Ατομική Διπλωματική Εργασία

ΥΛΟΠΟΙΗΣΗ ΕΡΓΑΛΕΙΟΥ ΓΙΑ ΤΗΝ ΠΡΟΣΟΜΟΙΩΣΗ ΠΛΗΘΥΣΜΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΜΟΝΤΕΛΟΠΟΙΗΜΕΝΩΝ ΣΤΗΝ ΑΛΓΕΒΡΑ ΔΙΕΡΓΑΣΙΩΝ PALPS

Διονυσία Αγαθοκλέους

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Δεκέμβριος 2011

Ευχαριστίες

Ένα μεγάλο ευχαριστώ θα ήθελα να δώσω στην επιβλέπουσα καθηγήτρια μου κα. Άννα Φιλίππου η οποία με βοήθησε σε μεγάλο βαθμό στην διεκπεραίωση της διπλωματικής μου εργασίας.

Περίληψη

Η παρούσα εργασία ασχολείται με το θέμα της ανάπτυξης λογισμικού για την προσομοίωση πληθυσμιακών συστημάτων μοντελοποιημένα στην άλγεβρα διεργασιών PALPS και έχει σαν γενικότερο σκοπό την μελέτη και ανάλυση πληθυσμών μέσω του εργαλείου αυτού καθώς και την εξαγωγή συμπερασμάτων για την μελλοντική κατάσταση ενός ή περισσότερων πληθυσμών, που μπορεί να προκύψει κάτω από ένα υποσύνολο κανόνων σημασιολογίας που έχει η άλγεβρα διεργασιών PALPS. Η άλγεβρα διεργασιών PALPS είναι επέκταση της άλγεβρα διεργασιών CCS η οποία παρουσιάζει ως μέθοδο επικοινωνίας μεταξύ των διεργασιών του συστήματος τον δυαδικό συγχρονισμό και οι διεργασίες της κτίζονται από ένα σύνολο ατομικών ενεργειών με τη χρήση τελεστών σύνθεσης διεργασιών. Η PALPS παίρνει αυτούς τους κανόνες και τους προσαρμόζει με την έννοια του χώρου. Επίσης για την ανάπτυξη του συστήματος μελέτησα το κύκλο ανάπτυξης λογισμικού στο οποίο και αναλύονται οι απαιτήσεις του λογισμικού καθώς και τα χαρακτηριστικά του λογισμικού. Γίνεται επίσης ανάλυση και επεξήγηση των αλγόριθμων που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Πληθυσμιακή Οικολογία

Η πληθυσμιακή οικολογία αποτελεί τον κλάδο της οικολογίας που ασχολείται με την δομή και την δυναμική των πληθυσμών. Μελετά τις διάφορες ποσοτικές μεταβολές ενός πληθυσμού στο χρόνο και στον βιότοπο που επιβιώνει. Υπάρχουν πολλές οικολογικές και βιολογικές διαδικασίες οι οποίες προσδιορίζουν αλλά και μεταβάλλουν τον τρόπο διάταξης των ατόμων του πληθυσμού στον βιότοπο τους. Για την κατανόηση των διαδικασιών αυτών απαιτείται η ύπαρξη ενός κατάλληλου προτύπου, το οποίο να έχει την ικανότητα να προβλέπει την κατάσταση στην οποία θα βρίσκεται το σύστημα που μελετάται μετά την εκτέλεση διάφορων ενεργειών. Για να είναι αξιόπιστα τα αποτελέσματα ενός τέτοιου προτύπου, πρέπει να γίνει σωστή παρατήρηση και μελέτη των βιολογικών και οικολογικών διαδικασιών του πληθυσμού ή των πληθυσμών που θα εξετασθούν.

Το κύριο πρόβλημα με το οποίο καταπιάνεται η οικολογία πληθυσμών είναι πως τα χαρακτηριστικά και οι ενέργειες πληθυσμών μπορούν να αντληθούν από τα χαρακτηριστικά και τις ενέργειες των ατόμων του πληθυσμού. Οι μελέτες που πραγματοποιούνται γύρω από το θέμα της δυναμικής των πληθυσμών ασχολούνται με τον τρόπο συνύπαρξης πληθυσμών δύο ή περισσότερων ειδών. Για

σκοπούς μιας τέτοιας μελέτης απαιτείται η απεικόνιση του κάθε ατόμου από την γέννηση του, την αύξηση του , την αναπαραγωγή του , μέχρι και τον θάνατο του. Η απεικόνιση αυτή αν και είναι αρκετά ακριβή , επιτρέπει στους επιστήμονες της πληθυσμιακής οικολογίας να εξερευνήσουν και να ερμηνεύσουν τον τρόπο που η δυναμική ενός πληθυσμού ή ενός οικοσυστήματος προκύπτει μέσω της αλληλεπίδρασης τους με τα υπόλοιπα άτομα με τα οποία συνυπάρχουν στο βιότοπο στο οποίο ζουν.

1.2 Τεχνικές ανάλυσης συστημάτων

Οι τεχνικές μοντελοποίησης και ανάλυσης ενός ή περισσότερων πληθυσμών που συνυπάρχουν σε ένα περιβάλλον αποτελούν ένα σημαντικό σημείο για τις επιστήμες της βιολογίας και της οικολογίας. Με την βοήθεια αυτών των τεχνικών μπορούν να γίνουν διάφορα πειράματα και μετρήσεις για τον τρόπο συμπεριφοράς ενός πληθυσμού καθώς επίσης και της κατάστασης στην οποία θα επέλθει ο πληθυσμός μετά από κάποιο χρονικό διάστημα. Έτσι μπορούν οι επιστήμονες να μελετούν τις συμπεριφορές των πληθυσμών και να εξάγουν συμπεράσματα για τον τρόπο επιβίωσης τους και γενικά για τον τρόπο που ζουν μέσα σε ένα περιβάλλον. Ένα από τα σημαντικότερα μοντέλα που προτάθηκαν για την περιγραφή πληθυσμιακών συστημάτων είναι οι άλγεβρες διεργασιών. Οι άλγεβρες διεργασιών αποτελούν πρότυπα με την ικανότητα μοντελοποίησης και ανάλυσης πληθυσμιακών συστημάτων. Επιτρέπουν την προδιαγραφή των μεμονωμένων συνιστωσών ενός συστήματος και παρέχουν τη δυνατότητα κατανόησης πώς οι συνιστώσες αυτές αλληλεπιδρούν μεταξύ τους, συμβάλλοντας στην γενική συμπεριφορά του συστήματος. Εξάγονται έτσι συμπεράσματα για ένα σύνθετο σύστημα από τις ιδιότητες των επιμέρους συνιστωσών του. Η PALPS(Process Algebra with Location for Population Systems) είναι ένα πρότυπο αλγεβρών διεργασιών για την μελέτη και ανάλυση πληθυσμών. Κύριο στοιχείο της γλώσσας αυτή μας επιτρέπει να εξάγουμε συμπεράσματα για την κατάσταση του χώρου στον οποίο συμβιώνουν συγκεκριμένοι πληθυσμοί, μετά τις αλληλεπιδράσεις των ατόμων των πληθυσμών στην προσπάθεια τους να επιβιώσουν.

1.3 Σκοπός της Διπλωματικής Εργασίας

Πρωταρχικός στόχος της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη λογισμικού το οποίο θα προσομοιώνει πληθυσμιακά συστήματα μοντελοποιημένα στην άλγεβρα διεργασιών PALPS, με στόχο την εξαγωγή συμπερασμάτων για την μελλοντική κατάσταση ενός ή περισσότερων πληθυσμών, που μπορεί να προκύψει κάτω από συγκεκριμένες συνθήκες. Η συμπεριφορά των ατόμων ενός πληθυσμού κυβερνάται από ένα σύνολο κανόνων τους οποίους και ενσωμάτωσα στο λογισμικό. Οι κανόνες αυτοί επιτρέπουν την αλληλεπίδραση του ενός ατόμου με τα υπόλοιπα του πληθυσμού, καθώς επίσης και με το περιβάλλον του. Το λογισμικό θα παίρνει σαν είσοδο ένα πληθυσμιακό σύστημα μοντελοποιημένο στην άλγεβρα διεργασίας PALPS και θα παρουσιάζει στο χρήστη κάθε πιθανή αντίδραση και την δυσδιάστατη υπόσταση του πληθυσμιακού συστήματος ανά πάσα στιγμή. Επίσης του παρέχεται η δυνατότητα να προσομοιώσει κάποια αντίδραση που επιλέγει ο ίδιος, ή μία τυχαία αντίδραση, και να παρακολουθήσει τις αλλαγές που επιφέρει αυτή η αντίδραση στο πληθυσμιακό σύστημα που εξετάζει. Στο τέλος της προσομοίωσης δίνεται η δυνατότητα στο χρήστη να εξάγει συμπεράσματα για το πληθυσμιακό σύστημα το οποίο προσομοίωσε κατά πόσο αυξομειώθηκε. Αν δηλαδή έχουν επιβιώσει τα άτομα στο πληθυσμό ή έχουν πεθάνει μετά την εκτέλεση κάποιων ενεργειών.

1.3 Δομή Διπλωματικής Εργασίας

Στο έγγραφο θα βρείτε κάποιους ορισμούς γύρω από τα πληθυσμιακά συστήματα και την άλγεβρα διεργασίας PALPS που προηγήθηκε της ανάπτυξης του λογισμικού καθώς και τα κύρια χαρακτηριστικά του που ακολούθησα. Συγκεκριμένα το **πρώτο κεφάλαιο** αποτελεί μια εισαγωγή με το σκοπό και το κίνητρο αυτής της διπλωματικής εργασίας. Εξηγεί την σημαντικότητα των τεχνικών μοντελοποίησης και ανάλυσης πληθυσμιακών συστημάτων και δίνει ένα σύντομο ορισμό για τις άλγεβρες διεργασιών. Ακολουθούν μια πιο εκτενή αναφορά για τις άλγεβρες διεργασιών και τις τεχνικές αναπαράστασης πληθυσμιακών συστημάτων που μελετήθηκαν στο **δεύτερο κεφάλαιο**. Το **τρίτο κεφάλαιο** περιγράφει την άλγεβρα

διεργασιών PALPS, την σύνταξη της και σημασιολογία της. Δίνει επίσης ένα παράδειγμα μοντελοποίησης πληθυσμιακού συστήματος. Στο **κεφάλαιο 4** παρουσιάζεται ο κύκλος ανάπτυξης του λογισμικού, αναλύονται οι απαιτήσεις του λογισμικού καθώς και τα χαρακτηριστικά του λογισμικού ενώ στο **κεφάλαιο 5** θα βρείτε αναλυτικά την επεξήγηση των αλγόριθμων που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος. Στο **κεφάλαιο 6** θα βρείτε σε ποιούς τομείς μπορεί να χρησιμοποιηθεί το λογισμικό που αναπτύχθηκε καθώς και τις προοπτικές του στο μέλλον και κάποια γενικά συμπεράσματα τόσο για τη χρήση της άλγεβρας διεργασιών PALPS, όσο και για την χρήση του λογισμικού που έχει αναπτυχθεί.

Κεφάλαιο 2

Πληθυσμιακά Συστήματα και Άλγεβρες Διεργασιών

2.1 Πληθυσμιακά Συστήματα

Ένα πληθυσμιακό σύστημα ορίζεται ως ένα σύνολο ατόμων του ίδιου είδους που ζουν και αναπτύσσονται στην ίδια περιοχή. Κάθε φυσικός πληθυσμός παρουσιάζει ποσοτικές μεταβολές στον χρόνο, οι οποίες μπορούν να χαρακτηριστούν μικρές, μεγάλες, συχνές, σπάνιες ή περιοδικές. Οι μεταβολές αυτές μπορεί να προέρχονται από αλλαγές στον ρυθμό γεννήσεων, θανάτων ή και των δύο, που μπορεί να προκύψουν, είτε λόγω της συνύπαρξης τους με άλλους πληθυσμούς, είτε λόγω κάποιου ενδογενούς παράγοντα όπως η κακή συνεργασία μεταξύ τους στον καταμερισμό τους στις διάφορες εργασίες για την επιβίωση τους.

Πολλοί επιστήμονες ασχολήθηκαν με τη μελέτη πληθυσμιακών συστημάτων. Στόχος τους ήταν η κατανόηση της συμπεριφοράς των πληθυσμών μέσω των σημαντικότερών συστατικών τους. Υπάρχουν πολλοί και διάφοροι τρόποι μοντελοποίησης και ανάλυσης πληθυσμών. Με την βοήθεια αυτών των τεχνικών μπορούν να γίνουν διάφορα πειράματα και μετρήσεις για τον τρόπο συμπεριφοράς ενός πληθυσμού καθώς επίσης και της κατάστασης στην οποία θα επέλθει ο πληθυσμός μετά από κάποιο χρονικό διάστημα. Έτσι μπορούν οι επιστήμονες να μελετούν τις συμπεριφορές των πληθυσμών και να εξάγουν συμπεράσματα για τον τρόπο επιβίωσης τους και γενικά για τον τρόπο που ζουν μέσα σε ένα περιβάλλον.

Μερικά σημαντικά πρότυπα τα οποία χρησιμοποιήθηκαν για την μοντελοποίηση πληθυσμών είναι τα Petri Nets. Οι άλγεβρες διεργασιών είναι επίσης ένα πρότυπο το οποίο έχει προταθεί για τη μοντελοποίηση και ανάλυση συστημάτων όπου ο υπολογισμός επιτυγχάνεται μέσω του παραλληλισμού και της ανταλλαγής μηνυμάτων. Στο επόμενο υποκεφάλαιο γίνεται αναφορά στις άλγεβρες διεργασιών και τον τρόπο που αυτές μπορούν να χρησιμοποιηθούν στην μοντελοποίηση πληθυσμιακών συστημάτων.

2.2 Άλγεβρες Διεργασιών

Οι άλγεβρες διεργασιών περιλαμβάνουν μια γλώσσα προδιαγραφής (μοντελοποίησης συστημάτων) που αποτελείται από ένα σύνολο τελεστών για σύνθεση συστημάτων σε μεγαλύτερα συστήματα. Επίσης έχουν μια αυστηρά διατυπωμένη σημασιολογία που απεικονίζει τις δυνατές λειτουργίες και αλληλεπιδράσεις ενός συστήματος διατυπωμένου στη γλώσσα προδιαγραφής.
Ασχολούνταν με το λειτουργικό κομμάτι των συστημάτων, όπως την εξωτερική τους συμπεριφορά, τον έλεγχο ροής, τον συγχρονισμό και την επικοινωνία μεταξύ των συστημάτων.

Οι άλγεβρες διεργασιών είναι ένας μαθηματικοποιημένος τρόπος να εκφράζουμε την συμπεριφορά παράλληλων συστημάτων ή σύμφωνα με τους Bergstra & Klop οι οποίοι έχουν επινοήσει τον όρο *άλγεβρα διεργασιών* είναι ένα πρότυπο το οποίο έχει την ικανότητα να εξάγει κάποια συμπεράσματα για ένα σύνθετο σύστημα από τις ιδιότητες των επιμέρους συνιστωσών του. Έτσι λοιπόν για να ορίσουμε μια άλγεβρα διεργασιών χρειαζόμαστε την σύνταξη με βάση την οποία θα κτίζουμε το σύστημα και τους κανόνες σημασιολογίας που ορίζουν την λειτουργία κάθε τύπου διεργασίας. Η χρήση τους για την μοντελοποίηση πληθυσμιακών συστημάτων ήταν αναμενόμενη καθώς τα πληθυσμιακά συστήματα αποτελούν παράδειγμα παράλληλων συστημάτων.

Κορυφαία παραδείγματα αλγεβρών διεργασιών αποτελούν οι CSP, CCS, ACP. Οι διεργασίες αυτές για να περιγράψουν και να αναλύσουν συστήματα που

επικοινωνούν μεταξύ τους και λειτουργούν παράλληλα. Στηρίζονται δηλαδή στην ιδέα ότι τα δύο πιο σημαντικά δομικά στοιχεία για την κατανόηση των πολύπλοκων δυναμικών συστημάτων είναι ο παραλληλισμός και η μεταξύ τους επικοινωνία. Και για να το πετύχουν αυτό,βασίστηκαν πάνω στην ιδέα ότι, μπορεί κάποιος να επαληθεύσει ολόκληρο το σύστημα με το να αναλύσει και να περιγράψει τα δομικά του στοιχεία.

Καθώς η ποικιλία των αλγεβρών διαδικασιών είναι μεγάλη, υπάρχουν κάποια κοινά χαρακτηριστικά για όλες τις άλγεβρες διεργασιών όπως είναι οι κανόνες σημασιολογίας και η επικοινωνία μεταξύ των ανεξάρτητων διεργασιών(ανταλλαγή μηνυμάτων). Σημαντικός σταθμός σε σχέση με την εργασία μου είναι η μελέτη της άλγεβρας διεργασιών CCS (Calculus of Communicating Systems),που επινοήθηκε από τον Milner.Επεκτάσεις που ήδη έχουν πραγματοποιηθεί γύρω από την CCS, είναι η WSCCS[5] και η TPCCS[7], οι οποίες εντάσσουν τις έννοιες της πιθανότητας, του χρόνου και της προτεραιότητας. Στην CCS η επιλογή μεταξύ διαφορετικών μεταβάσεων γίνεται μη-ντετερμινιστικά . Στο επόμενο υποκεφάλαιο θα γίνει πιο αναλυτική επεξήγηση για την άλγεβρα διεργασιών CCS.

2.3 CCS- Calculus Of Communicating Systems

Το κύριο πρόσωπο στην ιστορία των άλγεβρων διεργασιών είναι χωρίς αμφιβολία ο Robin Milner, ο οποίος γεννήθηκε το 1934 και δημιούργησε την Άλγεβρα Διεργασιών CCS κατά τα τέλη του 1970 με αρχές του 1980. Η CCS (Calculus of Communicating Systems) παρουσιάζει ως μέθοδο επικοινωνίας μεταξύ των διεργασιών του συστήματος τον δυαδικό συγχρονισμό και οι διεργασίες της κτίζονται από ένα σύνολο ατομικών ενεργειών με τη χρήση τελεστών σύνθεσης διεργασιών.

Οι ενέργειές της CCS μοντελοποιούν αδιαίρετες επικοινωνίες μεταξύ ακριβώς δύο συμμετεχόντων. Η τυπική γλώσσα περιλαμβάνει βασικά στοιχεία για την περιγραφή της παράλληλης σύνθεσης, της επιλογής μεταξύ ενεργειών και του περιορισμού

εμβέλειας. Η CCS είναι χρήσιμη για την εκτίμηση της ορθότητας ιδιοτήτων ενός συστήματος όπως τα αδιέξοδα (deadlocks) και τα ζωντανά αδιέξοδα (livelocks).

Η άλγεβρα διεργασιών CCS αποτελείται από ένα σύνολο τελεστών για την σύνθεση μικρών υποσυστημάτων σε μεγαλύτερα συστήματα, καθώς επίσης και την σημασιολογία μέσω της οποίας δίνονται οι αλληλεπιδράσεις του συστήματος διατυπωμένες με την σωστή σύνταξη της αντίστοιχης γλώσσας.

Η γλώσσα CCS αποτελείται από διεργασίες οι οποίες κτίζονται, με την βοήθεια τελεστών, από ένα σύνολο ατομικών ενεργειών(actions).

Μια ενέργεια στην CCS μπορεί να είναι:

- Ενέργεια εισόδου: α
- Ενέργεια εξόδου: $\bar{\alpha}$
- Εσωτερική Ενέργεια: τ

Όπως και σε κάθε άλγεβρα διεργασιών η εσωτερική ενέργεια τ, επιτυγχάνεται με τον συνδυασμό δύο συμπληρωματικών ενεργειών (α, $\bar{\alpha}$) , μιας ενέργειας εισόδου και μιας ενέργειας εξόδου. Με την εκτέλεση εσωτερικών ενεργειών, διάφορες διεργασίες σε ένα σύστημα ή και ολόκληρα συστήματα, έχουν την ικανότητα να επικοινωνούν μεταξύ τους μέσω καναλιών. Η συμπεριφορά ενός συστήματος είναι το αποτέλεσμα των ενεργειών που εκτελούν τα επιμέρους κομμάτια του είτε αυτόνομα, είτε ως αποτέλεσμα της μεταξύ τους επικοινωνίας.

Εάν θεωρήσουμε ότι το Λ είναι το σύνολο όλων των καναλιών της γλώσσας τότε, οι τελεστές που σχηματίζουν της γλώσσας CCS είναι οι πιο κάτω:

- 0: τερματισμός/αδιέξοδο
- α.E: διαδοχή- εκτέλεσε την ενέργεια α και συνέχισε σύμφωνα με την E
- E+F: επιλογή-συνέχισε σύμφωνα με την διεργασία E ή F
- E | F: ταυτοχρονισμός-εκτέλεσε παράλληλα τις διεργασίες E και F
- E \ L: περιορισμός-το υποσύνολο L του Λ περιέχει ενέργειες οι οποίες είναι δεσμευμένες αποκλειστικά για την διεργασία E.
- E[F]:μετονομασία- Οι ενέργειες της διεργασίας E μετονομάζονται σύμφωνα με την f σύναρτηση. Π.χ. f(a)=b.

Σύνταξη της CCS

Εκτός από το σύνολο των ενεργειών και των διεργασιών που υπάρχουν στη γλώσσα, υπάρχει και ένα σύνολο κανόνων που ορίζουν τη λειτουργική σημασιολογία κάθε διεργασίας . Οι κανόνες αυτοί ορίζουν τις δυνατές μεταβάσεις κάθε διεργασίας όπου μια μετάβαση έχει την πιο κάτω μορφή:

$E \xrightarrow{a} F$

Η πιο πάνω πρόταση ερμηνεύεται ως εξής: Η διεργασία E μπορεί να εκτελέσει την ενέργεια α και εξελίσσεται στην διεργασία F.

Οι εκφράσεις της γλώσσας ερμηνεύονται σαν συστήματα μεταβάσεων με ετικέτες(labeled transition systems).

$$\frac{συνθήκες\ μεταβάσεων}{συμπέρασμα}\ επιμέρους\ συνθήκη$$

Πιο κάτω ορίζονται οι κανόνες για κάθε ένα από τους τελεστές της γλώσσας:

- Τελεστής Διαδοχής: Σύμφωνα με τον τελεστή διαδοχής εάν είμαστε σε μια κατάσταση α.E, όπου α ανήκει στο σύνολο των ενεργειών και E στο σύνολο των διεργασιών, τότε μπορεί να εκτελεστεί η ενέργεια α και να φτάσουμε στην διεργασία E.

$$\text{Act}\quad \frac{-}{\alpha.E \xrightarrow{\alpha} E}$$

- Τελεστής Επιλογής: Σύμφωνα με τον **Sum1** κανόνα, εάν ισχύει ότι από την διεργασία Ε μέσω της ενέργειας α μπορούμε να φτάσουμε στην διεργασία Ε΄, τότε στην περίπτωση επιλογής μεταξύ της Ε και F διεργασίας, επιλέγεται η Ε και με την εκτέλεση της ενέργειας α μετατρέπεται σε Ε΄.

$$\text{Sum1} \quad \frac{E \xrightarrow{\alpha} E'}{E + F \xrightarrow{\alpha} E'}$$

Αντίστοιχα στον κανόνα **Sum2**, εάν ισχύει ότι από την διεργασία F μέσω της ενέργειας α μπορούμε να φτάσουμε στην διεργασία F΄, τότε στην περίπτωση επιλογής μεταξύ της Ε και F διεργασίας, επιλέγεται η F και με την εκτέλεση της ενέργειας α μετατρέπεται σε F΄.

$$\text{Sum2} \quad \frac{F \xrightarrow{\alpha} F'}{E + F \xrightarrow{\alpha} F'}$$

- Τελεστής Ταυτοχρονισμού: Σύμφωνα με τον κανόνα **Com1**, εάν ισχύει ότι από την διεργασία Ε μέσω της ενέργειας α μπορούμε να φτάσουμε στην διεργασία Ε΄, τότε στην περίπτωση ταυτόχρονης εκτέλεσης των διεργασιών Ε και F, με την εκτέλεση της ενέργειας α μπορούμε να φθάσουμε στην κατάστασης ταυτοχρονισμού της διεργασίας Ε’ με την διεργασία F (Ε΄| F).

$$\text{Com1} \quad \frac{E \xrightarrow{\alpha} E'}{E \mid F \xrightarrow{\alpha} E' \mid F}$$

Αντίστοιχα και στον κανόνα **Com2**, εάν ισχύει ότι από την διεργασία F μέσω της ενέργειας α μπορούμε να φτάσουμε στην διεργασία F΄, τότε στην περίπτωση ταυτόχρονης εκτέλεσης των διεργασιών Ε και F, με την εκτέλεση της ενέργειας α μπορούμε να φθάσουμε στην κατάστασης ταυτοχρονισμού της διεργασίας Ε με την διεργασία F’(Ε| F΄).

$$\text{Com}_2 \quad \frac{F \xrightarrow{a} F'}{E \mid F \xrightarrow{a} E \mid F'}$$

- Στον κανόνα **Com3** πραγματοποιείται η επικοινωνία μεταξύ των διεργασιών μέσω της εσωτερικής ενέργειας. Εάν ισχύει ότι από την διεργασία E μέσω της ενέργειας α μπορούμε να φτάσουμε στην διεργασία E′ και από την διεργασία F μέσω της ενέργειας $\bar{\alpha}$ μπορούμε να φτάσουμε στην διεργασία F′ τότε λόγω της ταυτόχρονης εκτέλεσης των 2 ενεργειών εισόδου και εξόδου, εκτελείται η εσωτερική ενέργεια τ και φθάνουμε στη κατάσταση E′ | F′.

$$\text{Com3} \quad \frac{E \xrightarrow{\alpha} E' \ , \ F \xrightarrow{\bar{\alpha}} F'}{E \mid F \xrightarrow{\tau} E' \mid F'}$$

- Τελεστής Περιορισμού: Εάν ισχύει ότι από την διεργασία E μέσω της ενέργειας α μπορούμε να φτάσουμε στην διεργασία E′ τότε με την εκτέλεση της ενέργειας α φθάνει στην κατάσταση E′ δεδομένου του περιορισμού η διεργασία E να μην εκτελέσει καμιά ενέργεια που ανήκει στα κανάλια του L.

$$\text{Res} \quad \frac{E \xrightarrow{\alpha} E'}{E \setminus L \xrightarrow{\alpha} E' \setminus L} \quad \alpha, \bar{\alpha} \text{ δεν ανήκουν στο } L$$

- Τελεστής μετονομασίας: Εάν ισχύει ότι από την διεργασία E μέσω της ενέργειας α μπορούμε να φτάσουμε στην διεργασία E′ τότε μπορούμε να μετονομάσουμε τα κανάλια του E με βάση την συνάρτηση f.

$$\text{Rel} \quad \frac{E \xrightarrow{\alpha} E'}{E[f] \xrightarrow{f(\alpha)} E'[f]}$$

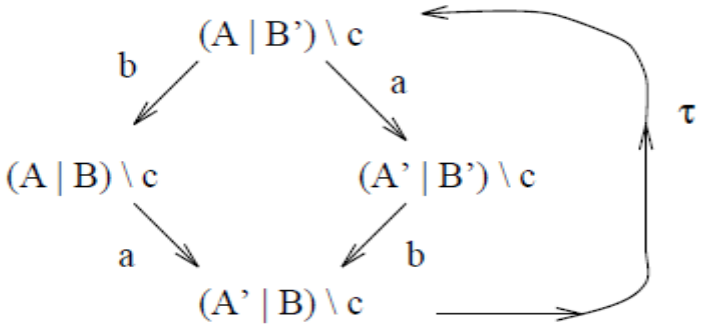
Επίσης, η συμπεριφορά ενός συστήματος μπορεί να μεταφραστεί σε γράφο με βάρη, ο οποίος ονομάζεται σύστημα μεταβάσεων. Οι κορυφές του γράφου αντιστοιχούν στις ήδη εκτελεσμένες ενέργειες.

Παράδειγμα μοντελοποίησης ενός συστήματος που αποτελείται από τις διεργασίες A, A΄,B και B΄ , οι οποίες ορίζονται ως εξής:

$A \overset{def}{=} \alpha.A'$ $\quad$ $B \overset{def}{=} c.B'$

$A' \overset{def}{=} \overline{c}.A$ $\quad$ $B' \overset{def}{=} b.B$

Εάν έχουμε το σύστημα (A|B)\c τότε μπορούμε να παρατηρήσουμε την συμπεριφορά των ενεργειών μετατρέποντας τα στο πιο κάτω γράφο:



Κεφάλαιο 3

Η Άλγεβρα Διεργασίας PALPS

3.1 PALPS

Επέλεξα να ασχοληθώ με την άλγεβρα διεργασιών PALPS η οποία μας παρέχει την δυνατότητα να εξάγουμε συμπεράσματα για την κατάσταση του χώρου στον οποίο συμβιώνουν συγκεκριμένοι πληθυσμοί, μετά τις αλληλεπιδράσεις των ατόμων που υπάρχουν στο πληθυσμό στην προσπάθεια τους να επιβιώσουν.

Η γλώσσα PALPS η οποία παρουσιάζεται σε αυτό το υποκεφάλαιο είναι από τα αρχικά Process Algebra With Location for Population Systems.Η γλώσσα αυτή χρησιμοποιεί κάποιους από τους τελεστές της CCS του Milner, προσαρμόζοντας σε αυτούς τους κανόνες την έννοια του χώρου. Η έννοια του χώρου αποτελεί ένα σημαντικό σημείο στην κατανόηση του τρόπου που ένας πληθυσμός επιβιώνει.

Η PALPS μας επιτρέπει, επίσης, να παράγουμε spatially-explicit ατομικά μοντέλα και να αναλύσουμε τη συμπεριφορά τους. Η διεργασία έχει δύο επίπεδα: στο πρώτο επίπεδο, μπορούμε να ορίσουμε τη συμπεριφορά ενός ατόμου που επιβιώνει στο πληθυσμό, καθώς, στο δεύτερο επίπεδο, μπορούμε να καθορίζουμε ένα σύστημα από πληθυσμούς διάφορων ειδών με την πληροφορία της αρχικής τους τοποθεσία στο περιβάλλον στο οποίο ζουν. Αυτά τα είδη στο κύκλο ζωής τους μπορούν να αλλάζουν την τοποθεσία τους, αν το θέλουν, και αλληλεπιδρούν μεταξύ τους με διάφορους τρόπους όπως π.χ. να κατατρώγονται μεταξύ τους.

Η άλγεβρα διεργασιών PALPS είναι ειδικά σχεδιασμένη να ταιριάζει με τη γνωστή προσέγγιση προσωπικότητας των πρότυπων πληθυσμών αφού καθιστούν δυνατό να περιγραφεί η εξέλιξη κάθε ατόμου του πληθυσμού σαν διαδικασία και συνεπώς, να συνθέσουμε ένα σύνολο προσωπικοτήτων (όπως επίσης και το περιβάλλον τους) σε ένα ολοκληρωμένο οικολογικό σύστημα. Χαρακτηριστικά όπως ο χρόνος, η πιθανότητα και η στοχαστική συμπεριφορά μπορούν να αξιοποιηθούν για να παραχθούν πιο ακριβή πρότυπα, τα οποία αν συνδεθούν με εργαλεία ανάλυσης, μπορούν να χρησιμοποιηθούν για την ανάλυση και πρόβλεψη της συμπεριφοράς τους. Σκοπός της δημιουργίας της PALPS είναι η εισαγωγή ενός πλαισίου μιας άλγεβρας διεργασιών που να είναι εξοπλισμένη με την έννοια της τοποθεσίας για να καταστήσει πιθανό τα πρότυπο της στα οικολογικά συστήματα. Συγκεκριμένα, η άλγεβρα διεργασιών θα είναι συγκεκριμένη σε τομέα και θα συσχετίζει άτομα με πληροφορίες βάση της θέσης τους. Έτσι θα επιτρέπει την εξερεύνηση συμπεριφοράς ενός ατόμου ή ολόκληρων πληθυσμών εξαρτώμενη στη τοποθεσία στην οποία βρίσκονται. Η καινοτομία του προτύπου αυτού είναι ότι συνδέει τις πληροφορίες τοποθεσίας με τη συγκεκριμένες συμπεριφορές όπως η αναπαραγωγή, διαρπαγή και η μετακίνηση ενός ατόμου ή και πολλών πληθυσμών σε ένα σύστημα. Στη συνέχεια θα δούμε τη σύνταξη της γλώσσας μαζί με τους κανόνες σημασιολογίας που την διέπουν αφού πρώτα γίνει αναφορά σε ένα σύνολο ενεργειών της.

3.2 PALPS –Σύνταξη και Σημασιολογία της γλώσσας

Το σύστημα υλοποιεί ένα υποσύνολο της PALPS. Πιο κάτω θα παρουσιαστούν με λεπτομέρεια οι ενέργειες που έχω υλοποιήσει τόσο και η σύνταξη της γλώσσας και η σημασιολογία της.

Η γλώσσα αποτελείται από διεργασίες οι οποίες κτίζονται, με την βοήθεια τελεστών, από ενός συνόλου ατομικών ενεργειών (actions).
Μια ενέργεια μπορεί να είναι:

- Ενέργεια εισόδου(input): a

- Ενέργεια εξόδου(output): $\bar{\alpha}$
- Εσωτερική ενέργεια: τ
- Ενέργεια μετακίνησης: move
- Ενέργεια αναπαραγωγής: reproduce

Ενέργεια Μετακίνησης: Κατά την ενέργεια μετακίνησης κάθε άτομο μπορεί να αφήνει μια συγκεκριμένη τοποθεσία του χώρου και να φτάνει σε μια γειτονική της.

Ενέργεια Αναπαραγωγής: Κάθε άτομο έχει την ικανότητα να αναπαράγει συγκεκριμένο αριθμό απογόνων.

Όπως και σε κάθε άλγεβρα διεργασιών η εσωτερική ενέργεια επιτυγχάνεται με τον συνδυασμό δυο συμπληρωματικών ενεργειών $(\alpha, \bar{\alpha})$,μιας ενέργειας εισόδου και μιας ενέργειας εξόδου. Με την εκτέλεση εσωτερικών ενεργειών, διάφορες διεργασίες σε ένα σύστημα ή και ολόκληρα συστήματα έχουν την ικανότητα να επικοινωνούν μεταξύ τους.

Για την ανάπτυξη του συστήματος που υλοποίησα χρειάστηκε να ορίσω την σχέση Γειτνίασης(Neighbors) και την έννοια του χώρου που θεωρώ ότι υπάρχουν στη γλώσσα για την ενέργεια της μετακίνησης.Έτσι, λοιπόν, θεωρώ ότι ο χώρος που βρίσκεται ένας πληθυσμός σαν ένα πίνακα $n \times n$, τότε κάθε θέση του πίνακα ορίζεται σαν μια συγκεκριμένη τοποθεσία που την συμβολίζω με l(i,j) , όπου i και j καθορίζουν τις συντεταγμένες της τοποθεσίας, την γραμμή και την στήλη του πίνακα. Πιο κάτω φαίνεται ένας πίνακας 4×4 που αντιπροσωπεύει ένα «κόσμο» διαχωρισμένο σε 4 επιμέρους τοποθεσίες τα l1,l2,l3 και l4.

l1	l2
l3	l4

Συνεπώς αφού ο κόσμος στον οποίο είναι κατανεμημένες οι οντότητες του πληθυσμού είναι διαχωρισμένος, απαιτείται η ύπαρξη της ενέργειας move, με την οποία μια οντότητα μπορεί να κινηθεί από μια τοποθεσία σε μια άλλη. Στο σημείο αυτό φαίνεται η αναγκαιότητα της ύπαρξης μιας σχέσης γειτνίασης Neighbors την οποία υλοποίησα για το σύστημα μου. Η σχέση γειτνίασης Neighbors θα καθορίσει σε κάθε οντότητα που βρίσκεται στην τοποθεσία l προς μια κατεύθυνση μπορεί να κινηθεί.

Πιο κάτω ορίζεται η σχέση γειτνίασης Neighbors σύμφωνα με την οποία, κάθε τοποθεσία l(i,j) είναι γειτονική με τις εξής τοποθεσίες:

- l(i-1,j), την τοποθεσία πάνω από την l(i,j)
- l(i+1,j) , την τοποθεσία κάτω από την l(i,j)
- l(i,j-1) , την τοποθεσία αριστερά από την l(i,j)
- l(i,j+1) , την τοποθεσία δεξιά από την l(i,j)

Επομένως κάθε οντότητα μπορεί να μετακινηθεί από την l τοποθεσία στην l′ με την εκτέλεση της ενέργειας move αν και μόνο αν (l,l′)∈ Neighbors.

Σύνταξη της PALPS

Το υποσύνολο της PALPS με το οποίο έχω ασχοληθεί χωρίζεται σε δύο επίπεδα. Στο $1^{ο}$ επίπεδο η σύνταξη αφορά ένα μεμονωμένο άτομο του συνόλου των ατόμων που συνυπάρχουν στο κόσμο, ενώ το $2^{ο}$ επίπεδο αφορά ολόκληρους πληθυσμούς.

Αρχικά θα δοθεί η σύνταξη της γλώσσας σε ατομικό επίπεδο και στη συνέχεια σε επίπεδο πληθυσμών. Οι τελεστές οι οποίοι ορίζουν την σύνταξη της γλώσσας παρουσιάζονται πιο κάτω:

- 0:τερματισμός
- +:επιλογή
- . : διαδοχή
- **|** : ταυτοχρονισμός

Έστς ότι το σύνολο Actions αντιπροσωπεύει τις ενέργεις που μπορεί να εκτελέσει μια διεργασία και ορίζεται: Actions={α,$\overline{a}$, move, prey, rep}.
Ακολουθεί η γραμματική που δίνει την σύνταξη των διεργασιών της γλώσσας σε επίπεδο ενός ατόμου. Όπου n∈ Actions Το P απεικονίζει μια διεργασία, μια οντότητα ενός πληθυσμού.
P ::= 0|n.P|P1+P2

- 0: Διεργασία η οποία έχει τερματίσει
- n.P: Διεργασία που μπορεί να εκτελέσει την ενέργεια n και μπορεί να πάρει τις τιμές (α,$\overline{a}$, move, rep)και να συμπεριφερθεί όπως η διεργασία P.
- P1+P2: Διεργασία η οποία δίνει επιλογές ανάμεσα στις διεργασίες P1 και P2.

Σε επίπεδο πληθυσμού άτομα και ολόκληρα συστήματα πληθυσμών εκτελούν τις ενέργειες τους παράλληλα με την βοήθεια του τελεστή ταυτοχρονισμού. Η σύνταξη της γλώσσας δίνεται από την πιο κάτω γραμματική:

S::=0|P:[s,L]|R:[s]|S1|S2|S\L|[S]

Τα πληθυσμιακά συστήματα είναι κτισμένα με βάση την τοποθεσία των ατόμων:

- P:[s,L] συμβολίζει ένα άτομο, τι είδος είναι(s) και σε ποια τοποθεσία βρίσκεται (L). Το άτομο αυτό εκτελεί τις ενέργειες του παράλληλα με τις ενέργειες που εκτελούν άλλα άτομα που συνυπάρχουν στο ίδιο κόσμο.
- R:[s] ορίζει τι είδος είναι και δημιουργεί ένα νέο άτομο του είδους αυτού του ατόμου.(R=**!a.P**)
- S1 και S2: Αποτελούν συστήματα, συνθέσεις δηλαδή πολλών ατόμων μαζί τα οποία αλληλεπιδρούν μεταξύ τους κατά τη συνύπαρξη τους στον κόσμο, εκτελώντας ταυτόχρονα διαφορετικές ενέργειες ή επικοινονώντας μεταξύ τους.

Σημασιολογία της PALPS

Εκτός από το σύνολο των ενεργειών και των διεργασιών που υπάρχουν στην γλώσσα, υπάρχει και ένα σύνολο μεταβάσεων με ετικέτες, με το οποίο μια διεργασία εκτελώντας μια ενέργεια που αναπαρίσταται σαν ετικέτα, μπορεί να εξελιχθεί σε άλλη διεργασία. Το σύνολο των μεταβάσεων αυτών αποτελούν το σύνολο των κανόνων που καθορίζει την σημασιολογία της γλώσσας και έχουν τη μορφή:

$$E \xrightarrow{a} E'$$

και η οποία ερμηνεύεται ως εξής: εάν η διεργασία Ε εκτελέσει την ενέργεια α τότε μπορεί να εξελιχθεί στην διεργασία Ε΄.

Οι κανόνες για κάθε ένα από τους τελεστές της γλώσσας σε **ατομικό επίπεδο** ορίζονται πιο κάτω:

Τελεστής Διαδοχής:

$$\frac{-}{\alpha.E \xrightarrow{\alpha} E}$$

Τελεστής Επιλογής:

1) $\frac{E \xrightarrow{\alpha} E'}{E+F \xrightarrow{\alpha} E'}$

2) $\frac{F \xrightarrow{\alpha} F'}{E+F \xrightarrow{\alpha} F'}$

Επεξήγηση των κανόνων σημασιολογίας της γλώσσας σε ατομικό επίπεδο:

- Τελεστής Διαδοχής: Εάν είμαστε σε μια κατάσταση α.Ε, όπου α ανήκει στο σύνολο των ενεργειών και Ε στο σύνολο των διεργασιών, τότε μπορεί να εκτελεστεί η ενέργεια α και να φτάσουμε στην διεργασία Ε΄.

- Τελεστής Επιλογής: Σύμφωνα με τον πρώτο κανόνα επιλογής, εάν ισχύει ότι από την διεργασία Ε μέσω της ενέργειας α μπορούμε να φτάσουμε στην διεργασία Ε΄, τότε στην περίπτωση επιλογής μεταξύ της Ε και F διεργασίας, επιλέγεται η Ε και με την εκτέλεση της ενέργειας α μετατρέπεται σε Ε΄.

Αντίστοιχα και με το δεύτερο κανόνα επιλογής, εάν ισχύει ότι από την διεργασία F μέσω της ενέργειας α μπορούμε να φτάσουμε στην διεργασία F′, τότε στην περίπτωση επιλογής μεταξύ της E και F διεργασίας, επιλέγεται η F και με την εκτέλεση της ενέργειας α μετατρέπεται σε F′.

Για την μελέτη ενός ή και περισσότερων ατόμων ταυτόχρονα απαιτείται η σύνθεση των ατόμων των πληθυσμών με την βοήθεια του τελεστή ταυτοχρονισμού . Πιο κάτω θα παρουσιάσω πως μπορούμε να αναπαραστήσουμε πολλά άτομα διαφορετικά ή ίδιου πληθυσμού να αλληλεπιδράσουν μεταξύ τους εκτελώντας διαφορετικές ενέργειες. Η άλγεβρα διεργασίας PALPS χρησιμοποιεί κανόνες που έχουν σχέση με την έννοια του χώρου. Η έννοια του χώρου είναι ένα σημαντικό σημείο στην κατανόηση πως ένας πληθυσμός επιβιώνει. Έτσι πιο κάτω φαίνεται ένα άτομο που απεικονίζεται από την διεργασία (P), την τοποθεσία στην οποία βρίσκεται (l) και το είδος του πληθυσμού στο οποίο ανήκει (s):

P:[l, s]

Οι πιο κάτω κανόνες είναι οι κανόνες που χρειάστηκε να υλοποιήσω για την ανάπτυξη του συστήματος.

Table 2: **Transition rules for systems**

(Loc) $$\frac{P \xrightarrow{e,\theta} P'}{P{:}[\![\mathbf{s},\ell]\!] \xrightarrow{e@\ell,\theta@\ell} P'{:}[\![\mathbf{s},\ell]\!]}$$

(Move) $$\frac{P \xrightarrow{e,move} P', (\ell,\ell') \in \mathbf{Nb}}{P{:}[\![\mathbf{s},\ell]\!] \xrightarrow{e,\tau} P'{:}[\![\mathbf{s},\ell']\!]}$$

(Par1) $$\frac{S_1 \xrightarrow{e,\beta} S_1'}{S_1|S_2 \xrightarrow{e,\beta} S_1'|S_2}$$

(Par2) $$\frac{S_1 \xrightarrow{e_1,a@\ell} S_1', S_2 \xrightarrow{e_2,\overline{a}@\ell} S_2'}{S_1|S_2 \xrightarrow{e_1\wedge e_2,\tau} S_1'|S_2'}$$

(RepA) $$\frac{\ell \in \mathbf{Loc}}{(!a.P){:}[\![\mathbf{s}]\!] \xrightarrow{true,a@\ell} P{:}[\![\mathbf{s},\ell]\!]|(!a.P){:}[\![\mathbf{s}]\!]}$$

The semantics of PALPS is defined in terms of a structural operational semantics, which is given on the above table. The rules in Table 2 describe the behavior of systems.

Ο κανόνας (Loc) ενσωματώνει πληροφορίες για την τοποθεσία στις ενέργειες των ατόμων που βρίσκονται εκεί και έτσι όταν οι διεργασίες εκτελέσουν την ενέργεια στη τοποθεσία αυτή τότε επηρεάζονται ανάλογα.

Ο κανόνας (move) εξηγεί ότι κάθε άτομο μπορεί μη ντετερμινιστικά να μετακινηθεί σε μια γειτονική τοποθεσία(πάνω, κάτω, δεξιά, αριστερά) από τη θέση στην οποία βρίσκεται.

Οι κανόνες (Par1)και (Par2) προσομοιώνουν την σημασιολογία της παράλληλης σύνθεσης. Για παράδειγμα στο κανόνα (Par1) όταν ένα άτομο εκτελέσει κάποια ενέργεια τότε μπορεί να επηρεάσει μόνο αυτό και τα άλλα άτομα να μείνουν ανεπηρέαστα.
Στο κανόνα (Par2) όταν δύο άτομα ή και περισσότερα βρίσκονται στην ίδια τοποθεσία και εκτελέσουν μια ενεργεία εισόδου και μια ενέργεια εξόδου τότε μπορούν να επηρεαστούν αυτά τα άτομα με την εφαρμογή της εσωτερικής ενέργειας.

Ο κανόνας (rep) δίνει τη σημασιολογία της δημιουργίας ακόμη ένα ατόμου. Επίσης σε αυτό το κανόνα μπορούμε να παρατηρήσουμε πως το άτομο που παράγει νέα ανεξάρτητα άτομα νέου είδους μπορεί να παραμείνει στο περιβάλλον για επιπλέον χρήση. Η αρχική τοποθεσία των παραγόμενων ατόμων είναι αυτή που το άτομο εκτέλεσε την ενέργεια της αναπαραγωγής. ((!α.P):[s])::= rep.

3.3 Παράδειγμα Μοντελοποίησης Πληθυσμιακού Συστήματος

Με την χρήση της άλγεβρας διεργασίας PALPS μπορούν να μοντελοποιηθούν συστήματα που αποτελούνται από ένα ή περισσότερους πληθυσμούς των οποίων τα άτομα μπορούν να εκτελούν τις ενέργειες της μετακίνησης, της αναπαραγωγής, του παραλληλισμού, την επιλογή ανάμεσα σε 2 διεργασίες, την αρπαγή(prey) την οποία δεν έχω υλοποιήσει στο σύστημα μου. Πριν τη μοντελοποίηση απαιτείται η επιλογή των ενεργειών που θα εκτελούν τα άτομα καθώς επίσης και η σειρά με την οποία θα τις εκτελούν. Στο παράδειγμα πιο κάτω θα γίνεται επαναλαμβανόμενη

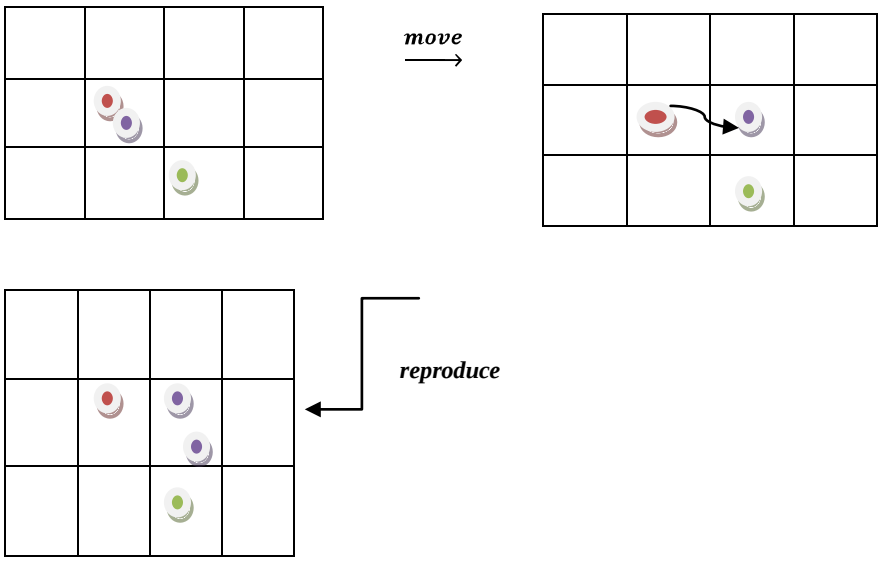
εκτέλεση ενεργειών μιας διεργασίας μέχρι να παρατηρήσουμε και να μελετήσουμε τον τρόπο που συμπεριφέρονται τα άτομα που συνυπάρχουν στον κόσμο.

Στο παράδειγμα που θα ακολουθήσω ένα τέτοιο σύστημα, συγκεκριμένα τα άτομα του πληθυσμού θα μπορούν να αναπαράγονται και να μετακινούνται. Οι δύο ενέργειες εκτελούνται συνεχώς εναλλάξ ξεκινώντας από την ενέργεια της μετακίνησης.

Σε περίπτωση που ένα άτομο εκτελέσει την ενέργεια της αναπαραγωγής τότε παράγει ένα νέο άτομο. Επίσης τα άτομα που απομένουν στην θέση μετά την εκτέλεση μιας αναπαραγωγής είναι το νέο άτομο μαζί με αυτό που το δημιούργησε.

Κατά την διαδικασία της μετακίνησης ισχύει ότι κάθε άτομο έχει την ικανότητα να μετακινείται προς οποιαδήποτε τοποθεσία βρίσκεται πάνω, κάτω, δεξιά ή αριστερά της τοποθεσίας που βρίσκεται το άτομο.

Πιο κάτω απεικονίζεται το σχήμα όπου επιτυγχάνεται με την σειρά μιας ενέργειας μετακίνησης και ακολουθεί μια αναπαραγωγής.



Στην πρώτη εκτέλεση της μετακίνησης στο δεύτερο πίνακα βλέπουμε ένα από τα άτομα που βρίσκονται στη τοποθεσία 6 να μετακινείται και να πηγαίνει σε μια γειτονική τοποθεσία στη 7.

Στη εκτέλεση της αναπαραγωγής βλέπουμε το άτομο το οποίο μετακινήθηκε και βρίσκετε στη θέση 7 να διπλασιάζεται. Στη τοποθεσία αυτή βρίσκεται το νέο άτομο μαζί με αυτό που το δημιούργησε.

Το άτομο (P) που υπάρχει στο αρχικό κόσμο μπορεί να εκτελέσει την ενέργεια της μετακίνησης και μετά την ενέργεια της αναπαραγωγής:

P=move.reproduce.P (αναδρομική διεργασία)
Το σύστημα που μελετήσαμε μπορεί να δημιουργηθεί από την σύνθεση των ατόμων με την βοήθεια του ταυτοχρονισμού όπως παρουσιάζεται πιο κάτω:

System= P:[l_6,s]| P1:[l_6,s1]| P2:[l_{11},s2]
P=move.reproduce.P
P1=move.0
P2=rep.a.0
s=move.0
s1=move.rep.0
s2=rep.move.move.rep.rep.move.0

Η διεργασία P είναι αναδρομική. Εκτελεί την ενέργεια της μετακίνησης και μετά την ενέργεια της αναπαραγωγής. Η διεργασία P1 εκτελεί μια φορά την ενέργεια της μετακίνησης και μετά «πεθαίνει». Η διεργασία P2 εκτελεί μια φορά την ενέργεια της αναπαραγωγής, μετά το κανάλι 'α' και μετά «πεθαίνει». Το άτομο που αναπαράγει είναι του είδους s2. Στο πάνω παράδειγμα είδαμε την προσομοίωση του ατόμου P.

Κεφάλαιο 4

Ανάπτυξη λογισμικού

4.1 Ανάλυση απαιτήσεων και προδιαγραφών του λογισμικού

Ακολουθούν κάποια χαρακτηριστικά που πρέπει να έχει το λογισμικό που θα αναπτυχθεί.

1. Το λογισμικό χρειάζεται να αναγνωρίζει την σύνταξη της άλγεβρας διεργασιών PALPS.

2. Το πληθυσμιακό σύστημα θα δίνεται στο λογισμικό υπό μορφή κειμένου, μέσα σε ένα αρχείο κειμένου ή μπορεί να το δίνει και ο ίδιος ο χρηστής σε ένα πεδίο κειμένου το οποίο θα υπάρχει στη φόρμα.

3. Το λογισμικό πρέπει να παρουσιάζει όλες τις πιθανές αντιδράσεις του πληθυσμιακού συστήματος, και να δίνει τη δυνατότητα στον χρήστη να προσομοιώσει μια, δείχνοντας του τις αλλαγές που προκαλεί αυτή η αντίδραση στο σύστημα.

4. Το λογισμικό πρέπει να παρουσιάζει όλες τις πιθανές αντιδράσεις του πληθυσμιακού συστήματος, και να δίνει τη δυνατότητα στον χρήστη να προσομοιώσει μια σειρά από τυχαίες αντιδράσεις πατώντας ένα κουμπί , δείχνοντας του τις αλλαγές που προκαλεί κάθε ενέργεια στο σύστημα. Η διαδικασία

πρέπει να τερματίζει δίνοντας ο χρήστης ένα αριθμό επαναλήψεων ή όταν το σύστημα έχει φθάσει σε αδιέξοδο.

5. Το λογισμικό πρέπει να παρουσιάζει τη δυσδιάστατη αναπαράσταση του πληθυσμιακού συστήματος και τις ενέργειες που μπορούν να εκτελεστούν από αυτό.

7. Το λογισμικό πρέπει να παρουσιάζει τον ορισμό των διεργασιών πατώντας πάνω σε κάθε διεργασία που είναι μέσα στο δυσδιάστατο πίνακα και σε ξεχωριστό χώρο στη φόρμα.

8. Το λογισμικό πρέπει να είναι σε θέση να ξεχωρίζει τις διεργασίες του διαφορετικού είδους με διαφορετικά χρώματα.

9. Το λογισμικό πρέπει να εμφανίζει σε κάποιο χώρο στη φόρμα ποιες ενέργειες εκτελεστήκαν μέχρι κάποια συγκεκριμένη στιγμή από το σύστημα και να τις αποθηκεύει σε κάποιο αρχείο .

10. Το λογισμικό πρέπει να εμφανίζει μηνύματα λάθους στη σύνταξη του κειμένου που δίνεται σαν είσοδο, και περιέχει το πληθυσμιακό σύστημα γραμμένο σε PALPS και να δίνει τη δυνατότητα στο χρήστη να διορθώσει τη λάθος σύνταξη.

11.Η σύνταξη που θα δέχεται εξής:
System=(process1|..| process$_n$), ορισμός του συστήματος
(όνομα διεργασίας-processes):(κείμενο), για τη δήλωση των διεργασιών
(όνομα είδους-species):(κείμενο), για τη δήλωση των τύπων του πληθυσμού

Σε οποιαδήποτε σειρά και αν γραφτεί η πιο πάνω σύνταξη θα γίνεται δεκτή από το parser. Δεν είναι απαραίτητο να είναι πρώτα ο ορισμός του συστήματος και μετά ο ορισμός των διεργασιών και μετά για τα είδη.

Π.χ: System=((P:[s,2])|(P1:[s,3])%{eat}|(P2:[s,4]))%{a}| P4:[s,5]

P=(b.(move).0)+b.'a.move.move.P

P1=b.(move.rep.0)+b.'a.move.move.P

P2=c.move.rep.move.0

P4=move.rep.move.0

s=rep.move.0

4.2 Επιλογή γλώσσας προγραμματισμού

Για την επιλογή της γλώσσα προγραμματισμού με επηρέασε πολύ η ευκολία στη διαχείριση κειμένου, καθώς ο χρήστης θα δίνει σαν είσοδο το πληθυσμιακό σύστημα υπό μορφή κειμένου. Σημαντικό επίσης παράγοντα θεώρησα την ευκολία στην αποτελμάτωση, κάτι που με οδήγησε στο συμπέρασμα ότι πρέπει να χρησιμοποιήσω μια Interpreted γλώσσα προγραμματισμού. Μια τέτοια γλώσσα είναι η java, με την οποία είχα και προηγούμενη εμπειρία, οπότε δεν θα αντιμετώπιζα δυσκολίες στην εκμάθηση της γλώσσας ούτε στην εκμάθηση της έννοιας της αντικειμενοστρέφιας. Κάποια άλλα πλεονεκτήματα της java είναι ο μεγάλος αριθμός έτοιμων βιβλιοθηκών που θα είχα στη διάθεση μου, η ευκολία στη ανάπτυξη γραφικής διπροσωπίας, η εύκολη διαχείριση της μνήμης και το γεγονός ότι είναι ανεξάρτητη πλατφόρμας. Τέλος αποφάσισα να χρησιμοποιήσω το εργαλείο NetBeans IDE 6.9.1 για την ανάπτυξη του λογισμικού, το οποίο είναι ένα εργαλείο που σου προσφέρει τη δυνατότητα δημιουργία φορμών αυτόματα.

4.3 Χαρακτηρίστηκα λογισμικού και δυνατότητες χρήσης.

Ακολουθούν κάποια χαρακτηριστικά που έχει το λογισμικό που έχει αναπτυχθεί:

(Χαρακτηριστικό 1): Με το πάτημα ενός κουμπιού θα εμφανίζεται ένα παράθυρο πλοήγησης στα αρχεία του υπολογιστή του χρήστη, μέσω του οποίου ο χρήστης θα επιλέγει το αρχείο κειμένου που περιέχει το πληθυσμιακό σύστημα. Αν ο χρήστης δεν επιθυμεί να εισάγει το σύστημα από το παράθυρο πλοήγησης τότε η φόρμα θα περιέχει χώρο ένα πεδίο κειμένου για να μπορεί ο χρήστης να δώσει το σύστημα

εκεί και μέσω του parser που έχει υλοποιηθεί θα αναγνωρίζει αν έχει δοθεί σωστά η σύνταξη του συστήματος αλλιώς θα βγάζει τα κατάλληλα μηνύματα.

(Χαρακτηριστικό 2): Για τη παρουσίαση όλων των πιθανών ενεργειών έχουμε ένα πεδίο κειμένου, μέσα στο οποίο θα τις παρουσιάζουμε. Ο χρήστης θα μπορεί να δει με την επίδραση της κάθε ενέργειας ποιες διεργασίες επηρεάζονται και τι αλλαγές θα συμβούν στο σύστημα. Οι διεργασίες που επηρεάζονται θα εμφανίζονται με διαφορετικό χρώμα από τις άλλες διεργασίες. Επίσης θα παρουσιάζεται το πώς αλλάζει η διεργασία μετά την εκτέλεση της διεργασίας. Ο χρήστης θα μπορεί να επιλέξει την ενέργεια που επιθυμεί μέσω μίας λίστας με το πάτημα ενός κουμπιού η ενέργεια που επέλεξε θα εφαρμόζεται.

(Χαρακτηριστικό 3): Με το πάτημα ενός κουμπιού ο χρήστης θα ενεργοποιεί μια διαδικασία τυχαίας επιλογής ενεργειών, αφού πρώτα δώσει τον αριθμό επαναλήψεων. Η διαδικασία αυτή θα συνεχίζεται μέχρι συμπληρωθεί ο αριθμός επαναλήψεων, ή μέχρι να μην υπάρχουν άλλες δυνατές ενέργειες.

(Χαρακτηριστικό 4): Για την παρουσίαση τόσο της δυσδιάστατης αναπαράστασης του πληθυσμιακού συστήματος όσο και της αναπαράστασης των διεργασιών χρειάζεται να έχουμε κάποιο χώρο πάνω στη φόρμα, πάνω στον οποίο θα ζωγραφίζεται αυτή η αναπαράσταση. Εκεί θα μπορεί να βλέπει ο χρήστης τη θέση κάθε διεργασίας ακόμη και της κινήσεις της. Επίσης οι διεργασίες διαφορετικού είδους από τις άλλες θα εμφανίζονται με διαφορετικό χρώμα.

(Χαρακτηριστικό 5): Για την παρουσίαση των μηνυμάτων λάθους στην σύνταξη του κειμένου εισόδου, θα εμφανίζονται διάλογοι όπου θα αναγράφονται τα μηνύματα.

(Χαρακτηριστικό 6): Το λογισμικό θα αναγνωρίζει γραμμές που έχουν την συγκεκριμένη σύνταξη της γλώσσας PALPS που ορίσαμε πιο πάνω μέσω του parser που έχει υλοποιηθεί.

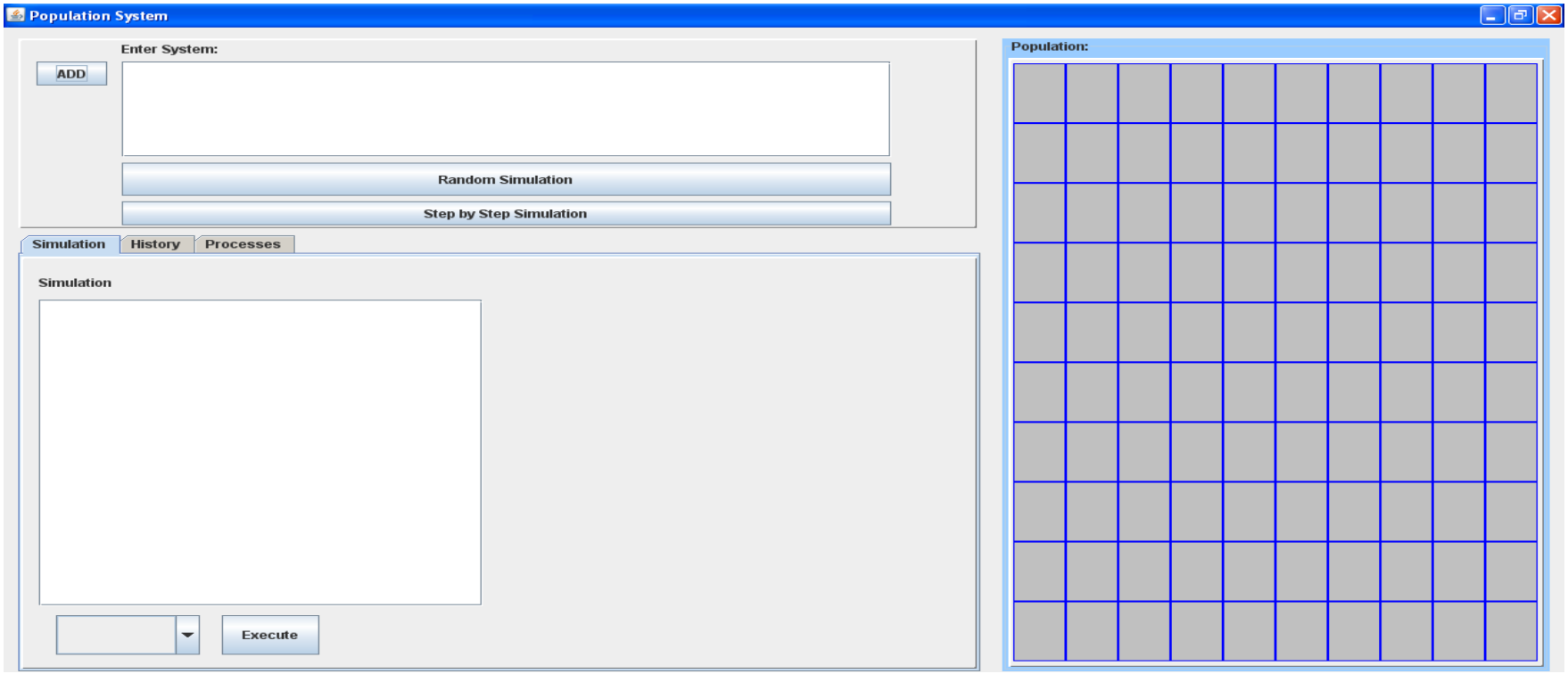
(Χαρακτηριστικό 7): Ο χρήστης θα έχει τη δυνατότητα να αποθηκεύει σε ένα αρχείο κειμένου τις αντιδράσεις που έχουν προσομοιωθεί μέχρι στιγμής από το λογισμικό, με την σειρά που έχουν προσομοιωθεί, θα εμφανίζονται επίσης και σε ξεχωριστό χώρο της φόρμας.

(Χαρακτηριστικό 8): Ο χρήστης θα έχει τη δυνατότητα πατώντας σε κάποια διεργασία που βρίσκεται στο δυσδιάστατο πίνακα να βλέπει το συγκεκριμένο ορισμό της διεργασίας σε κάποιο χώρο της φόρμας και αν πατήσει πάνω στο πίνακα θα μπορεί να δει τον ορισμό του συστήματος. Επίσης ο ορισμός της κάθε διεργασίας αναγράφεται σε ξεχωριστό χώρο πάνω στη φόρμα και έτσι ο χρήστης θα μπορεί να δει την συγκεκριμένη κατάσταση της κάθε διεργασίας.

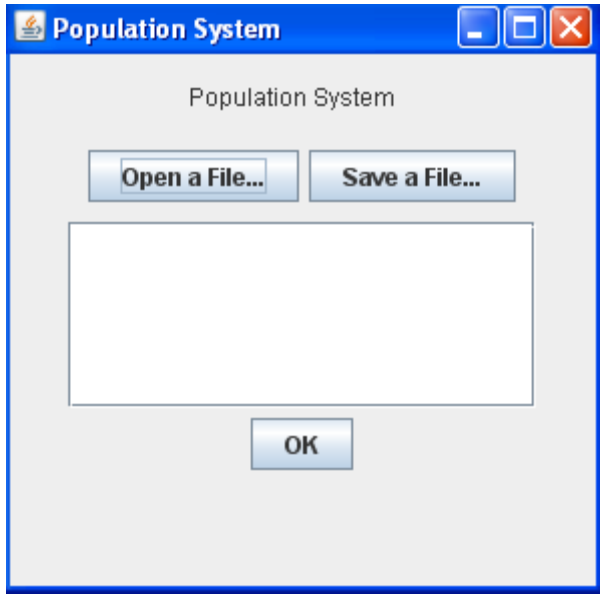
(Χαρακτηριστικό 9): Ο χρήστης θα έχει την δυνατότητα με το πάτημα ενός κουμπιού να επαναφέρει το σύστημα στην αρχική του κατάσταση.

Στη συνέχεια γίνεται εισαγωγή κάποιων φορμών του λογισμικού που υλοποίησα μαζί με την επεξήγηση τους.

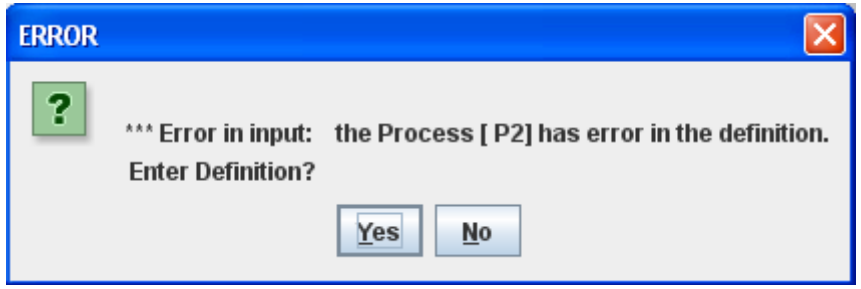
Η φόρμα του λογισμικού θα έχει την εξής δομή κατά την εκκίνηση:



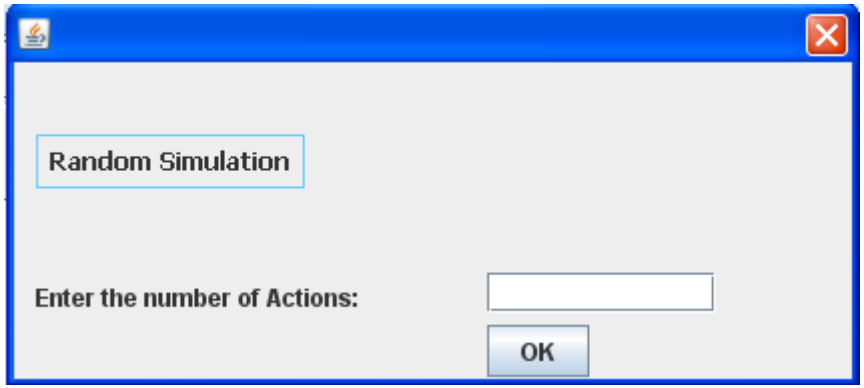
Κατά την εκκίνηση του προγράμματος θα εμφανίζεται η πιο πάνω φόρμα στο χρήστη. Ο χρήστης μπορεί να εισάγει το σύστημα για να ξεκινήσει η προσομοίωση με το κουμπί ADD ή να γράψει στο πεδίο κειμένου το σύστημα που ο ίδιος επιθυμεί. Με το κουμπί Random Simulation ανοίγεται μια νέα φόρμα στην οποία ζητάει από το χρήστη να δώσει ένα αριθμό επαναλήψεων για το πόσες φόρες θα εκτελέσει μια τυχαία ενέργεια το σύστημα. Με το κουμπί Step by Step μπορούμε να δούμε βήμα βήμα την επίδραση της κάθε ενέργειας στο σύστημα καθώς και την αναπαράσταση του συστήματος στο πίνακα Population που είναι τοποθετημένος στα δεξιά της φόρμας.



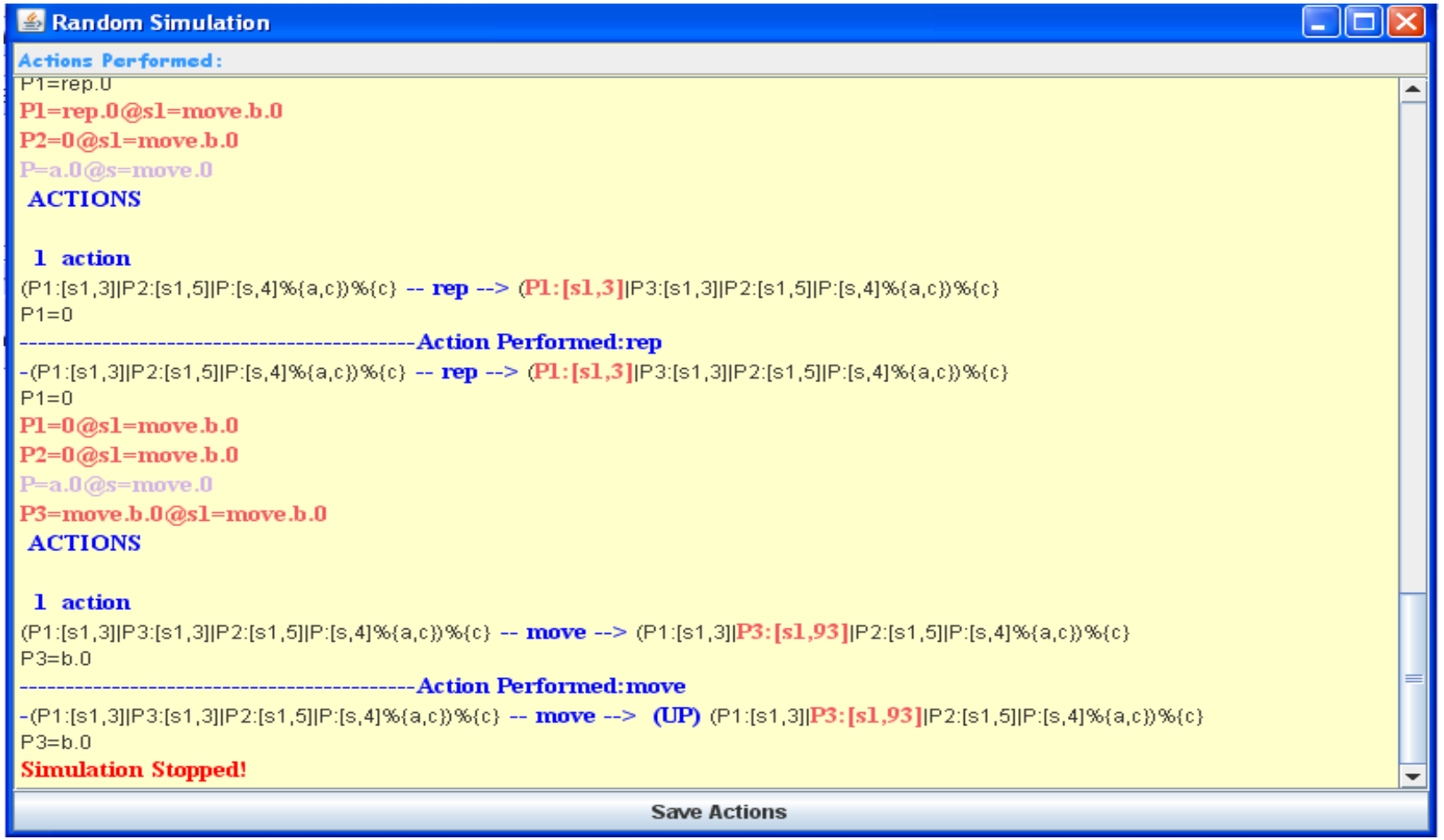
Εδώ φαίνεται η φόρμα που ανοίγεται όταν πατήσει ο χρήστης το κουμπί ADD από την κυρίως φόρμα. Όταν πατήσει το Open File ανοίγεται ένα παράθυρο πλοήγησης στα αρχεία του υπολογιστή του χρήστη μέσω του οποίου ο χρήστης μπορεί να επιλέξει το αρχείο κειμένου που περιέχει το πληθυσμιακό σύστημα. Με το πάτημα του κουμπιού OK επιστρέφει στη κυρίως φόρμα.



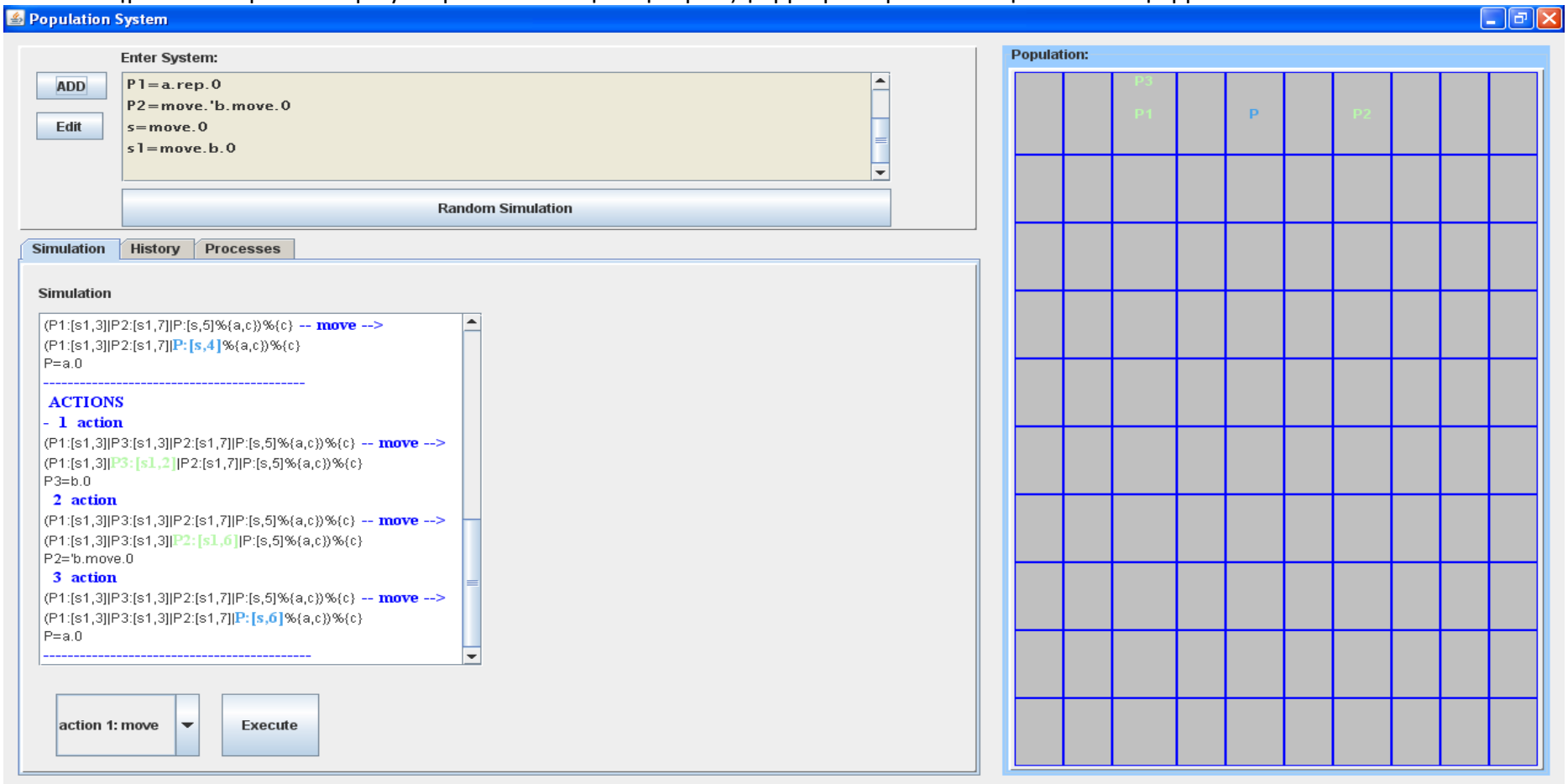
Όταν το σύστημα που έχει δοθεί από τον χρήστη για είσοδο έχει λάθος στη σύνταξη τότε ο χρήστης μπορεί να ενημερωθεί από τη φόρμα που παρουσιάζεται.



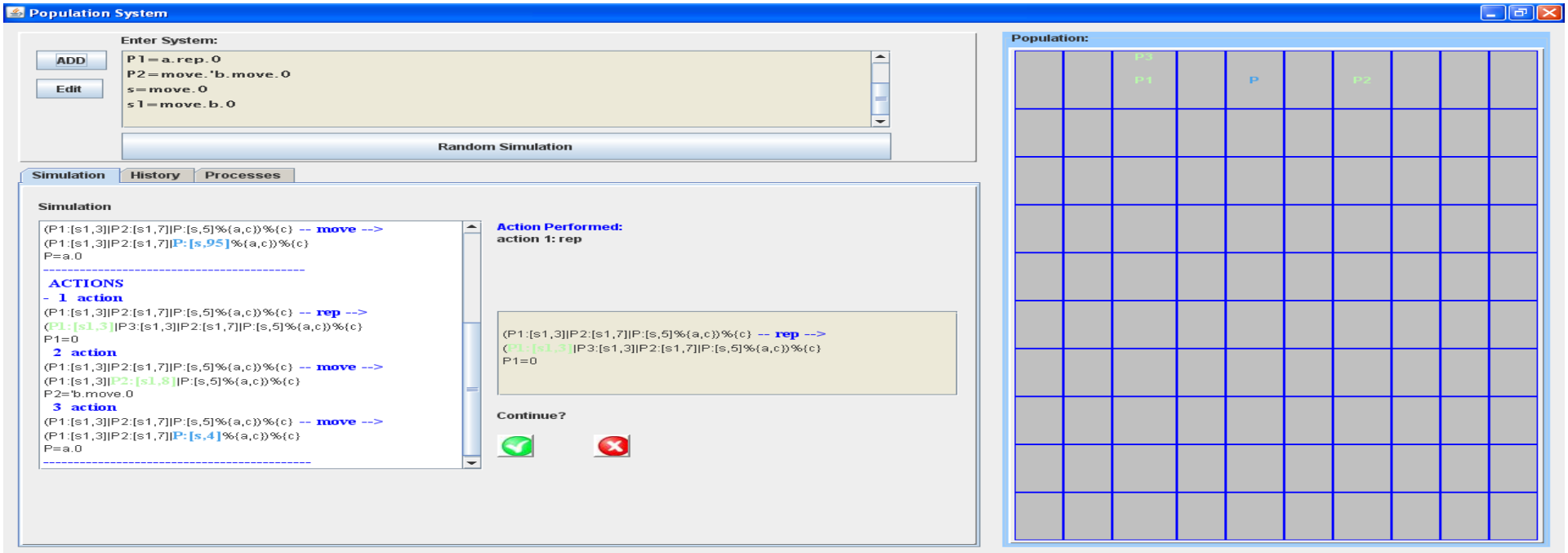
Εδώ φαίνεται η φόρμα που ανοίγεται όταν πατήσει ο χρήστης το κουμπί Random Simulation από την κυρίως φόρμα(προηγούμενη σελίδα). Για να μπορέσει ο χρήστης να δώσει τον αριθμό επαναλήψεων και να προσομοιωθεί το σύστημα με τυχαίες ενέργειες μέχρι να ολοκληρωθεί ο αριθμός επαναλήψεων. Πιο κάτω παρουσιάζεται η φόρμα που δείχνει τις τυχαίες ενέργειες.



Με το πάτημα του κουμπιού Step by Step όταν εκκινήσει η κυρίως φόρμα μετατρέπεται στην πιο κάτω φόρμα:

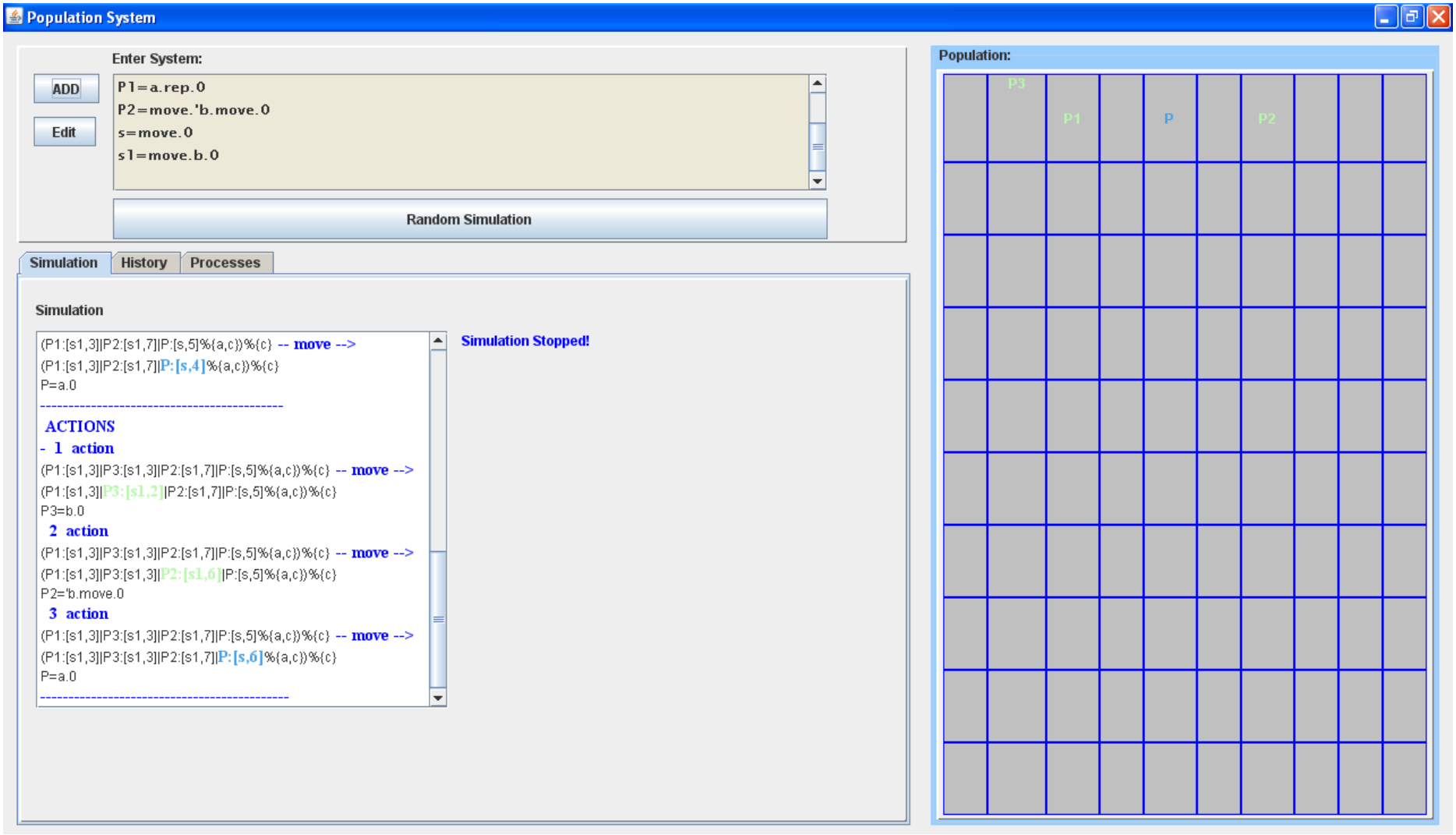


Με το πάτημα του κουμπιού execute τότε εφαρμόζεται η ενέργεια στο σύστημα και γίνονται οι κατάλληλες αλλαγές στο σύστημα και στο πίνακα δίπλα. Η φόρμα μετατρέπεται όπως φαίνεται πιο κάτω:



Παρουσιάζεται η ενέργεια που εκτελέστηκε καθώς επίσης και η παρουσίαση του συστήματος μετά την εκτέλεση του συστήματος. Ο χρήστης έχει τη δυνατότητα να συνεχίσει τη προσομοίωση με το πάτημα ενός κουμπιού ή και τερματίσει. Αν πατήσει το κουμπί 'Χ' τότε η φόρμα έχει την πιο κάτω μορφή. Αν όμως επιθυμεί να συνεχίσει την προσομοίωση τότε η φόρμα θα πάρει τη μορφή που είχε στη προηγούμενη σελίδα.

Όταν τερματίσει η προσομοίωση τότε εμφανίζεται το κατάλληλο μήνυμα και το σύστημα μένει όπως έχει.



Κεφάλαιο 5

Σύστημα Υλοποίησης

5.1 Επεξήγηση Συστήματος Υλοποίησης

5.2 Επεξήγηση Αλγόριθμων που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος

5.3 Αποτελέσματα Συστήματος

Στο υποκεφάλαιο αυτό θα γίνει ακριβής εξήγηση των δομών που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος, τους αλγόριθμους που υλοποίησα και τις κλάσεις – αντικείμενα. Όπως έχει αναφερθεί στο Κεφάλαιο 3 μέσα από τη σημασιολογία της άλγεβρας διεργασίας PALPS, ένας πληθυσμός μπορεί να εκτελέσει τις ενέργειες της μετακίνησης, της αναπαραγωγής, της αρπαγής και όλα τα άτομα που ανήκουν στο πληθυσμό μπορούν να εκτελέσουν τις πιο πάνω ενέργειες ταυτόχρονα και να συγχρονίζονται μεταξύ τους με το τελεστή ταυτοχρονισμού. Εκτός από τις ενέργειες που μπορεί να εκτελέσει ένας πληθυσμός, κάθε αυθαίρετο άτομο μπορεί να εκτελέσει ενέργειες τύπου ‘α’, $(a,\bar{a}) \in Ch$ οι οποίες είναι δραστηριότητες όπως το να τρώει, παρατηρεί το περιβάλλον, και να αναπαράγεται. Το σύστημα που έχω αναπτύξει μπορεί να υλοποιήσει τις ενέργειες αυτές ακόμη και τις ενέργειες της μετακίνησης και της αναπαραγωγής. Η μετακίνηση είναι μια πολύ σημαντική ενέργεια στην προσπάθεια επιβίωσης ενός πληθυσμού. Τα άτομα έχουν την ιδιότητα να φθάνουν πιο γρήγορα στη τοποθεσία όπου υπάρχει τροφή και η οποία θα τους βοηθήσει να τραφούν. Επίσης ένα άτομο που θέλει να εκτελέσει την ενέργεια της αρπαγής τότε πρέπει να μετακινηθεί για να φθάσει σε μια περιοχή όπου υπάρχουν άτομα διαφορετικού είδους από αυτό. Το ίδιο και η ενέργεια της αναπαραγωγής αποτέλεσε μια σημαντική ενέργεια στην προσπάθεια της επιβίωσης ενός πληθυσμού. Ένας πληθυσμός είναι σημαντικό να αναπαράγεται και να

εξαπλώνεται και να δημιουργεί απογόνους. Τα πιο πάνω ήταν σημαντικά κίνητρα τα οποία με ώθησαν να υλοποιήσω ένα υποσύνολο της PALPS.

Η γενική ιδέα της υλοποίησης ήταν η ανάπτυξη ενός συστήματος το οποίο θα προσομοιώνει όλες τις πιθανές ενέργειες ενός πληθυσμιακού συστήματος μοντελοποιημένο στην άλγεβρα διεργασίας PALPS. Κάθε άτομο που ανήκει στο πληθυσμό εκτελεί όλες τις πιθανές ενέργειες μαζί με όλα τα υπόλοιπα άτομα. Στο σύστημα που υλοποίησα ο κόσμος στον οποίο συνυπάρχουν τα άτομα είναι ένας 10×10 δυσδιάστατος πίνακας και συνεπώς υπάρχουν 100 διαφορετικές θέσεις. Κάθε άτομο έχει τη δυνατότητα να μετακινείται πάνω, κάτω, δεξιά και αριστερά τοποθεσία από αυτή που βρίσκεται. Ο 10×10 κόσμος παρουσιάζει κυκλικές ιδιότητες. Αυτό σημαίνει ότι όταν ένα άτομο βρίσκεται για παράδειγμα σε μια από τις τοποθεσίες της πρώτης γραμμής του πίνακα και κινηθεί προς τα πάνω τότε θα βρεθεί στην αντίστοιχη στήλη στην τελευταία γραμμή του πίνακα. Η αντίστοιχη σχέση παρουσιάζεται με την δεξιά και αριστερή πλευρά του κόσμου.

Η κύρια ιδέα της υλοποίησης μπορεί να χωριστεί σε δύο σημαντικά μέρη.

Στο πρώτο μέρος επιτυγχάνεται η υλοποίηση του συντακτικού αναλυτή (parser) μαζί με το κτίσιμο του συντακτικού δέντρου του οποίου οι ακραίοι κόμβοι του απεικονίζουν όλες τις πιθανές ενέργειες που κάθε διεργασία μπορεί να εκτελέσει.

Ο συντακτικός αναλυτής (parser) ελέγχει και μετά συνθέτει τις λεκτικές μονάδες (tokens) με βάση τη σύνταξη της γλώσσας, ώστε να προκύψει μια αφηρημένη μορφή του προγράμματος (συντακτικό δέντρο), κατάλληλη για περαιτέρω επεξεργασία. Ένας συντακτικός αναλυτής χρειάζεται μια αναλυτική γραμματική, γραμμένη σε μορφή κοντά στη BNF. Μια αναλυτική γραμματική ορίζει τις νόμιμες μορφές που μπορεί να έχει μια συμβολοσειρά, ελέγχοντας κατά πόσο η συγκεκριμένη συμβολοσειρά ανήκει στην γλώσσα.

Η σύνταξη της γλώσσας ορίζεται από τις πιο κάτω γραμματικές:

- **System ::= P:expr | (System) | System%rest | System|System**
- **expr ::= [digit,letter]**
- **restr ::= {channels}**
- **channels ::= channel | channel,channel**

- **channel ::= [a-z]***
- **P ::= actions.P | 0 | P+P | (P) | letter**
- **Actions ::= channel | ‘channels | move | rep**
- **digit ::= [1-9]***
- **letter ::= [A-Z,a-z]***

Κάθε γραμματική που ορίσαμε πιο πάνω, παρουσιάζεται πιο κάτω η υλοποίηση της.

Κλάσεις για την δημιουργία του συντακτικού δέντρου:

Ένα συντακτικό δέντρο αποτελείται από λεκτικές μονάδες με βάση της σύνταξη της γλώσσας . Στην περίπτωση μας, οι ακραίοι κόμβοι του συντακτικού δέντρου απεικονίζουν όλες τις πιθανές ενέργειες που κάθε διεργασία μπορεί να εκτελέσει. Πιο κάτω παρουσιάζεται η κλάση Node που χρησιμεύει για το κτίσιμο του δέντρου.

```
public class Node {
        Node previous;
         Node left;
         Node right;
         String value;
         Boolean action=false;
         Integer x=-1;
         Integer y=-1;
         Person p=null;
         Species s=null;
         Set<String> restricted_actions=null;
}
```

Η κλάση Node χρησιμοποιείται για την απεικόνιση των ατόμων στο πληθυσμό και τις ενέργειες που μπορεί να εκτελέσει κάθε άτομο. Τα πεδίο previous, left και right χρησιμεύουν για το κτίσιμο του δέντρου. Το previous δείχνει το προηγούμενο του κόμβο. Το δέντρο που κτίζεται είναι δυαδικό επομένως είναι απαραίτητο να υπάρχουν 2 δείχτες που να δείχνουν το αριστερό και το δεξί κόμβο του δέντρου. Το πεδίο value

αντιπροσωπεύει την τιμή που έχει ένας κόμβος για παράδειγμα μπορεί να είναι ένας τελεστής ‘|’ ή μια ενέργεια’α’. Το πεδίο action υποδηλώνει κατά πόσο ο συγκεκριμένος κόμβος του δέντρου είναι μια ενέργεια ‘α’ ή ‘move’ή ‘rep’. Αν είναι μια ενέργεια τότε έχει τιμή true διαφορετικά false. Τα πεδία x και y χρησιμοποιούνται για να κρατούν τις συντεταγμένες του χώρου στο οποίο βρίσκεται ένα άτομο. Το πεδίο p(Person) χρησιμοποιείται για να κρατά το όνομα του ατόμου-διεργασία και επίσης κρατάει την ρίζα του κόμβου του ορισμού του. Το πεδίο s(Species) χρησιμοποιείται για να κρατά το όνομα του είδους ενός ατόμου και επίσης κρατάει την ρίζα του κόμβου του ορισμού του.(χρησιμεύει για την ενέργεια της αναπαραγωγής). Το πεδίο restricted_actions κρατάει τις ενέργειες που δεν πρέπει να εκτελέσει μια διεργασία.

Π.χ αν έχουμε το πιο κάτω σύστημα:

System=P:[s,5]|P1:[s1,7]

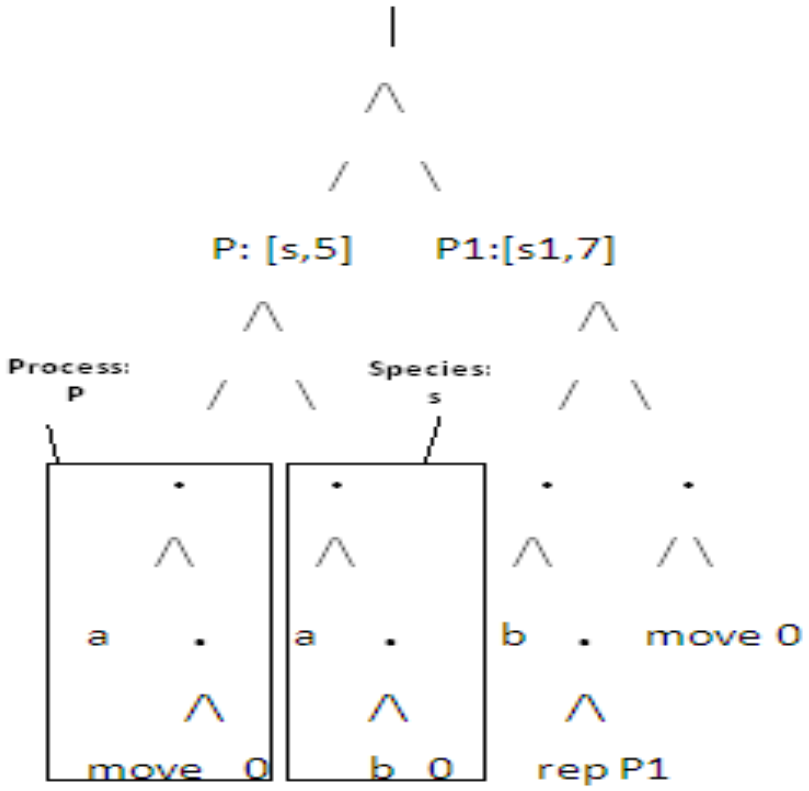
P=a.move.0

P1=b.rep.P1

S=a.b.0

S1=move.0

Η δομή του συντακτικού δέντρου πρέπει να είναι όπως δείχνει το σχήμα:



Στην παράγραφο αυτή θα παρουσιαστούν με ψευδοκώδικα οι αλγόριθμοι των πιο πάνω γραμματικών. Για κάθε γραμματική/κατηγορία ορίζουμε μια αναδρομική διαδικασία η οποία με δεδομένο εισόδου μια συμβολοσειρά ελέγχει κατά πόσο η συμβολοσειρά ικανοποιεί τη γραμματική για την συγκεκριμένη κατηγορία και ταυτόχρονα χτίζεται και το συντακτικό δέντρο .

Παρουσιάζεται ο ψευδοκώδικας του κανόνα **System** βήμα προς βήμα πιο κάτω:

```
Boolean ParseSystem(String statement,Node node) {
                boolean flag=false;
                Boolean flag1=false;
           int countLeft = 0; //αριθμός αριστερών
      παρενθέσεων
           if(statement doesn’t contain ‘|’ or ‘%’ or
      ‘(’or ‘)’)
                flag1=true;
           if(node ==null)
                node=new Node();
           for (int i = 0; i< statement.length(); i++) {
                char c=statement.charAt(i);
           if(c equals ‘(’)
                     countLeft++;
           else if(c equals ‘)’)
                if(countLeft>0)
                     countLeft--;
           else
                     error;
           else if countLeft==0

           if c equals ‘|’
```

```
     str,str1=split statement into 2 by the symbol
'|'

          node.value="|";

     flag=ParseSystem(str,node.left)              &
ParseSystem(str1,node.right);

     if(flag==false) return error;
     return flag;
     if c equals '%'
          str,str1=split  statement  into  2  by  the
symbol '%'
     if(str1 contains "|")
               flag=ParseSystem(str,node.right)    &
ParseRestriction(str1,node) ;
     else
               flag=ParseSystem(str,node)          &
ParseRestriction(str1,node.right) ;
     if(flag==false) return error;
     return flag;
     if c equals ':' & flag1=true
          if(definedProcess(str)==false)
               print « the process 'str' hasn't been
defined. » ;
          node.value=statement;
          flag=ParsePerson(str,node.left)          &
expr(str1,node.right) ;
     if(flag=false)return error ;
     return flag;
     else if c=EOF && countLeft>0
          return Error;
     else if c=EOF && countLeft==0
     str=RemoveInitialLastParenthesis(str);
```

```
            flag=ParseSystem(str,node);
        }
```

Η βασική ιδέα του πιο πάνω ψευδοκώδικα είναι ότι παίρνει σαν παράμετρο μια συμβολοσειρά και βάση της γραμματικής που αντιπροσωπεύει System ::= P:expr | (System) | System%rest | System|System, έχουμε 4 περιπτώσεις. Η 1η περίπτωση είναι (System), ελέγχουμε αν η συμβολοσειρά έχει ίσο αριθμό παρενθέσεων. Αν όντως έχει ίσο αριθμό τότε καλεί ξανά τη συνάρτηση ParseSystem με την ίδια συμβολοσειρά αλλά χωρίς τις παρενθέσεις. Η 2η περίπτωση είναι System|System, ελέγχουμε ένα-ένα χαρακτήρα της συμβολοσειράς μέχρι να βρούμε το σύμβολο '|'. Όταν το βρούμε τότε χωρίζουμε τη συμβολοσειρά σε 2 μέρη και τότε ξανακαλούμε 2 φορές τη συνάρτηση ParseSystem με παράμετρο το ένα μέρος της συμβολοσειράς και την άλλη φορά με το άλλος μέρος της συμβολοσειράς. Η 3η περίπτωση είναι System%rest, ελέγχουμε κάθε χαρακτήρα της συμβολοσειράς μέχρι να βρούμε το σύμβολο '%'. Είναι ίδια η διαδικασία με της 2ης περίπτωσης, μόνο που η διαφορά είναι ότι δεν καλεί 2 φόρες τη συνάρτηση ParseSystem. Όταν σπάσει τη συμβολοσειρά σε 2 μέρη βάση του συμβόλου '%' τότε το πρώτο μέρος περνιέται σαν παράμετρος στη συνάρτηση ParseSystem και το δεύτερο μέρος περνιέται σαν παράμετρος στη συνάρτηση ParseRestriction. Και η τελευταία περίπτωση είναι P:expr, ελέγχουμε ένα-ένα χαρακτήρα της συμβολοσειράς μέχρι να βρούμε το σύμβολο ':'. Όταν το βρούμε χωρίζουμε τη συνάρτηση σε 2 μέρη βάση του συμβόλου ':'. Το πρώτο μέρος παίρνεται σαν παράμετρος στη συνάρτηση ParsePerson και το δεύτερο μέρος παίρνεται σαν παράμετρος στη συνάρτηση expr. Για κάθε περίπτωση αν δεν βρεθεί το σύμβολο που θα χωρίσει τη συμβολοσειρά τότε συνεχίζει στην επόμενη περίπτωση. Όταν φθάσει στη τελευταία περίπτωση και δεν βρεθεί το σύμβολο τότε επιστρέφει τη λογική τιμή false που σημαίνει ότι η συμβολοσειρά δεν ακολουθεί .

Παρουσιάζεται ο ψευδοκώδικας του κανόνα ParseRestriction βήμα προς βήμα πιο κάτω:

```
Boolean ParseRestriction(String statement, Node node) {
        Boolean flag = false, flag1 = false;

          if(node==null)
               node=new Node();
```

```
          if (statement  contains "|") {
              s1,s2=split str1 into two strings by ‘|’
               node.value=”|”;
       return           ParseRestriction(s1,node)           &
ParseSystem(s2,node.left);
               }

         if (statement[0]!=’{’)
                   print("Left '{' misplaced.");

          for(int i=1;i<statement.lenght;i++){
                   str=str+c;
                   char c=statement.charAt(i);

              if (c == ‘}’) {
                  return restr(str);
              }
          }
     return error;

}
```

Η πιο πάνω συνάρτηση χρησιμεύει για να ελέγξουμε όταν μια συμβολοσειρά ακολουθεί αυτή τη σύνταξη “ {restr}|System” και καλείται από την συνάρτηση ParseSystem όταν βρει το σύμβολο ’%’.Η συνάρτηση αυτή ελέγχει αν υπάρχει στη συμβολοσειρά το σύμβολο ‘|’ και αν όντως υπάρχει τότε χωρίζει τη συμβολοσειρά σε 2 μέρη βάση του συμβόλου ‘|’.Το πρώτο μέρος περνιέται σαν παράμετρος στην συνάρτηση ParseRestriction και το δεύτερο μέρος περνιέται σαν παράμετρος στην συνάρτηση ParseSystem. Αν δεν υπάρχει το σύμβολο ‘|’ στη συμβολοσειρά τότε γίνεται έλεγχος για το πρώτο χαρακτήρα της συμβολοσειράς αν είναι το σύμβολο ‘{’ και συνεχίζει μέχρι το τέλος της συμβολοσειράς να βρει το σύμβολο ‘}’, όταν το βρει τότε καλείται η συνάρτηση restr με παράμετρο την ίδια συμβολοσειρά αλλά χωρίς τα σύμβολα ‘{}’.

Παρουσιάζεται ο ψευδοκώδικας του κανόνα expr βήμα προς βήμα πιο κάτω:

```
Boolean expr(String statement,Node node){
     Boolean flag = false, flag1 = false;
     for (int i = 0; i < statement.length(); i++) {

          char c=statement.charAt(i);// next character is not EOF
          if(i==0 && c isn't '['
               print "Misplaced '['".

          read until you reach  ','  and save string as location
     continue reading until you reach ']' and save string as
     species
               flag=Digit(location);

                    if(definedSpecies(str)==false)
                              print    « the    species 'species'
               hasn't been defined. » ;//έλεγχος μέσα στο αρχείο αν
               είναι ορισμένη η συμβολοσειρά  species

                    flag=flag & letter(species);

return flag;
}
```

Παρόμοια η βασική ιδέα του πιο πάνω ψευδοκώδικα είναι ότι παίρνει για παράμετρο μια συμβολοσειρά για να ελέγξει αν ακολουθεί την εξής σύνταξη «[digit,letter]». Ελέγχουμε το πρώτο χαρακτήρα της συμβολοσειράς αν είναι το σύμβολο '[' και μετά συνεχίζουμε μέχρι να φθάσουμε στο σύμβολο '] '. Όταν φθάσουμε στο σύμβολο ']' χωρίζουμε τη συμβολοσειρά σε 2 μέρη βάση του συμβόλου ',' . Μετά καλούμε τη συνάρτηση digit() με παράμετρο το πρώτο μέρος της συμβολοσειράς. Το δεύτερο μέρος το ελέγχουμε αν έχει ορισθεί στο αρχείο κειμένου που δώσαμε για είσοδο. Στη συνέχεια καλούμε τη συνάρτηση letter() με παράμετρο το δεύτερο μέρος της συνάρτησης.

Παρουσιάζεται ο ψευδοκώδικας του κανόνα restr βήμα προς βήμα πιο κάτω:

```
Boolean restr(String statement,Node node){

                    return channels(statement);
}
```

Παρουσιάζεται ο ψευδοκώδικας του κανόνα channels βήμα προς βήμα πιο κάτω:

```
Boolean channels(String statement){
     if(statement contains ‘,’)
               str,str1=split statement into two by the symbol ‘,’
          return channel(str)&channels(str1);
     else
          return channel(statement);

}
```

Παρουσιάζεται ο ψευδοκώδικας του κανόνα channel βήμα προς βήμα πιο κάτω:

```
Boolean channel(String statement){
     if(statement matches ("[a-z]*"))//regular expression
          return true;
     else
          return false;
}
```

Παρουσιάζεται ο ψευδοκώδικας του κανόνα parsePerson βήμα προς βήμα πιο κάτω:

```
static Boolean ParsePerson(String statement,Node node) {
        Boolean flag = false;
        If(node == null)node=new Node();
        int mikos_string=statement.length();
        int lbrackets=0;
        for (int i = 0; i < mikos_string; i++) {
                char c=statement.charAt(i);
                if (c equals to '(') {
                    lbrackets++;
                }
                if (c equals to ')') {
                      if(lbrackets>0)
                            lbrackets--;
```

```
                    if(lbrackets equals to 0 && next character
after c is EOF)
                          return ParseSystem(str,node);//(P)
                    }
                else
                     error;
          else

          str=str+c;//append character to the new string str


        if(lbrackets==0)
           if (c equals to '+') {
              node.value="+";
              return             Person(str,node.left)             &
Person(str1,node.right);
          }
          if (c equals to '.') {
              node.value=".";
              return             actions(str,node.left)            &
Person(str1,nide.right);
          }
          if (c equals to EOF) {
                    if (statement matches("[0]")) {
                          node.value=statement;
                          return true;
                    } else {
                          Node.value=statement;
                          return letter(statement);
                    }
        }
return false;}
```

Ο ψευδοκώδικας ParsePerson() ακολουθεί την ίδια λογική με όλες τις άλλες συναρτήσεις που υλοποιούν τις γραμματικές του συντακτικού αναλυτή. Παίρνει σαν παράμετρο μια συμβολοσειρά και ελέγχει αν ακολουθεί αυτή τη σύνταξη «**actions.P | 0 | P+P | (P) | letter».** Έχουμε, δηλαδή, 5 περιπτώσεις. Η 1η περίπτωση είναι (P),

ελέγχουμε αν η συμβολοσειρά έχει ίσο αριθμό παρενθέσεων. Αν όντως έχει ίσο αριθμό τότε καλεί ξανά τη συνάρτηση ParsePerson() με την ίδια συμβολοσειρά αλλά χωρίς τις παρενθέσεις. Η 2η περίπτωση είναι **P+P**, ελέγχουμε κάθε χαρακτήρα της συμβολοσειράς μέχρι να βρούμε το σύμβολο '+'. Όταν το βρούμε τότε χωρίζουμε τη συμβολοσειρά σε 2 μέρη και τότε ξανακαλούμε 2 φορές τη συνάρτηση ParsePerson() με παράμετρο το ένα μέρος της συμβολοσειράς και την άλλη φορά το άλλος μέρος της συμβολοσειράς. Η 3η περίπτωση είναι **actions.P**, ελέγχουμε κάθε χαρακτήρα της συμβολοσειράς μέχρι να βρούμε το σύμβολο '.'. Όταν βρούμε το σύμβολο '.' τότε χωρίζουμε τη συμβολοσειρά σε 2 μέρη βάση του συμβόλου '.' το πρώτο μέρος περνιέται σαν παράμετρος στη συνάρτηση actions και το δεύτερο μέρος περνιέται σαν παράμετρος στη συνάρτηση ParsePerson. Και η τελευταία περίπτωση ελέγχουμε αν η συμβολοσειρά ταιριάζει με τη συμβολοσειρά «0» ή αν είναι απλά είναι ένα αλφαριθμητικό.

Παρουσιάζεται ο ψευδοκώδικας του κανόνα actions βήμα προς βήμα πιο κάτω:

```
Boolean actions(String statement,Node node) {

        if (statement matches("move")) {
                node.value=statement;
            flag = true;
        } else if (statement matches("rep")) {
                Node.value=statement;
            flag = true;
        } else{
            if(statement matches ("[a-z]*|'[a-z]*"))

                 print ("Wrong Action.  Found \""
                    + statement + "\" instead of move, rep, [a-z],
or '[a-z].");
            else
      node.value=statement;
       return true;
        }
    }
```

Η συνάρτηση actions ελέγχει αν η συμβολοσειρά που περνιέται σαν παράμετρος ταιριάζει με τις συμβολοσειρές των ενεργειών που όρισα για την γλώσσα PALPS. Ο πρώτος έλεγχος που γίνεται είναι αν η συμβολοσειρά ταιριάζει με την συμβολοσειρά της ενέργειας «move» διαφορετικά ελέγχει αν ταιριάζει με την συμβολοσειρά «rep». Αν δεν ταιριάζει με τις πιο πάνω ενέργειες τότε γίνεται έλεγχος αν είναι ένα απλό κανάλι της μορφής [a-z] αν είναι ενέργεια εισόδου ή αν είναι ενέργεια εξόδου ‘[a-z].

Παρουσιάζεται ο ψευδοκώδικας του κανόνα digit βήμα προς βήμα πιο κάτω:

```
Boolean digit(String st) {
        if (st matches("[1-9][0-9]{0,2}")) {//ελέγχει αν το αλφαριθμητικό είναι ακέραιος αριθμός μονοψήφιος ή διψήφιος
            return true;
        }
        return false;
  }
```

Παρουσιάζεται ο ψευδοκώδικας του κανόνα letter βήμα προς βήμα πιο κάτω:

```
private static Boolean letter(String st) {
        if (st matches("[a-z]*[0-9]*|[A-Z]*[0-9]*")) {
            return true;
        }
        return false;
    }
```

Όταν ολοκληρωθεί η πρώτη φάση και είναι σωστή η σύνταξη του κειμένου που έχει δοθεί για είσοδο τότε περνάμε στο δεύτερο μέρος της υλοποίησης το οποίο έχει να κάνει με την σημασιολογία της γλώσσας και τους κανόνες της. Πιο κάτω παρουσιάζεται η κυρίως συνάρτηση του λογισμικού η οποία δείχνει τη συνολική ροή του προγράμματος. Οι συναρτήσεις που χρησιμοποιεί θα αναλυθούν πιο κάτω αναλυτικότερα.

```
void main(){
FindActions( root);
    do{
   InternalAction();
   RestrictedDublicateAction(root);
    if (actions.size() == 0) {
```

```
        Print("Population System is in deadlock. No more actions. " );
        exit(0);
    }
      //Ask User what action to use from the array "actions"
      action=InitializeAction();

            if (action matches("rep")) {
                root = Reproduction(root, ...);

            }
            if (action matches("move")) {
                root = MoveAction(root, ...);
            }

            root = DeleteActions( root, ...);
        }
        actions = null;

    FindActions( root);
    if(actions.isEmpty()){
      Print ("Population died!");
    }
    }while(!actions.isEmpty());
}
```

Όπως έχουμε αναφέρει το δεύτερο μέρος έχει να κάνει με την σημασιολογία της γλώσσας και τους κανόνες της. Για την υλοποίηση αυτών των κανόνων χρειάστηκε να προσπελάσουμε το συντακτικό δέντρο για να βρούμε τις πιθανές ενέργειες που μπορεί να εκτελέσει κάθε άτομο που ανήκει στο πληθυσμό. Όλες οι ενέργειες αποθηκεύονται σε ένα δυσδιάστατο πίνακα με το όνομα actions. Αυτό επιτυγχάνετε με τις πιο κάτω συναρτήσεις:

void FindActions(Node node)

void findActionsIndividual(Node node,..)

Στη συνέχεια αφού βρούμε όλες τις ενέργειες που μπορεί να εκτελέσει το σύστημα, θα προσπελάσουμε το πίνακα ‘actions’ που αρχικοποιήσαμε στις προηγούμενες διαδικασίες για την εύρεση ενεργειών, για να προσθέσουμε την εσωτερική ενέργεια. Η εσωτερική ενέργεια όπως έχει αναφερθεί στο κεφάλαιο 3 επιτυγχάνεται όταν μια ενέργεια εισόδου και μια ενέργεια εξόδου εκτελούνται στην ίδια τοποθεσία από δύο ή περισσότερα διαφορετικά άτομα.

public static void InternalAction() {}

Μετά αφού γίνει ο έλεγχος αν χρειάζεται να προστεθεί η εσωτερική ενέργεια, τότε γίνεται έλεγχος για τις δεσμευμένες ενέργειες. Τότε αφαιρούνται ενέργειες από το πίνακα ‘actions’.

public static Node RemoveRestrictedActions(Node node){}

Αφού γίνει η επιλογή της ενεργείας που θα εκτελεστεί τότε γίνεται έλεγχος αν είναι οι ενέργειες της μετακίνησης ή της αναπαραγωγής για να γίνει η κατάλληλη τροποποίηση στο συντακτικό δέντρο.

private static Node Reproduction(Node root,..){}

private static Node MoveAction(Node root,…){}

Στο τέλος διαγράφεται από το συντακτικό δέντρο η ενέργεια που εκτελέστηκε.

private static Node DeleteActions(String action, Node root, ..){}

Η όλη διαδικασία επαναλαμβάνεται μέχρι όλα τα άτομα του πληθυσμού να μην έχουν άλλη ενέργεια να εκτελέσουν. Αν όμως υπάρχουν διεργασίες που είναι αναδρομικές τότε η προσομοίωση θα τερματίσει μόνο από τον χρήστη.

Πιο κάτω θα δοθεί μια πιο λεπτομερής αναφορά για τις συναρτήσεις που αναφέραμε καθώς επίσης θα δοθεί σε μορφή ψευδοκώδικα ο τρόπος λειτουργίας τους:

```
public static void FindActions(Node node) {

        if (node != null) {
            if (node.value.equals("|")) {
            Κλήση της συνάρτησης FindActions με παράμετρο τον
      αριστερό κόμβο
            Κλήση της συνάρτησης FindActions με παράμετρο το δεξή
      κόμβο
            }else if(node.p!=null){//έχει βρει ένα άτομο
            Κλήση της συνάρτησης  findActionsIndividual() για εύρεση
      των ενεργειών αυτού του ατόμου;

      }

   }
}
```

Η αναδρομική συνάρτηση FindActions χρησιμεύει στο να βρίσκει τις διεργασίες μέσα στο συντακτικό δέντρο και στη συνέχεια σε κάθε διεργασία καλείται η συνάρτηση findActionsIndividual για εύρεση των ενεργειών αυτής της διεργασίας.

Πριν προχωρήσουμε στον ψευδοκώδικα της findActionsIndividual θα πρέπει να γίνει αναφορά στο πίνακα actions. Κάθε γραμμή του πίνακα actions είναι τύπου Individual:

```
static class Individual {
      String nameOfAction;
      String []nameOfProcess;
      Node  nodesChoice=null;
      int movement=-1;
      int position=-1;
 }
```

Το πεδίο nameOfAction αντιπροσωπεύει την ενέργεια που θα εκτελέσει το σύστημα.
Ο πίνακας nameOfProcess περιέχει τις διεργασίες που επηρεάζονται από την εκτέλεση της ενέργειας nameOfAction. Το πεδίο nodesChoice χρησιμεύει όταν η διεργασία είναι τύπου P1+P2 και έτσι μέσω αυτού του πεδίου κρατάμε το κόμβο που θα διαγραφεί από το συντακτικό δέντρο αν εκτελεστεί η ενέργεια nameOfAction και η ρίζα της έχει ως τελεστή το ‘+’ .Το πεδίο movement παίρνει τις τιμές 1-4 που σημαίνει αν το άτομο κινηθεί πάνω, κάτω, δεξιά και αριστερά. Την τιμή αυτή την παίρνει μόνο αν το πεδίο nameOfAction είναι ίσο με «move».Το πεδίο position είναι η τοποθεσία του ατόμου και αυτό μας χρειάζεται για την συνάρτηση της εσωτερικής ενέργειας. Ο πίνακας action θα έχει την πιο κάτω μορφή:

Actions				
Όνομα ενέργειας	**Όνομα Διεργασίας**	**Κόμβος choice**	**Τιμή κίνησης**	**Τοποθεσία ατόμου**
‘a’	**P**	**Null**	**-1**	**(1,1)**
‘b’	**P3**	**..**	**-1**	**(2,3)**
Move	**P6**	**..**	**1(πάνω)**	**(1,1)**
_t	**P,P6**	**Null**	**-1**	**(-1,-1)**

```
private static void findActionsIndividual(Node node, Node choice,
String Process) {

        if (node.value.equals("+")) {
      Κλήση της συνάρτησης  findActionsIndividual( node.left,
    node.right, Process).
       Κλήση της συνάρτησης  findActionsIndividual( node.right,
    node.left, Process
        } else if (node.value.equals(".")) {
      Κλήση της συνάρτησης  findActionsIndividual(node.left,
    choice, Process);
    } else if (node.action==true) {
      Αν είναι η ενέργεια move τότε
          Βάζουμε στο άτομο μια τυχαία τιμή από το 1-
      4.(αριστερά, δεξιά, πάνω, κάτω )

    αποθήκευση ενέργειας στο πίνακα actions μαζί με το όνομα της
διεργασίας  τον κόμβο choice, την κίνηση και την τοποθεσία του
ατόμου.

}
}
```

Η συνάρτηση **findActionsIndividual** έχει ως παραμέτρους τον κόμβο που προσπελάζεται μέχρι να φθάσει στο κόμβο που είναι η ενέργεια (Node node), τον κόμβο που πρόκειται να διαγραφεί από το συντακτικό δέντρο αν επιλεγεί η συγκεκριμένη ενέργεια (Node choice) το όνομα της διεργασίας(String Process)και η τοποθεσία ενός ατόμου. Οι παράμετροι αυτοί αποθηκεύονται στο πίνακα actions.Στόχος της πιο πάνω συνάρτησης είναι να αρχικοποιήσει το πίνακα actions.

Εκτός από την υλοποίηση των πιο πάνω συναρτήσεων γίνεται και χρήση της πιο κάτω συνάρτησης η οποία ανατρέχει το πίνακα actions και ελέγχει αν χρειάζεται το σύστημα να εκτελέσει την εσωτερική ενέργεια. Αν όντως χρειάζεται τότε προσθέτει στο πίνακα actions την εσωτερική ενέργεια μαζί με τις διεργασίες που επηρεάζονται από αυτή.

```
public static void InternalAction() {
```

```
       Individual item = null, item1 = null;
        Individual newItem=null;
for (int i = 0; i < actions.size(); i++) {
     item = ανάκτηση 'i' στοιχείου από το action πίνακα
          for (int j = 0; j < actions.size(); j++) {
               item1 = ανάκτηση 'j' στοιχείου από το action πίνακα
          if    (item1.nameOfAction    is    complementary    with
     item.nameOfAction) {
          String Process= ανάκτηση διεργασίας από το item
          String Process1= ανάκτηση διεργασίας από το item1;
               if(item.position==item1.position    &&    Process!=
          Process1)
               if(newItem.nameOfProcesses    δεν    περιέχει    τις
διεργασίες Process, Process1)
     αποθήκευση στο newItem τις διεργασίες  Process, Process1.

  }
 }

Πρόσθεση στο πίνακα actions το newItem.
}
```

Στόχος της πιο πάνω συνάρτησης είναι η αποθήκευση ενός νέου αντικείμενου στο τέλος του πίνακα actions. Το νέο αντικείμενο θα αντιπροσωπεύει την εσωτερική ενέργεια μαζί με όλες τις διεργασίες που επηρεάζονται από αυτή. Το σημαντικότερο σημείο σε αυτή την συνάρτηση είναι ότι παίρνει δύο στοιχεία από το πίνακα actions και συγκρίνει τις ενέργειες που αντιπροσωπεύουν αν είναι συμπληρωματικές τότε συνεχίζει και ελέγχει αν είναι στην ίδια τοποθεσία τα άτομα που ανήκουν στο πληθυσμό και τότε αν οι διεργασίες που είναι συμπληρωματικές μεταξύ τους δεν υπάρχουν ήδη μέσα στο πίνακα με τις διεργασίες τότε αποθηκεύονται. Στο τέλος αφού συγκρίνονται όλες οι ενέργειες του πίνακα actions τότε γίνεται η εισαγωγή tounewItem.

Μετά την διαδικασία για την εισαγωγή της εσωτερικής ενέργειας, γίνεται χρήση της πιο κάτω συνάρτησης για τις δεσμευμένες ενέργειες κάθε διεργασίας. Αυτή η συνάρτηση θα έχει ως αποτέλεσμα την αφαίρεση κάποιων ενεργειών από το πίνακα actions.

```
public static Node RemoveRestrictedActions(Node node){
    if(node!=null){
        if(node.value.equals("|") ){
            RemoveRestrictedActions (node.left);
            RemoveRestrictedActions (node.right);
           }else if(node.p!=null &&
                                node.restricted_actions!=null){
            String Process=node.p.name;
            for(int i=0;i<node.restricted_actions.size();i++){
                String raction=ανάκτηση δεσμευμένης ενέργειας 'i'από
το πίνακα restricted_actions του συγκεκριμένου κόμβου node;
                for(int j=0;j<actions.size();j++){
                      Individual item= ανάκτηση 'j' στοιχείου από το
                 action πίνακα;
                    if(raction equals item.nameOfAction)){

if( item.nameOfProcess.contains(Process))
 {
     deleteProcess(item.nameOfProcess,Process);
  }
  }
}
```

Ο τρόπος που λειτουργεί η πιο πάνω σύναρτηση είναι παρόμοιος με τις πιο πάνω αναδρομικές συναρτήσεις. Το κοινό τους σημείο είναι η προσπέλαση του συντακτικού δέντρου αναδρομικά. Εδώ καλείται αναδρομικά η συνάρτηση μέχρι ως ότου φθάσει στο κόμβο που αντιπροσωπεύει μια διεργασία «P:[s,l]», τότε γίνεται έλεγχος αν η λίστα του κόμβου που περιέχει τις δεσμευμένες ενέργειες δεν είναι άδεια να προχωρήσει με το να παίρνουμε μια-μια τις δεσμευμένες ενέργειες και τις ελέγχουμε αν είναι ίσες με αυτές του πίνακα action. Αν όντως είναι ίσες τότε διαγράφουμε τη διεργασία που εκτελεί αυτή την ενέργεια, διαγράφεται βασικά ολόκληρη η γραμμή του πίνακα action.

Τέλος θα γίνει επεξήγηση των τριών τελευταίων σημαντικότερων συναρτήσεων οι οποίες χρησιμοποιούνται μετά την επιλογή της ενέργειας που θα εκτελέσει το σύστημα. Θα δούμε δηλαδή πως επηρεάζεται το συντακτικό δέντρο μετά την εφαρμογή της ενέργειας και τις τροποποιήσεις που πρέπει να κάνουμε ανάλογα με την ενέργεια που επιλέγει.

Πρώτα θα δούμε τη συνάρτηση για την εκτέλεση της αναπαραγωγής ενός ατόμου.

```
    private  static Node Reproduction(Node root,String Person){
        Node node=root;
        if (node != null) {
            if (node.value.equals("|")) {
            node.left=Reproduction(node.left,Person,flag);
            node.left.previous=node;
            node.right=Reproduction(node.right,Person,flag);
            node.right.previous=node;
            }else if(node.p!=null ){
                if(node.p.name.equals(Person)){
           Node offspring=new Node();
                Initialize characteristics of offspring
           Node operatorNode=new Node();
                operatorNode.value='|';
           PutOffspringInSyntaxTree(node,operatorNode,offspring);
}}}
```

Το σημαντικό αυτής της συνάρτησης είναι η δημιουργία ενός νέου κόμβου που αντιπροσωπεύει το νέο άτομο που αναπαρήχθην με τα χαρακτηριστικά της ίδιας τοποθεσίας με του 'πατέρα' του, να έχει ορισμό αυτό του είδους(species) του πατέρα του. Εκτός από την δημιουργία αυτού του κόμβου πρέπει να τοποθετηθεί στο συντακτικό δέντρο. Για αυτό και η συνάρτηση PutOffspringInSyntaxTree(node,operatorNode,offspring); τοποθετεί στο κόμβο που βρίσκεται η διεργασία που εκτέλεσε την ενέργεια της αναπαραγωγής ένα νέο κόμβο που αντιπροσωπεύει το τελεστή ταυτοχρονισμού '|'. Στο αριστερό του κόμβο τοποθετείται η διεργασία που εκτέλεσε την ενέργεια της αναπαραγωγής και στα δεξιά τον απόγονο. Και έτσι ο απόγονος μπορεί να θεωρηθεί ως ένα αυθαίρετο άτομο όπως τα υπόλοιπα.

Μετά από την συνάρτηση της αναπαραγωγής, θα δούμε τη συνάρτηση της μετακίνησης.

```
private   static   Node   MoveAction(Node   root,String   Person,int
movement){
Node node=root;
int choice=0;
        if (node != null) {
```

```
            if (node.value.equals("|")) {
            node.left=MoveAction(node.left,Person,movement,flag);
                        node.left.previous=node;
            node.right=MoveAction(node.right,Person,movement,flag);
                        node.right.previous=node;
            }else if(node.p!=null ){
                if(node.p.name.equals(Person)){
      choice=movement;

                        switch(choice){
                            case 0: //panw
                                node.x=node.x-1;
            checkEdgePosition();
                            return       node.changePosition(node.x,
node.y);
                            case 1://aristera

                 node.y=node.y-1;
                            checkEdgePosition();

                             return      node.changePosition(node.x,
node.y);

                            case 2://katw

                                 node.x=node.x+1;
                  checkEdgePosition();

                             return      node.changePosition(node.x,
node.y);

                            case 3://deksia

                node.y=node.y+1;
                            checkEdgePosition();
                 return node.changePosition(node.x, node.y);

        }
return node;

}
```

Το ίδιο και σε αυτή την σύναρτηση γίνεται αναδρομή μέχρι να φθάσει στο κόμβο που αντιπροσωπεύει μια διεργασία.Το σημαντικό αυτής της συνάρτησης είναι οι 4 περιπτώσεις μετακίνησης του ατόμου(πάνω, κάτω, δεξιά, αριστερά). Ανάλογα με τη κίνηση αλλάζουν και οι συντεταγμένες του ατόμου. Για παράδειγμα αν πάει πάνω τότε μειώνεται κατά 1 ο δείκτης της γραμμής και της στήλης μένει ίδια. Σε κάθε περίπτωση γίνεται έλεγχος αν είναι στα άκρα του πίνακα, ούτως ώστε όταν γίνει η μετακίνηση να εφαρμοστεί η κυκλική ιδιότητα και να αλλάξουν οι συντεταγμένες ανάλογα.

Τελευταία συνάρτηση θα δούμε τη συνάρτηση που διαγράφεται ο κόμβος που αντιπροσωπεύει μια ενέργεια.

```
private static Node DeleteActions(String action, Node root, Node nodeChoice, String nameOfProcess)
```

Επίσης η συνάρτησης αυτή είναι αναδρομική. Γίνεται κλήση της συνάρτησης μέχρι να φθάσει στο κόμβο που αντιπροσωπεύει τον τελεστή '+ ' αν υπάρχει και τότε διαγράφεται το αριστερό ή το δεξί υποκλάδι, ανάλογα με τη παράμετρο (Node nodeChoice) που έχει η συνάρτηση. Μετά ξανακαλείται η συνάρτηση μέχρι να φθάσει στο κόμβο που αντιπροσωπεύει τον τελεστή διαδοχής '.' Και τότε διαγράφεται ο αριστερό τους κόμβος και αυτού παίρνει τη θέση του ο δεξί κόμβος.

Όλες οι πιο πάνω συναρτήσεις βρίσκονται σε ένα βρόγχο επανάληψης μέχρι όλα τα άτομα πεθάνουν στο πληθυσμό ή όταν ο χρήστης τερματίσει την προσομοίωση.

5.3 Αποτελέσματα Συστήματος

Σε αυτό το υποκεφάλαιο θα τρέξουμε ένα παράδειγμα πληθυσμιακού συστήματος για να δούμε την αποτελεσματικότητα του συστήματος καθώς και για την εξαγωγή κάποιων συμπερασμάτων. Τη σύστημα που δίνομαι για είσοδο αποτελείται από 3 άτομα (P, P1, P2) τα οποία συμβιώνουν σε ένα κόσμο 10×10. Ο ορισμός των διεργασιών αυτών παρουσιάζεται πιο κάτω καθώς και το είδος του πληθυσμού κάθε διεργασίας (s, s1):

P=move.a.0+a.'b.'a.0

P1= a.rep.P1

P2=move.'b.move.0

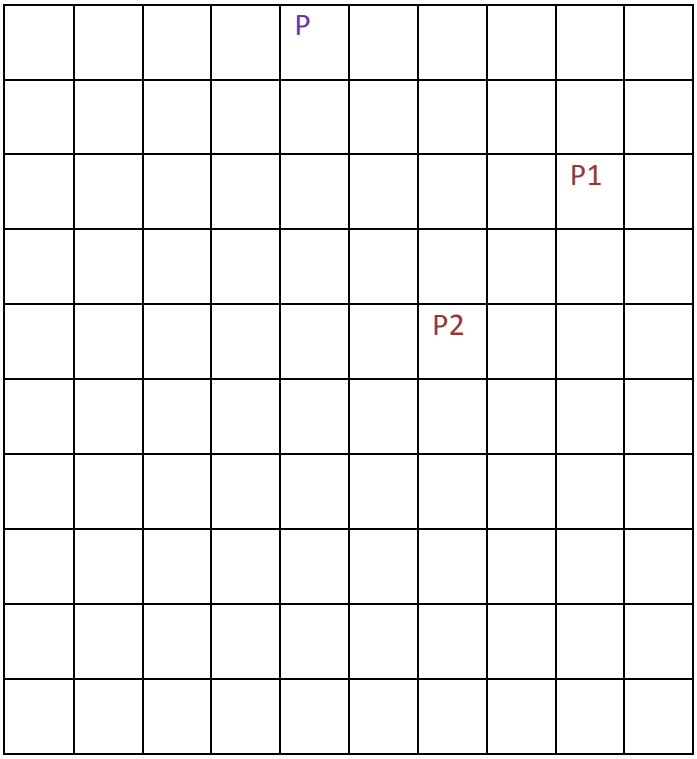
s=move.0

s1=move.b.0

Το πληθυσμιακό σύστημα ορίζεται ως εξής:

System=((P:[s,5])%{a,c}|(P1:[s1,39])|(P2:[s1,57]))%{c}

Ο κόσμος στον οποίο ζει ένας πληθυσμός συμβολίζεται ως ένα πλέγμα 10×10.



Βλέπουμε ότι η αρχική τοποθεσία κάθε διεργασίας ορίζεται στον ορισμό του συστήματος. Για παράδειγμα η αρχική τοποθεσία της P είναι η θέση 5, για την P1 είναι η 39 και για την P2 είναι η 57.

Αφού η αρχικοποίηση του πληθυσμού έγινε, θα δούμε τις ενέργειες που μπορεί να εκτελέσει το σύστημα.

Βλέπουμε ότι οι ενέργειες που μπορεί να εκτελέσει το σύστημα είναι οι (α, move, move).

ACTIONS----------

1 action

(P1:[s1,3]|P2:[s1,7]|P:[s,5]%{a,c})%{c} -- a --> (P1:[s1,3]|P2:[s1,7]|P:[s,5]%{a,c})%{c}

P1=rep.0

2 action

(P1:[s1,3]|P2:[s1,7]|P:[s,5]%{a,c})%{c} -- move --> (P1:[s1,3]|P2:[s1,8]|P:[s,5]%{a,c})%{c}

P2='b.move.0

3 action

(P1:[s1,3]|P2:[s1,7]|P:[s,5]%{a,c})%{c} -- move --> (P1:[s1,3]|P2:[s1,7]|P:[s,95]%{a,c})%{c}

P=a.0

Επιλέγεται η πρώτη ενέργεια και έχει ως αποτέλεσμα να επηρεάζεται μόνο η διεργασία P1.

(P1:[s1,3]|P2:[s1,7]|P:[s,5]%{a,c})%{c} -- a --> (**P1:[s1,3]**|P2:[s1,7]|P:[s,5]%{a,c})%{c}

P1=rep.P1

Και στη συνέχεια το σύστημα μπορεί να εκτελέσει τις πιο κάτω ενέργειες (rep, move, move):

ACTIONS----------

- 1 action

(P1:[s1,3]|P2:[s1,7]|P:[s,5]%{a,c})%{c}--rep-->

(P1:[s1,3]|P3:[s1,3]|P2:[s1,7]|P:[s,5]%{a,c})%{c}

P1= a.rep.P1

2 action

(P1:[s1,3]|P2:[s1,7]|P:[s,5]%{a,c})%{c} -- move --> (P1:[s1,3]|P2:[s1,8]|P:[s,5]%{a,c})%{c}

P2='b.move.0

3 action

(P1:[s1,3]|P2:[s1,7]|P:[s,5]%{a,c})%{c} -- move --> (P1:[s1,3]|P2:[s1,7]|P:[s,6]%{a,c})%{c}

P=a.0

Επιλέγεται η τρίτη ενέργεια και έχει ως αποτέλεσμα να επηρεάζεται μόνο η διεργασία P και να μην εμφανίζεται μετά στο σύστημα για το λόγο η επόμενη ενέργεια που μπορεί να εκτελέσει είναι η «α» η οποία είναι δεσμευμένη ενέργεια για τη διεργασία αυτή.

(P1:[s1,3]|P2:[s1,8]|P:[s,5]%{a,c})%{c}--move--> (P1:[s1,3]|P2:[s1,8]|**P:[s,4]**%{a,c})%{c}

P=a.0

Το σύστημα μπορεί να εκτελέσει τις πιο κάτω ενέργειες:

ACTIONS----------

1 action

(P1:[s1,3]|P2:[s1,7]|P:[s,6]%{a,c})%{c}--rep-->

(P1:[s1,3]|P3:[s1,3]|P2:[s1,7]|P:[s,6]%{a,c})%{c}

P1=a.rep.P1

2 action

(P1:[s1,3]|P2:[s1,7]|P:[s,6]%{a,c})%{c} -- move --> (P1:[s1,3]|P2:[s1,17]|P:[s,6]%{a,c})%{c}

P2='b.move.0

Εκτελείται η πρώτη ενέργεια και βλέπουμε πως δημιουργείται ακόμη ένα άτομο στο πληθυσμό από τη διεργασία P1 το P3, η οποία εκτέλεσε την ενέργεια της αναπαραγωγής και στην συνέχεια η διεργασία αυτή επειδή είναι αναδρομική έχει τον αρχικό της ορισμό.

Το P3 έχει διαφορετική συμπεριφορά από τη διεργασία την οποία το δημιούργησε.

Το σύστημα συνεχίζει την προσομοίωση του μέχρι όλα τα άτομα στο πληθυσμό φθάσουν σε μια αδρανές κατάσταση. Μπορεί ακόμη να μην σταματήσει ποτέ την προσομοίωση γιατί κάποιες διεργασίες είναι αναδρομικές. Σε αυτή την περίπτωση δεν θα σταματήσει η προσομοίωση γιατί η διεργασία P1 είναι αναδρομική.

Κεφάλαιο 6

Συμπεράσματα

6.1 Συμπεράσματα

Βασικός στόχος γύρω από την μελέτη των πληθυσμιακών συστημάτων είναι η κατανόηση του τρόπου που ζουν διάφοροι πληθυσμοί. Ο τρόπος που ζει ένας πληθυσμός εξάγει πολλά συμπεράσματα για την μελλοντική κατάσταση του πληθυσμού όπως κατά πόσο αυξήθηκε ή μειώθηκε ένας πληθυσμός μετά από κάποιο χρονικό διάστημα. Σε κλάδους όπου μελετάται η πληθυσμιακή οικολογία μπορούν να δώσουν απαντήσεις σε ερωτήματα τύπου πόσο καιρό θα καταφέρει ο πληθυσμός να αυξηθεί κατά κάποιο ποσοστό ή ποιοι είναι οι λόγοι εξάλειψης του πληθυσμού. Για να μπορέσουν όμως να δοθούν αυτές τις απαντήσεις απαιτείται η χρήση τεχνικών μοντελοποίησης μέσω προτύπων όπως είναι οι άλγεβρες διεργασιών. Το λογισμικό που υλοποίησα κάνει χρήση τεχνικών μοντελοποίησης μέσω της άλγεβρας διεργασίας PALPS.

Στην εργασία μου μελέτησα μια νέα άλγεβρα διεργασιών PALPS, η οποία είναι επέκταση της CCS που αποτελεί ένα αξιόλογο τρόπο μοντελοποίησης πληθυσμών για την διεξαγωγή πειραμάτων που έχουν σχέση με πληθυσμιακά συστήματα. Η PALPS χρησιμοποιεί κάποιους από τους τελεστές της CCS, προσαρμόζοντας σε αυτούς τους κανόνες την έννοια του χώρου, επιτυγχάνοντας έτσι την μοντελοποίηση των ενεργειών της μετακίνησης, της αναπαραγωγής, της αρπαγής. Μέσω αυτής της μοντελοποίησης μπορούν να απαντηθούν ερωτήματα μετά από πόσο χρόνο ή μετά από πόσες ενέργειες μπορεί να προκύψει αύξηση ή μείωση του πληθυσμού; Το πιο πάνω ερώτημα είναι και η σημαντικότητα των αποτελεσμάτων του προγράμματος μου κατά την υλοποίηση ενός υποσυνόλου κανόνων της σημασιολογίας της PALPS συμπεριλαμβανομένων και τις ενέργειες της μετακίνησης και

της αναπαραγωγής . Η παραγωγή κάποιων πιθανών καταστάσεων του συστήματος που δημιουργούνται μετά την εκτέλεση κάποιων ενεργειών μπορούμε να δούμε αν ένας πληθυσμός έχει επιβιώσει ή πεθάνει.

Συνοψίζοντας, κατέληξα στο συμπέρασμα ότι οι άλγεβρες διεργασιών έχουν την ικανότητα να επιτρέπουν την προδιαγραφή των μεμονωμένων συνιστωσών ενός συστήματος και να παρέχουν την δυνατότητα κατανόησης του πώς οι συνιστώσες αυτές αλληλεπιδρούν μεταξύ τους συμβάλλοντας στην γενική συμπεριφορά του συστήματος. Εξάγουν έτσι συμπεράσματα για ένα σύνθετο σύστημα από τις ιδιότητες των επιμέρους συνιστωσών του, κάτι πολύ σημαντικό για το πεδίο των πληθυσμιακών συστημάτων.

Σε θέματα εκπαίδευσης θα μπορούσε να χρησιμοποιηθεί κατά την εκμάθηση μοντελοποίηση των πληθυσμιακών συστημάτων στην άλγεβρα διεργασιών PALPS καθώς θα δίνει την δυνατότητα στο εκπαιδευόμενο να παρακολουθήσει και να μελετήσει το μοντέλο που έχει δημιουργήσει εύκολα με την χρήση του λογισμικού, και να εξετάσει την ορθότητα του μοντέλου που έχει δημιουργήσει.

Θα μπορούσε επίσης να χρησιμοποιηθεί τόσο για την προσομοίωση πληθυσμιακών και βιολογικών συστημάτων σε ερευνητικά εργαστήρια βιολογίας. Θα ήταν χρήσιμο σε ένα ερευνητικό εργαστήρι καθώς δίνει την δυνατότητα να παρακολουθεί κανείς την κατάσταση στην οποία βρίσκεται το πληθυσμιακό σύστημα ανά πάσα στιγμή και να προσομοιώσει αντιδράσεις επιλέγοντάς τις ή αφήνοντας το λογισμικό να επιλέξει τυχαία.

6.2 Μελλοντικές Επεκτάσεις

Τη δεδομένη στιγμή το λογισμικό προσομοιώνει τυχαίες αντιδράσεις σε ένα πληθυσμιακό σύστημα, ή αντιδράσεις που επιλέγει ο χρήστης. Αυτές οι τυχαίες επιλογές αντιδράσεων γίνονται χωρίς πιθανότητες ή την τιμή κάποιων βαρών μέσα από μια λίστα από πιθανές αντιδράσεις. Στο μέλλον είναι δυνατόν να προστεθούν βάρη που να επηρεάζουν την επιλογή των τυχαίων αντιδράσεων ή έννοιες όπως είναι ο χρόνος, η προτεραιότητα και η πιθανότητα, φέρνοντας έτσι την προσομοίωση πιο κοντά στα πραγματικά δεδομένα, καθώς ξέρουμε ότι κάποιες ενέργειες είναι πιο πιθανόν να συμβούν από κάποιες άλλες. Παρόμοια θα μπορούσαν να ενταχθούν έννοιες του φαγητού, του επίπεδου δυσκολίας της

κάθε ενέργειας, της ηλικίας των ατόμων ή της δύναμης που έχει κάθε άτομο ούτως ώστε να μπορεί να πραγματοποιεί τις διάφορες ενέργειες που απαιτούνται για την επιβίωση του, όπως η συλλογή τροφής και φύλαξη της φωλιάς του.

Επιπλέον, επέκταση της παρούσας εργασίας θα μπορούσε να ήταν η υλοποίηση περισσότερων ενεργειών εκτός από τις ενέργειες της μετακίνησης και της αναπαραγωγής όπως για παράδειγμα ενέργειες όπως είναι η αρπαγή, η μόλυνση ή και του θανάτου. Επιπρόσθετα σε συνδυασμό με την έννοια του χώρου, θα μπορούσε να ενταχθεί σε κάθε τοποθεσία η έννοια του φαγητού ή το επίπεδο δυσκολίας της κάθε ενέργειας που πραγματοποιείται από ένα άτομο. Για παράδειγμα η ενέργεια της μετακίνησης μπορεί να χρειάζεται λιγότερη ενέργεια από ένα άτομο να πραγματοποιηθεί παρά η ενέργεια της αναπαραγωγής.

Τέλος, κατά την υλοποίηση του συστήματος, στο κόσμο που επιβιώνει ένας πληθυσμός είναι ένα πλέγμα 10×10, θα μπορούσαμε με δυναμικό τρόπο να επεκτείνουμε τον κόσμο αυτό όταν τα άτομα στο πληθυσμό αυξάνονται και δεν έχουν αρκετό χώρο να κινηθούν. Ακόμη θα ήταν καλό η εύρεση μεθόδων οι οποίες θα βοηθούν στη χρήση λιγότερου χώρου μνήμης κατά την αποθήκευση πληροφοριών και έτσι θα έχει ως αποτέλεσμα την μελέτη πιο μεγάλων συστημάτων με περισσότερα άτομα. Επιπρόσθετα, το λογισμικό που ανάπτυξα κάνει χρήση κάποιων γραφικών, καλό θα ήταν η εισαγωγή περισσότερων φορμών ή ακόμη και καλύτερη αναπαράσταση του χώρου που επιβιώνει ένας πληθυσμός μέσω κάποιων 3-D εικονιδίων που θα απεικονίζει ένα άτομο.

Βιβλιογραφία

[1] Α. Φιλίππου, Σημειώσεις του μαθήματος ΕΠΛ664 Ανάλυση και Επαλήθευση Συστημάτων, Πανεπιστήμιο Κύπρου, 2009.

[2] R. Milner, Communication and Concurrency, Prentice Hall, December 1989.

[3] X. Efthmiou and A. Philippou, A Process Calculus for Spatially-Explicit Ecological Systems. In P. Milazzo and M. de J. Perez Jimenez, editors, Proceedings of the 1st International Workshop on Application of Membrane Computing, Concurrency and Agent-based Modelling in Population Biology (AMCA-POP 2010), p. 84-89.

[4] Π. Ευθυμίου, Πληθυσμιακά Συστήματα και Άλγεβρες Διεργασιών, Διπλωματική Εργασία, Πανεπιστήμιο Κύπρου, Μάιος 2010.

[5] J.C.M. Baeten & W.P. Weijland, Process algebra, Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press 1990.

[6] R.J. van Glabbeek, Notes on the methodology of CCS and CSP, Report CS-R8624, Centre for Mathematics and Computer Science, Amsterdam 1986; Theoretical Computer Science 177(2), 1997, p. 329-349.

[7] R. Milner, Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, editor, Handbook of Theoretical Computer Science, chapter 19, p. 1201–1242.

[8] W.J. Fokkink, Introduction to Process Algebra, Texts in Theoretical Computer Science, An EATCS Series, Springer (January 2000).

[9] S. Kulick, Process algebra, CCS, and bisimulation decidability, Technical Report 94-06, Institute for Research in Cognitive Science, 1994.