και θα διαχωρίζονται με ",":

  *credit = Credit[**(ccReq, ccResp, ccState)**, CreditCheckOP] ~*

  ενώ σε αντίθετη περίπτωση θα δηλώνεται ένα ID:

  *credit = Credit[**ccReq**, CreditCheckOP] ~*

- **ROLE_OP:**Εάν δηλώνονται πολλά ID's, τότε αυτά θα εμπερικλείονται σε παρενθέσεις και θα διαχωρίζονται με ",":

  *buyer = Buyer[(po, poAck, poResp, poRej, poState),**(PoReqOP, DisplayOP)**]*

  ενώ σε αντίθετη περίπτωση θα δηλώνεται ένα ID:

  *buyer = Buyer[(po, poAck, poResp, poRej, poState),**PoReqOP**]*

- **D:**Θα ξεκινάει με ένα ID, στη συνέχεια θα βρίσκει "=" καιαμέσως μετά θα βρίσκει μια δραστηριότητα ACTIVITY.
- **ACTIVITY**: Θα μπορεί να είναι μια βασική δραστηριότητα BAείτε μια συνθήκη CONDITION, είτε μια επανάληψηREPETITION, είτε μια μονάδα εργασίας WORK_UNIT,είτε μια δραστηριότητα διαδοχής SEQUENCE_ACTIVITY, είτε μια επιλογή CHOICE, είτε μια γενική επιλογή GENERAL_CHOICE, είτε ένας παραλληλισμός PARALLELISM, είτε ένας τελεστής καθυστέρησης DELAY που ακολουθείται από μια δραστηριότητα ACTIVITY, είτε μια δραστηριότητα προθεσμίαςDEADLINE.
- **BA**: Θα μπορεί να είναι είτε καμιά ενέργεια "skip", είτε θα ξεκινά με τη λέξη "comm", θα ακολουθείται από "(" και θα μπορεί να συνεχίζει είτε με ένα αίτημα REQUEST, είτε με μια απάντηση ANSWER, είτε με ένα αίτημα-απάντησηREQUEST_ANSWER. Με το τέλος ενό ς από αυτά τα τρία, θα ακολουθεί ")".
- **REQUEST**: Θα ξεκινά με τη λέξη "request", θα ακολουθεί "(", μετά θα βρίσκει ID, θα ακολουθεί "." και στη συνέχεια θα βρίσκει ID, μετά "," θα ξαναβρεί ID, το οποίο θα ακολουθείται με ".", ενώ στη συνέχεια θα βρίσκει και πάλι ID, θα ακολουθείται από ",", μετά από ID και ")".Η τιμή της μεταβλητής που βρίσκεται

στο δεύτερο ID του ρόλου που βρίσκεται στο πρώτο ID, θα αποστέλνεται στη μεταβλητή που βρίσκεται στο τέταρτο ID του ρόλου που βρίσκεται στο τρίτο ID, μέσω του καναλιού που βρίσκεται στο πέμπτο ID. Στη συνέχεια ακολουθεί το τμήμα rec, το οποίο και θα προσπαθήσουμε να εξηγήσουμε πιο συνοπτικά από τα υπόλοιπα: Εάν δηλώνονται πολλά ID"."ID":="EXP , τότε αυτά θα εμπερικλείονται σε παρενθέσεις και θα διαχωρίζονται με “,”, ενώ σε αντίθετη περίπτωση θα δηλώνεται μια φορά το ID"."ID":="EXP και θα ακολουθείται από “,”. Σε αυτό το σημείο γίνεται η ανάθεση μιας έκφρασης σε μια μεταβλητή που ανήκει στο ρόλο πο υ βρίσκεται στο πρώτο ID. Ομοίως, γίνεται και η ανάθεση μιας έκφρασης σε μια μεταβλητή που ανήκει στο ρόλο που βρίσκεται στο τρίτο ID, με τη διαφορά ότι εάν το ID"."ID":="EXP υπάρχει μόνο μία φορά, το “,” που ακολουθεί δε σημαίνει ότι θα υπάρξει κι άλλη ανάθεση μιας έκφρασης σε μια μεταβλητή που ανήκει σε κάποιο ρόλο, αλλά θα ακολουθείται από ένα OP και “)”που σηματοδοτεί το τέλος του REQUEST.

- **ANSWER**: Θα ξεκινά με τη λέξη “answer”, θα ακολουθεί “(“, μετά θα βρίσκει ID, θα ακολουθεί “.” και στη συνέχεια θα βρίσκει ID, μετά “,” θα ξαναβρεί ID, το οποίο θα ακολουθείται με “.”, ενώ στη συνέχεια θα βρίσκει και πάλι ID, θα ακολουθείται από “,”, μετά από ID και “)”. Η τιμή της μεταβλητής που βρίσκεται στο τέταρτο ID του ρόλου που βρίσκεται στο τρίτο ID, θα αποστέλνεται στη μεταβλητή που βρίσκεται στο δεύτερο ID του ρόλου που βρίσκεται στο πρώτο ID, μέσω του καναλιού που βρίσκεται στο πέμπτο ID. Στη συνέχεια ακολουθεί το τμήμα rec, το οποίο και θα προσπαθήσουμε να εξηγήσουμε πιο συνοπτικά από τα υπόλοιπα: Εάν δηλώνονται πολλά ID"."ID":="EXP , τότε αυτά θα εμπερικλείονται σε παρενθέσεις και θα διαχωρίζονται με “,”, ενώ σε αντίθετη περίπτωση θα δηλώνεται μια φορά το ID"."ID":="EXP και θα ακολουθείται από “,”. Σε αυτό το σημείο γίνεται η ανάθεση μιας έκφρασης σε μια μεταβλητή που ανήκει στο ρόλο που βρίσκεται στο πρώτο ID. Ομοίως, γίνεται και η ανάθεση μιας έκφρασης σε μια μεταβλητή που ανήκει στο ρόλο που βρίσκεται στο τρίτο ID, με τη διαφορά ότι εάν το ID"."ID":="EXP υπάρχει μόνο μία φορά, το “,” που ακολουθεί δε σημαίνει ότι θα υπάρξει κι άλλη ανάθεση μιας έκφρασης σε μια μεταβλητή που ανήκει σε κάποιο ρόλο, αλλά θα ακολουθείται από ένα OP και “)”που σηματοδοτεί το τέλος του ANSWER

- **REQUEST_ANSWER**: Θα ξεκινά με τη λέξη “requestAnswer” θα ακολουθεί “(“και στη συνέχεια θα βρίσκει τη λέξη “request”, θα ακολουθεί “(“, μετά θα βρίσκει ID, θα ακολουθεί “.” και στη συνέχεια θα βρίσκει ID, μετά “,” θα ξαναβρεί ID, το οποίο θα ακολουθείται με “.”, ενώ στη συνέχεια θα βρίσκει και πάλι ID, θα ακολουθείται από “,”, μετά από ID και “)”που σηματοδοτεί το τέλος του REQUEST. Έπειτα θα βρίσκει τη λέξη “answer”, θα ακολουθεί “(“, μετά θα βρίσκει ID, θα ακολουθεί “.” και στη συνέχεια θα βρίσκει ID, μετά “,” θα ξαναβρεί ID, το οποίο θα ακολουθείται με “.”, ενώ στη συνέχεια θα βρίσκει και πάλι ID, θα ακολουθείται από “,”, μετά από ID και “)”που σηματοδοτεί το τέλος του ANSWER. Σε αυτή την αλληλεπίδραση, στέλνεται αρχικά ένα αίτημα από τον ρόλο που βρίσκεται στο πρώτο ID, στο ρόλο που βρίσκεται στο τρίτο ID, μέσω του καναλιού που βρίσκεται στο πέμπτο ID και ακολούθως στέλνεται η απάντηση από τον ρόλο που βρίσκεται στο όγδοο ID, στο ρόλο που βρίσκεται στο έκτο ID, μέσω του καναλιού που βρίσκεται στο δέκατο ID. Στη συνέχεια ακολουθεί το τμήμα rec, το οποίο και θα προσπαθήσουμε να εξηγήσουμε πιο συνοπτικά από τα υπόλοιπα: Εάν δηλώνονται πολλά ID"."ID":="EXP , τότε αυτά θα εμπερικλείονται σε παρενθέσεις και θα διαχωρίζονται με “,”, ενώ σε αντίθετη περίπτωση θα δηλώνεται μια φορά το ID"."ID":="EXP και θα ακολουθείται από “,”. Σε αυτό το σημείο γίνεται η ανάθεση μιας έκφρασης σε μια μεταβλητή που ανήκει στο ρόλο που βρίσκεται στο πρώτο ID. Ομοίως, γίνεται και η ανάθεση μιας έκφρασης σε μια μεταβλητή που ανήκει στο ρόλο που βρίσκεται στο τρίτο ID, με τη διαφορά ότι εάν το ID"."ID":="EXP υπάρχει μόνο μία φορά, το “,” που ακολουθεί δε σημαίνει ότι θα υπάρξει κι άλλη ανάθεση μιας έκφρασης σε μια μεταβλητή που ανήκει σε κάποιο ρόλο, αλλά θα ακολουθείται από ένα OP και “)”που σηματοδοτεί το τέλος του REQUEST_ANSWER.
- **A**: Μπορεί να ακολουθείται από IDκαι στη συνέχεια ένα από “;” ή “+” ή “||” το οποίο θα ακολουθείται από τη δραστηριότητα ACTIVITY, είτε θα είναι ένα IDτο οποίο θα ακολουθείται από τη δήλωση D.
- **OP**: Θα παίρνει ένα από τα “e” ή DELAYή την καθυστέρηση απροσδιόριστου χρόνου δ που δηλώνεται με το“delta”.
- **DELAY**: Ξεκινά με τη λέξη “xr”, ακολουθεί “(“, μετά ένα ψηφίο DIGITκαι στη συνέχεια “)”.

- **EXP**: Αποτελείται από ένα TERMκαι στη συνέχεια ακολουθείται 0 ή πολλές φορές με “+” TERMή “-” TERM.
- **TERM**: Αποτελείται από ένα FACTORκαι στη συνέχεια ακολουθείται 0 ή πολλές φορές με “*” FACTORή “/” FACTOR.
- **FACTOR**: Αποτελείται από ένα IDτο οποίο μπορεί να ακολουθείταιμια φορά ή καθόλου από “. ID”,είτε από ένα IDτο οποίο εμπερικλείεται σε εισαγωγικά, είτε από ένα PRIMARYείτε από “true” είτε από“false” είτε από “(“ το οποίο ακολουθείται από EXP και στη συνέχεια από”)”
- **CONDITION**:Αποτελείται από ένα φρουρό GUARD και στη συνέχεια ακολουθείται από ένα “?” και ένα ACTIVITY.
- **REPETITION**:Αποτελείται από ένα φρουρό GUARD και στη συνέχεια ακολουθείται από ένα “*” και ένα ACTIVITY.
- **WORK_UNIT**:Αποτελείται από ένα φρουρό GUARD και στη συνέχεια ακολουθείται από ένα “:”, έπειτααπό ένα ACTIVITY, ακολούθως από ένα “:” και ξανά από ένα φρουρό GUARD.
- **SEQUENCE**: Θα ξεκινά με τη λέξη “sequence”, θα ακολουθεί “(“ και στη συνέχεια θα βρίσκει ένα ACTIVITY το οποίο θα ακολουθείται από ένα “;” και θα επαναλαβάνεται μία ή περισσότερες φορές και έπειτα θα τελειώνει με ένα ACTIVITY.
- **CHOICE**: Θα ξεκινά με τη λέξη “choice”, θα ακολουθεί “(” και στη συνέχεια θα βρίσκει ένα ACTIVITY το οποίο θα ακολουθείται από ένα “+” και θα επαναλαβάνεται μία ή περισσότερες φορές και έπειτα θα τελειώνει με ένα ACTIVITY.
- **GENERAL_CHOICE**: Θα ξεκινά με τη λέξη “case”, θα ακολουθεί “(” και στη συνέχεια θα βρίσκει αρχικά ένα GUARD, μετά θα ακολουθείται από “:” έπειτα θα βρίσκει ένα ACTIVITY, στη συνέχεια “,”, το οποίο θα ακολουθείται από ένα GUARD, ακολούθως θα βρίσκει “:” και θα τελειώνει με ένα ACTIVITY.
- **PARALLELISM**: Θα ξεκινά με τη λέξη “parallelism”, θα ακολουθεί “(” και στη συνέχεια θα βρίσκει ένα ACTIVITY το οποίο θα ακολουθείται από ένα “||” και θα επαναλαβάνεται μία ή περισσότερες φορές και έπειτα θα τελειώνει με ένα ACTIVITY.
- **DEADLINE**: Θα ξεκινά με τη λέξη “deadline”, θα ακολουθεί “(” και στη συνέχεια θα βρίσκει αρχικά ένα ACTIVITY, μετά θα ακολουθείται από “,”

έπειτα θα βρίσκει ένα DIGIT, στη συνέχεια “,”, το οποίο θα ακολουθείται από ένα ACTIVITY και ακολούθως θα βρίσκει “,” και θα τελειώνει με ένα ACTIVITY.

- **GUARD**: Αποτελείται από ένα BOOL_OR.
- **BOOL_OR**: Αποτελείται από ένα BOOL_ANDκαι στη συνέχεια ακολουθείται 0 ή πολλές φορές με “|” BOOL_AND.
- **BOOL_AND**:Αποτελείται από ένα COMP_EXPκαι στη συνέχεια ακολουθείται 0 ή πολλές φορές με “&” COMP_EXP.
- **COMP_EXP:**Αποτελείται από ένα SIMPLE_EXPκαι στη συνέχεια ακολουθείται 0 ή πολλές φορές με “==” SIMPLE_EXPή “!=” SIMPLE_EXP.
- **SIMPLE_EXP**: Αποτελείται είτε από ένα EXP,είτε από “(” το οποίο ακολουθείται από GUARD και στη συνέχεια από “)”

3.5.4 Αλγόριθμος για τη συντακτική ανάλυση μιας T-CDL χορογραφίας

Περιγραφή Αλγορίθμου:

Αρχικά ελέγχουμε κατά πόσο έχει δοθεί η χορογραφία στη μορφή $C[\overline{R}, A]$ και αποθηκεύουμε τους ρόλους της χορογραφίας $\overline{R}$, καθώς και την κύρια συνάρτησή της Ασε ένα πίνακα κατακερματισμού. Ακολούθως, εξετάζουμε κατά πόσο έχουν δηλωθεί οι ρόλοι, σύμφωνα με τη μορφή $R ::= r[\overline{x}, \overline{op}]$ και αποθηκεύουμε σε πίνακα κατακερματισμού τις μεταβλητές $\overline{x}$ και σε ξεχωριστό πίνακα κατακερματισμού τις συμπεριφορές $\overline{op}$ μαζί με το όνομα του ρόλου στο οποίο ανήκουν.

Σε περίπτωση που δεν έχει δηλωθεί έστω και ένας ρόλος ο οποίος αναφέρεται στην χορογραφία, ο χρήστης ενημερώνεται με το ανάλογο μήνυμα. Επίσης, η δήλωση των ρόλων απαιτούμε να γίνεται πριν από τη δήλωση της κύριας συνάρτησης της χορογραφίας και σε περίπτωση που γίνει το αντίθετο, ο χρήστης ενημερώνεται και πάλι με το αντίστοιχο μήνυμα. Για να ξεχωρίσουμε τη δήλωση των ρόλων από τη δήλωση της κύριας συνάρτησης, ο χρήστης θα πρέπει μόλις τελειώσει από τη δήλωση του

τελευταίου ρόλου, να εισάξει τον ειδικό χαρακτήρα “~”, διαφορετικά ο συντακτικός αναλυτήςθα συνεχίζει να ψάχνει δηλώσεις όμοιες με αυτές των ρόλων της χορογραφίας.

Η κύρια συνάρτηση της χορογραφίας δηλώνεται ως μια δραστηριότητα και αποθηκεύεται στο αντίστοιχοπίνακα κατακερματισμού. Μία δραστηριότητα στην T-CDL, μπορεί να είναι είτε μια βασική δραστηριότητα (BasicActivity), είτε μια συνθήκη, είτε μια επανάληψη, είτε μια μονάδα εργασίας,είτε μια δραστηριότητα διαδοχής, είτε μια επιλογή, είτε μια γενική επιλογή, είτε ένας παραλληλισμός, είτε ένας τελεστής καθυστέρησης, είτε μια δραστηριότητα προθεσμίας.

Μια βασική δραστηριότητα μπορεί να είναι η λέξη “skip”, είτε μια ανάθεση, είτε ένα αίτημα, είτε μια απάντηση, είτε ένα αίτημα – απάντηση.

Για τις βασικές δραστηριότητες ανάθεση, αίτημα, απάντηση και αίτημα – απάντηση, καθώς και σε άλλες περιπτώσεις γίνεται έλεγχος κατά πόσο οι μεταβλητές ανήκουν στον ρόλο που θα υπάρχει στο πρόγραμμα. Επιπλέον ελέγχοι για αυτό το κομμάτι αναφέρονται στο Υποκεφάλαιο 3.5.5 καθώς και στο παράρτημα Γ.

Λόγω της μορφής που έχει η σύνταξη της γραμματικής της T-CDL, όπως φαίνεται και στους Πίνακες 4.1 - 4.3, ενδέχεται η δήλωση μιας δραστηριότητας να περιέχει κι άλλες δραστηριότητες οι οποίες δεν έχουν ακόμη δηλωθεί. Σε περίπτωση που φτάσουμε στο EOFκαι υπάρχει έστω και μια δραστηριότητα η οποία δεν έχει δηλωθεί, θα εμφανίζεται στο χρήστη το ανάλογο μήνυμα.

3.5.5 Δομές που χρησιμοποιήθηκαν

Για την υλοποίηση του parserχρησιμοποίησαμε 5 πίνακες κατακερματισμού:

1. **chRoles_HT:** Αποθηκεύει τα ονόματα των ρόλων της χορο γραφίας με τη μορφή κλάσης ChRoles, στην οποία αποθηκεύεται το όνομα του ρόλου της χορογραφίας, το είδος του καθώς και την γραμμή και στήλη στην οποία βρίσκεται ο ρόλος. Στην κλάση αυτή υπάρχει και μια μεταβλητή isDefinedη οποία χρησιμοποιείται πιο μετά για να ελέγξουμε αν όντως έχουν δηλωθεί όλοι οι ρόλοι.

2. **mainFun_HT:**Αποθηκεύει τοόνομα της κύριας συνάρτησηςτης χορογραφίας με τη μορφή κλάσης MainFun, στην οποία αποθηκεύεται το όνομα της συνάρτησης, το είδος της καθώς και την γραμμή και στήλη στην οποία βρίσκεται η κύρια συνάρτηση. Στην κλάση αυτή υπάρχει και μια μεταβλητή isDefinedη οποία χρησιμοποιείται πιο μετά για να ελέγξουμε αν όντως έχει δηλωθεί η κύρια συνάρτηση.
3. **roleX_HT:** Αποθηκεύει τα ονόματα των μεταβλητών για κάποιο ρόλο της χορογραφίας με τη μορφή κλάσης RoleX στην οποία αποθηκεύεται το όνομα της μεταβλητής, το είδος της, καθώς και το όνομα του ρόλου στον οποίο ανήκει.
4. **roleX_OP:** Αποθηκεύει τα ονόματα των συμπεριφορών για κάποιο ρόλο της χορογραφίας με τη μορφή κλάσης RoleOp στην οποία αποθηκεύεται το όνομα της συμπεριφοράς, το είδος της, καθώς και το όνομα του ρόλου στον οποίο ανήκει.
5. **activities_HT:** Αποθηκεύει τα ονόματα των δραστηριοτήτων με τη μορφή κλάσης Activitiesστην οποία αποθηκεύεται το όνομα της δραστηριότητας, το είδος της καθώς και την γραμμή και στήλη στην οποία βρίσκεται η δραστηριότητα. Στην κλάση αυτή υπάρχει και μια μεταβλητή isDefinedη οποία χρησιμοποιείται πιο μετά για να ελέγξουμε αν όντως έχει δηλωθεί η δραστηριότητα.

Επίσης χρησιμοποιήσαμε 9ArrayLists:

1. **roleNames:** Αποθηκεύει τα ονόματα των ρόλων της χορογραφίας.
2. **roleX:** Αποθηκεύει τα ονόματα των μεταβλητών για κάποιο ρόλο της χορογραφίας με τη μορφή κλάσης RoleX
3. **roleOP:** Αποθηκεύει τα ονόματα των μεταβλητών για κάποιο ρόλο της χορογραφίας με τη μορφή κλάσης RoleOp
4. **rolesFInAssgnm:** Αποθηκεύει τα ονόματα των ρόλωνF που δίνονται στις βασικές δραστηριότητες αίτημα, απάντηση, αίτημα – απάντηση.
5. **rolesTInAssgnm:** Αποθηκεύει τα ονόματα των ρόλων T που δίνονται στις βασικές δραστηριότητες αίτημα, απάντηση, αίτημα – απάντηση.
6. **varX:**Αποθηκεύει τα ονόματα των μεταβλητών που ανήκουν στο ρόλο Fκαι που δίνονται στις βασικές δραστηριότητες αίτημα, απάντηση, αίτημα – απάντηση.
7. **varY:**Αποθηκεύει τα ονόματα των μεταβλητών που ανήκουν στο ρόλο Tκαι που δίνονται στις βασικές δραστηριότητες αίτημα, απάντηση, αίτημα – απάντηση.
8. **varXIA:**Αποθηκεύει τα ονόματα των μεταβλητών που ανήκουν στο ρόλο Fκαι που δίνονται στιςαναθέσεις των βασικών δραστηριοτήτωναίτημα, απάντηση, αίτημα – απάντηση.
9. **varYIA:**Αποθηκεύει τα ονόματα των μεταβλητών που ανήκουν στο ρόλο Tκαι που δίνονται στιςαναθέσεις των βασικών δραστηριοτήτωναίτημα, απάντηση, αίτημα – απάντηση.

3.5.6 Μεθόδοι που χρησιμοποιήθηκαν

Για την υλοποίηση του συντακτικού αναλυτή χρησιμοποιήσαμε τις ακόλουθες μεθόδους:

1. **dump():** Ημέθοδος αυτή ανήκει στην κλάση SimpleNodeκαι μας τυπώνει σε δεντρική μορφή τη γραμματική που έχουμε δώσει στο εργαλείο JavaCC.
2. **intcheckChorRoleDecl(StringmainFunName):** Ησυνάρτησηαυτήδέχεται σαν παράμετρο το όνομα της κύριας συνάρτησης της χορογραφίας και ελέγχει κατά πόσο έχουν όλοι οι ρόλοι της εν λόγω χορογραφίας δηλωθεί. Εάν έχουν δηλωθεί τότε επιστρέφει 1, διαφορετικά εμφανίζει ότι ο συγκεκριμένος ρόλος δεν έχει δηλωθεί και πως η δήλωσή τους πρέπει να γίνει πριν από τη δήλωση της κύριας συνάρτησης της χορογραφίας.

3. **intcheckActivityDecl():** Ησυνάρτησηαυτήελέγχει κατά πόσο έχουν δηλωθεί όλες οι δραστηριότητες τησεν λόγω χορογραφίας, μόλις φτάσει στο τέλος του αρχείου δηλαδή στο EOF. Εάν έχουν δηλωθεί τότε επιστρέφει 1, διαφορετικά εμφανίζει ότι η συγκεκριμένηδραστηριότητα δεν έχει δηλωθεί.
4. **intassignCheckNoComma(Stringf1, Stringt1, Stringft, Stringxy, int line, int column, int xyLine, int xyCol, int sectionCount, intiSection):** Η συνάρτησηαυτήδέχεταισανπαραμέτρουςτουςρόλουςf, tκαι ένα άλλο ρόλοft, μια μεταβλητή, τη γραμμή και στήλη του ρόλο υ ft, τη γραμμή και στήλη της μεταβλητής xy, ένα μετρητή που θα χρησιμοποιήσουμε για να ξέρουμε από ποια από τις βασικές δραστηριότητες (αίτημα, απάντησηή αίτημα-απάντηση) θα πάρουμε τις μεταβλητές από τους πίνακες κατακερματισμού, για να κάνουμε τους απαραίτητους ελέγχους, καθώς και μια ακέραια τιμή που καθορίζει σε ποιο τμήμα βρισκόμαστε (αίτημα, απάντηση, ανάθεση).Εξετάζουμετο ρόλο ftγια να δούμε εάν είναι ο ίδιος με το ρόλο fή t, θέτουμε μια μεταβλητή role1ίση με 1 εάν μιλάμε για το ρόλο fκαι ίση με 2 αν μιλάμε για το ρόλο T. Ακολούθως, αποθηκεύουμε τον ρόλο αυτό καθώς και τη μεταβλητή xyστις αντίστοιχες δομές και εξετάζουμε κατά πόσο ανήκει η μεταβλητή xyστο ρόλο ft. Σε περίπτωση που ο ρόλος δεν ταιριάζει ούτε με το ρόλο f, ούτε με το ρόλο t, εμφανίζεται στο χρήστη το ανάλογο μήνυμα. Έπειτα, επιστρέφουμε την τιμή της μεταβλητής role1.
5. **voidassignCheckWithComma(Stringf1, Stringt1, Stringft, Stringxy, int f1Line, int f1Col, int t1Line, int t1Col, int ftLine, int ftCol, int xyLine, int xyCol, int sectionCount, intvarArrayList):** Η συνάρτησηαυτήδέχεταισανπαραμέτρουςτουςρόλους f, tκαι ένα άλλο ρόλο ft, μια μεταβλητή,τη γραμμή και στήλη των ρόλωνf1, t1 και ft, τη γραμμή και στήλη της μεταβλητής xy, ένα μετρητή που θα χρησιμοποιήσουμε για να ξέρουμε από ποια από τις βασικές δραστηριότητες (αίτημα, απάντηση ή αίτημα-απάντηση) θα πάρουμε τις μεταβλητές από τους πίνακες κατακερματισμού, για να κάνουμε τους απαραίτητους ελέγχους, καθώς και μια ακέραια τιμή που καθορίζει αν μιλάμε για το ρόλο f ή το ρόλοt.Εξετάζουμετο ρόλο ftγια να δούμε εάν είναι ο ίδιος με το ρόλο fή t ανάλογα με την τιμή που έχει η μεταβλητή varArrayList, αποθηκεύουμε τον ρόλο αυτό καθώς και τη μεταβλητή xyστις αντίστοιχες δομές και εξετάζουμε κατά πόσο ανήκει η μεταβλητή xyστο ρόλο ft.και σε αντίθετη

περίπτωση του εμφανίζεται το αντίστοιχο μήνυμα. Σε περίπτωση που ο ρόλος δεν ταιριάζει ούτε με το ρόλο f, ούτε με το ρόλο t, ανάλογα με την τιμή της μεταβλητής varArrayListεμφανίζεται στο χρήστη το ανάλογο μήνυμα.

6. **intassignCheckSection2(Stringf1, Stringt1, Stringft, Stringxy, int f1Line, int f1Col, int t1Line, int t1Col, int ftLine, int ftCol, int xyLine, int xyCol, int sectionCount, booleanassignRole, intcheckVarList, intiSection):** Η συνάρτησηαυτήδέχεταισανπαραμέτρουςτουςρόλους f, tκαι ένα άλλο ρόλο ft, μια μεταβλητή, τη γραμμή και στήλη των ρόλων f1, t1 και ft, τη γραμμή και στήλη της μεταβλητής xy, ένα μετρητή που θα χρησιμοποιήσουμε για να ξέρουμε από ποια από τις βασικές δραστηριότητες (αίτημα, απάντηση ή αίτημα-απάντηση) θα πάρουμε τις μεταβλητές από τους πίνακες κατακερματισμού, για να κάνουμε τους απαραίτητους ελέγχους, μια boolean μεταβλητή, μιαακέραια τιμή καθορίζει αν μιλάμε για το ρόλο f ή το ρόλοt και μια ακέραια τιμή που καθορίζει σε ποιο τμήμα βρισκόμαστε (αίτημα, απάντηση, ανάθεση).Εξετάζουμετο ρόλο ftγια να δούμε εάν είναι ο ίδιος με το ρόλο fή t, ανάλογα με την τιμή που έχει η μεταβλητή varArrayList και εάν η μεταβλητή assignRoleέχει την τιμή truetότε θέτουμε τη μεταβλητή role2 ίση με 1 εάν μιλάμε για το ρόλο fκαι ίση με 2 αν μιλάμε για το ρόλο t. Ακολούθως, αποθηκεύουμε τον ρόλο αυτό καθώς και τη μεταβλητή xyστις αντίστοιχες δομές και εξετάζουμε κατά πόσο ανήκει η μεταβλητή xyστο ρόλο ft. Σε περίπτωση που ο ρόλος δεν ταιριάζει ούτε με το ρόλο f, ούτε με το ρόλο t, εμφανίζεται στο χρήστη το ανάλογο μήνυμα. Έπειτα, επιστρέφουμε την τιμή της μεταβλητής role2.
7. **voidcheckVarCorrectness(StringsenderReceiver, Stringvar, intsenderReceiverLine, intsenderReceiverCol, intvarLine, intvarCol, intiSection):** Ησυνάρτησηαυτήδέχεταισανπαραμέτρουςέναρόλο, μιαμεταβλητή, τηγραμμήκαιστήλητουρόλουft, τηγραμμήκαιστήλητηςμεταβλητήςxyκαιτοτμήμαστοοποίοβρισκόμαστε (αίτημα, απάντηση, ανάθεση) καιελέγχεικατάπόσοημεταβλητήvarανήκειστορόλοsenderReceiverκαιεάνδενανήκειεπιστρέφειτοανάλογομήνυμαστοχρήστη.
8. **voidisVarDuplicate(intcheckRoleVar):** Η συνάρτησηαυτήδέχεταισανπαραμέτρο μια ακέραια τιμή που του καθορίζει αν

μιλάμε για το ρόλο F ή το ρόλοT και ελέγχουμε εάν μια μεταβλητή ενός ρόλου έχει ήδη πάρει τιμή στο τμήμα της ανάθεσης. Εάν όντως ισχύει αυτό , τότε εμφανίζο με στο χρήστη το ανάλογο μήνυμα που το ν παραπέμπει να χρησιμοποιήσει μια άλλη μεταβλητή.

9. **voidisVarAlreadyInReqAns(Stringft, Stringxy, int sectionCount, intcheckRoleVar, intcheckSection):** Η συνάρτησηαυτήδέχεταισανπαραμέτρουςέναρόλο, μια μεταβλητή, ένα μετρητή που θα χρησιμοποιήσο με για να ξέρο με από ποια από τις βασικές δραστηριότητες (αίτημα, απάντηση ή αίτημα-απάντηση) θα πάρουμε τις μεταβλητές από τους πίνακες κατακερματισμού, για να κάνουμε τους απαραίτητους ελέγχους, μια ακέραια τιμή που του καθορίζει αν μιλάμε για το ρόλο f ή το ρόλοt και μια ακέραια τιμή που καθορίζει σε ποιο τμήμα βρισκόμαστε (αίτημα, απάντηση, ανάθεση) και σε περίπτωση που ο χρήστης χρησιμοποιεί στο τμήμα της ανάθεσης μια μεταβλητή για κάποιο ρόλο, ίδια με τη μεταβλητή του ιδίου ρόλου στο τμήμα της αίτησης/απάντησης, του εμφανίζεται το ανάλογο μήνυμα.

3.5.7 Παραδείγματα Εκτέλεσης:

I. Δίνεται το αρχείο “Είσοδος1” και τρέχοντας το πρόγραμμα για τη συντακτική ανάλυση, παίρνουμε το “Αποτέλεσμα1” όπως φαίνεται πιο κάτω. Σε αυτή την περίπτωση, μετά το όνομα της χορογραφίας θα έπρεπε να υπήρχε ‘[‘.

```
1 PurchaseOrder(buyer, seller, credit, inv), purchaseOrder]
2 buyer = Buyer[(po, poAck, poResp, poRej, poState),(PoReqOP, DisplayOP)]
3 seller = Seller[(po, poAck, poResp, poRej, ccReq, ccResp, icReq, icResp,
4                          poState, ccState, icState, msg), PoHandleOP]
5
6 inv = Inv[(icReq, icResp, icState), InvCheckOP]
7
8 credit = Credit[(ccReq, ccResp, ccState), CreditCheckOP] ~
9
10 purchaseOrder = sequence(poRequest; checkCreditInv; finalAnswer)
11
12
13 poRequest = comm(request(Buyer.po, Seller.poAck, seller_to_buyer),
14                 (Buyer.poAck :="unknown", Buyer.poRej:="unknown"),
15                 (Seller.poResp:= "received", Seller.poRej:="unknown"),
16                 xr(6))
17
18 checkCreditInv = deadline(parallelism(creditCheck || invCheck), 5, poExc, poFinal)
19
20 poExc = Seller.icReq:=false
21 poFinal = Seller.icReq:=true
22
23 creditCheck = xr(3)comm(requestAnswer(request(Seller.ccReq, Credit.ccReq, seller_to_credit),
24                                        answer(Seller.ccResp, Credit.ccResp, credit_to_seller),
25                                        Credit.ccState:="sent",
26                                        Seller.ccState:="received",
27                                       xr(9)))
28
29 invCheck = xr(3)comm(requestAnswer(request(Seller.icReq, Inv.icReq, seller_to_inventory),
30                                        answer(Seller.icResp, Inv.icResp, inventory_to_seller),
31                                        Inv.icState:="sent",
32                                        Seller.icState:="received",
33                                        e))
34
35 finalAnswer =  case (16 / 2 + 4 * (4+2): poResponse, Seller.icReq == false : poReject )
```

Σχήμα 3.8: Είσοδος1

```
Exception in thread "main" ParseException: Encountered " "(" "( "" at line 1, column 14.
Was expecting:
    "[" ...

        at Parser.generateParseException(Parser.java:3979)
        at Parser.jj_consume_token(Parser.java:3861)
        at Parser.choreography(Parser.java:592)
        at Parser.code(Parser.java:562)
        at Parser.tcdl_file(Parser.java:519)
        at Parser.main(Parser.java:50)
```

Σχήμα 3.9: Αποτέλεσμα1

II. Δίνεται το αρχείο “Είσοδος 2” και τρέχοντας το πρόγραμμα για τη συντακτική ανάλυση, παίρνουμε το “Αποτέλεσμα 2” όπως φαίνεται πιο κάτω. Σε αυτή την περίπτωση, απουσιάζει από τη χορογραφία η κύρια συνάρτηση, δηλαδή ένα ID.

```
PurchaseOrder[(buyer, seller, credit, inv), ]
buyer = Buyer[(po, poAck, poResp, poRej, poState),(PoReqOP, DisplayOP)]
seller = Seller[(po, poAck, poResp, poRej, ccReq, ccResp, icReq, icResp,
                         poState, ccState, icState, msg), PoHandleOP]

inv = Inv[(icReq, icResp, icState), InvCheckOP]

credit = Credit[(ccReq, ccResp, ccState), CreditCheckOP] ~

purchaseOrder = sequence(poRequest; checkCreditInv; finalAnswer)


poRequest = comm(request(Buyer.po, Seller.poAck, seller_to_buyer),
                 (Buyer.poAck :="unknown", Buyer.poRej:="unknown"),
                 (Seller.poResp:= "received", Seller.poRej:="unknown"),
                 xr(6))

checkCreditInv = deadline(parallelism(creditCheck || invCheck), 5, poExc, poFinal)

poExc = Seller.icReq:=false
poFinal = Seller.icReq:=true

creditCheck = xr(3)comm(requestAnswer(request(Seller.ccReq, Credit.ccReq, seller_to_credit),
                                      answer(Seller.ccResp, Credit.ccResp, credit_to_seller),
                                      Credit.ccState:="sent",
                                      Seller.ccState:="received",
                                     xr(9)))

invCheck = xr(3)comm(requestAnswer(request(Seller.icReq, Inv.icReq, seller_to_inventory),
                                      answer(Seller.icResp, Inv.icResp, inventory_to_seller),
                                      Inv.icState:="sent",
                                      Seller.icState:="received",
                                      e))

finalAnswer =  case (16 / 2 + 4 * (4+2): poResponse, Seller.icReq == false : poReject )
```

Σχήμα 3.8: Είσοδος 2

```
Exception in thread "main" ParseException: Encountered " "]" "] "" at line 1, column 45.
Was expecting:
    <ID> ...

        at Parser.generateParseException(Parser.java:3979)
        at Parser.jj_consume_token(Parser.java:3861)
        at Parser.mainFun(Parser.java:665)
        at Parser.choreography(Parser.java:595)
        at Parser.code(Parser.java:562)
        at Parser.tcdl_file(Parser.java:519)
        at Parser.main(Parser.java:50)
```

Σχήμα 3.9: Αποτέλεσμα 2

III. Δίνεται το αρχείο “Είσοδος 3” και τρέχοντας το πρόγραμμα για τη συντακτική ανάλυση, παίρνουμε το “Αποτέλεσμα 3” όπως φαίνεται πιο κάτω. Σε αυτή την περίπτωση, απουσιάζει η δήλωση του ρόλου invπου υπάρχει στη χορογραφία.

```
1 PurchaseOrder[(buyer, seller, credit, inv), purchaseOrder]
2 buyer = Buyer[(po, poAck, poResp, poRej, poState),(PoReqOP, DisplayOP)]
3 seller = Seller[(po, poAck, poResp, poRej, ccReq, ccResp, icReq, icResp,
4                          poState, ccState, icState, msg), PoHandleOP]
5
6
7
8 credit = Credit[(ccReq, ccResp, ccState), CreditCheckOP] ~
9
10 purchaseOrder = sequence(poRequest; checkCreditInv; finalAnswer)
11
12
13 poRequest = comm(request(Buyer.po, Seller.poAck, seller_to_buyer),
14                  (Buyer.poAck :="unknown", Buyer.poRej:="unknown"),
15                  (Seller.poResp:= "received", Seller.poRej:="unknown"),
16                  xr(6))
17
18 checkCreditInv = deadline(parallelism(creditCheck || invCheck), 5, poExc, poFinal)
19
20 poExc = Seller.icReq:=false
21 poFinal = Seller.icReq:=true
22
23 creditCheck = xr(3)comm(requestAnswer(request(Seller.ccReq, Credit.ccReq, seller_to_credit),
24                                        answer(Seller.ccResp, Credit.ccResp, credit_to_seller),
25                                        Credit.ccState:="sent",
26                                        Seller.ccState:="received",
27                                       xr(9)))
28
29 invCheck = xr(3)comm(requestAnswer(request(Seller.icReq, Inv.icReq, seller_to_inventory),
30                                        answer(Seller.icResp, Inv.icResp, inventory_to_seller),
31                                        Inv.icState:="sent",
32                                        Seller.icState:="received",
33                                        e))
34
35 finalAnswer =  case (16 / 2 + 4 * (4+2): poResponse, Seller.icReq == false : poReject )
```

Σχήμα 3.12: Είσοδος3

```
Role "buyer" is now declared !
Role "seller" is now declared !
Role "credit" is now declared !
Role inv at line 1, column 39 has not been declared yet. The role declaration must been done before the main function's "purchaseOrder" declaration
```

Σχήμα 3.13: Αποτέλεσμα 3

IV. Δίνεται το αρχείο “Είσοδος 4” και τρέχοντας το πρόγραμμα για τη συντακτική ανάλυση, παίρνουμε το “Αποτέλεσμα 4” όπως φαίνεται πιο κάτω. Σε αυτή την περίπτωση, η μεταβλητή “po” του ρόλου Buyer εμφανίζεται τόσο στο τμήμα της ανάθεσης, όσο και στο τμήμα της αίτησης.

```
1 PurchaseOrder[(buyer, seller, credit, inv), purchaseOrder]
2 buyer = Buyer[(po, poAck, poResp, poRej, poState),(PoReqOP, DisplayOP)]
3 seller = Seller[(po, poAck, poResp, poRej, ccReq, ccResp, icReq, icResp,
4                          poState, ccState, icState, msg), PoHandleOP]
5 inv = Inv[(icReq, icResp, icState), InvCheckOP]
6 credit = Credit[(ccReq, ccResp, ccState), CreditCheckOP] ~
7 purchaseOrder = sequence(poRequest; checkCreditInv; finalAnswer)
8 poRequest = comm(request(Buyer.po, Seller.poAck, seller_to_buyer),
9                  (Buyer.po :="unknown", Buyer.poRej:="unknown"),
10                 (Seller.poResp:= "received", Seller.poRej:="unknown"),
11                 xr(6))
12 checkCreditInv = deadline(parallelism(creditCheck || invCheck), 5, poExc, poFinal)
13 poExc = Seller.icReq:=false
14 poFinal = Seller.icReq:=true
15 creditCheck = xr(3)comm(requestAnswer(request(Seller.ccReq, Credit.ccReq, seller_to_credit),
16                                        answer(Seller.ccResp, Credit.ccResp, credit_to_seller),
17                                        Credit.ccState:="sent",
18                                        Seller.ccState:="received",
19                                       xr(9)))
20 invCheck = xr(3)comm(requestAnswer(request(Seller.icReq, Inv.icReq, seller_to_inventory),
21                                        answer(Seller.icResp, Inv.icResp, inventory_to_seller),
22                                        Inv.icState:="sent",
23                                        Seller.icState:="received",
24                                        e))
25 finalAnswer =  case (16 / 2 + 4 * (4+2): poResponse, Seller.icReq == false : poReject )
26 poResponse = comm(request(Buyer.poResp, Seller.poResp, seller_to_buyer),
27                  Buyer.poState:="complete",
28                  (Seller.poState:="complete", Seller.msg:=Seller.poRej),
29                  e)
30 poReject =  comm(request(Buyer.poRej, Seller.poRej, seller_to_buyer),
31                  Buyer.poState:="complete",
32                  Seller.po:="complete",
33                  e)
```

Σχήμα 3.14: Είσοδος 4

```
Role "buyer" is now declared !
Role "seller" is now declared !
Role "inv" is now declared !
Role "credit" is now declared !
Main Function "purchaseOrder" is now declared !
Activity "poRequest" is now declared !
Variable "po" of role "Buyer" in the assignment section, in line 9, column 41 can't be the same with variable "po" in the request section.
```

Σχήμα 3.15: Αποτέλεσμα 4

V. Δίνεται το αρχείο “Είσοδος 5” και τρέχοντας το πρόγραμμα για τη συντακτική ανάλυση, παίρνουμε το “Αποτέλεσμα 5” όπως φαίνεται πιο κάτω. Σε αυτή την περίπτωση, η μεταβλητή “msg” δεν ανήκει στο ρόλοSeller.

```
PurchaseOrder[(buyer, seller, credit, inv), purchaseOrder]
buyer = Buyer[(po, poAck, poResp, poRej, poState),(PoReqOP, DisplayOP)]
seller = Seller[(po, poAck, poResp, poRej, ccReq, ccResp, icReq, icResp,
                        poState, ccState, icState), PoHandleOP]
inv = Inv[(icReq, icResp, icState), InvCheckOP]
credit = Credit[(ccReq, ccResp, ccState), CreditCheckOP] ~
purchaseOrder = sequence(poRequest; checkCreditInv; finalAnswer)
poRequest = comm(request(Buyer.po, Seller.poAck, seller_to_buyer),
                 (Buyer.poAck :="unknown", Buyer.poRej:="unknown"),
                 (Seller.poResp:= "received", Seller.poRej:="unknown"),
                 xr(6))
checkCreditInv = deadline(parallelism(creditCheck || invCheck), 5, poExc, poFinal)
poExc = Seller.icReq:=false
poFinal = Seller.icReq:=true
creditCheck = xr(3)comm(requestAnswer(request(Seller.ccReq, Credit.ccReq, seller_to_credit),
                                      answer(Seller.ccResp, Credit.ccResp, credit_to_seller),
                                      Credit.ccState:="sent",
                                      Seller.ccState:="received",
                                     xr(9)))
invCheck = xr(3)comm(requestAnswer(request(Seller.icReq, Inv.icReq, seller_to_inventory),
                                      answer(Seller.icResp, Inv.icResp, inventory_to_seller),
                                      Inv.icState:="sent",
                                      Seller.icState:="received",
                                      e))
finalAnswer =  case (16 / 2 + 4 * (4+2): poResponse, Seller.icReq == false : poReject )
poResponse = comm(request(Buyer.poResp, Seller.poResp, seller_to_buyer),
                  Buyer.poState:="complete",
                  (Seller.poState:="complete", Seller.msg:=Seller.poRej),
                  e)
poReject =  comm(request(Buyer.poRej, Seller.poRej, seller_to_buyer),
                  Buyer.poState:="complete",
                  Seller.po:="complete",
                  e)
```

Σχήμα 3.16: Είσοδος 5

```
Role "buyer" is now declared !
Role "seller" is now declared !
Role "inv" is now declared !
Role "credit" is now declared !
Main Function "purchaseOrder" is now declared !
Activity "poRequest" is now declared !
Activity "checkCreditInv" is now declared !
Activity "poExc" is now declared !
Activity "poFinal" is now declared !
Activity "creditCheck" is now declared !
Activity "invCheck" is now declared !
Activity "finalAnswer" is now declared !
Activity "poResponse" is now declared !
The variable "msg" in the assignment section in line 28, column 71 doesn't belong to role "Seller".
```

Σχήμα 3.17: Αποτέλεσμα 5

VI. Δίνεται το αρχείο “Είσοδος 6” και τρέχοντας το πρόγραμμα για τη συντακτική ανάλυση, παίρνουμε το “Αποτέλεσμα 6” όπως φαίνεται πιο κάτω. Σε αυτή την περίπτωση,μετά το “e” υπάρχει μια επιπλέον παρένθεση ενώ θα έπρεπε να έβρισκε EOFκαι να τέλειωνε το πρόγραμμα.

```
PurchaseOrder[(buyer, seller, credit, inv), purchaseOrder]
buyer = Buyer[(po, poAck, poResp, poRej, poState),(PoReqOP, DisplayOP)]
seller = Seller[(po, poAck, poResp, poRej, ccReq, ccResp, icReq, icResp,
                        poState, ccState, icState, msg), PoHandleOP]
inv = Inv[(icReq, icResp, icState), InvCheckOP]
credit = Credit[(ccReq, ccResp, ccState), CreditCheckOP] ~
purchaseOrder = sequence(poRequest; checkCreditInv; finalAnswer)
poRequest = comm(request(Buyer.po, Seller.poAck, seller_to_buyer),
                 (Buyer.poAck :="unknown", Buyer.poRej:="unknown"),
                 (Seller.poResp:= "received", Seller.poRej:="unknown"),
                 xr(6))
checkCreditInv = deadline(parallelism(creditCheck || invCheck), 5, poExc, poFinal)
poExc = Seller.icReq:=false
poFinal = Seller.icReq:=true
creditCheck = xr(3)comm(requestAnswer(request(Seller.ccReq, Credit.ccReq, seller_to_credit),
                                      answer(Seller.ccResp, Credit.ccResp, credit_to_seller),
                                      Credit.ccState:="sent",
                                      Seller.ccState:="received",
                                     xr(9)))
invCheck = xr(3)comm(requestAnswer(request(Seller.icReq, Inv.icReq, seller_to_inventory),
                                      answer(Seller.icResp, Inv.icResp, inventory_to_seller),
                                      Inv.icState:="sent",
                                      Seller.icState:="received",
                                      e))
finalAnswer =  case (16 / 2 + 4 * (4+2): poResponse, Seller.icReq == false : poReject )
poResponse = comm(request(Buyer.poResp, Seller.poResp, seller_to_buyer),
                  Buyer.poState:="complete",
                  (Seller.poState:="complete", Seller.msg:=Seller.poRej),
                  e)
poReject =  comm(request(Buyer.poRej, Seller.poRej, seller_to_buyer),
                  Buyer.poState:="complete",
                  Seller.po:="complete",
                  e))
```

Σχήμα 3.18: Είσοδος 6

```
Role "buyer" is now declared !
Role "seller" is now declared !
Role "inv" is now declared !
Role "credit" is now declared !
Main Function "purchaseOrder" is now declared !
Activity "poRequest" is now declared !
Activity "checkCreditInv" is now declared !
Activity "poExc" is now declared !
Activity "poFinal" is now declared !
Activity "creditCheck" is now declared !
Activity "invCheck" is now declared !
Activity "finalAnswer" is now declared !
Activity "poResponse" is now declared !
Activity "poReject" is now declared !
Exception in thread "main" ParseException: Encountered " ")" ") "" at line 33, column 37.
Was expecting:
    <EOF>

        at Parser.generateParseException(Parser.java:3979)
        at Parser.jj_consume_token(Parser.java:3861)
        at Parser.tcdl_file(Parser.java:520)
        at Parser.main(Parser.java:50)
```

Σχήμα 3.19: Αποτέλεσμα 6

VII. Δίνεται το αρχείο “Είσοδος 7” και τρέχοντας το πρόγραμμα για τη συντακτική ανάλυση, παίρνουμε το “Αποτέλεσμα 7” όπως φαίνεται πιο κάτω. Σε αυτή την περίπτωση,η δραστηριότητα“poFinal”δεν έχει οριστεί και επομένως δεν μπορεί να χρησιμοποιηθεί.

```
1  PurchaseOrder[(buyer, seller, credit, inv), purchaseOrder]
2  buyer = Buyer[(po, poAck, poResp, poRej, poState),(PoReqOP, DisplayOP)]
3  seller = Seller[(po, poAck, poResp, poRej, ccReq, ccResp, icReq, icResp,
4                          poState, ccState, icState, msg), PoHandleOP]
5  inv = Inv[(icReq, icResp, icState), InvCheckOP]
6  credit = Credit[(ccReq, ccResp, ccState), CreditCheckOP] ~
7  purchaseOrder = sequence(poRequest; checkCreditInv; finalAnswer)
8  poRequest = comm(request(Buyer.po, Seller.poAck, seller_to_buyer),
9                   (Buyer.poAck :="unknown", Buyer.poRej:="unknown"),
10                  (Seller.poResp:= "received", Seller.poRej:="unknown"),
11                  xr(6))
12 checkCreditInv = deadline(parallelism(creditCheck || invCheck), 5, poExc, poFinal)
13 poExc = Seller.icReq:=false
14
15 creditCheck = xr(3)comm(requestAnswer(request(Seller.ccReq, Credit.ccReq, seller_to_credit),
16                                         answer(Seller.ccResp, Credit.ccResp, credit_to_seller),
17                                         Credit.ccState:="sent",
18                                         Seller.ccState:="received",
19                                        xr(9)))
20 invCheck = xr(3)comm(requestAnswer(request(Seller.icReq, Inv.icReq, seller_to_inventory),
21                                         answer(Seller.icResp, Inv.icResp, inventory_to_seller),
22                                         Inv.icState:="sent",
23                                         Seller.icState:="received",
24                                         e))
25 finalAnswer =  case (16 / 2 + 4 * (4+2): poResponse, Seller.icReq == false : poReject )
26 poResponse = comm(request(Buyer.poResp, Seller.poResp, seller_to_buyer),
27                   Buyer.poState:="complete",
28                   (Seller.poState:="complete", Seller.msg:=Seller.poRej),
29                   e)
30 poReject =  comm(request(Buyer.poRej, Seller.poRej, seller_to_buyer),
31                   Buyer.poState:="complete",
32                   Seller.po:="complete",
33                   e)
```

Σχήμα 3.20: Είσοδος 7

```
Role "buyer" is now declared !
Role "seller" is now declared !
Role "inv" is now declared !
Role "credit" is now declared !
Main Function "purchaseOrder" is now declared !
Activity "poRequest" is now declared !
Activity "checkCreditInv" is now declared !
Activity "poExc" is now declared !
Activity "creditCheck" is now declared !
Activity "finalAnswer" is now declared !
Activity "poResponse" is now declared !
Activity "poReject" is now declared !
Activity "poFinal" has not been declared yet!
```

Σχήμα 3.21: Αποτέλεσμα 7

Κεφάλαιο4

Συμπεράσματα

4.1 Τελικά συμπεράσματα

Μέσα από αυτή τη διπλωματική εργασία, υλοποιήσαμε ένα πρόγραμμα το οποίο θα κάνει parsingτις χορογραφίες που είναι γραμμένες στην T-CDL, βάσει της γραμματικής που δόθηκε στους Πίνακες 4.1, 4.2 και 4.3.

Ένα μεγάλο μέρισμα του χρόνου που είχα στη διάθεσή μου για την εκπόνηση αυτής της διπλωματικής, το πέρασα μελετώντας το εργαλείο JavaCC, πάνω στο οποίο βασιζόταν η διπλωματική μου.

Κατ'αρχάς το εργαλείο αυτό δουλεύει με το CommandPromptκαι αν και βρήκα τις εντολές με τις οποίες θα έτρεχα το πρόγραμμά μου, ωστόσο ήταν πιο δύσκολο να γίνει ο εντοπισμός των όποιων λαθώνθα προέκυπταν στην πορεία της υλοποίησης. Για το σκοπό αυτό βρήκα τον τρόπο που μπορώ να εισάξω τον JavaCCως pluginEclipse και που περιέγραψα στο Υποκεφάλαιο 2.3.1.

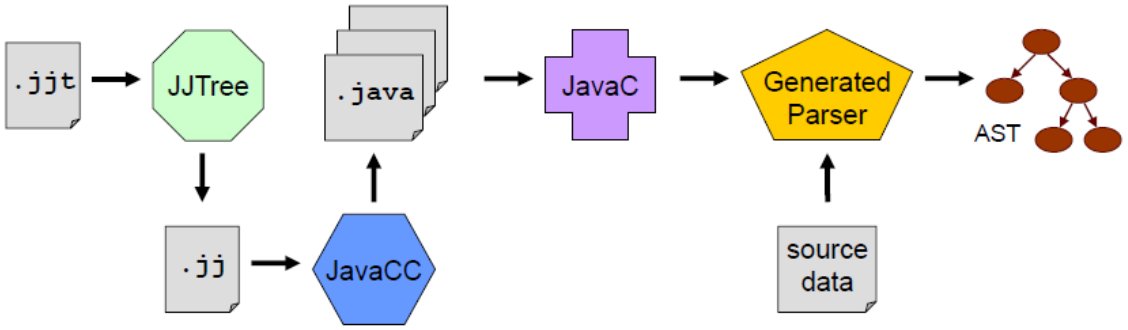
Έπειτα, έπρεπε να μάθω τη σύνταξή του εργαλείου αυτού, τον τρόπο που μπορώ να γράψω τους κανόνες της γραμματικής για την οποία θέλω να υλοποιήσω τον parser καθώς και το πώς μπορώ να εισαγάγω κομμάτια κώδικα Java στους κανόνες της γραμματικής μου, που θα χρησιμοποιούσα για τους διάφορους ελέγχους που χρειάστηκε να κάνω. Πολύ βοηθητικές μου φάνηκαν οι ασκήσεις που υπήρχαν στο

διαδίκτυο του Πανεπιστημίου DublinCityUniversity[9] οι οποίες ξεκινούσαν από απλά προγραμματάκια και εμβάθυναν σιγά σιγά.

4.2 Μελλοντική Εργασία

Μελλοντικά θα μπορούσε να χρησιμοποιηθεί αυτός ο συντακτικός αναλυτής και μαζί με τον αλγόριθμο μετάφρασης που προτάθηκε στην προηγούμενη διπλωματική [1] θα μπορούσε να γίνει η μετάφραση από μια χορογραφία δοσμένη σε T-CDLσε κώδικα Promela. Το αρχείο με τον κώδικα Promela που θα παραχθεί, θα αποτελεί είσοδο του εργαλείου Spinτο οποίο θα ελέγχει κατά πόσο η χορογραφία που δόθηκε ήταν ορθή και σύμφωνα με τις απαιτήσεις που υπάρχουν.

Από την εμπειρία που είχα με το συγκεκριμένο εργαλείο, το JavaCC, κάποιος δε θα δυσκολευτεί ιδιαίτερα στο να συνεχίσει και να ολοκληρώσει την υλοποίηση του αλγορίθμου μετάφρασης. Κατ' αρχάς, ο κώδικας για τη συντακτική ανάλυση είναι γραμμένος στο αρχείο .jjt. Μεταγλωττίζοντας το αρχείο μέσω του JJTree, μια JavaCCγραμματική εμπλουτίζεται με την εισαγωγή κώδικα που κτίζει δέντρα (tree-buildingcode).Επομένως, όταν παράγεται ένας συντακτικός αναλυτής από την πλέον εμπλουτισμένη JavaCCγραμματική και όταν αυτός ο παραγόμενος συντακτικός αναλυτής εκτελείται έχοντας ως είσοδο ένα αρχείο ο κώδικας του οποίου στηρίζεται στη γραμματική αυτή τότε παράγεται ένα Αφαιρετικά Συντακτικό Δένρο (AbstractSyntaxTree - AST)



Σχήμα 4.1: Στάδια μεταγλώττισης ενός .jjtαρχείου [10]

Η μεγάλη δυσκολία έγγειται στο γεγονός ότι, κάποιος πρέπει να κατανοήσει τον τρόπο λειτουργίας ενός ASTκαι να βρει ένα τρόπο ώστε να έχει πρόσβαση στις πληροφορίες που μπορεί να δώσει. Για αρχή υπάρχουν κάποια παραδείγματα στο [10] τα οποία ίσως βοηθήσουν στην καλύτερη κατανόηση.

Στη συνέχεια θα πρέπει να γίνει υλοποίηση της συνάρτησης Pr, όπως δίνεται μέσα από τη διατριβή [1], για την εξαγωγή των όψεων ενορχήστρωσης˙η συνάρτηση Pr, με παραμέτρους μια δραστηριότητα και ένα ρόλο, εξάγει τις δραστηριότητες που αφορούν το ρόλο αυτό. Ακολούθως, γίνεται η μετάφραση των όψεων ενορχήστρωσης σε κώδικα Promela αφού προηγηθεί η μελέτη της συμπεριφοράς του κάθε ρόλου μέσα σε κάθε αλληλεπίδραση της χορογραφίας, όπως δίνεται στο [1]. Με την ολοκλήρωση της μετάφρασης, ολοκληρώνεται και η δημιουργία ενός αντιπροσωπευτικού μοντέλου το οποίο θα μπορούμε να τρέξουμε στο εργαλείο μοντελο-ελέγχου Spin. Έτσι θα είμαστε σε θέση να επιβεβαιώσουμε κάποιες ιδιότητες της χορογραφίας όπως το κατά πόσο επιστρέφει ορθό αποτέλεσμα, εάν περιέχει αδιέξοδα, εάν ικανοποιεί τις αρχικές προδιαγραφές κλπ.

4.3 Κέρδος που αποκόμισα από την διπλωματική αυτή εργασία

Μέσα στα πλαίσια της ολοκλήρωσης της διπλωματικής μου εργασίας κατάφερα να αποκομίσω κάποια πολύ σημαντικά οφέλη τα οποία θα μου φανούν σίγουρα χρήσιμα στην μετέπειτά μου πορεία. Έμαθα λοιπόν, τη σωστή μεθοδολογία για την εκπόνηση μιας εργασίας ξεκινώντας από το ερευνητικό επίπεδο. Επίσης κάτι πολύ σημαντικό είναι ότι εμπλούτισα τις γνώσεις μου σχετικά με τη συντακτική ανάλυση και το ποια βήματα πρέπει να ακολοθήσει κανείς ώστς να υλοποιήσει ένα συντακτικό αναλυτή. Κλείνοντας λοιπόν, βελτίωσα αρκετά τις προγραμματιστικές μου ικανότητες και έμαθα να χειρίζομαι το εργαλείο JavaCC,όπου με την βοήθειά του κατάφερα να υλοποιήσω ένα συντακτικό αναλυτή.

4.4 Προβλήματα και παραδοχές

Ο αρχικός στόχος αυτής της διπλωματικής εργασίας, ήταν η συντακτική ανάλυση ενός αρχείου που θα περιέχει μια χορογραφία γραμμένη στη γλώσσα T-CDLέτσι ώστε να διαπιστωθεί κατά πόσο ακολουθεί ορθά τη σύνταξη της γλώσσας αυτής και στη συνέχεια θα προχωρούσαμε στη μετάφραση της χορογραφίας αυτής, σε κώδικα Promela, σύμφωνα με τον αλγόριθμο που προτάθηκε σε προηγούμενη διατριβή [1]. Ο λόγος για τον οποίο καταλήξαμε στο πρώτο κομμάτι του αρχικού μας στόχου, δηλαδή στην υλοποίηση ενός προγράμματος για συντακτική ανάλυση, οφειλόταν στο γεγονός ότι αρκετό μέρος του χρόνου κατά την εκπόνηση της διπλωματικής εργασιάς, αφιερώθηκε στην κατανόηση και στην επίλυση ορισμένων δυσκολιών που αντιμετώπισα και που αναφέρω πιο κάτω.

Μέσα στα πλαίσια της ολοκλήρωσης αυτής της εργασίας, ένα από τα σημαντικότερα προβλήματα που αντιμετώπισα ήταν η κατανόηση του εργαλείου JavaCC, πάνω στο οποίο βασιζόταν η διπλωματική μου και όπως έχω προαναφέρει, σημαντικό ρόλο στην κατανόηση συνέβαλλαν οι ασκήσεις που υπήρχαν στο διαδίκτυο του Πανεπιστημίου DublinCityUniversity[9].

Συγκεκριμένα έπρεπε αρχικά να κατανοήσω πώς με το εργαλείο αυτό θα μπορούσα να γράψω τη δική μου γραμματική, καθώς και πώς μπορώ αφού γράψω τους κανόνες της γραμματικής να έχω πρόβαση στις τιμές των tokensπου θα αντιστοιχούσαν σε κομμάτια του προγράμματος, ώστε να προβώ σε μετέπειτα επεξεργασία. Επιπλέον, ήταν πολύ σημαντικό να κατανήσω το πώς θα ενσωμάτωνα και θα συσχέτιζα τον κώδικα σε Javaμε τους κανόνες της γραμματικής που θα έγραφα στο JavaCC.

Καθώς προχωρούσα με την υλοποίηση του προγράμματος για συντακτική ανάλυση της γλώσσας T-CDL, αντιμετώπισα δυσκολίες όσο αφορά την αριστερή αναδρομή που υπήρχε σε μεγάλο βαθμό στη σύνταξη της T-CDLόπως φαίνεται στον

πίνακα 2.2. Η λύση που δίνεται για την αφαίρεση της αριστερής αναδρομής στο Υποκεφάλαιο 2.3.6, δε δούλευε ακριβώς όπως θα έπρεπε αλλοιώνοντας την ίδια τη γραμματική. Για το λόγο αυτό έκανα προσαρμογές στη σύνταξη και τις οποίες ανέφερα στο Υποκεφάλαιο 3.5.2

Κλείνοντας, θα ήθελα να σημειώσω ότι ο JavaCC είναι ένα πολύ εύκολο και εύχρηστο εργαλείο το οποίο αξίζει να μάθει κανείς για να κάνει συντακτική ανάλυση μιας γλώσσας καθώς και για την υλοποίηση ενός διερμηνέα, ανεξάρτητα από τις δυσκολίες που συνάντησα κατά την πορεία της υλοποίησης του συντακτικού αναλυτή για τη γλώσσα T-CDL.

Βιβλιογραφία

[1] K.Kouppanou, “Μοντελοποίηση Και Ανάλυση Υπηρεσιών Διαδικτύου με Χρονικούς Περιορισμούς μέσω της Άλγεβρας ΔιεργασιώνΤ-CDLκαι το Εργαλείο Spin”. Διατριβή Μάστερ, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου, 2011.

[2] M. P. Papazoglou, P. Traveso, S. Dustdar, andF. Leymann, “Service- Oriented Computing Research Roadmap,” in *Proceedings of Service Oriented Computing*, 2006.

[3] H. Haas, W3C, A. Brown, Microsoft, “Web Services Glossary”, 2004 Available: http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

[4] A. Barros, M. Dumas, and P. Oaks. “A Critical Overview of the Web Services Choreography Description Language”, BPTrends, 2005.

[5] H. Yang, X. Zhao, and Z. Qiu, “Towards the Formal Model and Verification of Web Services Choreography Description Language”, In *WS-FM 2006, LNCS 4184.* Springer, 2006.

[6] M. Mordechai Ben-Ari, “Principles of the Spin Model Checker”, Springer, 2008

[7] SourceForge.Net, 2012
Available: http://www.eclipse-javacc.sourceforge.net

[8] Java Compiler Compiler [tm] (JavaCC [tm]) - The Java Parser Generator.
Available:http://www.javacc.java.net/doc/javaccgrm.html

[9] G. Hamilton, Dublin City University, “CA448- Compiler Construction”
Available:http://www.computing.dcu.ie/~hamilton/teaching/CA448/exercises.html

[10] R. Mak, Department of Computer Science San Jose State University, “CS 153: Concepts of Compiler Design”
Available:http://www.cs.sjsu.edu/~mak/CS153/lectures/CS153-111026.ppt

Παράρτημα Α

Κώδικαςαλγορίθμου υλοποίησης Συντακτικού Αναλυτήγια T-CDLχορογραφίες

Α.1Κύριο Πρόγραμμα

```
// Parser.jjt
/**
* JJTree template file created by SF JavaCC plugin 1.5.17+ wizard for JavaCC 1.5.0+
 */
options
{
  JDK_VERSION = "1.5";
  LOOKAHEAD = 5;
  static = true;
}

PARSER_BEGIN(Parser)
import java.io.*;
import java.util.*;
public class Parser
{
        // hashtables
        public static Hashtable chRoles_HT = new Hashtable();
        public static Hashtable mainFun_HT = new Hashtable();
        public static Hashtable roleX_HT = new Hashtable();
        public static Hashtable roleOp_HT = new Hashtable();
        public static Hashtable activities_HT = new Hashtable();

        // array lists
        public static ArrayList<String> roleNames = new ArrayList<String>();
        public static ArrayList<RoleX> roleX = new ArrayList<RoleX>();
        public static ArrayList<RoleOp> roleOp = new ArrayList<RoleOp>();
        public static ArrayList<String> rolesFInAssgnm = new ArrayList<String>();
        public static ArrayList<String> rolesTInAssgnm = new ArrayList<String>();
        public static ArrayList<String> varX = new ArrayList<String>();
        public static ArrayList<String> varY = new ArrayList<String>();
        public static ArrayList<VarXInAssgnm> varXIA = new ArrayList<VarXInAssgnm>();
        public static ArrayList<VarYInAssgnm> varYIA = new ArrayList<VarYInAssgnm>();


        // variables
        public static int roleNameCount = -1;
        public static int i = 1;
        public static int mainFunctionIsDefined = 0;
        public static int sectionCount = 0;
        public static int rolefInAsgnmFound = 0, roletInAsgnmFound = 0;
        public static int rolefInAsignFound = 0, roletInAsignFound = 0;
        public static int role1f = 0, role2f = 0;
```

```
	public static void main(String args []) throws ParseException, FileNotFoundException
	{
		String temp, temp1, temp4, temp6, temp8;
		ChRoles temp2;
		MainFun temp3;
		RoleX temp5;
		RoleOp temp7;
		Activities temp9;

		if ( args.length < 1 )
		{
			System.out.println("Please pass in the filename for a parameter.");
			System.exit(1);
		}

		Parser parser= new Parser( new
FileInputStream("C://Users//Christie//Dropbox//ADE//ADE//THESIS_PARSER//src//" +
args[0] + ".txt") );
		//System.out.println("Reading from standard input...");
		SimpleNode root = parser.tcdl_file();
		//root.dump(" ");
	}

	// Methods
	public static int checkChorRoleDecl(String mainFunName)
	{
		Enumeration t1;
		String temp;
		boolean chRolesNotDeclared = false;
		ChRoles temp1;

		t1 = chRoles_HT.keys();
		// getting data from chRoles Hashtable
		while ( t1.hasMoreElements() )
		{

			temp = (String)t1.nextElement();
			temp1 = (ChRoles)chRoles_HT.get(temp);

			if ( temp1.value != null )
			{
				if(temp1.isDefined == 0)
				{
					chRolesNotDeclared = true;
					System.out.println("Role " + temp + " at line " +
temp1.line + ", column " + temp1.column + " has not been declared yet. The role declaration
must been done before " +
							"the main function's \"" + mainFunName + "\"
declaration");
				}
			}
		}
		// afou koita3oume gia ola ta choreography roles
		if(chRolesNotDeclared)
```

```
                        System.exit(1);

                return 1;
        }

        public static int checkActivityDecl()
        {
                Enumeration t1;
                String temp;
                boolean activityNotDeclared = false;
                Activities temp1;

                t1 = activities_HT.keys();
                // getting data from activities Hashtable
                while ( t1.hasMoreElements() )
                {

                        temp = (String)t1.nextElement();
                        temp1 = (Activities)activities_HT.get(temp);

                        if ( temp1.value != null )
                        {
                                if(temp1.isDefined == 0)
                                {
                                        activityNotDeclared = true;
                                        System.out.println("Activity \"" + temp + "\" has not
been declared yet!");
                                }
                        }
                }
                // afou koita3oume gia ola ta activities
                if(activityNotDeclared)
                        System.exit(1);

                return 1;
        }

        public static int assignCheckNoComma(String f1, String t1, String ft, String xy, int line,
int column, int xyLine, int xyCol, int sectionCount, int iSection)
        {
                String section = null;
                VarXInAssgnm vX;
                VarYInAssgnm vY;

                switch(iSection)
                {
                        case 0:
                                section = "request";
                                break;

                        case 1:
                                section = "answer";
                                break;
```

```
                        case 2:
                                section = "request answer";
                                break;

                }

                if(f1.equals(ft))
                {
                        role1f = 1;         // for role f
                        rolesFInAssgnm.add(ft);
                   vX = new VarXInAssgnm("variable in assignment", xy, ft, sectionCount,
xyLine, xyCol);
                        varXIA.add(vX);

                   //varX.add(x1.image);
                        checkVarCorrectness(ft, xy, line, column, xyLine, xyCol, 2);

                }
                else
                        if(t1.equals(ft))
                        {
                                role1f = 2;         // for role t
                                rolesTInAssgnm.add(ft);
                                vY = new VarYInAssgnm("variable in assignment", xy, ft,
sectionCount, xyLine, xyCol);
                                varYIA.add(vY);

                                //varY.add(x1.image);
                                checkVarCorrectness(ft, xy, line, column, xyLine, xyCol, 2);
                        }
                        else
                        {
                                System.out.println("\"" + ft + "\" at line " + line + ", column " +
column + " is not a valid role! You may use either role \"" + f1 + "\" either role \"" + t1 + "\"
that you " +
                                "used in the " + section + " section ! ");
                                System.exit(1);
                        }

                return role1f;
        }

        public static void assignCheckWithComma(String f1, String t1, String ft, String xy, int
f1Line, int f1Col, int t1Line, int t1Col, int ftLine, int ftCol, int xyLine, int xyCol, int
sectionCount, int varArrayList)
        {
                VarXInAssgnm vX;
                VarYInAssgnm vY;

                if(varArrayList == 0)
                        if(f1.equals(ft))
                        {
                           rolesFInAssgnm.add(ft);
```

```
                        vX = new VarXInAssgnm("variable in assignment", xy, ft,
sectionCount, xyLine, xyCol);
                            varXIA.add(vX);

                        //varX.add(xy);
                            checkVarCorrectness(ft, xy, ftLine, ftCol, xyLine, xyCol, 2);
                    }
                    else
                    {
                            System.out.println("Role \"" + ft + "\" at line " + ftLine + ",
column " + ftCol + " must be the same with role \"" + f1 + "\" at line "
                                                                    + f1Line + ", column "
+ f1Col + " in the assignment section.");
                            System.exit(1);

                    }
            else
                    if(varArrayList == 1)
                            if(t1.equals(ft))
                            {
                               rolesTInAssgnm.add(ft);
                               vY = new VarYInAssgnm("variable in assignment", xy, ft,
sectionCount, xyLine, xyCol);
                                    varYIA.add(vY);

                               //varX.add(xy);
                                    checkVarCorrectness(ft, xy, ftLine, ftCol, xyLine,
xyCol, 2);
                            }
                            else
                            {
                                    System.out.println("Role \"" + ft + "\" at line " + ftLine
+ ", column " + ftCol + " must be the same with role \"" + t1 + "\" at line "
                                                                        + t1Line + ",
column " + t1Col + "\" in the assignment section.");
                                    System.exit(1);

                            }
        }

        public static int assignCheckSection2(String f1, String t1, String ft, String xy, int
f1Line, int f1Col, int t1Line, int t1Col, int ftLine, int ftCol, int xyLine, int xyCol, int
sectionCount, boolean assignRole, int checkVarList, int iSection)
        {
                String section = null;
                VarXInAssgnm vX;
                VarYInAssgnm vY;

                switch(iSection)
                {
                        case 0:
                                section = "request";
                                break;
```

```
                    case 1:
                        section = "answer";
                        break;

                    case 2:
                        section = "request answer";
                        break;
                }

                if(checkVarList == 1)    // we're referring to role t
                {
                    if(t1.equals(ft))
                    {
                        if(assignRole == true)
                            role2f = 2;         // for role t

                        rolesTInAssgnm.add(ft);
                        vY = new VarYInAssgnm("variable in assignment", xy, ft,
sectionCount, xyLine, xyCol);
                        varYIA.add(vY);

                        //varY.add(y1.image);
                        checkVarCorrectness(ft, xy, ftLine, ftCol, xyLine, xyCol, 2);
                    }
                    else
                        if(f1.equals(ft))
                        {
                            System.out.println("Role's \"" + ft + "\" variables at
line " + ftLine + ", column " + ftCol + " have already been assigned ! Please assign role's \"" +
                            t1 + "\" variables. ");
                            System.exit(1);
                        }
                        else
                        {
                            System.out.println("\"" + ft + "\" at line " + ftLine + ",
column " + ftCol + " is not a valid role! You may use role \"" + t1 + "\" that you " +
                            "used in the " + section + " section at line " + t1Line +
", column " + t1Col);
                            System.exit(1);

                        }
                }
                else
                    if(checkVarList == 0)    // we're referring to role f
                    {
                        if(f1.equals(ft))
                        {
                            if(assignRole == true)
                                role2f = 1;         // for role f

                            rolesFInAssgnm.add(ft);
                            vX = new VarXInAssgnm("variable in assignment",
xy, ft, sectionCount, xyLine, xyCol);
                            varXIA.add(vX);
```

```
                                        varX.add(xy);
                                        checkVarCorrectness(ft, xy, ftLine, ftCol, xyLine,
xyCol, 2);
                                }
                                else
                                        if(t1.equals(ft))
                                        {
                                                System.out.println("Role's \"" + ft + "\"
variables at line " + ftLine + ", column " + ftCol + " have already been assigned ! Please assign
role's \"" +
                                                f1 + "\" variables. ");
                                                System.exit(1);
                                        }
                                        else
                                        {
                                                System.out.println("\"" + ft + "\" at line " +
ftLine + ", column " + ftCol + " is not a valid role! You may use role \"" + f1 + "\" that you " +
                                                "used in the " + section + " section at line " +
f1Line + ", column " + f1Col);
                                                System.exit(1);

                                        }
                        }

                return role2f;
        }

        public static void checkVarCorrectness(String senderReceiver, String var, int
senderReceiverLine, int senderReceiverCol, int varLine, int varCol, int iSection)
        {
                String roleName, section = null;
                int roleFound = 0, varMatched = 0;
                int i;

                switch(iSection)
                {
                        case 0:
                                section = "request";
                                break;

                        case 1:
                                section = "answer";
                                break;

                        case 2:
                                section = "assignment";
                                break;

                        case 3:
                                section = "comparison";
                                break;
                }
```

```
		for (i = 0; i < roleNames.size(); i++)
		{
			roleName = roleNames.get(i);

			if(roleName.equals(senderReceiver))
			{
				roleFound = 1;

				for(int k=0; k<roleX.size(); k++)
				{
					// if we 're refering to the specific role
					if(roleName.equals(roleX.get(k).roleName))
					{
						if(roleX.get(k).value.equals(var))
						{
							varMatched = 1;
							break;	// since we found the match
wanted
						}
						else
							continue;		// continue to the next
value that might be matched to the wanted one
					}
				}

				if(varMatched == 0)
				{
					System.out.println("The variable \"" + var + "\" in the "
+ section + " section in line " + varLine + ", column " + varCol +
									" doesn't
belong to role " + "\"" + senderReceiver + "\".");

					System.exit(1);
				}

			}
		}
		if(roleFound == 0)
		{
			System.out.println("The role \"" + senderReceiver + "\" provided in the
" + section + " section, in line " + senderReceiverLine + ", column " + senderReceiverCol + "
hasn't been defined!");
			System.exit(1);
		}
	}

	public static void isVarDuplicate(int checkRoleVar)
	{
		int i, j;
		boolean duplicates = false;

		// get the variables of Role F
		if(checkRoleVar == 0)
			for(i=0; i<varXIA.size(); i++)
```

```
                                        for(j=i+1; j<varXIA.size(); j++)
                                                if(j!=i &&
varXIA.get(j).value.equals(varXIA.get(i).value))
                                                {
                                                        duplicates = true;
                                                        System.out.println("Variable \"" +
varXIA.get(j).value + "\" of role \"" + varXIA.get(j).roleName + "\" in the assignment section,
in line " + varXIA.get(j).line
                                                                                                + ",
column " + varXIA.get(j).column + "  has already been assigned. Please choose a different
variable.");
                                                        System.exit(1);
                                                }
                        else    // get the variables of Role T
                                for(i=0; i<varYIA.size(); i++)
                                        for(j=i+1; j<varYIA.size(); j++)
                                                if(j!=i &&
varYIA.get(j).value.equals(varYIA.get(i).value))
                                                {
                                                        duplicates = true;
                                                        System.out.println("Variable \"" +
varYIA.get(j).value + "\" of role \"" + varYIA.get(j).roleName + "\" in the assignment section,
in line " + varYIA.get(j).line
                                                                                                + ",
column " + varYIA.get(j).column + "  has already been assigned. Please choose a different
variable.");
                                                        System.exit(1);
                                                }
        }

        public static void isVarAlreadyInReqAns(String ft, String xy, int sectionCount, int
checkRoleVar, int checkSection)
        {
                String section = "null";
                int i, j;

                switch(checkSection)
                {
                        case 0:
                                section = "request";
                                break;

                        case 1:
                                section = "answer";
                                break;
                }

                // get the variables of Role F
                if(checkRoleVar == 0)
                {
                        for(i=0; i<varXIA.size(); i++)
                        {
                                // if we 're referring to the specific role
                                if(ft.equals(varXIA.get(i).roleName))
```

```
                                {
                                        if(varXIA.get(i).value.equals(xy))
                                        {
                                                if(sectionCount ==
varXIA.get(i).sectionCount)
                                                {
                                                        System.out.println("Variable \"" +
varXIA.get(i).value + "\" of role \"" + ft + "\" in the assignment section, in line " +
varXIA.get(i).line

        + ", column " + varXIA.get(i).column + " can't be the same with variable \"" + xy + "\"
in the " + section + " section.");
                                                        System.exit(1);
                                                }
                                        }
                                }
                        }
                }
                else    // get the variables of Role T
                {
                        for(j=0; j<varYIA.size(); j++)
                        {
                                if(ft.equals(varYIA.get(j).roleName))
                                {
                                        if(varYIA.get(j).value.equals(xy))
                                        {

                                                if(sectionCount ==
varYIA.get(j).sectionCount)
                                                {
                                                        System.out.println("Variable \"" +
varYIA.get(j).value + "\" of role \"" + ft + "\" in the assignment section, in line " +
varYIA.get(j).line

        + ", column " + varYIA.get(j).column + " can't be the same with variable \"" + xy + "\"
in the " + section + " section.");
                                                        System.exit(1);
                                                }
                                        }
                                }
                        }
                }
        }
}

PARSER_END(Parser)

SKIP :
{
  " "
| "\n"
| "\t"
| "\r"
}
```

```
TOKEN : /* OPERATORS */
{
        //< ADDMULOP: "+" | "-" | "*" | "/">
        < PLUS : "+" >
|       < MINUS : "-" >
|       < MULT : "*" >
|       < DIV : "/" >
|       < EQOP: "==" | "!=">
}

TOKEN : /* RESERVED WORDS*/
{
        < NO_ACTION: "skip">
|       < COMM: "comm">
|       < REQUEST: "request">
|       < ANSWER: "answer">
|       < REQUEST_ANSWER: "requestAnswer">
|       < CONDITION: "condition">
|       < REPETITION: "repetition" >
|  < WORK_UNIT: "workUnit" >
|       < SEQUENCE: "sequence">
|       < CHOICE: "choice" >
|       < PARALLELISM: "parallelism">
|       < CASE: "case">
|       < XR: "xr" >
|  < SKIPPED: "e">
|       < UNSPECIFIED: "delta">
|       < DEADLINE: "deadline">
|       < TRUE: "true">
|       < FALSE: "false">
}

TOKEN :
{

        < LETTER: (< UNDERSCORE > | < SMALL_CAPS > | < CAPS >) >
|       < ID: (< LETTER >)+ (< LETTER > | < DIGIT > | < UNDERSCORE >)*>
|       < PRIMARY: (< DIGIT >)+>
|       < #CAPS : ["A"-"Z"]>
|       < #SMALL_CAPS : ["a"-"z"]>
|       < #DIGIT : [ "0"-"9" ] >
|       < #UNDERSCORE : "_">
|       < L_PAR: "(" >
|       < R_PAR: ")" >
|       < L_BRACK : "[" >
|       < R_BRACK : "]" >
|       < QUOTES: "\"">
|       < COMMA: ",">
|       < DOT: ".">
|       < ASSIGN: ":=">
|       < #ROLE_ASSIGN: "::=">
}
```

```
SimpleNode tcdl_file() :
{}
{
	code() < EOF >
	{
		// molis vrei EOF koitazei na dei kata poso exoun oristei ola ta activities
		if (checkActivityDecl() == 1)
		{
			// end successfully
			System.out.println("Parsing end successfully...");
			return jjtThis;
		}
		else
		{

		}
	}

}

void code() :
{}
{
	choreography() declaration()
}

void choreography() :
{}
{
	< ID >< L_BRACK > chorRoles() < COMMA > mainFun() < R_BRACK >
}

void chorRoles() :
{
	Token tok1, tok2, tok3;
}
{
	< L_PAR > ( tok1 = < ID >
				{
					jjtThis.value=tok1.image;
					chRoles_HT.put(tok1.image, new
ChRoles("choreography role", tok1.image, tok1.beginLine, tok1.beginColumn));
				} < COMMA >)+ tok2 = < ID >
					{
						jjtThis.value=tok2.image;
						chRoles_HT.put(tok2.image,
new ChRoles("choreography role", tok2.image, tok2.beginLine, tok2.beginColumn));
					} < R_PAR >

|	tok3 = < ID >
	{
		jjtThis.value=tok3.image;
		chRoles_HT.put(tok3.image, new ChRoles("choreography role", tok3.image,
tok3.beginLine, tok3.beginColumn));
```

```
        }
}

void mainFun() :
{
        Token tok1;
}
{
        tok1 = < ID >
        {
                jjtThis.value=tok1.image;
                mainFun_HT.put(tok1.image, new MainFun("main function", tok1.image,
tok1.beginLine, tok1.beginColumn));
        }
}

void declaration() :
{
        String s, s1;
        MainFun mf;
        ChRoles ch;
        Token tok1=null, tok2 = null, tok3 = null, tok4;
        Enumeration t ,t1;
        int check;
}
{
        tok1 = < ID > "=" tok2 = < ID >< L_BRACK >
        {
                t1 = chRoles_HT.keys();
                // getting data from chRoles Hashtable
                while ( t1.hasMoreElements() )
                {
                        s1 = (String)t1.nextElement();
                        ch = (ChRoles)chRoles_HT.get(s1);

                        if ( ch.value != null )
                        {
                                if ( s1.equals(tok1.image) )
                                {
                                        if(ch.isDefined == 1)
                                                System.out.println("Role " + "\"" + s1 + "\" at
line " + ch.line + ", column " + ch.column + " has already been declared !");
                                        else // Role hasn't been declared yet
                                        {
                                                ch.isDefined = 1;
                                                System.out.println("Role " + "\"" + s1 + "\"" +
" is now declared !");
                                        }

                                }
                        }
                }

                jjtThis.value=tok2.image;
```

```
                roleNames.add(tok2.image);
                roleNameCount++;

        }roleX() < COMMA > roleOp() < R_BRACK > (declaration() | "~" d())
}

void roleX() :
{
        Token tok1, tok2, tok3;
        String roleName;
        RoleX rX;
                //retrieve the role name from the array list
        roleName = roleNames.get(roleNameCount);
}
{
        < L_PAR > ( tok1 = < ID >
                                {

                                        jjtThis.value=tok1.image;
                                        rX = new RoleX("role variable", tok1.image,
roleName);
                                        roleX.add(rX);
                                }< COMMA >)+ tok2 = < ID >
                                                        {
                                                                jjtThis.value=tok2.image;
                                                                rX = new RoleX("role
variable", tok2.image, roleName);
                                                                roleX.add(rX);
                                                        } < R_PAR >

|       tok3 = < ID >
        {
                jjtThis.value=tok3.image;
                rX = new RoleX("role variable", tok3.image, roleName);
                roleX.add(rX);
        }
}

void roleOp() :
{
        Token tok1, tok2, tok3;
        String roleName;
        RoleOp rOp;
        //retrieve the role name from the array list
        roleName = roleNames.get(roleNameCount);
}
{
        < L_PAR > ( tok1 = < ID >
                                {
                                        jjtThis.value=tok1.image;
                                        rOp = new RoleOp("role variable", tok1.image,
roleName);
                                        roleOp.add(rOp);
                                }< COMMA >)+ tok2 = < ID >
```

```
                                                        {
                                                                jjtThis.value=tok2.image;
                                                                rOp = new RoleOp("role
variable", tok2.image, roleName);
                                                                roleOp.add(rOp);
                                                        } < R_PAR >

|       tok3 = < ID >
        {
                jjtThis.value=tok3.image;
                rOp = new RoleOp("role variable", tok3.image, roleName);
                roleOp.add(rOp);
        }
}

void d() :
{
        Token tok1;
        MainFun mf;
        Activities a;
        String s, s1;
        Enumeration t, t1;
        int isActivity = 0;
}
{
        tok1 = < ID > "="
        {
                t = mainFun_HT.keys();
                // getting data from MainFun Hashtable
                while ( t.hasMoreElements() == true )
                {
                        s = (String)t.nextElement();
                        mf = (MainFun)mainFun_HT.get(s);

                        if ( mf.value != null )
                        {
                                jjtThis.value=tok1.image;
                                // ean to id mas einai i main function
                                if(s.equals(tok1.image))
                                {
                                        // check if the choreography roles have been
successfully declared
                                        if(checkChorRoleDecl(tok1.image) == 1)
                                        {
                                                if(mf.isDefined == 1)
                                                {
                                                        System.out.println("Main Function " +
"\"" + s + "\" at line " + mf.line + ", column " + mf.column + " has already been declared !");
                                                        System.exit(1);  //Exit due to the above
error
                                                }
                                                else // Main Function hasn't been declared yet
                                                {
                                                        mf.isDefined = 1;
```

```
                                                mainFunctionIsDefined = 1;
                                                isActivity = -1;            // we assure
that the id provided isn't the main function
                                                System.out.println("Main Function " +
"\"" + s + "\"" + " is now declared !");
                                        }
                                }
                        }
                        // to id den einai to main function
                        else
                        {
                                // ean exoun dilw8ei oi roloi + efoson to id != mainFun
                                if(checkChorRoleDecl(tok1.image) == 1)
                                {
                                        if(mainFunctionIsDefined == 0)
                                        {
                                                System.out.println("Main Function \""
+ s + "\" has not been declared yet. The main function's declaration must been done" +
                                                " after the role's declaration");
                                                System.exit(1);
                                        }
                                }
                        }
                }
        }

        t1 = activities_HT.keys();
        // getting data from activities Hashtable
        while ( t1.hasMoreElements() == true )
        {
                s1 = (String)t1.nextElement();
                a = (Activities)activities_HT.get(s1);

                if ( a.value != null )
                {
                        // trying to find if the specific activity from the hashtable is the
same as the id provided
                        if ( a.value.equals(tok1.image) )
                        {
                                isActivity = 1;

                                if(a.isDefined == 1)
                                        System.out.println("Activity " + "\"" + a.value
+ "\" at line " + a.line + ", column " + a.column + " shas already been declared !");
                                else // Activity hasn't been declared yet
                                {
                                        a.isDefined = 1;
                                        System.out.println("Activity " + "\"" + a.value
+ "\"" + " is now declared !");
                                }

                        }
                }
        }
```

```
                    // check if the id provided doesn't match with an activity and the main function
has declared
                    if(isActivity == 0)
                    {
                            System.out.println("\"" + tok1.image + "\" at line " + tok1.beginLine +
", column " + tok1.beginColumn + " has not declared yet!");
                            System.exit(1);
                    }


          } activity() (d())*          // otan to activity vrei kapoio termatiko kanona, tote 8a
perimenw tin epomeni dilwsi
                                                  // kapoiou activity h' poli apla mporei na eftase sto
EOF gi'auto to d() mporei na iparxei h' oxi
}

void activity() :
{
          Token tok1 = null, tok2 = null;
}
{
          basicActivity()
          |          sequenceActivity()
          |          parallelism()
          |          tok1 = < ID >
                    {
                            jjtThis.value = tok1.image;
                            activities_HT.put(tok1.image, new Activities("activity", tok1.image,
tok1.beginLine, tok1.beginColumn));
                    }
          | condition()
          | repetition()
          | workUnit()
          | generalChoice()
          | delay() activity()
          | deadlineActivity()
}

void basicActivity() :
{}
{
          < NO_ACTION >
          |          < COMM >< L_PAR > (request() | answer() | requestAnswer()) < R_PAR>
          |          assignment()
}

void assignment() :
{
          Token r1 = null, r2 = null, r3 = null, x1 = null, x2 = null, x3 = null;
}
{
          < L_PAR > (r1 = < ID >< DOT > x1 = < ID >< ASSIGN > exp()
          {
```

```
                checkVarCorrectness(r1.image, x1.image, r1.beginLine, r1.beginColumn,
x1.beginLine, x1.beginColumn, 2);
        }       < COMMA >)+ r2 = < ID >< DOT > x2 = < ID >< ASSIGN > exp() <
R_PAR>
                {
                        checkVarCorrectness(r2.image, x2.image, r2.beginLine,
r2.beginColumn, x2.beginLine, x2.beginColumn, 2);
                }
        | r3 = < ID >< DOT > x3 = < ID >< ASSIGN > exp()
        {
                checkVarCorrectness(r3.image, x3.image, r3.beginLine, r3.beginColumn,
x3.beginLine, x3.beginColumn, 2);
        }
}

void request() :
{
        Token f1 = null, f2 = null, f3 = null, f4 = null, x = null, x1 = null, x2 = null, x3 = null, t1
= null, t2 = null, t3 = null, t4 = null, y = null, y1 = null, y2 = null, y3 = null, c = null;
        int i, j, role1 = 0, role2 = 0;
        String roleF = null, roleT = null;
        VarXInAssgnm vX;
        VarYInAssgnm vY;
}
{
        < REQUEST >< L_PAR> f1 = < ID >< DOT > x = < ID >< COMMA > t1 = < ID ><
DOT > y = < ID >< COMMA >
        c = < ID >< R_PAR >< COMMA >
        {
                // increasing the section Counter
                sectionCount++;

                // checking if the variables belong to the roles provided
                checkVarCorrectness(f1.image, x.image, f1.beginLine, f1.beginColumn,
x.beginLine, x.beginColumn, 0);
                checkVarCorrectness(t1.image, y.image, t1.beginLine, t1.beginColumn,
y.beginLine, y.beginColumn, 0);
        }
                ( < L_PAR > f2 = < ID >< DOT > x1 = < ID >< ASSIGN > exp()
                {
                        role1 = assignCheckNoComma(f1.image, t1.image, f2.image,
x1.image, f2.beginLine, f2.beginColumn, x1.beginLine, x1.beginColumn, sectionCount, 0);
                }
                        (< COMMA > f3 = < ID >< DOT > x2 = < ID >< ASSIGN > exp()
                        {
                                if(role1 == 1)    // we're referring to role F
                                        assignCheckWithComma(f1.image, t1.image,
f3.image, x2.image, f1.beginLine, f1.beginColumn, t1.beginLine, t1.beginColumn,
f3.beginLine, f3.beginColumn, x2.beginLine, x2.beginColumn, sectionCount, 0);
                                else
                                        if(role1 == 2)    // we're referring to role T
```

```
                                        assignCheckWithComma(f1.image, t1.image,
f3.image, x2.image, f1.beginLine, f1.beginColumn, t1.beginLine, t1.beginColumn,
f3.beginLine, f3.beginColumn, x2.beginLine, x2.beginColumn, sectionCount, 1);
                        })* < R_PAR>
                |       f4 = < ID >< DOT > x3 = < ID >< ASSIGN > exp()
                        {
                                role1 = assignCheckNoComma(f1.image, t1.image, f4.image,
x3.image, f4.beginLine, f4.beginColumn, x3.beginLine, x3.beginColumn, sectionCount, 0);
                        }) ","
                                ( < L_PAR > t2 = < ID >< DOT > y1 = < ID >< ASSIGN >
exp()
                                {
                                        if(role1 == 1)    // if role f's variable(s) has(have) been
assigned before (',' section)
                                                role2 = assignCheckSection2(f1.image,
t1.image, t2.image, y1.image, f1.beginLine, f1.beginColumn, t1.beginLine, t1.beginColumn,
t2.beginLine, t2.beginColumn, y1.beginLine, y1.beginColumn, sectionCount, true, 1, 0);
                                        else
                                                if(role1 == 2)    // if role t's variable(s)
has(have) been assigned before (',' section)
                                                        role2 =
assignCheckSection2(f1.image, t1.image, t2.image, y1.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t2.beginLine, t2.beginColumn, y1.beginLine, y1.beginColumn,
sectionCount, true, 0, 0);
                                }
                                        (< COMMA > t3 = < ID >< DOT > y2 = < ID ><
ASSIGN > exp()
                                        {
                                                if(role2 == 2)    // we're referring to role T
                                                        role2 =
assignCheckSection2(f1.image, t1.image, t3.image, y2.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t3.beginLine, t3.beginColumn, y2.beginLine, y2.beginColumn,
sectionCount, false, 1, 0);
                                                else
                                                        if(role1 == 2)    // we're referring to
role F
                                                                role2 =
assignCheckSection2(f1.image, t1.image, t3.image, y2.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t3.beginLine, t3.beginColumn, y2.beginLine, y2.beginColumn,
sectionCount, false, 0, 0);
                                        })* < R_PAR>
                                |       t4 = < ID >< DOT > y3 = < ID >< ASSIGN > exp()
                                        {
                                                if(role1 == 1)    // if role f's variable(s)
has(have) been assigned before (',' section)
                                                        role2 =
assignCheckSection2(f1.image, t1.image, t4.image, y3.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t4.beginLine, t4.beginColumn, y3.beginLine, y3.beginColumn,
sectionCount, true, 1, 0);
                                                else
                                                        if(role1 == 2)    // if role t's variable(s)
has(have) been assigned before (',' section)
                                                                role2 =
assignCheckSection2(f1.image, t1.image, t4.image, y3.image, f1.beginLine, f1.beginColumn,
```

```
t1.beginLine, t1.beginColumn, t4.beginLine, t4.beginColumn, y3.beginLine, y3.beginColumn,
sectionCount, true, 0, 0);
                                        }) "," op()
                                                {
                                                        for(i = 0; i < rolesFInAssgnm.size();
i++)
                                                        {
                                                                roleF =
rolesFInAssgnm.get(i);

                                                                for(j = 0; j <
rolesTInAssgnm.size(); j++)
                                                                {
                                                                        roleT =
rolesTInAssgnm.get(j);

                                                                        // check if there any of
role F's variables in assignment section is the same with variable x in request section or with
variable u in answer section

        isVarAlreadyInReqAns(f1.image, x.image, sectionCount, 0, 0);
                                                                        // check if there any of
role T's variables in assignment section is the same with variable y in request section or with
variable v in answer section

        isVarAlreadyInReqAns(t1.image, y.image, sectionCount, 1, 0);


                                                                        // check if there any
duplicates on the role F's variables
                                                                        isVarDuplicate(0);
                                                                        // check if there any
duplicates on the role T's variables
                                                                        isVarDuplicate(1);

                                                                }
                                                        }
                                                }
}


void answer() :
{
        Token f1 = null, f2 = null, f3 = null, f4 = null, x = null, x1 = null, x2 = null, x3 = null, t1
= null, t2 = null, t3 = null, t4 = null, y = null, y1 = null, y2 = null, y3 = null, c = null;
        int i, j, role1 = 0, role2 = 0;
        String roleF = null, roleT = null;
        VarXInAssgnm vX;
        VarYInAssgnm vY;
}
{
        < ANSWER >< L_PAR> f1 = < ID >< DOT > x = < ID >< COMMA > t1 = < ID ><
DOT > y = < ID >< COMMA >
        c = < ID >< R_PAR >< COMMA >
```

```
	{
		// increasing the section counter
		sectionCount++;

		// checking if the variables belong to the roles provided
		checkVarCorrectness(f1.image, x.image, f1.beginLine, f1.beginColumn,
x.beginLine, x.beginColumn, 1);
		checkVarCorrectness(t1.image, y.image, t1.beginLine, t1.beginColumn,
y.beginLine, y.beginColumn, 1);
	}
		( < L_PAR > f2 = < ID >< DOT > x1 = < ID >< ASSIGN >< QUOTES > exp()
< QUOTES >
		{
			role1 = assignCheckNoComma(f1.image, t1.image, f2.image,
x1.image, f2.beginLine, f2.beginColumn, x1.beginLine, x1.beginColumn, sectionCount, 1);
		}
			(< COMMA > f3 = < ID >< DOT > x2 = < ID >< ASSIGN ><
QUOTES > exp()
			{
				if(role1 == 1)	// we're referring to role F
					assignCheckWithComma(f1.image, t1.image,
f3.image, x2.image, f1.beginLine, f1.beginColumn, t1.beginLine, t1.beginColumn,
f3.beginLine, f3.beginColumn, x2.beginLine, x2.beginColumn, sectionCount, 0);
				else
					if(role1 == 2)	// we're referring to role T

						assignCheckWithComma(f1.image, t1.image,
f3.image, x2.image, f1.beginLine, f1.beginColumn, t1.beginLine, t1.beginColumn,
f3.beginLine, f3.beginColumn, x2.beginLine, x2.beginColumn, sectionCount, 1);

			} < QUOTES >)* < R_PAR>
		|	f4 = < ID >< DOT > x3 = < ID >< ASSIGN >< QUOTES > exp() <
QUOTES >
			{
				role1 = assignCheckNoComma(f1.image, t1.image, f4.image,
x3.image, f4.beginLine, f4.beginColumn, x3.beginLine, x3.beginColumn, sectionCount, 1);
			}) ","
				( < L_PAR > t2 = < ID >< DOT > y1 = < ID >< ASSIGN ><
QUOTES > exp() < QUOTES >
				{
					if(role1 == 1)	// if role f's variable(s) has(have) been
assigned before (',' section)
						role2 = assignCheckSection2(f1.image,
t1.image, t2.image, y1.image, f1.beginLine, f1.beginColumn, t1.beginLine, t1.beginColumn,
t2.beginLine, t2.beginColumn, y1.beginLine, y1.beginColumn, sectionCount, true, 1, 1);
					else
						if(role1 == 2)	// if role t's variable(s)
has(have) been assigned before (',' section)
							role2 =
assignCheckSection2(f1.image, t1.image, t2.image, y1.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t2.beginLine, t2.beginColumn, y1.beginLine, y1.beginColumn,
sectionCount, true, 0, 1);
				}
```

```
                                        (< COMMA > t3 = < ID >< DOT > y2 = < ID ><
ASSIGN >< QUOTES > exp()
                                        {
                                                if(role2 == 2)    // we're referring to role T
                                                        role2 =
assignCheckSection2(f1.image, t1.image, t3.image, y2.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t3.beginLine, t3.beginColumn, y2.beginLine, y2.beginColumn,
sectionCount, false, 1, 1);
                                                else
                                                        if(role1 == 2)    // we're referring to
role F
                                                                role2 =
assignCheckSection2(f1.image, t1.image, t3.image, y2.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t3.beginLine, t3.beginColumn, y2.beginLine, y2.beginColumn,
sectionCount, false, 0, 1);
                                        }       < QUOTES >)* < R_PAR>
                                |       t4 = < ID >< DOT > y3 = < ID >< ASSIGN ><
QUOTES > exp() < QUOTES >
                                        {
                                                if(role1 == 1)    // if role f's variable(s)
has(have) been assigned before (',' section)
                                                        role2 =
assignCheckSection2(f1.image, t1.image, t4.image, y3.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t4.beginLine, t4.beginColumn, y3.beginLine, y3.beginColumn,
sectionCount, true, 1, 1);
                                                else
                                                        if(role1 == 2)    // if role t's variable(s)
has(have) been assigned before (',' section)
                                                                role2 =
assignCheckSection2(f1.image, t1.image, t4.image, y3.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t4.beginLine, t4.beginColumn, y3.beginLine, y3.beginColumn,
sectionCount, true, 0, 1);
                                        }) ","  op()
                                                {
                                                        for(i = 0; i < rolesFInAssgnm.size();
i++)
                                                        {

                                                                roleF =
rolesFInAssgnm.get(i);

                                                                for(j = 0; j <
rolesTInAssgnm.size(); j++)
                                                                {

                                                                        roleT =
rolesTInAssgnm.get(j);

                                                                // check if there any of
role F's variables in assignment section is the same with variable x in request section or with
variable u in answer section

        isVarAlreadyInReqAns(f1.image, x.image, sectionCount, 0, 1);
```

```
                                                                    // check if there any of
role T's variables in assignment section is the same with variable y in request section or with
variable v in answer section

        isVarAlreadyInReqAns(t1.image, y.image, sectionCount, 1, 1);

                                                                    // check if there any
duplicates on the role F's variables
                                                                    isVarDuplicate(0);
                                                                    // check if there any
duplicates on the role T's variables
                                                                    isVarDuplicate(1);
                                                                }
                                                            }
                                                        }
}


void requestAnswer() :
{
        Token f1 = null, f2 = null, f3 = null, f4 = null, f5 = null, x = null, u = null, x1 = null, x2
= null, x3 = null, t1 = null, t2 = null, t3 = null, t4 = null, t5 = null, y = null, v = null, y1 = null,
y2 = null, y3 = null, cf = null, ct = null;
        Enumeration e;
        String roleF = null, roleT = null;
        RoleX r;
        VarXInAssgnm vX;
        VarYInAssgnm vY;
        int i, j, role1 = 0, role2 = 0, rolefFound = 0, roletFound = 0, varXMatched = 0,
varUMatched = 0;
}
{
        < REQUEST_ANSWER >< L_PAR>< REQUEST >< L_PAR> f1 = < ID >< DOT > x
= < ID >< COMMA > t1 = < ID >< DOT > y = < ID >< COMMA >
        ct = < ID >< R_PAR >< COMMA >
        {
                //increasing the section counter
                sectionCount++;

                // checking if the variables belong to the roles provided
                checkVarCorrectness(f1.image, x.image, f1.beginLine, f1.beginColumn,
x.beginLine, x.beginColumn, 0);
                checkVarCorrectness( t1.image, y.image, t1.beginLine, t1.beginColumn,
y.beginLine, y.beginColumn, 0);
        }
                < ANSWER >< L_PAR> f2 = < ID >< DOT > u = < ID >< COMMA > t2 = <
ID >< DOT > v = < ID >< COMMA >
                cf = < ID >< R_PAR >< COMMA >
                {
                        if(f1.image.equals(f2.image))
                        {
                                if(t1.image.equals(t2.image))
                                {
                                        if(!(x.image.equals(u.image)))
```

```
                                        {
                                                if(!(y.image.equals(v.image)))
                                                {
                                                        // checking if the variables belong to
the roles provided
                                                        checkVarCorrectness(f2.image,
u.image, f2.beginLine, f2.beginColumn, u.beginLine, u.beginColumn, 1);
                                                        checkVarCorrectness(t2.image,
v.image, t2.beginLine, t2.beginColumn, v.beginLine, v.beginColumn, 1);

                                                }
                                                else    // y == v
                                                {
                                                        System.out.println("Variable \"" +
v.image + "\" of role \"" + t2.image + "\" in the answer section, in line " + v.beginLine + ",
column "

        + v.beginColumn + " can't be the same with the one in the request section. Please
choose a different variable.");
                                                        System.exit(1);
                                                }
                                        }
                                        else    // x == u
                                        {
                                                System.out.println("Variable \"" + u.image +
"\" of role \"" + f2.image + "\" in the answer section, in line " + u.beginLine + ", column " +
u.beginColumn +

        " can't be the same with the one in the request section. Please choose a different
variable.");
                                                System.exit(1);
                                        }
                                }
                                else    // t1 != t2
                                {
                                        System.out.println("Role \"" + t1.image + "\" in the
request section, in line " + t1.beginLine + ", column " + t1.beginColumn +

        "must be the same with role \"" + t2.image + "\" in the answer section in line " +
t2.beginLine + ", column " +

         t2.beginColumn + ".");
                                        System.exit(1);
                                }
                        }
                        else    //f1 != f2
                        {
                                System.out.println("Role \"" + f1.image + "\" in the request
section, in line " + f1.beginLine + ", column " + f1.beginColumn +

        "must be the same with role \"" + f2.image + "\" in the answer section in line " +
f2.beginLine + ", column " +

        f2.beginColumn + ".");
```

```
                        System.exit(1);
                    }
                }
                    ( < L_PAR > f3 = < ID >< DOT > x1 = < ID >< ASSIGN >< QUOTES
> exp() < QUOTES >
                    {
                        role1 = assignCheckNoComma(f1.image, t1.image, f3.image,
x1.image, f3.beginLine, f3.beginColumn, x1.beginLine, x1.beginColumn, sectionCount, 2);
                    }
                        (< COMMA > f4 = < ID >< DOT > x2 = < ID >< ASSIGN ><
QUOTES > exp()
                        {
                            if(role1 == 1)    // we're referring to role F
                                assignCheckWithComma(f1.image, t1.image,
f4.image, x2.image, f1.beginLine, f1.beginColumn, t1.beginLine, t1.beginColumn,
f4.beginLine, f4.beginColumn, x2.beginLine, x2.beginColumn, sectionCount, 0);
                            else
                                if(role1 == 2)    // we're referring to role T

                                    assignCheckWithComma(f1.image,
t1.image, f4.image, x2.image, f1.beginLine, f1.beginColumn, t1.beginLine, t1.beginColumn,
f4.beginLine, f4.beginColumn, x2.beginLine, x2.beginColumn, sectionCount, 2);
                        }
                        < QUOTES >)* < R_PAR>
                    |   f5 = < ID >< DOT > x3 = < ID >< ASSIGN >< QUOTES >
exp() < QUOTES >
                        {
                            role1 = assignCheckNoComma(f1.image, t1.image,
f5.image, x3.image, f5.beginLine, f5.beginColumn, x3.beginLine, x3.beginColumn,
sectionCount, 2);
                        }) ","
                            ( < L_PAR > t3 = < ID >< DOT > y1 = < ID ><
ASSIGN >< QUOTES > exp() < QUOTES >
                            {
                                if(role1 == 1)    // if role f's variable(s)
has(have) been assigned before (',' section)
                                    role2 =
assignCheckSection2(f1.image, t1.image, t3.image, y1.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t3.beginLine, t3.beginColumn, y1.beginLine, y1.beginColumn,
sectionCount, true, 1, 2);
                                else
                                    if(role1 == 2)    // if role t's variable(s)
has(have) been assigned before (',' section)
                                        role2 =
assignCheckSection2(f1.image, t1.image, t3.image, y1.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t3.beginLine, t3.beginColumn, y1.beginLine, y1.beginColumn,
sectionCount, true, 0, 2);
                            }
                                (< COMMA > t4 = < ID >< DOT > y2 = < ID
>< ASSIGN >< QUOTES > exp()
                                {
                                    if(role2 == 2)    // we're referring to
role T
```

```
                                                        role2 =
assignCheckSection2(f1.image, t1.image, t4.image, y2.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t4.beginLine, t4.beginColumn, y2.beginLine, y2.beginColumn,
sectionCount, false, 1, 2);
                                                else
                                                        if(role1 == 2)    // we're
referring to role F
                                                                role2 =
assignCheckSection2(f1.image, t1.image, t4.image, y2.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t4.beginLine, t4.beginColumn, y2.beginLine, y2.beginColumn,
sectionCount, false, 0, 2);
                                        } < QUOTES >)* < R_PAR >
                                |       t5 = < ID >< DOT > y3 = < ID >< ASSIGN ><
QUOTES > exp() < QUOTES >
                                        {
                                                if(role1 == 1)    // if role f's variable(s)
has(have) been assigned before (',' section)
                                                        role2 =
assignCheckSection2(f1.image, t1.image, t5.image, y3.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t5.beginLine, t5.beginColumn, y3.beginLine, y3.beginColumn,
sectionCount, true, 1, 2);
                                                else
                                                        if(role1 == 2)    // if role t's
variable(s) has(have) been assigned before (',' section)
                                                                role2 =
assignCheckSection2(f1.image, t1.image, t5.image, y3.image, f1.beginLine, f1.beginColumn,
t1.beginLine, t1.beginColumn, t5.beginLine, t5.beginColumn, y3.beginLine, y3.beginColumn,
sectionCount, true, 0, 2);
                                        }) ","  op() < R_PAR >
                                                {
                                                        for(i = 0; i <
rolesFInAssgnm.size(); i++)
                                                        {
                                                                roleF =
rolesFInAssgnm.get(i);

                                                                for(j = 0; j <
rolesTInAssgnm.size(); j++)
                                                                {
                                                                        roleT =
rolesTInAssgnm.get(j);

                                                                        // check if
there any of role F's variables in assignment section is the same with variable x in request
section

        isVarAlreadyInReqAns(f1.image, x.image, sectionCount, 0, 0);
                                                                        // check if
there any of role F's variables in assignment section is the same with variable u in answer
section

        isVarAlreadyInReqAns(f2.image, u.image, sectionCount, 0, 1);
```

```
                                                                    // check if
there any of role T's variables in assignment section is the same with variable y in request
section

        isVarAlreadyInReqAns(t1.image, y.image, sectionCount, 1, 0);
                                                                    // check if
there any of role T's variables in assignment section is the same with variable v in answer
section

        isVarAlreadyInReqAns(t2.image, v.image, sectionCount, 1, 1);

                                                                    // check if
there any duplicates on the role F's variables

        isVarDuplicate(0);
                                                                    // check if
there any duplicates on the role T's variables

        isVarDuplicate(1);
                                                                        }
                                                                    }
                                                                }
}

void op() :
{}
{
        < SKIPPED >    | delay() |          < UNSPECIFIED >
}

void delay() :
{
}
{
        < XR > < L_PAR >< PRIMARY >< R_PAR >
}

void exp() :
{}
{
                term() (< PLUS > term() | < MINUS > term())*
}

void term() :
{}
{
                factor() (< MULT > factor() | < DIV > factor())*
}

void factor() :
{
        Token tok1 = null, r = null, x = null;
}
{
```

```
                tok1 = < ID >(< DOT > x = < ID >
                {
                        checkVarCorrectness(tok1.image, x.image, tok1.beginLine,
tok1.beginColumn, x.beginLine, x.beginColumn, 3);
                })?
        |       < QUOTES >< ID >< QUOTES >
        |       < PRIMARY >
        |       < TRUE >
        |       < FALSE >
        |       < L_PAR > exp() < R_PAR >
}


void condition() :
{}
{
        < CONDITION >< L_PAR > guard() "?" activity() < R_PAR >
}

void repetition() :
{}
{
        < REPETITION >< L_PAR > guard() "*" activity() < R_PAR >
}

void workUnit() :
{}
{
        < WORK_UNIT >< L_PAR > guard() ":" activity() ":" guard() < R_PAR >
}

void sequenceActivity() :
{}
{
        < SEQUENCE >< L_PAR > (activity() ";" )+ activity() < R_PAR >
}

void choice() :
{}
{
        < CHOICE >< L_PAR > (activity() "+" )+ activity() < R_PAR >
}

void generalChoice() :
{}
{
        < CASE >< L_PAR > guard() ":" activity() < COMMA > guard() ":" activity() <
R_PAR >
}

void parallelism() :
{}
{
        < PARALLELISM >< L_PAR > (activity() "||" )+ activity() < R_PAR >
```

```
}

void deadlineActivity() :
{}
{
        < DEADLINE >< L_PAR > activity() < COMMA >< PRIMARY >< COMMA >
activity() < COMMA > activity() < R_PAR >
}


void guard() :
{}
{
        boolOr()
}

void boolOr() :
{}
{
        boolAnd() ("|" boolAnd())*
}

void boolAnd() :
{}
{
        compExp() ("&" compExp())*
}

void compExp() :
{}
{
        simpleExp() (< EQOP > simpleExp())*
}

void simpleExp() :
{
        Token r = null, x = null;
}
{
        exp()
|       < L_PAR > guard() < R_PAR >
}
```

A.2Κλάσεις που χρειάζονται για να γίνει compileτο αρχείο Parser,jjt

```
// ChRoles.java
public class ChRoles extends Object {
        String kind, value;
        int line, column, isDefined;

        public ChRoles(String iKind, String iValue, int iLine, int iColumn) {
                kind = iKind;
                value = iValue;
                line = iLine;
                column = iColumn;
                isDefined = 0;
        }
}

// MainFun.java
public class ChRoles extends Object {
        String kind, value;
        int line, column, isDefined;

          public ChRoles(String iKind, String iValue, int iLine, int iColumn) {
                kind = iKind;
                value = iValue;
                line = iLine;
                column = iColumn;
                isDefined = 0;
        }
}

// RoleX.java

public class RoleX extends Object {
        String kind, value, roleName;

        public RoleX(String iKind, String iValue, String iRoleName) {
                kind = iKind;
                value = iValue;
                roleName = iRoleName;
        }
}


//RoleOp.java
 public class RoleOp extends Object {
        String kind, value, roleName;

        public RoleOp(String iKind, String iValue, String iRoleName) {
                kind = iKind;
                value = iValue;
                roleName = iRoleName;
        }
}
```

```
// RoleNames.java
import java.util.ArrayList;

public class RoleNames extends Object {
	String roleName;
	ArrayList<RoleX> rX = new ArrayList<RoleX>();
	ArrayList<RoleOp> rOp = new ArrayList<RoleOp>();
	boolean copyRoleX, copyRoleOp;

	public RoleNames(String iRoleName, ArrayList<RoleX> iRx, ArrayList<RoleOp>
	iRop) {
		roleName = iRoleName;
		copyRoleX = rX.addAll(iRx);
		copyRoleOp = rOp.addAll(iRop);
	}
}

// Activities,java
public class Activities extends Object {
	String kind, value;
	int line, column, isDefined;

	public Activities(String iKind, String iValue, int iLine, int iColumn) {
		kind = iKind;
		value = iValue;
		line = iLine;
		column = iColumn;
		isDefined = 0;
	}
}

// VarXInAssngm.java
public class VarXInAssgnm extends Object {
	String kind, value, roleName;
	int sectionCount, line, column;

	public VarXInAssgnm(String iKind, String iValue, String iRoleName, int
	iSectionCount, int iLine, int iColumn) {
		kind = iKind;
		value = iValue;
		sectionCount = iSectionCount;
		line = iLine;
		column = iColumn;
		roleName = iRoleName;
	}
}
```

```
// VarYInAssgnm.java
public class VarYInAssgnm extends Object {

	String kind, value, roleName;
	int sectionCount, line, column;

	public VarYInAssgnm(String iKind, String iValue, String iRoleName, int
	iSectionCount, int iLine, int iColumn) {
		kind = iKind;
		value = iValue;
		sectionCount = iSectionCount;
		line = iLine;
		column = iColumn;
		roleName = iRoleName;
	}
}
```

Παράρτημα Β

Πιο κάτω φαίνεται η γραμματική με τη μορφή δέντρου (Σχήμα Β2), για την είσοδο(Σχήμα Β1).Αυτό ίσως φανεί χρήσιμο σε μελλο ντική εργασία, όπο υ καλό θα ήταν να βλέπουμε την αναπαράσταση της γραμματικής σε δεντρική μορφή, κάτι που θα μπορούσε να βοηθήσει εν μέρη και για τη διάσχιση του ASTδέντρου. Αυτό το επιτυγχάνουμε με την κλήση της μεθόδου dump..

```
1 PurchaseOrder[(buyer, seller, credit, inv), purchaseOrder]
2 buyer = Buyer[(po, poAck, poResp, poRej, poState),(PoReqOP, DisplayOP)]
3 seller = Seller[(po, poAck, poResp, poRej, ccReq, ccResp, icReq, icResp,
4                          poState, ccState, icState), PoHandleOP]
5
6 inv = Inv[(icReq, icResp, icState), InvCheckOP]
7
8 credit = Credit[(ccReq, ccResp, ccState), CreditCheckOP] ~
9
10 purchaseOrder = poRequest; checkCreditInv; finalAnswer
11 poRequest = skip
```

Σχήμα Β1: Είσοδος 1

```
tcdl_file
 code
  choreography
   chorRoles
   mainFun
  declaration
   roleX
   roleOp
   declaration
    roleX
    roleOp
    declaration
     roleX
     roleOp
     declaration
      roleX
      roleOp
      declaration
       d
        activity
         A
          activity
           A
            activity
             A
              d
               activity
                basicActivity
```

Σχήμα Β2: Δεντρική Αναπαράσταση Γραμματικής βάσει της Εισόδου 1