Ατομική Διπλωματική Εργασία

ΠΛΗΘΥΣΜΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΑΛΓΕΒΡΕΣ ΔΙΕΡΓΑΣΙΩΝ

Πολυξένη Ευθυμίου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2010

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΛΗΘΥΣΜΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΑΛΓΕΒΡΕΣ ΔΙΕΡΓΑΣΙΩΝ

Πολυξένη Ευθυμίου

Επιβλέπουσα Καθηγήτρια

κα. Άννα Φιλίππου

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

Μάιος 2010

Ευχαριστίες

Ένα μεγάλο ευχαριστώ χρωστώ στην επιβλέπουσα καθηγήτριά μου κα Άννα Φιλίππου, η οποία έπαιξε πολύ σημαντικό ρόλο στην διεκπεραίωση της διπλωματικής μου εργασίας. Με τις πολύτιμες συμβουλές της και την άψογη καθοδήγησή της, κατάφερα να ολοκληρώσω την παρούσα διπλωματική εργασία. Την ευχαριστώ για την εμπιστοσύνη που έδειξε προς το πρόσωπό μου και την βοήθεια και στήριξη που μου παρείχε σε περιπτώσεις που τις χρειαζόμουν.

Περίληψη

Η παρούσα εργασία ασχολείται με το θέμα ‘Συστήματα Πληθυσμών και Άλγεβρες Διεργασιών’ και είχε σαν γενικότερο στόχο της, την χρήση ενός προτύπου άλγεβρας διεργασιών για την μελέτη και ανάλυση πληθυσμών καθώς και την αξιολόγηση του συγκεκριμένου προτύπου για μελέτη εφαρμογών από το πεδίο των πληθυσμιακών συστημάτων.

Στόχος της μελέτης που πραγματοποιείται γύρω από πληθυσμιακά συστήματα είναι η κατανόηση της συμπεριφοράς των πληθυσμών μέσω των σημαντικότερων συστατικών τους όπως πόροι, εχθροί, περιβάλλον. Για την σωστή κατανόηση τους απαιτείται η ύπαρξη κάποιου προτύπου το οποίο θα μπορεί να προβλέψει τις πιθανές μελλοντικές καταστάσεις του συστήματος, αφού πρώτα ρυθμιστεί σωστά με παραμέτρους και χαρακτηριστικά που μπορούν να ληφθούν μέσω της παρατήρησης του πληθυσμού. Η εργασία μου επικεντρώνεται στις άλγεβρες διεργασιών, πρότυπα τα οποία έχουν την δυνατότητα μοντελοποίησης και ανάλυσης πληθυσμιακών συστημάτων. Σημαντικοί σταθμοί της μελέτης των αλγεβρών διεργασιών, σε σχέση με την εργασία μου αποτελούν η γλώσσα CCS, η επέκτασή της WSCCS η οποία χρησιμοποιήθηκε για την ανάλυση πληθυσμών μυρμηγκιών, καθώς επίσης και η TPCCS.

Για την ολοκλήρωση της εργασίας μου με βάση την γλώσσα WSCCS, δημιούργησα μια νέα άλγεβρα διεργασιών την PALPS (Process Algebra with Location for Population Systems) και υλοποιώντας συγκεκριμένο σύστημα κατέληξα στο συμπέρασμα ότι μέσω των αλγεβρών διεργασιών μπορώ να προβλέψω τις πιθανές καταστάσεις του χώρου στον οποίο επιβιώνουν τα άτομα ενός πληθυσμού, με συγκεκριμένες πιθανότητες. Παρουσιάζοντας παραδείγματα ενεργειών και συμπεριφορών ενός πληθυσμού μέσω της γλώσσας που δημιούργησα, κατάληξα στο ότι η συμπεριφορά ενός πληθυσμού μπορεί με επιτυχία να απεικονιστεί με την επιλογή του κατάλληλου προτύπου όπως είναι οι άλγεβρες διεργασιών.

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 βιότοπό 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 διαφόρων 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 εξεταστούν.

Το κύριο πρόβλημα με το οποίο καταπιάνεται η οικολογία πληθυσμών είναι πώς 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 πληθυσμών, ασχολούνται με τον τρόπο συνύπαρξης πληθυσμών δύο ή περισσότερων ειδών. Για σκοπούς μιας τέτοιας μελέτης απαιτείται η απεικόνιση του κάθε ατόμου του πληθυσμού από την γέννηση του, την αύξησή του, την αναπαραγωγή του, μέχρι και τον θάνατο. Αν και μια τέτοια

προσπάθεια είναι αρκετά ακριβή, επιτρέπει στους επιστήμονες της πληθυσμιακής οικολογίας να εξερευνήσουν και να ερμηνεύσουν το τρόπο που η δυναμική ενός πληθυσμού ή ενός οικοσυστήματος προκύπτει μέσω της αλληλεπίδρασης των ατόμων με το περιβάλλον τους και μέσω της αλληλεπίδρασης τους με τα υπόλοιπα άτομα με τα οποία συνυπάρχουν στον βιότοπο που ζουν.

1.2 Σκοπός της Διπλωματικής Εργασίας

Πρωταρχικός στόχος της παρούσας διπλωματικής εργασίας είναι η μελέτη των τεχνικών μοντελοποίησης και ανάλυσης πληθυσμών, που θα μπορούσαν να εφαρμοστούν στον κλάδο της πληθυσμιακής οικολογίας, 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 οι οποίοι επιτρέπουν την αλληλεπίδραση του ενός ατόμου με τα υπόλοιπα του πληθυσμού, καθώς επίσης και με το περιβάλλον του.

Η εργασία μου επικεντρώνεται στις άλγεβρες διεργασιών που αποτελούν πρότυπα με την ικανότητα της μοντελοποίησης και ανάλυσης πληθυσμιακών συστημάτων. Γενικότερος στόχος της εργασίας, ήταν η χρήση ενός προτύπου άλγεβρας διεργασιών για την μελέτη και ανάλυση πληθυσμών, καθώς και την αξιολόγηση του συγκεκριμένου προτύπου για μελέτη εφαρμογών από το πεδίο των πληθυσμιακών συστημάτων. Ίσως τα σημαντικότερα μοντέλα που προτάθηκαν για την περιγραφή πληθυσμιακών συστημάτων πριν από τις άλγεβρες διεργασιών, να είναι τα Petri Nets [1].

Για τις επιστήμες της βιολογίας και οικολογίας, οι τεχνικές μοντελοποίησης και ανάλυσης ενός ή περισσότερων πληθυσμών που συνυπάρχουν σε ένα περιβάλλον, αποτελούν ένα σημαντικό σημείο. 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 επιβίωσής τους και γενικά για τον τρόπο που ζουν μέσα σε ένα περιβάλλον.

Ο χώρος μέσα στον οποίο ένας πληθυσμός επιβιώνει παίζει καθοριστικό ρόλο στο τρόπο με τον οποίο αυξάνεται ή εξαλείφεται ένας πληθυσμός, αλλά και στον τρόπο που συμπεριφέρεται γενικότερα. Στόχος μου, ήταν η δημιουργία μια νέας άλγεβρας διεργασιών με την έννοια της τοποθεσίας, η οποία θα μου επέτρεπε να παρουσιάσω την σημαντικότητα της έννοιας του χώρου στον τρόπο που ένας πληθυσμός συμπεριφέρεται. Για παράδειγμα ένας πληθυσμός μπορεί να παρουσιάζει την ιδιότητα ότι για να αναπαράγεται χρειάζεται να βρίσκεται μόνος του σε μια τοποθεσία. Επίσης για την μόλυνση ατόμων ενός πληθυσμού από τα άτομα ενός άλλου πληθυσμού, η έννοια του χώρου αποτελεί ένα σημαντικό σημείο. Για να πραγματοποιηθεί η μόλυνση σε μια τοποθεσία, απαιτείται η ύπαρξη ενός μολυσματικού και ενός υγιούς τουλάχιστον ατόμου στην τοποθεσία αυτή. Το ίδιο συμβαίνει και κατά την συνύπαρξη δυο πληθυσμών που ο ένας επιβιώνει εις βάρος του άλλου. Για παράδειγμα εάν ένας πληθυσμός έχει την ιδιότητα της αρπαγής, τότε για να μπορέσει να αρπάξει και να καταναλώσει ένα άλλο άτομο, απαιτείται να υπάρχει στην τοποθεσία που βρίσκεται ένα τουλάχιστο θήραμα. Γενικεύοντας το τελευταίο σημείο ένας άλλος σημαντικός παράγοντας στην επιβίωση ενός πληθυσμού είναι η ύπαρξη φαγητού στον χώρο που επιβιώνει ο πληθυσμός. Έλλειψη φαγητού από τον χώρο επιβίωσης ενός πληθυσμού ίσως να έχει σαν αποτέλεσμα την εξάλειψη του πληθυσμού στην συγκεκριμένη τοποθεσία σε γρήγορους ρυθμούς. Όλα αυτά αποτελούν σημαντικά κίνητρα τα οποία μπορούν να ωθήσουν κάποιον στην μελέτη της συμπεριφοράς ενός πληθυσμού σε σχέση με την τοποθεσία στην οποία βρίσκονται τα άτομά του, εξάγοντας γενικά συμπεράσματα για τον τρόπο συμπεριφοράς και επιβίωσης του.

1.3 Δομή της Διπλωματικής Εργασίας

Στόχος του πρώτου μέρους της διπλωματικής μου εργασίας ήταν η εξοικείωση μου με τα πληθυσμιακά συστήματα και τις άλγεβρες διεργασιών. Μελέτες που έχουν ήδη πραγματοποιηθεί γύρω από το θέμα των αλγεβρών διεργασιών και των πληθυσμιακών συστημάτων παρουσιάζονται στο Κεφάλαιο 2. Συγκεκριμένα μελέτησα τις γλώσσες

CCS και WSCCS [7,3,4,5]. Στη συνέχεια μελέτησα κάποια άρθρα που αναφέρονταν κυρίως στον τρόπο μοντελοποίησης του πληθυσμού των μυρμηγκιών μέσω των αλγεβρών διεργασιών και κυρίως με την βοήθεια της WSCCS [6,3,4,5,8]. Ασχολήθηκα επίσης με την δημιουργία της άλγεβρας διεργασιών PALPS (Process Algebra with Location for Population Systems), εντάσσοντας σε αυτή την έννοια του χώρου. Η γλώσσα αυτή παρουσιάζεται στο Κεφάλαιο 3 μαζί με παραδείγματα. Στο Κεφάλαιο 4 ακολουθεί η περιγραφή του συστήματος που υλοποίησα στην γλώσσα προγραμματισμού C καθώς επίσης παρουσιάζονται και τα διάφορα πειράματα και αποτελέσματα. Στο πέμπτο και τελευταίο κεφάλαιο, γίνεται αξιολόγηση της γλώσσας που δημιούργησα για κατανόηση της συμπεριφοράς ενός πληθυσμού και παρουσιάζονται τα γενικά συμπεράσματα της εργασίας μου μαζί με μελλοντικές επεκτάσεις που θα μπορούσαν να πραγματοποιηθούν.

Κεφάλαιο 2

Πληθυσμιακά Συστήματα και Άλγεβρες Διεργασιών

2.1 Πληθυσμιακά Συστήματα

Καταρχήν ο όρος πληθυσμός αναφέρεται σε 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 συνύπαρξής τους με άλλους πληθυσμούς, είτε λόγω κάποιου ενδογενούς παράγοντα.

Παράδειγμα ενός ενδογενούς παράγοντα είναι η κακή συνεργασία μεταξύ των ατόμων ενός πληθυσμού στον καταμερισμό τους στις διάφορες εργασίες για επιβίωσή τους, όπως το μάζεμα της τροφής ή την φύλαξη της φωλιάς τους από πληθυσμούς που αποτελούν εχθρούς τους. Μειωμένα αποθέματα τροφών στις φωλιές τους κατά τις περιόδους που δεν μπορεί λόγο καιρικών συνθηκών να πραγματοποιηθεί συλλογή τροφής, μπορεί να επιφέρει τον θάνατο σε άτομα του πληθυσμού. Αλλαγές στον ρυθμό γεννήσεων και θανάτων ενός πληθυσμού, μπορεί να προκληθούν και από εξωγενείς παράγοντες όπως είναι οι ακραίες καιρικές συνθήκες ή άλλοι πληθυσμοί οι οποίοι

μπορεί να τρέφονται εις βάρος του, ή να έχουν την ικανότητα να τον μολύνουν, κάτι το οποίο μπορεί να επιταχύνει την εξάλειψή του, λόγο αύξησης των θανάτων και μείωσης του ρυθμού αναπαραγωγής. Αυτές οι μεταβολές μελετώνται από τον κλάδο της οικολογίας που ονομάζεται Δυναμική Πληθυσμών.

Το 1967 ο Chark et al. εισήγαγε για πρώτη φορά τον όρο «Life System», ενώ το 1981 προτάθηκε από τον Berryman ο όρος «Population System» (Πληθυσμιακό Σύστημα) [1]. Οι δύο αυτοί όροι ταυτίστηκαν και με την χρήση τους γίνεται αναφορά σε οποιοδήποτε πληθυσμό, μαζί με το περιβάλλον στο οποίο είναι ενεργός. Τα σημαντικότερα συστατικά ενός συστήματος πληθυσμού είναι [1]:

1. Ο ίδιος ο πληθυσμός: Οι οργανισμοί ενός πληθυσμού μπορεί να υποδιαιρούνται σε ομάδες σύμφωνα με διάφορα χαρακτηριστικά τους, όπως η ηλικία, το στάδιο ανάπτυξης στο οποίο βρίσκονται, το φύλο τους. Για παράδειγμα σύμφωνα με την ηλικία ο πληθυσμός των ανθρώπων μπορεί να διαχωριστεί σε τρεις ομάδες: τα παιδιά, τους έφηβους και τους ενήλικες.

2. Πόροι: Μπορεί να θεωρηθεί σαν πόρος οτιδήποτε βοηθά ένα πληθυσμό να επιβιώσει, όπως είναι η τροφή και ο χώρος διαμονής του (φωλιά, σπίτι, καταφύγιο).

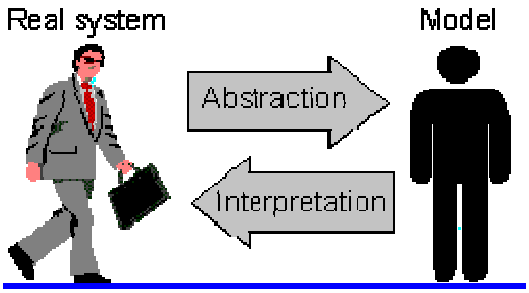
3. Εχθροί: Οτιδήποτε απειλεί ένα πληθυσμό. Για παράδειγμα για ένα πληθυσμό ζώων, εχθρός μπορεί να θεωρηθεί ένας άλλος αρπακτικός πληθυσμός ζώων, ενώ για ένα πληθυσμό μικροοργανισμών, πληθυσμοί με την ιδιότητα του παρασιτισμού εις βάρος τους θεωρούνται εχθροί του.

4. Περιβάλλον: Αναφέρεται κυρίως στην θερμοκρασία του αέρα, του νερού, του χώματος, μέσα στο οποίο επιβιώνει ο πληθυσμός, καθώς επίσης και στον τρόπο που αυτή η θερμοκρασία μεταβάλλεται με τον χρόνο.

2.2 Μοντελοποίηση Πληθυσμιακών Συστημάτων

Πολλοί επιστήμονες ασχολήθηκαν με τη μελέτη πληθυσμιακών συστημάτων. Στόχος τους ήταν η κατανόηση της συμπεριφοράς των πληθυσμών μέσω των σημαντικότερων συστατικών τους όπως αυτά αναφέρθηκαν στο προηγούμενο υποκεφάλαιο. Υπάρχουν πολλοί και διάφοροι τρόποι μοντελοποίησης και ανάλυσης πληθυσμών.

Η επιτυχία της μοντελοποίησης ενός πληθυσμού εξαρτάται άμεσα από τη επιλογή του μοντέλου που θα χρησιμοποιηθεί. Ένα μοντέλο συνδέεται με την πραγματικότητα μέσω της διαδικασία της αφαιρετικότητας (abstraction) και της ερμηνείας (interpretation) [1].



Σχήμα 2.1: Συσχέτιση Πραγματικού Συστήματος με Πρότυπο [1]

Η αφαιρετικότητα αναφέρεται στην λήψη των σημαντικότερων συστατικών ενός πραγματικού συστήματος. Αποτελεί κάποιου είδους γενίκευσης του πραγματικού συστήματος. Όσο αφορά την ερμηνεία ισχύει ότι οι διάφοροι παράμετροι του προτύπου καθώς επίσης και η συμπεριφορά του, ταυτίζονται με τα χαρακτηριστικά και την συμπεριφορά του πραγματικού συστήματος που απεικονίζει [1].

Παρόλα αυτά έχει ειπωθεί ότι: «*Models are always wrong ... but many of them are useful» (Τα μοντέλα είναι πάντοτε λανθασμένα … αλλά πολλά από αυτά είναι χρήσιμα)*[1]. Πολλές φορές η πραγματικότητα μπορεί να ανατραπεί από οποιοδήποτε παράγοντα ο οποίος δεν ήταν αναμενόμενος και ο οποίος δεν λήφθηκε υπόψη κατά την μοντελοποίηση. Αν για παράδειγμα προσπαθούμε να προβλέψουμε το ποσοστό αύξησης της πυκνότητας κάποιου πληθυσμού σε 10 χρόνια, τα αποτελέσματα μας δεν είναι απόλυτα αξιόπιστα αφού ένας εξωτερικός παράγοντας μπορεί να τα ανατρέψει. Για παράδειγμα μια απότομη αλλαγή της θερμοκρασίας μπορεί να έχει σαν αποτέλεσμα

τον θάνατο ενός μεγάλου ποσοστού του πληθυσμού. Τα πρότυπα λοιπόν δεν μπορούν να δώσουν ποτέ το 100% της πραγματικότητας αν και πολλές φορές είναι χρήσιμα στην ανάλυση και μοντελοποίηση της συμπεριφοράς ενός συστήματος.

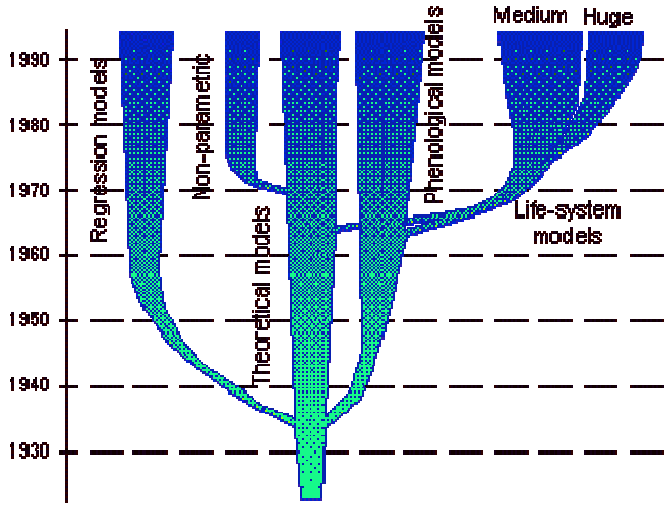
Κατά την μελέτη μου συνάντησα αρκετά μοντέλα ανάλυσης και μοντελοποίησης πληθυσμών. Κάποια από αυτά αναφέρονται πιο κάτω. Τα θεωρητικά μοντέλα (theoretical models) ξεκίνησαν από τα εκθετικά (exponential) και λογιστικά (logistic) μοντέλα (Berryman και Millstein) [2]. Τα πρότυπα αυτά είναι από τα σημαντικότερα πρότυπα που βασίζονται στην αναπαραγωγή των οργανισμών. Σημαντικότερο πλεονέκτημά τους είναι η απλότητά τους. Το εκθετικό πρότυπο ονομάζεται και 'μυθολογικό' αφού προβλέπει την αύξηση ενός πληθυσμού, υποθέτοντας ότι κάθε άτομο έχει ίσα ποσοστά αναπαραγωγής, ενώ στην πραγματικότητα το ποσοστό της αναπαραγωγής μπορεί να επηρεάζεται από παράγοντες όπως η θερμοκρασία και η ηλικία.

Για την μοντελοποίηση πληθυσμών χρησιμοποιήθηκαν επίσης τα πρότυπα οπισθοδρόμησης (regression models) [2]. Τα πρότυπα αυτά βασίζονται στην σχέση που υπάρχει μεταξύ της τιμής που προσπαθούμε να προβλέψουμε (π.χ. της πληθυσμιακής πυκνότητας) και ενός ή περισσότερων παραγόντων όπως είναι η θερμοκρασία και η ύπαρξη φαγητού. Η εγκυρότητα του συγκεκριμένου μοντέλου μπορεί να χαθεί όταν συμβούν γεγονότα τα οποία δεν ήταν αναμενόμενα, όπως για παράδειγμα η απότομη αλλαγή της θερμοκρασίας. Τα αυτοαναδρομικά πρότυπα (autoregressive) είναι πρότυπα οπισθοδρόμησης στα οποία η τιμή της εξαρτώμενης μεταβλητής, προβλέπεται από τις τιμές της, στις προηγούμενες χρονικές περιόδους ή από τις τιμές της σε γειτονικές θέσεις στον χώρο[1]. Βασισμένα στην οπισθοδρόμηση είναι και τα πιθανολογικά πρότυπα (Stochastic models) [1].

Συνεχίζοντας, τα ποιοτικά μοντέλα μπορούν να θεωρηθούν υποκατηγορία των θεωρητικών μοντέλων [2]. Αποτελούν εξολοκλήρου θεωρητικό εργαλείο και το γεγονός ότι δεν έχουν καμιά παράμετρο έχει σαν αποτέλεσμα πολλές φορές να καλούνται και μη-παραμετρικά (non-parametric) πρότυπα.

Πρότυπα τα οποία χρησιμοποιήθηκαν για την μοντελοποίηση πληθυσμών και τα οποία βασίζονται στον όρο ‘life-system’ είναι τα life-system πρότυπα τα οποία διαχωρίζονται σε μέτρια (medium) και τεράστια (huge) πρότυπα, ανάλογα με την πολυπλοκότητά τους. Τα πρότυπα αυτά περιέχουν στοιχεία τόσο από τα φαινολογικά όσο και από τα θεωρητικά πρότυπα. Τα τεράστια πρότυπα χρησιμοποιούν όλη την διαθέσιμη γνώση ενώ αντίθετα τα μέτρια πρότυπα χρησιμοποιούν μόνο τις σημαντικότερες πληροφορίες. Κατά διαστήματα εκφράστηκαν διάφορες απόψεις για τα πλεονεκτήματα και μειονεκτήματα των μέτριων και τεράστιων προτύπων [2]. Επικρατέστερη άποψη είναι ότι παρόλο που ο ρεαλισμός παίζει σημαντικό ρόλο στην ανάπτυξη ενός προτύπου και παρόλο που τα τεράστια life-system πρότυπα είναι ρεαλιστικά, η χρήση τους είναι περιορισμένη. Οι πιο σημαντικοί λόγοι περιορισμού της χρήσης τους είναι ότι ο χρόνος που χρειάζεται να ολοκληρωθεί ένα τέτοιο πρότυπο το καθιστά άχρηστο αφού μέχρι να ολοκληρωθεί τα προβλήματα πολλές φορές είτε λύνονται, είτε αντικαθιστώνται από άλλα. Επίσης τα τεράστια life-system πρότυπα μειονεκτούν έναντι των μέτριων στην επιτυχία συγκεκριμένων στόχων.

Στατιστικά πρότυπα χρησιμοποιήθηκαν για την ανάλυση του συσχετισμού μεταξύ της πληθυσμιακής πυκνότητας και των ποικίλων παραγόντων που έχουν την δυνατότητα να αλλάζουν στο χώρο και τον χρόνο. Τα πρότυπα αυτά έχουν την δυνατότητα να προβλέπουν τους αριθμούς πληθυσμών στον μέλλον και αντιπροσωπεύονται συνήθως από εξισώσεις [1].

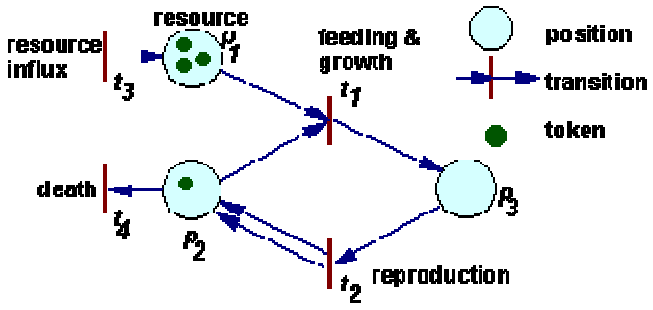


Σχήμα 2.2: Συσχετισμός κάποιων από τα πρότυπα που χρησιμοποιήθηκαν στην μοντελοποίηση πληθυσμών από το 1930 μέχρι το 1995 [2]

Άλλοι δυο τρόποι ανάλυσης και μοντελοποίησης πληθυσμών που συνάντησα κατά τη μελέτη μου, είναι τα Petri nets [1] και το μοντέλο Forrester [1] τα οποία παρουσιάζονται με μεγαλύτερη λεπτομέρεια στην συνέχεια.

Το μοντέλο Petri Nets επινοήθηκε το 1939 από τον Carl Adam Petri. Αποτελεί ένα γραφικό και μαθηματικό εργαλείο μοντελοποίησης το οποίο μπορεί να χρησιμοποιηθεί για να μιμηθεί τις διάφορες αλληλεπιδράσεις που συμβαίνουν μέσα σε ένα πληθυσμό καθώς επίσης και τις αλληλεπιδράσεις του με άλλους πληθυσμούς. Ένα Petri Δίκτυο αποτελείται από τις θέσεις (positions), τις μεταβάσεις (transitions), τα τόξα εισαγωγής που συνδέουν τις θέσεις με τις μεταβάσεις και τα τόξα παραγωγής που ξεκινώντας από τις μεταβάσεις καταλήγουν σε κάποια θέση. Οι θέσεις μπορούν να περιέχουν σημεία (tokens), διαμορφώνοντας έτσι την τρέχουσα κατάσταση του συστήματος που απεικονίζει το δίκτυο. Η κατάσταση του συστήματος μεταβάλλεται με τις «πυρκαγιές μετάβασης» που πραγματοποιούνται εάν υπάρχουν διαθέσιμα σημεία στις θέσεις εισαγωγής, τα οποία θα αφαιρεθούν για να προστεθούν στις θέσεις παραγωγής.

Πιο κάτω παρουσιάζεται παράδειγμα ενός Petri δικτύου:



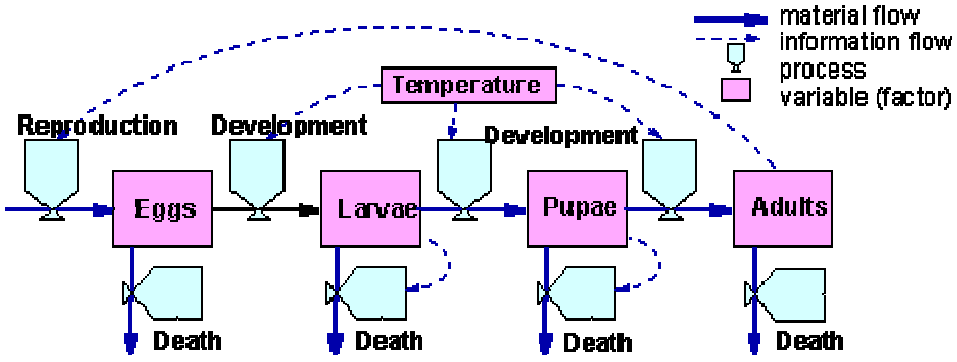
Σχήμα 2.3 : Παράδειγμα Petri Net [1]

Στο σχήμα οι θέσεις απεικονίζονται με τους μεγάλους γαλάζιους κύκλους και συμβολίζονται με το γράμμα p, ενώ οι μεταβάσεις παρουσιάζονται μέσω των κόκκινων γραμμών και συμβολίζονται με το γράμμα t. Τα τόξα που φτάνουν σε μια μετάβαση είναι τα τόξα εισαγωγής ενώ τα τόξα που φεύγουν από μια μετάβαση είναι τα τόξα παραγωγής. Τα σημεία απεικονίζονται από τους πράσινους μικρούς κύκλους που περιέχονται σε κάποιες θέσεις. Η ιδέα ενός Petri Net βασίζεται σε μια ακολουθία από μεταβάσεις «πυρκαγιάς». Όταν μια μετάβαση λάβει χώρο τότε αφαιρείται από κάθε

θέση της οποίας φεύγει τόξο εισαγωγής προς την συγκεκριμένη μετάβαση, ένα σημείο. Εάν περισσότερα από ένα βέλος πηγαίνει από τη θέση στη μετάβαση, ο αριθμός σημείων που αφαιρούνται από εκείνη την θέση είναι ίσος με τον αριθμό βελών. Αντίστοιχα τα νέα σημεία αυτά τοποθετούνται στις θέσεις που υποδεικνύονται από τα βέλη παραγωγής που κατευθύνονται από τη μετάβαση στην νέα θέση. Ο αριθμός σημείων που τοποθετούνται αντιστοιχεί στον αριθμό βελών που φτάνουν στην θέση.

Στο πιο πάνω παράδειγμα μπορεί να πραγματοποιηθεί η μετάβαση t1 (feeding & growth) με αποτέλεσμα να μετακινηθεί ένα σημείο από την θέση p1 και ένα από την p2 και να προστεθεί ένα σημείο στην p3. Τότε θα μπορεί να εκτελεστεί η μετάβαση t2 (reproduction) κατά την οποία θα μετακινηθεί το μοναδικό σημείο από την θέση p3 και θα προστεθεί ένα σημείο στην p2.

Άλλο ένα μοντέλο για την ανάλυση πληθυσμών είναι το Forrester το οποίο επινοήθηκε το 1961 από τον Forrester [1]. Το πρότυπο αυτό βασίζεται στην αναλογία δεξαμενής-σωλήνων (tank-pipe). Το σύστημα αποτελείται από ένα σύνολο δεξαμενών που συνδέονται με σωλήνες και διεξόδους (διαδικασίες) μέσω των οποίων ρυθμίζεται και πραγματοποιείται η ροή του υγρού από μια δεξαμενή σε άλλη. Το υγρό ή όπως αλλιώς αναφέρεται το υλικό στις δεξαμενές, αποτελεί μια μεταβλητή ή ένα παράγοντα που μπορεί να επηρεάζεται από την ροή των πληροφοριών μέσω των διαδικασιών . Πιο κάτω παρουσιάζεται ένα παράδειγμα μοντέλου Forrester:



Σχήμα 2.4: Παράδειγμα Forrester διαγράμματος [1]

Στο σχήμα 2.4 οι δεξαμενές απεικονίζονται με τα ροζ κουτιά, τα οποία περιέχουν μέσα μια μεταβλητή ή ένα παράγοντα. Οι διαδικασίες ή αλλιώς διέξοδοι παρουσιάζονται με

γαλάζιο χρώμα. Η ροή του υλικού απεικονίζεται με την συνεχόμενη γραμμή που αποτελούν τους σωλήνες, ενώ με την διακεκομμένη γραμμή παρουσιάζεται η ροή πληροφοριών που επηρεάζει μέσω των διαδικασιών τον παράγοντα των δεξαμενών. Το διάγραμμα στο σχήμα 2.4 απεικονίζει ένα σύστημα πληθυσμού εντόμων με τέσσερα στάδια (αυγά, προνύμφες, χρυσαλίδες, και ενήλικοι). Η θερμοκρασία είναι ο παράγοντας που ρυθμίζει την μετάβαση μεταξύ των σταδίων αυτών. Η παραγωγή των αυγών εξαρτάται από τον αριθμό ενηλίκων, ενώ η θάνατος εμφανίζεται σε όλα τα στάδια ανάπτυξης. Ο θάνατος στο στάδιο των προνυμφών και των χρυσαλίδων εξαρτάται από την ροή των πληροφοριών.

Ένα διάγραμμα Forrester μπορεί να μετασχηματιστεί σε πρότυπο διαφορικής εξίσωσης [1]. Κάθε διαδικασία γίνεται ένας όρος σε μια διαφορική εξίσωση που καθορίζει τη δυναμική της μεταβλητής. Για παράδειγμα, ο αριθμός προνυμφών στο πιο πάνω παράδειγμα, επηρεάζεται από τρείς διαφορετικές διαδικασίες: την ανάπτυξη αυγών ED(T), την ανάπτυξη των προνυμφών LD(T), και τον θάνατο των προνυμφών LM(N). Τα ποσοστά ανάπτυξης των αυγών και των προνυμφών είναι συνάρτηση της θερμοκρασίας, ενώ ο θάνατος των προνυμφών είναι συνάρτηση του αριθμού των προνυμφών N . Η εξίσωση για τη δυναμική των προνυμφών είναι η ακόλουθη:

$$dN/dt = ED\ (T) - LD\ (T) - LM\ (N)$$

Ο όρος ED (T) είναι θετικός επειδή η ανάπτυξη αυγών αυξάνει τον αριθμό προνυμφών («υγρή» εισροή). Οι όροι LD (T) και LM (N) είναι αρνητικοί επειδή η ανάπτυξη προνυμφών σε χρυσαλίδες και ο θάνατος μειώνει τον αριθμό τους.

Το σημαντικότερο μειονέκτημα του Forrester μοντέλου είναι ότι είναι αδύνατο να εφαρμοστεί σε διαδικασίες που περιλαμβάνουν δύο ή περισσότερους συμμετέχοντες, όπως για παράδειγμα η ταυτόχρονη συνύπαρξη ατόμων από διαφορετικούς πληθυσμούς με διαφορετική συμπεριφορά ο καθένας. Αυτό είναι ένα σημείο στο οποίο τα Petri Nets υπερισχύουν του μοντέλου Forrester.

Μελέτη πραγματοποιήθηκε επίσης από πολλούς επιστήμονες γύρω από το θέμα των αλγεβρών διεργασιών. Οι άλγεβρες διεργασιών είναι 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.

2.3 Άλγεβρες Διεργασιών

Ο όρος «άλγεβρα διεργασιών» επινοήθηκε το 1982 από τον Bergstra & Klop. Οι Άλγεβρες Διεργασιών (Process Algebra) είναι ένα πρότυπο το οποίο έχει προταθεί αρχικά για τη μοντελοποίηση και ανάλυση κατανεμημένων υπολογιστικών συστημάτων. Βασικός στόχος τους, είναι η συνθετικότητα (compositionality). Έχουν δηλαδή 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. Για αυτό το λόγο μεγαλύτερη έμφαση δίνεται στις μεταβάσεις (action-based) παρά στις καταστάσεις (state-based) του συστήματος.

Μέχρις στιγμής έχει πραγματοποιηθεί αρκετή μελέτη γύρω από το θέμα των αλγεβρών διεργασιών. Οι κυριότερες άλγεβρες διεργασιών είναι:

- CCS, Calculus of Communicating Systems, που επινοήθηκε από τον Milner
- CSP, Communicating Sequential Processes, που επινοήθηκε από τον Hoare
- ACP, Algebra of Communicating Processes, που επινοήθηκε από τους Bergstra & Klop

Επεκτάσεις που ήδη έχουν πραγματοποιηθεί περιλαμβάνουν την ένταξη σε υπάρχουσες άλγεβρες διεργασιών της έννοιας της πιθανότητας, του χρόνου και της προτεραιότητας [7,4]. Σημαντικοί σταθμοί της μελέτης των αλγεβρών διεργασιών σε σχέση με την εργασία μου, αποτελούν η CCS [9], η επέκτασή της WSCCS η οποία χρησιμοποιήθηκε για την ανάλυση πληθυσμών μυρμηγκιών στα [3, 4, 5, 8] καθώς επίσης και στο [6] και η TPCCS [7] η οποία περιλαμβάνει την έννοια της πιθανότητας και του χρόνου .

Η άλγεβρα διεργασιών Timed Probabilistic Calculus of Communicating Systems (TPCCS) [7] αποτελεί επέκταση της CCS του Milner, με την ανάμειξη της έννοιας της πιθανότητας και της έννοιας του χρόνου. Στην CCS η επιλογή μεταξύ διαφορετικών μεταβάσεων γίνεται μη-ντετερμινιστικά. Στην TPCCS υπάρχει μια πιθανολογική επιλογή, η οποία ορίζεται στο άρθρο [7], σαν μια διανεμημένη πιθανότητα πάνω σε ένα σύνολο ενδεχόμενων καταστάσεων. Έτσι σε κάθε κατάσταση που μπορεί να βρεθεί το σύστημα έχει είτε μια πιθανολογική επιλογή, είτε μια μη-ντετερμινιστική επιλογή. Με την πιθανολογική επιλογή, η μετάβαση συμβαίνει λόγω κάποιας πιθανότητας ενώ με την μη-ντετερμινιστική επιλογή, λόγω της εκτέλεσης κάποιας ενέργειας όπως και στην CCS. Οι επιλογές με βάση τις πιθανότητες μας απαλλάσσουν από τις λεπτομέρειες για το πώς πραγματοποιούνται οι συγκεκριμένες επιλογές.

2.4 CCS - Calculus of Communicating Systems

Το κύριο πρόσωπο στην ιστορία των αλγεβρών 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 όπως θα επεξηγηθεί περαιτέρω πιο κάτω.[10]

Όπως κάθε άλγεβρα διεργασιών έτσι και η 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 (input) : a
- ενέργεια εξόδου(output): $\bar{a}$
- εσωτερική ενέργεια: τ

Όπως και σε κάθε άλγεβρα διεργασιών η εσωτερική ενέργεια τ, επιτυγχάνεται με τον συνδυασμό δύο συμπληρωματικών ενεργειών (a, $\bar{a}$), 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 ενεργειών της γλώσσας τότε, οι τελεστές που σχηματίζουν την σύνταξη της γλώσσας CCS είναι οι πιο κάτω [9,10]:

- 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 συνάρτηση. Π.χ. f(α) = β.

Πίνακας 2.1: Σύνταξη της CCS

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

$$E \xrightarrow{a} F$$

και η οποία ερμηνεύεται ως εξής: Εάν η διεργασία E εκτελέσει την ενέργεια α, τότε μπορεί να εξελιχθεί στην διεργασία F.

Με την βοήθεια της πιο πάνω σχέσης $\longrightarrow$ μπορεί να οριστεί το σύνολο των κανόνων που δίνουν την σημασιολογία την γλώσσας. Όλοι οι κανόνες της σημασιολογίας είναι της μορφής:

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

Οι συνθήκες μεταβάσεων είναι της μορφής $E \xrightarrow{a} F$.Εάν οι συνθήκες μεταβάσεων και οι επιμέρους συνθήκη ισχύουν τότε ισχύει το συμπέρασμα.

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

Τελεστής Διαδοχής:

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

Τελεστής Επιλογής (Summation):

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

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

Τελεστής Ταυτοχρονισμού (Composition):

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

$$\text{Com2} \quad \frac{E \xrightarrow{\alpha} F'}{E \mid F \xrightarrow{\alpha} E \mid F'}$$

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

Τελεστής Περιορισμού (Restriction):

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

Τελεστής Μετονομασίας (Relabeling):

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

Πίνακας 2.2: Κανόνες Σημασιολογίας CCS

Πιο κάτω δίνεται επεξήγηση για κάθε ένα από τους κανόνες σημασιολογίας της CCS:

- Τελεστής Διαδοχής: Σύμφωνα με τον τελεστή 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 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 (E+F), μπορούμε να επιλέξουμε την Ε και με την εκτέλεση της ενέργειας α να φτάσουμε στην διεργασία Ε'.

  Αντίστοιχα στον κανόνα Sum2 εάν ισχύει ότι από την διεργασία F μπορούμε να φτάσουμε στην F΄ με την εκτέλεση της ενέργειας α, τότε κατά την επιλογή μεταξύ της Ε και F, μπορούμε να επιλέξουμε την F και με την εκτέλεση της α ενέργειας να φτάσουμε στην διεργασία 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 φτάσουμε στην κατάσταση ταυτοχρονισμού της διεργασίας Ε' με της διεργασία F (Ε΄| F).

  Αντίστοιχα στον Com2 κανόνα, 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 φτάσουμε στην κατάσταση ταυτοχρονισμού της διεργασίας Ε με της διεργασία F΄ (Ε| F΄).

  Στον κανόνα Com3 πραγματοποιείται η επικοινωνία μέσω εσωτερικής ενέργειας. Εάν ισχύει ότι από την διεργασία Ε μέσω της ενέργειας εισόδου α μπορούμε να φτάσουμε στην διεργασία Ε΄ και ότι από την διεργασία F μέσω της ενέργειας εξόδου $\bar{\alpha}$ μπορούμε να φτάσουμε στην διεργασία F΄, τότε με εσωτερική ενέργεια κατά τον ταυτοχρονισμό των διεργασιών Ε και F μπορούμε να φτάσουμε στην κατάσταση Ε' | F', μέσω της εσωτερικής ενέργειας που πραγματοποιείται λόγω της ταυτόχρονης εκτέλεσης των ενεργειών α και $\bar{\alpha}$.

- Τελεστής Περιορισμού: Η επικοινωνία των συστημάτων με το περιβάλλον τους αλλά και μεταξύ τους στην CCS πραγματοποιείται μέσω συγχρονισμών που συμβαίνουν σε κανάλια. Εάν ισχύει ότι από την διεργασία Ε μέσω της εκτέλεσης της ενέργειας α μπορούμε να φτάσουμε στην διεργασία Ε΄ και ότι η ενέργεια α και η συμπληρωματική της $\bar{\alpha}$ δεν ανήκουν στα κανάλια του L, τότε η Ε μπορεί να φτάσει με την εκτέλεση της α στην Ε΄, με τον περιορισμό να μην εκτελέσει καμιά ενέργεια που ανήκει στα κανάλια του L.

- Τελεστής Μετονομασίας: Εάν ισχύει ότι από την διεργασία Ε μέσω της εκτέλεσης της ενέργειας α μπορούμε να φτάσουμε στην διεργασία Ε΄, τότε μπορούμε να μετονομάσουμε τα κανάλια του Ε με βάση την συνάρτηση f.

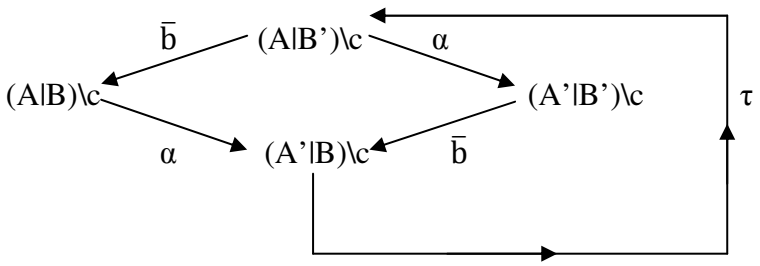
Η συμπεριφορά ενός συστήματος μπορεί να μεταφραστεί σε γράφο με βάρη, ο οποίος ονομάζεται σύστημα μεταβάσεων. Οι κορυφές του γράφου αντιστοιχούν σε διεργασίες, οι ακμές αντιστοιχούν στις μεταβάσεις των διεργασιών και τα βάρη αντιστοιχούν στις ήδη εκτελεσμένες ενέργειες.

Πιο κάτω παρουσιάζεται ένα παράδειγμα μοντελοποίησης ενός συστήματος που αποτελείται από τις διεργασίες A, A΄, B και B΄, οι οποίες ορίζονται ως εξής [9]:

$$A \stackrel{def}{=} \alpha.A' \qquad B \stackrel{def}{=} c.B'$$

$$A' \stackrel{def}{=} \bar{c}.A \qquad B' \stackrel{def}{=} \bar{b}.B$$

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

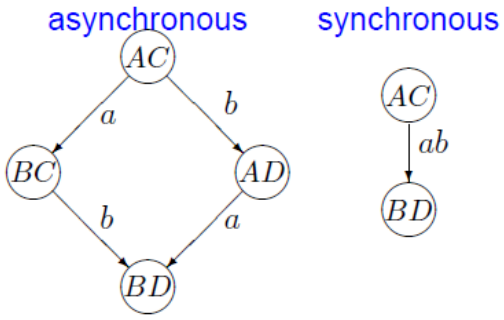


Σχήμα 2.5: Παράδειγμα στην CCS

2.5 WSCCS

Η WSCCS δημιουργήθηκε από τον Chris Tofts το 1992 και αποτελεί προέκταση της γλώσσας SCCS του Milner. Το όνομα της γλώσσα WSCCS προέρχεται από τα αρχικά των λέξεων: Weighted Synchronous Calculus of Communicating Systems, ενώ της γλώσσας SCCS από τα αρχικά: Synchronous Calculus of Communicating Systems. Στο προηγούμενο υποκεφάλαιο, έγινε μια σύντομη περιγραφή της γλώσσας CCS. Οι δυο γλώσσες SCCS και CCS έχουν προταθεί από τον Milner και διαφέρουν στο ότι η CCS αναφέρεται στον ασυγχρονισμό ενώ η SCCS στον συγχρονισμό. Η CCS αποτελεί υποκατηγορία της SCCS, και ο συγχρονισμός που παρουσιάζει σαν χαρακτηριστικό της

SCCS την μετατρέπει σε πιο δυνατή γλώσσα ενώ ταυτόχρονα ο προγραμματισμός γίνεται πιο εύκολος.



Σχήμα 2.6: Απεικόνιση του συγχρονισμού και ασυγχρονισμού

Οι μη πιθανολογικές άλγεβρες διεργασιών βασίζονται στον μη-ντετερμινισμό και στην τυχαία επιλογή της επόμενης κατάστασης που θα επέλθει το σύστημα. Η έννοια της πιθανότητας αναπτύχθηκε με στόχο να δώσει κάποιου είδους εκτίμηση στα μη-ντετερμινιστικά συστήματα καθώς επίσης και να βοηθήσει στην απόφαση επιλογής μεταξύ περισσότερων από ενός τρόπου εξέλιξης του συστήματος. Τα πιθανολογικά πρότυπα υπολογισμού συχνά προσφέρουν μεγαλύτερη ευελιξία και λύσεις, οι οποίες μπορεί να έχουν μεγαλύτερη ανοχή σφαλμάτων από ότι τα ντετερμινιστικά πρότυπα.

Στην CCS η διεργασία A+B, έχει την ικανότητα επιλογής τυχαία μεταξύ της διεργασίας A ή της διεργασίας B. Αυτό μπορεί να έχει σαν αποτέλεσμα την επιλογή συνεχώς της ίδιας διεργασίας, αφού η επιλογή δεν καθορίζεται από κάποιο παράγοντα. Στην WSCCS ο Tofts επέλεξε να μην ποσοτικοποιήσει τις επιλογές κατευθείαν με πιθανότητες, αλλά να χρησιμοποιήσει κάποιου είδους βάρη όπως ορίζεται και από το όνομα της γλώσσας Weighted Synchronous Calculus of Communicating Systems. Τα βάρη αυτά μπορούν να ερμηνευτούν σαν προτεραιότητες που δίνονται σε κάθε διεργασία [4].

Η έννοια της προτεραιότητας στην WSCCS που δηλώνεται από τα σύμβολα: ω, ω^1, ω^2, …, ω^n,… είναι δυναμική. Για παράδειγμα για τις διεργασίες A και B δίνονται τα πιο κάτω βάρη όταν απαιτείται η επιλογή μεταξύ των δύο:

$$1 : A + 2 : B$$

Τα βάρη είναι λειτουργικά πιο κατάλληλα από τις πιθανότητες, αφού δεν χρειάζονται οποιουδήποτε είδους κανονικοποίησης. Η κανονικοποίηση προκαλεί στην περίπτωση των πιθανοτήτων κάποιου είδους δυσκολία στην απόφαση κατά πόσο να επιτραπεί η εκτέλεση κάποιας διεργασίας. Χρησιμοποιώντας τα βάρη αντίθετα από τις πιθανότητες μια διεργασία δεν πρόκειται ποτέ να εκτελεστεί εάν καμιά από της ενέργειές της δεν επιτρέπεται [4].

Μέχρις στιγμής στην CCS η δυνατότητα αλλαγής κατάστασης του συστήματος γινόταν με την εκτέλεση μια ενέργειας μέσω των μεταβάσεων. Στην WSCCS η αλλαγής της κατάστασης μπορεί να γίνει είτε λόγο μιας επιλογής με βάρος: $\Sigma\{\mathrm{wi} : \mathrm{Ei}\} \xrightarrow{w_i} \mathrm{E_i}$, είτε της εκτέλεσης μιας ενέργειας όπως και στην CCS: $\alpha.\mathrm{E} \xrightarrow{\alpha} \mathrm{E}$

Οι τελεστές της WSCCS γλώσσας αποτελούνται από τους τελεστές της CCS : τον τελεστή τερματισμού, τον τελεστή διαδοχής, επιλογής, ταυτοχρονισμού, περιορισμού και μετονομασίας καθώς επίσης και τους επιπλέον τελεστές $\Sigma\{w_i : E_i\}$, $\Theta(E)$ και $\mu_i \tilde{x} \tilde{E}$.
Ο τελεστής $\Sigma\{w_i : E_i\}$ αποτελεί την σταθμική επιλογή μεταξύ των διεργασιών E_i ,των οποίων το βάρος δίνεται από το w_i που αντιστοιχεί σε κάθε μια. Ο τελεστής $\Theta(E)$ έχει την ικανότητα να παρουσιάζει τα κομμάτια της διεργασίας E με τα μεγαλύτερα βάρη, την μεγαλύτερη δηλαδή προτεραιότητα. Τέλος ο τελεστής $\mu_i \tilde{x} \tilde{E}$ δίνει την λύση x_i η οποία λαμβάνεται μετά από κάποιου είδους αναδρομής [4].

Επίσης λόγω του ότι η WSCCS αποτελεί γλώσσα συγχρονισμού υπάρχει και η έννοια του χρόνο η οποία δίνεται με την βοήθεια του συμβόλου √ και επιτρέπει να δίνουμε στις διεργασίες απλές χρονικές ιδιότητες.

2.6 Άλγεβρες Διεργασιών και Πληθυσμικά Συστήματα

Μελετώντας τα πληθυσμιακά συστήματα και τους τρόπους μοντελοποίησης τους μέσω των αλγεβρών διεργασιών, η μελέτη μου επικεντρώθηκε γύρω από τον τρόπο που η συμπεριφορά κυρίως του πληθυσμού των μυρμηγκιών, μοντελοποιείται με την βοήθεια των γλωσσών CCS και της επέκτασης της WSCCS[6,3,4,5].

Η άλγεβρα διεργασιών WSCCS του Chris Tofts, χρησιμοποιήθηκε από πολλούς επιστήμονες για την μελέτη της συμπεριφοράς του πληθυσμού των μυρμηγκιών [6,3,4,5]. Η WSCCS αποτελεί μια προέκταση της CCS, στην οποία προστίθεται ο τελεστής προτεραιότητας, και με την βοήθεια της οποίας μπορεί να περιγραφεί πως τα άτομα ενός πληθυσμού συμπεριφέρονται σύμφωνα με ένα σύνολο πιθανολογικών κανόνων. Η WSCCS αποτελεί ένα από τα σημαντικότερα, εάν όχι το σημαντικότερο πρότυπο δυναμικών συστημάτων. Για την καλύτερη κατανόηση των κανόνων της WSCCS ο Tofts σχεδίασε το εργαλείο Probability Workbench[6].

Ο D.J.T Sumpter, ο G.B. Blanchard και ο D.S. Broomhead συνδυάζοντας την WSCCS με τις αλυσίδες Markov, τις υπολογιστικές προσομοιώσεις και διαφορικές εξισώσεις, μπόρεσαν να καταλήξουν σε συμπεράσματα για τον τρόπο που τα άτομα ενός πληθυσμού μυρμηγκιών συμπεριφέρονται[6]. Αρχικά με την βοήθεια της WSCCS όρισαν τα μυρμήγκια και τις καταστάσεις στις οποίες μπορεί να βρεθεί το σύστημά τους, σύμφωνα με κάποιους πιθανολογικούς κανόνες. Τα μυρμήγκια, οι διεκπεραιωτές, (agents) αρχικά χαρακτηρίζονταν σαν ενεργά ή παθητικά και η κατάστασή τους μπορούσε να αλλάξει μέσω κάποιας μετάβασης σε t χρονικά βήματα με πιθανότητα $(1-\mathrm{p})^{\mathrm{t}-1}.\mathrm{p}$, όπου p είναι παράμετρος που αντιπροσωπεύει τις πιθανότητες. Για να καταφέρουν να μελετήσουν ολόκληρο τον πληθυσμό των μυρμηγκιών συνέθεσαν τα μυρμήγκια που αποτελούν τον πληθυσμό παράλληλα με την βοήθεια του τελεστή ταυτοχρονισμού όπως παρουσιάζεται πιο κάτω:

$$\mathrm{Colony}_n(i) \equiv \mathrm{Active}_1 \mid \ldots \mid \mathrm{Active}_i \mid \mathrm{Passive}_{i+1} \mid \ldots \mid \mathrm{Passive}_n$$

Με το πιο πάνω παρουσιάζεται πως η αποικία των n μυρμηγκιών απεικονίζεται με την βοήθεια των i Active και (n-i) Passive μυρμηγκιών, τα οποία μπορούν να πραγματοποιούν παράλληλα κάποιες ενέργειες.

Επόμενο βήμα ήταν ο ορισμός των ενεργειών (actions) του συστήματος ώστε τα άτομα του πληθυσμού να αλληλεπιδρούν μεταξύ τους. Οι ενέργειες μπορούν να ταυτιστούν με μηνύματα τα οποία λαμβάνουν τα μυρμήγκια από άλλα μυρμήγκια ή από άλλους πληθυσμούς. Μέσω αυτής της επικοινωνίας η συμπεριφορά των μυρμηγκιών αλλάξει.

Η κάθε ενέργεια χαρακτηρίζεται από ένα παράγοντα w, ο οποίος καθορίζει την σειρά προτεραιότητας της.

Στην συνέχεια πραγματοποίησαν ανάλυση του μοντέλου που πρότειναν μέσω των αλυσίδων Markov (βλέπε υποκεφάλαιο 2.7). Μια αλυσίδα Markov είναι μια χρονική διατεταγμένη ακολουθία από τυχαίες μεταβλητές, όπου η μεταβλητή t+1 της ακολουθίας, είναι μια υπόθεση πάνω στην τιμή της t-οστής μεταβλητής, αλλά δεν εξαρτάται από προηγούμενες τιμές.. Μέσω αυτών των αλυσίδων οι τρεις επιστήμονες απόδειξαν ότι μακροπρόθεσμα η δραστηριότητα τελειώνει με όλα τα μυρμήγκια παθητικά.

Επόμενο βήμα στην συγκεκριμένη μελέτη της συμπεριφοράς του πληθυσμού των μυρμηγκιών ήταν η απεικόνισή της μέσω διαφορικών εξισώσεων[6]. Το πιο πάνω μοντέλο της ενεργοποίησης και απενεργοποίησης των μυρμηγκιών, επεκτάθηκε στην συνέχεια από άλλους επιστήμονες εντάσσοντας διάφορες υποθέσεις σε αυτό, όπως: τα ενεργά μυρμήγκια γίνονται παθητικά όταν είναι κοντά σε παθητικά μυρμήγκια. Αφού η ανάλυση έγινε με την βοήθεια διαφορικών εξισώσεων στην συνέχεια μέσω προσομοιώσεων συσχετίστηκαν τα αποτελέσματα των μοντέλων.[6]

Συνεχίζοντας με την μελέτη που έγινε γύρω από το θέμα των πληθυσμιακών συστημάτων σε συνδυασμό με τις άλγεβρες διεργασιών, ο Chris Tofts, μελέτησε τον πληθυσμό των μυρμηγκιών, τα οποία είχαν την ικανότητα να κατανέμουν σωστά τις εργασίες τους μεταξύ τους. Η επιβίωση ενός πληθυσμού μυρμηγκιών εξαρτάται από διάφορες εργασίες τις οποίες πραγματοποιούν, όπως είναι η συλλογή τροφής, η φρούρηση της φωλιάς τους κ.α. Μέσα από διάφορους αλγόριθμους οι οποίοι εκφράστηκαν στην WSCCS, ο Tofts κατάφερνε κάθε φορά να αποδεικνύει, ότι υπάρχει τρόπος να κατανέμονται σωστά οι εργασίες στα μυρμήγκια έτσι ώστε ο πληθυσμός να παρουσιάζει μια σταθερότητα στον τρόπο που κατανέμονται οι εργασίες σε αυτόν. Θεώρησε ότι όλα τα μυρμήγκια εντάσσονται σε μια παραγωγική γραμμή όπου απαριθμούνται κατά σειρά οι εργασίες και κατασκεύασε στην WSCCS τον βασικό αλγόριθμο του μοντέλου του. Αρχικά απόδειξε ότι εάν η πιθανότητα να μετακινηθεί ένα μυρμήγκι από μια εργασία σε άλλη είναι μηδενική, τότε ο πληθυσμός μπορεί σε πεπερασμένο αριθμό μεταβάσεων να φτάσει σε μια σταθερή κατάσταση. Στο ίδιο

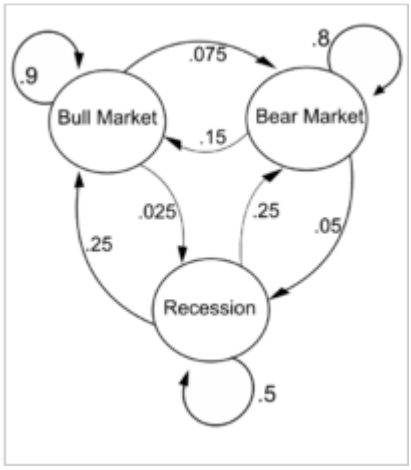
συμπέρασμα κατέληξε επεκτείνοντας την προηγούμενη σκέψη του και εντάσσοντας στις εργασίες βαθμό δυσκολίας. Και σε αυτή την περίπτωση χρησιμοποιώντας την WSCCS μπόρεσε να περιγράψει τον τρόπο που τα μυρμήγκια κατανέμονται σωστά στις εργασίες τους, ώστε στο τέλος να προκύπτει σταθερότητα στον πληθυσμό. Σε αντίστοιχο συμπέρασμα κατάληξε δίνοντας σε κάθε εργασία σύνολο υποεργασιών που έπρεπε να πραγματοποιηθούν για την ολοκλήρωση μιας εργασίας. Επομένως, σε κάθε περίπτωση μέσω της WSCCS στην οποία εκφραζόταν ο αλγόριθμος του μοντέλου, ήταν δυνατή η πλήρης περιγραφή της συμπεριφοράς των μυρμηγκιών κατά τον καταμερισμό τους στις εργασίες [5].

2.7 Αλυσίδες Markov

Οι αλυσίδες Markov χρησιμοποιήθηκαν για πρώτη φορά από τον Andrey Markov το 1906. Αποτελούν ένα σημαντικό μαθηματικό εργαλείο για στατιστική ανάλυση και μοντελοποίηση. Μια αλυσίδα Markov είναι μια τυχαία διακριτή διαδικασία, με την ιδιότητα ότι η επόμενη κατάσταση εξαρτάται μόνο από την τρέχουσα κατάσταση. Με την αναφορά σε μια τυχαία διακριτή διαδικασία εννοούμε ένα σύστημα του οποίου οι καταστάσεις αλλάζουν τυχαία μεταξύ των βημάτων, με αποτέλεσμα σε κάθε βήμα το σύστημα να βρίσκεται σε κάποια κατάσταση. Το κάθε βήμα σε μια αλυσίδα Markov μπορεί να είναι ίσο με κάποια διακριτή μονάδα μέτρησης, όπως είναι ο χρόνος. Η σημαντικότερη ιδιότητα των αλυσίδων είναι ότι η πιθανολογική κατανομή των καταστάσεων του συστήματος για κάθε επόμενο βήμα, εξαρτάται μόνο από την τρέχουσα κατάσταση.

Η μελλοντική κατάσταση του συστήματος στις αλυσίδες Markov είναι αδύνατο να προβλεφθεί αφού το σύστημα αλλάξει κατάσταση τυχαία. Οι στατιστικές ιδιότητες του μέλλοντος μπορούν παρόλα αυτά να προβλεφτούν και πολλές φορές αυτή είναι και η σημαντικότητα των αλυσίδων. Οι αλλαγές της κατάστασης του συστήματος καλούνται μεταβάσεις, και οι πιθανότητες που συνδέονται με τις διάφορες καταστάσεις καλούνται πιθανότητες μετάβασης. Μια αλυσίδα Markov αποτελείται από το σύνολο όλων των καταστάσεων και των πιθανοτήτων μετάβασης.

Πιο κάτω δίνεται ένα παράδειγμα μιας Markov αλυσίδας:



Σχήμα 2.7: Παράδειγμα Markov Αλυσίδας

Στο παράδειγμα υπάρχουν 3 διαφορετικές καταστάσεις οι οποίες συνδέονται με πιθανολογικές μεταβάσεις. Για παράδειγμα από την κατάσταση Bull Market με 90% πιθανότητα μπορεί να καταλήξει στην ίδια κατάσταση, με 7.5% να φτάσει στην κατάσταση Bear Market και με 2.5% πιθανότητα να φτάσει στην κατάσταση Recession. Από αυτά τα διαγράμματα μπορούμε να υπολογίσουμε για παράδειγμα τον μέσο όρο για το πόσο χρόνο χρειάζεται να φτάσει από την κατάσταση Recession στην κατάσταση Bull Market.

Κάθε αλυσίδα Markov μπορεί να απεικονιστεί και με την βοήθεια ενός πίνακα. Σύμφωνα με το πιο πάνω παράδειγμα μπορεί να δημιουργηθεί ένας 3×3 πίνακας, όπου για παράδειγμα στην θέση [1,1] βρίσκεται η πιθανότητα με την οποία από την πρώτη κατάσταση παραμένουμε στην ίδια κατάσταση. Αντίστοιχα στην θέση [2,3] βρίσκεται ο αριθμός 0.05, η πιθανότητα δηλαδή να κινηθούμε από την δεύτερη κατάσταση (Bear Market) στην τρίτη κατάσταση (Recession). Πιο κάτω παρουσιάζεται ο πίνακας του πιο πάνω παραδείγματος:

$$\begin{pmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{pmatrix}$$

Παρατηρώντας τον πίνακα αντιλαμβανόμαστε ότι το άθροισμα των πιθανοτήτων να κινηθούμε από μια κατάσταση προς όλες τις δυνατές επόμενες καταστάσεις είναι ίση με ένα. Το άθροισμα δηλαδή κάθε γραμμής του πίνακα είναι ίσο με ένα. Με την βοήθεια αυτού του πίνακα μπορεί μέσω τεχνικών των αλυσίδων αυτών και μέσω εξισώσεων, να επιτευχθεί ανάλυση ενός συστήματος.

Οι εφαρμογές των αλυσίδων Markov είναι πάρα πολλές. Μπορούν να χρησιμοποιηθούν στον κλάδο της οικονομίας, στον κλάδο της στατιστικής, της φυσικής, της χημείας, της πληροφορικής σε εφαρμογές του διαδικτύου, στην μουσική, καθώς επίσης και στην μαθηματική βιολογία. Μπορούν με επιτυχία να χρησιμοποιηθούν κατά την μοντελοποίηση και ανάλυση βιολογικών πληθυσμών. Οι καταστάσεις σε αυτή την περίπτωση είναι οι διάφορες καταστάσεις στις οποίες μπορεί να επέλθει ένα σύστημα που αποτελείται από την συνύπαρξη κάποιων πληθυσμών στον ίδιο κόσμο, ενώ η πιθανολογικές μεταβάσεις, δίνουν την πιθανότητα με την οποία το σύστημα από μια κατάσταση μπορεί να μεταβεί σε μια άλλη κατάσταση.

Κεφάλαιο 3

Άλγεβρα Διεργασιών με την Έννοια του Χώρου

3.1 PALPS

Κυριότερος στόχος της παρούσας διπλωματικής εργασίας ήταν η προσπάθεια δημιουργίας μιας άλγεβρας διεργασιών με την έννοια της τοποθεσίας, η οποία θα μας επιτρέπει να εξάγουμε συμπεράσματα για την κατάσταση του χώρου στον οποίο συμβιώνουν συγκεκριμένοι πληθυσμοί, μετά τις αλληλεπιδράσεις των ατόμων των πληθυσμών στην προσπάθειά τους να επιβιώσουν.

Την γλώσσα την οποία παρουσιάζω στο υποκεφάλαιο αυτό ονόμασα PALPS, από τα αρχικά Process Algebra with Location for Population Systems. Η γλώσσα αυτή χρησιμοποιεί κάποιους από τους τελεστές της WSCCS του Chris Tofts, η οποία περιγράφηκε στο προηγούμενο κεφάλαιο, προσαρμόζοντας σε αυτούς τους κανόνες την έννοια του χώρου και την έννοια των πιθανοτήτων. Η έννοια του χώρου αποτελεί ένα σημαντικό σημείο στην κατανόηση του τρόπου που ένας πληθυσμός επιβιώνει. Πολλές από της ενέργειες που πραγματοποιούν τα άτομα ενός πληθυσμού με συγκεκριμένες πιθανότητες, μπορούν να συσχετιστούν άμεσα με τον χώρο στον οποίο ζουν.

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

- ενέργεια εισόδου (input) : α

- ενέργεια εξόδου(output): $\overline{\alpha}$
- εσωτερική ενέργεια: τ
- ενέργεια μετακίνησης : move
- ενέργεια αναπαραγωγής: reproduce
- ενέργεια μόλυνσης: infect
- ενέργεια θανάτου: die

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 αναπαράγεται παράγοντας συγκεκριμένο αριθμό απογόνων.
Ενέργεια Μόλυνσης: Κάποιοι πληθυσμοί έχουν την δυνατότητα να εκκρίνουν δηλητήριο μολύνοντας άτομα άλλων πληθυσμών.
Ενέργεια Θανάτου: Κάθε άτομο οποιουδήποτε πληθυσμού μετά από κάποια χρονική περίοδο που ζει, πεθαίνει. Ο θάνατος μπορεί να προκληθεί από πολλούς και διάφορους λόγους (λόγω ηλικίας, μόλυνσης, έλλειψης τροφής κ.α.).

Πίνακας 3.1: Επεξήγηση Ενεργειών της PALPS

Όπως και σε κάθε άλγεβρα διεργασιών η εσωτερική ενέργεια επιτυγχάνεται με τον συνδυασμό δύο συμπληρωματικών ενεργειών $(\alpha, \overline{\alpha})$, 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 επεξήγηση της ενέργειας της μετακίνησης θα χρειαστεί να ορίσουμε την σχέση Γειτνίασης - Neighbors και την έννοια του χώρου που θεωρώ ότι υπάρχουν στην γλώσσα.

Θεωρώ ότι οι οντότητες οποιουδήποτε πληθυσμού ενός συστήματος που μπορεί να μοντελοποιηθεί από την γλώσσα αυτή, βρίσκονται κατανεμημένοι στα τμήματα ενός ορισμένου χώρου (location). Πιο συγκεκριμένα αν θεωρήσω ολόκληρο τον χώρο σαν ένα πίνακα n×n, ο οποίος θα αναφέρεται σαν ο 'κόσμος', τότε κάθε ένα από τα κελία του ορίζουν μια συγκεκριμένη τοποθεσία που την συμβολίζω με $\ell(i,j)$, όπου i και j καθορίζουν τις συντεταγμένες της τοποθεσίας, την γραμμή και την στήλη του πίνακα. Πιο κάτω απεικονίζεται ο διαχωρισμός του κόσμου ενός συστήματος σε 4 επιμέρους τοποθεσίες , τα ℓ_1, ℓ_2, ℓ_3 και ℓ_4.

ℓ_1	ℓ_2
ℓ_3	ℓ_4

Σχήμα 3.1: Διαχωρισμός το κόσμου σε 4 τοποθεσίες

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 η οποία θα καθορίσει σε κάθε οντότητα που βρίσκεται στην τοποθεσία ℓ προς ποια κατεύθυνση μπορεί να κινηθεί.

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

- $\ell(i-1 , j)$, την τοποθεσία πάνω από την $\ell(i,j)$
- $\ell(i+1 , j)$, την τοποθεσία κάτω από την $\ell(i,j)$
- $\ell(i , j-1)$, την τοποθεσία αριστερά από την $\ell(i,j)$
- $\ell(i , j+1)$, την τοποθεσία δεξιά από την $\ell(i,j)$
- $\ell(i,j)$, τον ίδιο της τον εαυτό

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.

Η σύνταξη και η σημασιολογία της γλώσσας θα δοθούν σε δύο επίπεδα. Στο πρώτο επίπεδο η σύνταξη θα αφορά ένα μεμονωμένο άτομο του συνόλου των ατόμων που συνυπάρχουν στον κόσμο, ενώ το δεύτερο επίπεδο θα αφορά ολόκληρους πληθυσμούς. Στο δεύτερο επίπεδο θα παρουσιαστεί πώς πολλά άτομα μαζί μπορούν να συνθεθούν και να αποτελέσουν ένα σύστημα στο οποίο να συνυπάρχουν πολλά άτομα που αλληλεπιδρούν μεταξύ τους.

Σύνταξη της PALPS:

Αρχικά θα δοθεί η σύνταξη της γλώσσας σε ατομικό επίπεδο και στην συνέχεια σε επίπεδο πληθυσμών. Οι τελεστές οι οποίοι ορίζουν την σύνταξη της γλώσσας παρουσιάζονται πιο κάτω και είναι ανάλογοι με τους αντίστοιχους τελεστές της WSCCS:

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

Ακολουθεί η γραμματική που δίνει την σύνταξη των διεργασιών της γλώσσας σε επίπεδο ενός ατόμου. Όπου α ∈ Actions. Το E απεικονίζει μια διεργασία, μια οντότητα ενός πληθυσμού.

$$E ::= 0 \mid \alpha . E \mid E_1 + E_2$$

- 0: Διεργασία η οποία έχει τερματίσει.
- α.E: Διεργασία που μπορεί να εκτελέσει την ενέργεια α και να συμπεριφερθεί όπως η διεργασία E.

• $E_1 + E_2$: Διεργασία η οποία δίνει επιλογή ανάμεσα στις διεργασίες E_1 και E_2.

Πίνακας 3.2: Σύνταξη Διεργασιών της PALPS σε ατομικό επίπεδο

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

$$N ::= E_i [\ell_i, p_i, Inf_i] \mid N_1 \mid N_2$$

• $E_i [\ell_i, p_i, Inf_i]$: Διεργασία που απεικονίζει ένα άτομο του πληθυσμού p_i στην τοποθεσία ℓ_i και το οποίο έχουν μολύνει μέχρις στιγμής Inf_i άτομα. Το άτομο αυτό εκτελεί τις ενέργειές του παράλληλα με τις ενέργειες που εκτελούν άλλα άτομα που συνυπάρχουν στον ίδιο κόσμο. • N_1 και N_2: Αποτελούν συστήματα, συνθέσεις δηλαδή πολλών ατόμων μαζί τα οποία αλληλεπιδρούν μεταξύ τους κατά τη συνύπαρξή τους στον κόσμο, εκτελώντας ταυτόχρονα διαφορετικές ενέργειες ή επικοινωνόντας μεταξύ τους.

Πίνακας 3.3: Σύνταξη Διεργασιών της PALPS σε επίπεδο πληθυσμού

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 αναπαριστάται σαν ετικέτα, μπορεί να εξελιχθεί σε άλλη διεργασία. Το σύνολο των μεταβάσεων αυτών αποτελούν το σύνολο των κανόνων που καθορίζει την σημασιολογία της γλώσσας στο επίπεδο των ατόμων και έχουν την μορφή:

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

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

Οι κανόνες για κάθε ένα από τους τελεστές της γλώσσας σε ατομικό επίπεδο αντιστοιχούν με αυτούς της WSCCS και ορίζονται στον πιο κάτω πίνακα:

Τελεστής Διαδοχής:

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

Τελεστής Επιλογής:

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

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

Πίνακας 3.3 : Κανόνες Σημασιολογίας της PALPS σε ατομικό επίπεδο

Ακολουθεί επεξήγηση του κάθε κανόνα σημασιολογίας της γλώσσας. Όλοι οι κανόνες αναφέρονται στο πώς συμπεριφέρεται ένα μόνο άτομο κάποιου πληθυσμού:

- Τελεστής Διαδοχής: Σύμφωνα με τον τελεστή 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 του τελεστή 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 (Ε + F), μπορούμε να επιλέξουμε την Ε και με την εκτέλεση της ενέργειας α να φτάσουμε στην διεργασία Ε'.

  Αντίστοιχα σύμφωνα με τον δεύτερο κανόνα του τελεστή 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 (E+F), μπορούμε να επιλέξουμε την F και με την εκτέλεση της ενέργειας α να φτάσουμε στην διεργασία F'.

Μέχρι στιγμής έχω παρουσιάσει την σημασιολογία της γλώσσας σε ατομικό επίπεδο χωρίς της ύπαρξη πιθανοτήτων. Σε επίπεδο πληθυσμών η σημασιολογία μπορεί να δοθεί με την βοήθεια των αλυσίδων Markov και των πιθανοτήτων. Ένα πληθυσμιακό σύστημα μπορεί να θεωρηθεί η συνύπαρξη ενός ή περισσότερων πληθυσμών σε ένα κόσμο. Η ύπαρξη πολλών ατόμων στον ίδιο κόσμο, έχει σαν αποτέλεσμα από μια συγκεκριμένη κατάσταση να υπάρχουν περισσότερες από μια πιθανές καταστάσεις, οι οποίες μπορεί να συμβούν με κάποια πιθανότητα. Οι αλυσίδες Markov όπως παρουσιάστηκαν στο υποκεφάλαιο 2.7, μπορούν να παρουσιάσουν τις καταστάσεις στις οποίες μπορεί να βρεθεί ένας ή περισσότεροι πληθυσμοί που συνυπάρχουν στον κόσμο, καθώς επίσης και τις πιθανολογικές μεταβάσεις με τις οποίες μια κατάσταση του συστήματος εξελίσσεται σε άλλη κατάσταση, σε μορφή γράφου. Στο επόμενο υποκεφάλαιο δίνεται ο τρόπος αναπαράστασης ενός πληθυσμιακού συστήματος στην PALPS, ενώ δίνονται επίσης οι κανόνες που καθορίζουν την σημασιολογία της γλώσσας στο επίπεδο των πληθυσμών. Η σημασιολογία στο επίπεδο των πληθυσμών, μπορεί να μεταφράζει τις διεργασίες σε μια αλυσίδα Markov, ώστε να μπορούμε στην συνέχεια να βρούμε τις πιθανότητες με τις οποίες μπορεί να προκύψει μια συγκεκριμένη κατάσταση του συστήματος. Με αυτό τον τρόπο επιτυγχάνεται η ανάλυση της συμπεριφοράς ενός συστήματος πληθυσμών.

3.2 Αναπαράσταση Πληθυσμιακού Συστήματος στην PALPS

Στο προηγούμενο υποκεφάλαιο έχω ορίσει τον τρόπο που ένα άτομο συμπεριφέρεται μέσω των κανόνων σημασιολογίας της γλώσσας. Για την μελέτη όμως ενός ή και περισσότερων πληθυσμών ταυτόχρονα, απαιτείται η σύνθεση των ατόμων των πληθυσμών με την βοήθεια του τελεστή ταυτοχρονισμού. Σε αυτό το υποκεφάλαιο θα παρουσιάσω τον τρόπο που μέσω της σύνταξης και σημασιολογίας της PALPS μπορούμε να μελετήσουμε και να αναπαραστήσουμε τον τρόπο που πολλά άτομα του

ίδιου ή και διαφορετικών πληθυσμών αλληλεπιδρούν εκτελώντας διαφορετικές ενέργειες.

Κύριως στόχος της διπλωματικής μου εργασίας ηταν να εντάξω σε μια άλγεβρα διεργασιών την έννοια του χώρου. Θεωρώ έτσι ότι το κάθε άτομο, που απεικωνίζεται από την διεργασία Ε χαρακτηρίζεται από την τοποθεσία στην οποία βρίσκεται ℓ_i, τον τύπο του πληθυσμού στον οποίο ανήκει p_i και τα άτομα που το έχουν μεχρι στιγμής μολύνει Inf_i και συμβολίζεται σαν:

$$E_i[\ell_i, p_i, Inf_i]$$

Στο ατομικό επίπεδο ισχύει ότι ένα ένα άτομο Ε εκτελώντας την ενέργεια α καταλήγει σε μια κατάσταση Ε'. Αντίστοιχα στο επίπεδο πληθυσμών το άτομο που τώρα απεικονίζεται ως $E_i[\ell_i, p_i, Inf_i]$, ισχύει ότι μπορεί να εκτελέσει την ενέργεια α και να καταλήξει στην κατάσταση $E_i'[\ell_i', p_i, Inf_i]$. Σε περίπτωση που η ενέργεια α είναι η ενέργεια της μετακίνησης, σημαίνει ότι η νέα τοποθεσία ℓ_i'πρέπει να είναι γειτονική με την προηγούμενη ℓ_i τοποθεσία. Σε αντίθετη περίπτωση η τοποθεσία του ατόμου δεν μεταβάλλεται. Τα πιο πάνω παρουσιάζονται σε μορφή κανόνων στον πιο κάτω πίνακα:

$$\frac{E \xrightarrow{\alpha} E'}{E_i[\ell_i, p_i, Inf_i] \xrightarrow{\alpha} E_i'[\ell_i, p_i, Inf_i]}$$

$$\frac{E \xrightarrow{move} E'}{E_i[\ell_i, p_i, Inf_i] \xrightarrow{\text{move}} E_i'[\ell_i', p_i, Inf_i]} \ (\ell_i, \ell_i') \in \text{Neighbors}$$

Πίνακας 3. :Συσχετισμός ατόμου σε ατομικό επίπεδο με ατόμου σε πληθυσμιακό επίπεδο

Σε επίπεδο πληθυσμών ένα σύνολο ατόμων N_1, έχει την ικανότητα να συγχρονίζει την εκτέλεση κάποιας ενέργειάς του με την εκτέλεση της ίδιας ή διαφορετικής ενέργειας

κάποιου άλλου συνόλου ατόμων N_2. Αυτό έχει σαν αποτέλεσμα να υπάρχει ο τελεστής ταυτοχρονισμού (|) η σημασιολογία για τον οποίο δίνεται πιο κάτω:

Τελεστής Ταυτοχρονισμού:

$$1)\quad \frac{N_1 \xrightarrow{\alpha} N_1'}{N_1 \mid N_2 \xrightarrow{\alpha} N_1' \mid N_2}$$

$$2)\quad \frac{N_2 \xrightarrow{\alpha} N_2'}{N_1 \mid N_2 \xrightarrow{\alpha} N_1 \mid N_2'}$$

$$3)\quad \frac{N_1 \xrightarrow{\alpha} N_1' \ ,\ N_2 \xrightarrow{\bar{\alpha}} N_2'}{N_1 \mid N_2 \xrightarrow{\tau} N_1' \mid N_2'}$$

Πίνακας 3. : Τελεστής Ταυτοχρονισμού σε πληθυσμιακό επίπεδο

- Τελεστής Ταυτοχρονισμού: Σύμφωνα με τον πρώτο κανόνα ταυτοχρονισμού, εάν ισχύει ότι από την διεργασία N_1 μέσω της ενέργειας α μπορούμε να φτάσουμε στην διεργασία N_1', τότε στην περίπτωση ταυτόχρονης εκτέλεσης των διεργασιών N_1 και N_2, με την εκτέλεση της ενέργειας α μπορούμε να φτάσουμε στην κατάσταση ταυτοχρονισμού της διεργασίας N_1' με την διεργασία N_2.

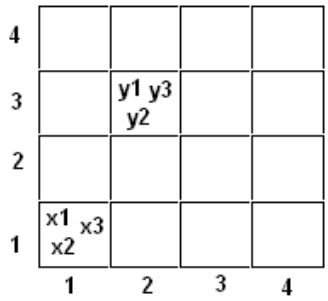
  Αντίστοιχα στον δεύτερο κανόνα ταυτοχρονισμού, εάν ισχύει ότι από την διεργασία N_2 μέσω της ενέργειας α μπορούμε να φτάσουμε στην διεργασία N_2', τότε στην περίπτωση ταυτόχρονης εκτέλεσης των διεργασιών N_1 και N_2, με την εκτέλεση της ενέργειας α μπορούμε να φτάσουμε στην κατάσταση ταυτοχρονισμού της διεργασίας N_1με της διεργασίας N_2'.

  Στον τρίτο κανόνα ταυτοχρονισμού πραγματοποιείται η επικοινωνία μέσω εσωτερικής ενέργειας. Εάν ισχύει ότι από την διεργασία N_1 μέσω της ενέργειας εισόδου α μπορούμε να φτάσουμε στην διεργασία N_1' και ότι από την διεργασία N_2 μέσω της ενέργειας εξόδου $\bar{\alpha}$ μπορούμε να φτάσουμε στην διεργασία

N_2',τότε με εσωτερική ενέργεια κατά τον ταυτοχρονισμό των διεργασιών N_1 και N_2,μπορούμε να φτάσουμε στην κατάσταση $N_1' \mid N_2'$, μέσω της εσωτερικής ενέργειας που πραγματοποιείται λόγο της ταυτόχρονης εκτέλεσης των ενεργειών α και $\overline{\alpha}$.

Ένα πληθυσμιακό σύστημα μπορεί να θεωρηθεί ως μια σύνθεση από τοποθεσίες πάνω στις οποίες συνυπάρχουν διάφοροι πληθυσμοί. Οι τοποθεσίες μπορούν να απεικονιστούν σαν ένα n×n πλέγμα, το οποίο θα αναφέρεται σαν ο 'κόσμος'. Ο κόσμος έχει κυκλικές ιδιότητες ώστε όταν ένα άτομο αφήσει την κάτω ή την αριστερά πλευρά του κόσμου, να επανεμφανίζεται στην πάνω και την δεξιά πλευρά αντίστοιχα.

Πιο κάτω απεικονίζεται ένας 4×4 κόσμος, στον οποίο συνυπάρχουν 2 διαφορετικοί πληθυσμοί:



Σχήμα 3.2: Αναπαράσταση 2 πληθυσμών σε 4x4 κόσμο

Τα πιο πάνω μπορούν να αναπαρασταθούν με την βοήθεια του τελεστή ταυτοχρονισμού, σύμφωνα με τον οποίο εκτελούνται παράλληλα οι ενέργειες κάθε ενός από τα 6 άτομα που συνυπάρχουν στον κόσμο. Δημιουργείται έτσι η κατάσταση N:

$$N ::= E_1[\ell_6, p_1, Inf_1] \mid E_2[\ell_6, p_1, Inf_2] \mid E_3[\ell_6, p_1, Inf_3]$$
$$\mid E_4[\ell_{13}, p_2, Inf_4] \mid E_5[\ell_{13}, p_2, Inf_5] \mid E_6[\ell_{13}, p_2, Inf_6]$$

Το κάθε $E_i[\ell_i, p_i, Inf_i]$ αντιστοιχεί σε ένα άτομο που βρίσκεται στην ℓ_i τοποθεσία. Σε αυτό το σημείο λόγω συνύπαρξης του ατόμου με άτομα άλλων πληθυσμών απαιτείται το κάθε E_i άτομο εκτός από την τοποθεσία του να κρατά πληροφορίες για το σε ποιο πληθυσμό ανήκει (p_i), καθώς επίσης και από ποιους ιούς έχει μέχρι στιγμής μολυνθεί.

Με την βοήθεια της κατάστασης N μπορούμε να δημιουργούμε συστήματα τα οποία να αποτελούνται από πολλά άτομα του ίδιου ή και διαφορετικού πληθυσμού έτσι ώστε να μπορούμε να απεικονίσουμε και να παρατηρήσουμε τον τρόπο που όλα τα άτομα αυτά αλληλεπιδρούν μεταξύ τους.

Οι διαφορετικοί πληθυσμοί που συνυπάρχουν στον κόσμο δημιουργούν το σύνολο $S=\{s_1, s_2, \dots, s_k\}$. Κάθε ένα από τα $s_1, s_2, \dots, s_k$ αντιπροσωπεύουν ένα πληθυσμό συγκεκριμένης κατηγορίας. Κάθε πληθυσμός μπορεί να ανήκει σε διαφορετική κατηγορία. Για παράδειγμα υπάρχουν αρπακτικοί-καταστροφικοί πληθυσμοί και δηλητηριώδης πληθυσμοί. Εάν ένας πληθυσμός s_j είναι δηλητηριώδες τότε τον ονομάζουμε ιό.

Ο κάθε πληθυσμός s_i που ανήκει στην κατηγορία των δηλητηριωδών πληθυσμών συσχετίζεται με την συνάρτηση infect, η οποία καθορίζει σε ποιους πληθυσμούς ο s_i μπορεί να μεταδώσει το δηλητήριό του μολύνοντάς τους:

$$\text{infect}(s_i) \subseteq S$$

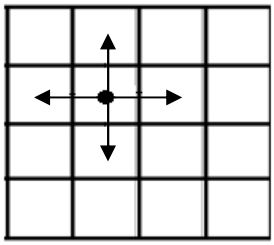
Όπως αναφέρθηκε πιο πριν ο κόσμος στον οποίο είναι κατανεμημένα τα άτομα των διαφόρων πληθυσμών είναι διαχωρισμένος σε διάφορες τοποθεσίες $L_{i,j}$. Επομένως απαιτείται η ύπαρξη μιας ενέργειας χάρη στην οποία παρουσιάζεται το φαινόμενο της μετακίνησης (move) των ατόμων από μια τοποθεσία σε άλλη. Στο σημείο αυτό φαίνεται η ύπαρξης της σχέσης γειτνίασης Neighbors η οποία ορίστηκε στο προηγούμενο υποκεφάλαιο, και η οποία καθορίζει σε κάθε άτομο που βρίσκεται στην τοποθεσία ℓ προς ποιες κατευθύνσεις μπορεί να κινηθεί.

Δεδομένου ότι οι πληθυσμοί με τους οποίους ασχολούμαι δεν μπορούν να πετάξουν, θεωρώ ότι ένα άτομο στο χώρο $L_{i,j,}$ μπορεί να μετακινηθεί μόνο στις τέσσερις τοποθεσίες που βρίσκονται γύρω του και απέχουν απόσταση ένα ή μπορεί να παραμείνει στο σημείο που βρίσκεται. Ισχύει ότι το άτομο στο $L_{i,j}$ μπορεί να μετακινηθεί σε οποιαδήποτε τοποθεσία $L_{i',j'}$ για την οποία ισχύει ότι $|i - i'| \leq 1$ και $j=j'$ ή $i=i'$ και $|j - j'| \leq 1$ ή να παραμείνει στην ίδια τοποθεσία όπου $i=i'$ και $j=j'$. Συγκεκριμένα οι τοποθεσίες στις οποίες μπορεί να μετακινηθεί ένα άτομο από την τοποθεσία $L_{i,j}$ εκτός

από το να παραμείνει στην ίδια θέση είναι όπως ορίζονται και στην σχέση Neighbors οι εξής:

- $L_{i-1, j}$, η τοποθεσία πάνω από την τοποθεσία που βρίσκεται
- $L_{i+1, j}$, η τοποθεσία κάτω από την τοποθεσία που βρίσκεται
- $L_{i, j+1}$, η τοποθεσία δεξιά από την τοποθεσία που βρίσκεται
- $L_{i, j-1}$, η τοποθεσία αριστερά από την τοποθεσία που βρίσκεται
- $L_{i, j}$, η ίδια τοποθεσία στην οποία βρίσκεται

Πιο κάτω παρουσιάζονται σε σχήμα οι επιλογές που έχει ένα άτομο σε συγκεκριμένη τοποθεσία όσο αφορά την μετακίνησή του. Οι πιθανότητες είτε να κινηθεί προς οποιαδήποτε κατεύθυνση, είτε να παραμείνει στο ίδιο σημείο είναι ίσες με 1/5. Για να αποφευχθούν επικαλύψεις θεωρούμε ότι το n είναι τουλάχιστον ίσο με το τρία :



Σχήμα 3.3: Δυνατοί τρόποι μετακίνησης του ατόμου στο χώρο $L_{1,1}$

Στην συνέχεια παρουσιάζονται οι διάφορες ενέργειες που μπορεί να εκτελέσουν τα άτομα των πληθυσμών του κόσμου και να αλλάξουν την κατάσταση του συστήματός τους. Καταρχήν έχουν την ικανότητα της μετακίνησης τους, σε άλλες τοποθεσίες του χώρου με την διαδικασία της μετακίνησης (Move) και την ικανότητα της αναπαραγωγής μέσω της διαδικασίας της αναπαραγωγής (Reproduce). Επιπλέον έχουν την ιδιότητα να μολύνουν άτομα άλλων πληθυσμών μέσω της διαδικασίας της μόλυνσης (Infect). Τέλος για οποιοδήποτε λόγο ανά πάσα στιγμή τα άτομα ενός πληθυσμού μπορεί να πεθάνουν μέσω της διαδικασίας θανάτου (Die). Όλες οι διαδικασίες μπορεί να εκτελεστούν με συγκεκριμένες πιθανότητες.

Για ευκολία της απεικόνισης των ενεργειών θα χρησιμοποιηθεί η κατάσταση N που ορίστηκε στην αρχή του υποκεφαλαίου, η οποία συνθέτει τα άτομα του συστήματος.

Για πιο ευανάγνωστες εξισώσεις το σύμβολο $\prod_{i \in I} \mathrm{E_i}[\ell_\mathrm{i}, \mathrm{p_i}, \mathrm{Inf_i}]$, αναφέρεται σε όλα τα άτομα και τις τοποθεσίες του κόσμου.

Επίσης η κάθε τοποθεσία αντί να περιγράφεται από την γραμμή και στήλη του πίνακα που αντιστοιχεί στον κόσμο, $\mathrm{L_{i,j}}$ όπου i={0, n-1} και j={0, n-1}, παρουσιάζεται σαν ℓ_i , όπου i={1, …,n×n}.

Επιπρόσθετα σε όλες τις ενέργειες όταν κάποιο άτομο δεν μπορεί να εκτελέσει κάποια ενέργεια α τότε συμβολίζω αυτή την ανικανότητα με τον πιο κάτω τρόπο:

$$\forall \mathrm{i} \in \mathrm{I}:\ \mathrm{E_i}[\ell_\mathrm{i}, \mathrm{p_i}, \mathrm{Inf_i}] \xrightarrow{\alpha} \mathrm{E_i}[\ell_\mathrm{i}, \mathrm{p_i}, \mathrm{Inf_i}]$$

Αφού το άτομο $\mathrm{E_i}[\ell_\mathrm{i}, \mathrm{p_i}, \mathrm{Inf_i}]$ δεν εκτελεί την ενέργεια α παραμένει στην αρχική του κατάσταση.

Πιο κάτω παρουσιάζονται με λεπτομέρεια οι κανόνες για κάθε ενέργεια ξεχωριστά, που καθορίζουν την σημασιολογία της γλώσσας σε επίπεδο πληθυσμών:

Ενέργεια Μετακίνησης - Move:

Κάθε άτομο έχει την δυνατότητα με την ενέργεια της μετακίνησης να μετακινείται από την τοποθεσία στην οποία βρίσκεται, προς κάποια γειτονική τοποθεσία όπως αυτή ορίζεται από την σχέση Neighbors. Σε κάθε εκτέλεση της ενέργειας της μετακίνησης το σύνολο I των ατόμων που συνυπάρχουν στον κόσμο, μπορεί να διαχωριστεί σε δυο υποσύνολα. Στο σύνολο των ατόμων που μπορούν να εκτελέσουν την ενέργεια μετακίνησης που συμβολίζεται με $\mathrm{I_1}$ και στο σύνολο των ατόμων που δεν μπορούν να εκτελέσουν την ενέργεια της μετακίνησης, που συμβολίζεται με το $\mathrm{I_2}$. Ισχύει δηλαδή ότι $\mathrm{I} = \mathrm{I_1} \cup \mathrm{I_2}$.

Ο κανόνας της μετακίνησης των ατόμων των πληθυσμών που συνυπάρχουν στον κόσμο και συνθέτουν ένα σύστημα είναι ο εξής:

$$I = I_1 \cup I_2$$

$$\forall i \in I_1 : E_i[\ell_i, p_i, Inf_i] \xrightarrow{\text{move}} E_i'[\ell_i', p_i, Inf_i]$$

$$\forall i \in I_2 : E_i[\ell_i, p_i, Inf_i] \xrightarrow{\cancel{\text{move}}} E_i[\ell_i, p_i, Inf_i]$$

$$\frac{\pi = p(\ell_1, \ell_1') \times \dots \times p(\ell_k, \ell_k') \qquad \forall(\ell_1', \dots, \ell_k') : (\ell_i, \ell_i') \in \text{Neighbors}}{\prod_{i \in I} E_i[\ell_i, p_i, Inf_i] \xrightarrow{\pi} \prod_{i \in I_1} E_i'[\ell_i', p_i, Inf_i] \mid \prod_{i \in I_2} E_i[\ell_i, p_i, Inf_i]} \text{ move}$$

Επομένως για μετά την εκτέλεση της ενέργειας της μετακίνησης με πιθανότητα π το σύνολο των ατόμων που συνυπάρχουν στον κόσμο απεικονίζεται από δύο υποσύνολα που δημιουργούνται: το υποσύνολο $\prod_{i \in I_1} E_i'[\ell_i', p_i, Inf_i]$το οποίο αναφέρεται στα άτομα που μετακινήθηκαν και έφτασαν σε μια γειτονική τοποθεσία τους, αλλάζοντας κατάσταση και το υποσύνολο $\prod_{i \in I_2} E_i[\ell_i, p_i, Inf_i]$ στο οποίο τα άτομα παραμένουν στην κατάσταση που ήταν αφού δεν έχουν εκτελέσει την ενέργεια της μετακίνησης.

Η πιθανότητα π με την οποία πραγματοποιείται η ενέργεια της μετακίνησης των ατόμων του συστήματος, είναι το γινόμενο των πιθανοτήτων της μετακίνησης κάθε ατόμου ξεχωριστά. Κάθε $p(\ell_1, \ell_1')$ αναφέρεται στην πιθανότητα ένα άτομο να μετακινηθεί από την ℓ_1τοποθεσία προς την ℓ_1', αρκεί οι δύο αυτές τοποθεσίες να είναι γειτονικές όπως ορίζεται από την σχέση Neighbors. Οι πιθανότητες αυτές αποτελούν στοιχεία τα οποία δίνονται από το άτομο που καθορίζει το μοντέλο. Αποτελούν δηλαδή παραμέτρους.

Ενέργεια Αναπαραγωγής – Reproduce:

Άλλο ένα φαινόμενο το οποίο μπορεί να εμφανιστεί κατά την συνύπαρξη πληθυσμών είναι η αναπαραγωγή. Ένα άτομο από οποιοδήποτε πληθυσμό, σε οποιαδήποτε τοποθεσία, έχει την ικανότητα να αναπαράγεται. Σε κάθε εκτέλεση της ενέργειας της αναπαραγωγής το σύνολο I των ατόμων που συνυπάρχουν στον κόσμο, μπορεί να διαχωριστεί σε δυο υποσύνολα. Στο σύνολο των ατόμων που μπορούν να εκτελέσουν την ενέργεια της αναπαραγωγής που συμβολίζεται με I_1 και στο σύνολο των ατόμων

που δεν μπορούν να εκτελέσουν την ενέργεια της αναπαραγωγής, που συμβολίζεται με το I_2. Επομένως, ισχύει ότι $I = I_1 \cup I_2$.

Καταρχήν θεωρώ ότι για να δημιουργηθούν απόγονοι ενός πληθυσμού σε μια τοποθεσία, απαιτείται να υπάρχει τουλάχιστον ένα άτομο από τον συγκεκριμένο πληθυσμό στην συγκεκριμένη τοποθεσία. Στην αντίθετη περίπτωση οι απόγονοι που παράγονται για τον συγκεκριμένο πληθυσμό είναι 0. Η πιθανότητα παραγωγής α απογόνων στην τοποθεσία i, από άτομα του πληθυσμού s_j δίνεται ως εξής: $f_{s_j}(\alpha_i)$. Η κάθε πιθανότητα ορίζεται από το άτομο που δημιουργά το μοντάλο. Επομένως η συνολική πιθανότητα με την οποία μπορεί να ολοκληρωθεί η ενέργεια της αναπαραγωγής είναι το γινόμενο των πιθανοτήτων παραγωγής α_i ατόμων από οποιοδήποτε πληθυσμό σε οποιαδήποτε i τοποθεσία.

Με βάση τα πιο πάνω ο κανόνας Reproduce ορίζεται ως εξής:

$$I = I_1 \cup I_2$$

$$\forall i \in I_1 : E_i[\ell_i, p_i, Inf_i] \xrightarrow{\text{reproduce}} E_i'[\ell_i, p_i, Inf_i]$$

$$\forall i \in I_2 : E_i[\ell_i, p_i, Inf_i] \xrightarrow{\text{~~reproduce~~}} E_i[\ell_i, p_i, Inf_i]$$

$$\pi = f_{s_j}(\alpha_1) \times \dots \times f_{s_j}(\alpha_k) \qquad \forall(\alpha_1, \dots, \alpha_k):\ f_{s_j}(\alpha_i) > 0$$

$$\frac{}{E_1[\ell_i, p_i, Inf_i] \mid E_2[\ell_i, p_i, Inf_i] \xrightarrow{\pi} E_1'[\ell_i, p_i, Inf_i] \mid E_2'[\ell_i, p_i, Inf_i] \mid \prod_{i\in\{1,2\}} \underbrace{E_i[\ell_i, p_i, Inf_i] \mid \dots \mid E_i[\ell_i, p_i, Inf_i]}_{\alpha_i}} \; reproduce$$

Άρα με το τέλος της εκτέλεσης της ενέργειας της μετακίνησης ο κόσμος αποτελείται από τα άτομα που εκτέλεσαν την ενέργεια της αναπαραγωγής, ανήκουν δηλαδή στο σύνολο I_1, και έφτασαν έτσι σε μια νέα κατάσταση $E_1'[\ell_i, p_i, Inf_i]$ χωρίς να αλλάξουν τοποθεσία, από το σύνολο των ατόμων που δεν εκτέλεσαν την ενέργεια και συνεπώς δεν άλλαξαν κατάσταση και όλα τα νέα άτομα τα οποία παράχθηκαν κατά την αναπαραγωγή. Τα άτομα αυτά είναι του τύπου του ατόμου που τα δημιούργησε. Αν δηλαδή ο πατέρας άνηκε σε ένα πληθυσμό Q τότε και οι απόγονοί του θα είναι τύπου

Q. Το κάθε α_i παρουσιάζει πόσα νέα άτομα δημιουργήθηκαν στην τοποθεσία ℓ_i από ένα συγκεκριμένο πληθυσμό.

Ενέργεια Μόλυνσης – Infect:

Στο σύνολο S των διαφόρων πληθυσμών που συνυπάρχουν στον κόσμο, υπάρχουν πληθυσμοί οι οποίοι έχουν την ιδιότητα να μολύνουν με κάποιο δηλητήριο που εκκρίνουν, κάποιους από τους υπόλοιπους πληθυσμούς. Οι πληθυσμοί αυτοί ανήκουν στο σύνολο V από τα αρχικά της λέξης Virus. Η συνάρτηση infect(s_i) καθορίζει ποιους από τους πληθυσμούς του συνόλου S μπορεί να μολύνει ο s_i πληθυσμός που ανήκει στο σύνολο V. Ισχύει ότι infect(s_i) $\subseteq$ S. Το σύνολο όλων των πληθυσμών που είναι ιοί απεικονίζονται από το σύνολο $I^{\ell,s}$. Το σύνολο των ατόμων που έχουν την ιδιότητα να μολύνουν, είναι δηλαδή ιοί τύπου s, σε συγκεκριμένη τοποθεσία ℓ, απεικονίζεται ως: $I_1^{\ell,s}$.

Επίσης σε κάθε τοποθεσία ℓ μπορεί να υπάρχει ένα σύνολο από υγιή άτομα τα οποία μπορούν να μολυνθούν από τον πληθυσμό s, και απεικονίζεται ως εξής: $I_2^{\ell,s}$. Από τα άτομα αυτού του συνόλου κάποια θα μολυνθούν από τον s και κάποια άλλα όχι. Αυτά τα οποία θα μολυνθούν απεικονίζονται από το σύνολο $J_2^{\ell,s}$. Κάθε $J_2^{\ell,s}$ αποτελεί υποσύνολο των ατόμων που είναι δυνατόν να μολυνθούν από τον s στην τοποθεσία ℓ.

Με βάση τα πιο πάνω ο κανόνας infect μπορεί να οριστεί όπως φαίνεται πιο κάτω:

$$\forall i \in I_1^{\ell,s} : I_1^{\ell,s} = \{i \in I \mid s_i = s, \ell_i = \ell, p_i \in V,\ E_i[\ell_i, s_i, Inf_i] \xrightarrow{\text{infect}} E_i'[\ell_i, s_i, Inf_i]\}$$

$$\forall i \in I_2^{\ell,s} : I_2^{\ell,s} = \{ j \in I \mid \ell_i = \ell, p_i \in \text{infect}(s)\}$$

$$\forall J_2^{\ell,s} \subseteq I_2^{\ell,s}$$

$$\pi = \prod_{\ell \in Loc} \prod_{s \in S} \prod_{\forall i \in J_2^{\ell,s}} p(i,s)$$

$$\frac{}{\prod_{i \in I} E_i[\ell_i, p_i, Inf_i] \xrightarrow{\pi} \prod_{\ell \in Loc} \prod_{s \in S} \left(\prod_{k \in J_2^{\ell,s}} E_k[\ell_k, p_k, Inf_k \cup \{s\}] \mid \prod_{\forall i \in I_1^{\ell,s}} E_i'[\ell_i, s_i, Inf_i] \mid \prod_{\forall i \notin \left(J_2^{\ell,s} \cup I_1^{\ell,s}\right)} E_i[\ell_i, p_i, Inf_i] \right)} \text{ infect}$$

Το σύνολο Loc που παρουσιάζεται στην εξίσωση είναι το σύνολο όλων των τοποθεσιών του κόσμου τον οποίο μελετούμε, ενώ το σύνολο S είναι το σύνολο όλων των πληθυσμών που συνυπάρχουν στον κόσμο που μελετούμε.

Η συνολική πιθανότητα με την οποία μπορεί να εκτελεστεί η ενέργεια της μόλυνσης είναι ίση με το γινόμενο των πιθανοτήτων κάποιος πληθυσμός s να μολύνει σε κάποια τοποθεσία ℓ κάποιο άτομο i από το σύνολο $J_2^{\ell,s}$.

Με το τέλος της ενέργειας της μόλυνσης το σύνολο των ατόμων που συνυπάρχουν στον κόσμο αποτελείται από το σύνολο των ατόμων που μολύνονται των οποίων αλλάζει η παράμετρος Inf αφού ο αριθμός των ατόμων που το μόλυναν αυξάνεται, από το σύνολο των ιών που εκτέλεσαν την ενέργεια της μόλυνσης και προχωρούν σε επόμενη κατάσταση, καθώς επίσης και από το σύνολο των ατόμων που δεν μπορούν να μολυνθούν από κανένα ιό και ούτε αποτελούν ιό, με αποτέλεσμα να παρουσιάζονται όπως ήταν αρχικά αφού η κατάστασή τους δεν επηρεάζεται.

Ενέργεια θανάτου - Die:

Τα άτομα οποιουδήποτε πληθυσμού ανά πάσα στιγμή υπάρχει πιθανότητα να πεθάνουν λόγω ηλικίας, δηλητηρίασης ή οποιουδήποτε άλλου εξωτερικού παράγοντα, όπως είναι η απότομη αλλαγή της θερμοκρασίας. Το αποτέλεσμα του θανάτου ενός ατόμου από μια τοποθεσία είναι η μείωση του μεγέθους του πληθυσμού στον οποίο ανήκει, στην τοποθεσία αυτή. Τα σύνολο των ατόμων I που συνυπάρχουν στον κόσμο χωρίζεται σε δύο υποσύνολα, το I_1 και το I_2. Στο I_1 ανήκουν τα άτομα που μπορούν να εκτελέσουν την ενέργεια του θανάτου ενώ στο I_2 ανήκουν τα άτομα που δεν μπορούν να εκτελέσουν την ενέργεια του θανάτου. Ισχύει ότι $I = I_1 \cup I_2$. Τα άτομα που μπορούν να εκτελέσουν την ενέργεια του θανάτου μπορεί να μην τα καταφέρουν στο τέλος να την εκτελέσουν αφού υπάρχει η έννοια των πιθανοτήτων. Το σύνολο των ατόμων που θα εκτελέσουν σίγουρα την ενέργεια του θανάτου ανήκουν στο σύνολο I_1' το οποίο είναι υποσύνολο του I_1.

Ο κανόνας Die ορίζεται ως εξής:

$$I = I_1 \cup I_2$$

$$\forall i \in I_1 : E_i[\ell_i, p_i, Inf_i] \xrightarrow{die} E_i'[\ell_i, p_i, Inf_i], \quad I_1' \subseteq I_1$$

$$\forall i \in I_2 : E_i[\ell_i, p_i, Inf_i] \xrightarrow{die} E_i[\ell_i, p_i, Inf_i]$$

$$\frac{\pi = \prod_{i \in I_1'} P_{die}(E_i[\ell_i, p_i, Inf_i]) \times \prod_{i \in I_1 - I_1'} (1 - P_{die}(E_i[\ell_i, p_i, Inf_i]))}{\prod_{i \in I} E_i[\ell_i, p_i, Inf_i] \xrightarrow{\pi} \prod_{i \in I_2} E_i[\ell_i, p_i, Inf_i] \mid \prod_{i \in I_1 - I_1'} E_i'[\ell_i, p_i, Inf_i]} die$$

Η πιθανότητα με την οποία ολοκληρώνεται η εκτέλεση της ενέργειας του θανάτου είναι ίση με το γινόμενο των πιθανοτήτων να πεθάνει κάθε ένα από τα άτομα που μπορούν να εκτελέσουν την ενέργεια του θανάτου και ανήκουν στο σύνολο I_1', και των πιθανοτήτων κάθε ατόμου που ανήκει στο I_1, μπορεί να εκτελέσει δηλαδή την ενέργεια του θανάτου αλλά δεν την εκτελεί.

Με το τέλος της εκτέλεσης της ενέργειας του θανάτου, απομένουν στον κόσμο τα άτομα τα οποία δεν μπορούσαν να εκτελέσουν την ενέργεια του θανάτου και τα άτομα που ενώ μπορούνσαν νατη εκτελέσουν, δεν την εκτέλεσαν. Τα υπόλοιπα άτομα διαγράφονται αφού πεθαίνουν. Όταν ένα άτομο πεθάνει θεωρώ ότι η επόμενη κατάσταση στην οποία φτάνει είναι ο τερματισμός 0.

3.3 Παράδειγμα Μοντελοποίησης Πληθυσμιακού Συστήματος

Με την χρήσης της άλγεβρας διεργασιών PALPS της οποία δόθηκε πιο πάνω η σύνταξη και σημασιολογία, μπορούν να μοντελοποιηθούν συστήματα που αποτελούνται από ένα ή περισσότερους πληθυσμούς των οποίων τα άτομα μπορούν να εκτελούν της ενέργειες της μετακίνησης, της αναπαραγωγής, της μόλυνσης και του θανάτου. Πριν την μοντελοποίηση απαιτείται η επιλογή των ενεργειών που θα εκτελούν τα άτομα καθώς επίσης και η σειρά με την οποία θα τις εκτελούν. Στην συνέχεια θα επιτυγχάνεται η

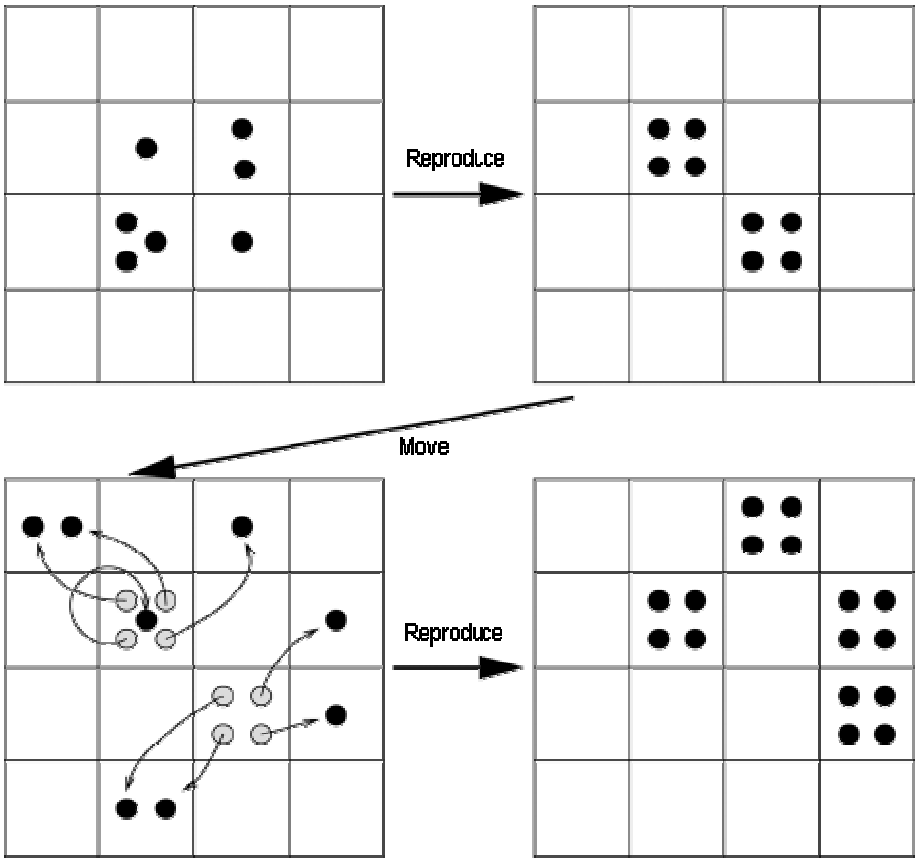
κυκλική εκτέλεση των ενεργειών αυτών ώστε να επιτευχθεί η παρατήρηση και η μελέτη του τρόπου που θα συμπεριφερθούν τα άτομα που συνυπάρχουν στον κόσμο.

Ακολουθεί ένα παράδειγμα ενός τέτοιου συστήματος. Στο παράδειγμα τα άτομα που συνυπάρχουν στον κόσμο έχουν την ικανότητα να αναπαράγονται και στην συνέχεια να μετακινούνται. Οι δύο ενέργειες εκτελούνται συνεχώς εναλλάξ ξεκινώντας από την ενέργεια της αναπαραγωγής.

Κατά την διαδικασία της αναπαραγωγής τα άτομα του πληθυσμού που θα εξεταστεί έχουν την ιδιότητα να αναπαράγονται μόνο εάν στην τοποθεσία που βρίσκονται δεν υπάρχουν άλλα άτομα. Σε περίπτωση που σε μια τοποθεσία υπάρχουν δύο ή περισσότερα άτομα τότε η ενέργεια της αναπαραγωγής αποτυγχάνει να εκτελεστεί. Σε περίπτωση που ένα άτομο εκτελέσει την ενέργεια τότε παράγει τέσσερα νέα άτομα. Επίσης τα άτομα που απομένουν στον κόσμο μετά την εκτέλεση μιας ενέργειας αναπαραγωγής είναι να νέα άτομα που δημιουργούνται στις τοποθεσίες που βρίσκονταν πριν την αναπαραγωγή τα άτομα που τα δημιούργησαν.

Κατά την διαδικασία της μετακίνησης ισχύει ότι κάθε άτομο έχει την ικανότητα να μετακινείται προς οποιαδήποτε τοποθεσία βρίσκεται πάνω, κάτω, δεξιά, αριστερά ή διαγώνια της τοποθεσίας που βρίσκεται το άτομο, ή και να παραμείνει στην ίδια τοποθεσία. Με την διαδικασία της μετακίνησης τα άτομα μπορούν να μετακινηθούν σε τοποθεσία που δεν υπάρχουν άλλα άτομα ώστε να μπορέσουν στην συνέχεια να αναπαραχθούν.

Τα πιο πάνω απεικονίζονται στο πιο κάτω σχήμα, όπου πραγματοποιούνται με την σειρά μια ενέργεια αναπαραγωγής, μια ενέργεια μετακίνησης και ακολουθεί άλλη μια ενέργεια αναπαραγωγής.



Σχήμα 3.4: Απεικόνιση της ενέργειας της αναπαραγωγής και της μετακίνησης

Στην πρώτη εκτέλεση της διαδικασίας της αναπαραγωγής παρατηρείται η παραγωγή τεσσάρων ατόμων στην $_{6}$ και $_{11}$ τοποθεσία, στις οποίες προηγουμένως υπήρχε μόνο ένα άτομο σε κάθε μια. Με την εκτέλεση της αναπαραγωγής στον κόσμο απομένουν μόνο τα 8 νέα άτομα που δημιουργήθηκαν. Στον αρχικό κόσμο στην $_{7}$ και $_{10}$ τοποθεσία επειδή υπήρχαν περισσότερα του ενός ατόμου δεν ήταν δυνατή η δημιουργία νέων ατόμων.

Στον τρίτο πίνακα παρουσιάζεται η κατάσταση του κόσμου μετά την εκτέλεση της ενέργειας της μετακίνησης του κάθε άτομου σε μια από τις γύρω του τοποθεσίες. Υπάρχει η δυνατότητα τα άτομα να παραμείνουν στην ίδια τοποθεσία όπως συμβαίνει με το ένα άτομο στην τοποθεσία στην $_{6}$.

Στον τελευταίο πίνακα ξαναδημιουργείται η κατάσταση που προκύπτει μετά την εκτέλεση της αναπαραγωγής. Και σε αυτό το σημείο παρατηρείται ότι η αναπαραγωγή πραγματοποιείται σε τοποθεσίες που υπάρχει μόνο ένα άτομο.

Στην συνέχεια παρουσιάζεται ο τρόπος που μπορεί το πιο πάνω παράδειγμα να οριστεί στην PALPS:

Το κάθε άτομο που υπάρχει στον αρχικό κόσμο μπορεί να εκτελέσει την ενέργεια της αναπαραγωγής ακολουθούμενη από την ενέργεια της μετακίνησης και να φτάσει ξανά στην ίδια κατάσταση, όπως παρουσιάζεται πιο κάτω:

$$\mathrm{E}\,[\ell\,,\mathrm{p}\,,\mathrm{Inf}\,] = reproduce.move.\,\mathrm{E}\,[\ell\,,\mathrm{p}\,,\mathrm{Inf}\,]$$

Συνθέτοντας όλα τα άτομα που συνυπάρχουν στον κόσμο με την βοήθεια του τελεστή ταυτοχρονισμού δημιουργείται το σύστημα με τον πληθυσμό που μελετάται. Στο πιο πάνω παράδειγμα θεωρώ ότι το σύνολο των ατόμων ανήκουν στον ίδιο πληθυσμό και άρα το p τους είναι το ίδιο. Η τοποθεσία τους εξαρτάται από την τοποθεσία στην οποία βρίσκονται, ενώ το Inf είναι όλων των ατόμων το ίδιο γιατί στο συγκεκριμένο παράδειγμα δεν εξετάζεται η ενέργεια της μόλυνσης. Επομένως το σύστημα μπορεί να δημιουργηθεί από την σύνθεση των ατόμων όπως παρουσιάζεται πιο κάτω:

$\mathrm{System}= \mathrm{E}_1\,[\ell_6\,,\mathrm{p}\,,\mathrm{Inf}\,] \mid \mathrm{E}_2\,[\ell_7\,,\mathrm{p}\,,\mathrm{Inf}\,] \mid \mathrm{E}_3\,[\ell_7\,,\mathrm{p}\,,\mathrm{Inf}\,] \mid \mathrm{E}_4\,[\ell_{10}\,,\mathrm{p}\,,\mathrm{Inf}\,] \mid$
$\mathrm{E}_5\,[\ell_{10}\,,\mathrm{p}\,,\mathrm{Inf}\,] \mid \mathrm{E}_6\,[\ell_{10}\,,\mathrm{p}\,,\mathrm{Inf}\,] \mid \mathrm{E}_7[\ell_{11},\mathrm{p}\,,\mathrm{Inf}\,]$

Κεφάλαιο 4

Σύστημα Υλοποίησης

4.1 Επεξήγηση Συστήματος Υλοποίησης

Στο υποκεφάλαιο αυτό θα παρουσιαστεί με λεπτομέρεια το σύστημα το οποίο υλοποίησα σε γλώσσα προγραμματισμού C. Όπως ανάφερα κατά την παρουσίαση της γλώσσας μου στο υποκεφάλαιο 3.2 του κεφαλαίου 3, μια από τις ενέργειες που μπορεί να εκτελέσει κάθε ένα από τα άτομα του κόσμου είναι η μετακίνηση. Στο πρόγραμμα υλοποίησα την ενέργεια της μετακίνησης, με στόχο την παραγωγή όλων των πιθανών καταστάσεων του κόσμου που μπορεί να δημιουργηθούν με την συνεχή εκτέλεση της ενέργειας της μετακίνησης. Κάθε κατάσταση παρουσιάζεται με την πιθανότητα, με την οποία μπορεί να προκύψει.

Η μετακίνηση μπορεί να αποτελέσει μια σημαντική ενέργεια στην προσπάθεια της επιβίωσης ενός πληθυσμού. Για παράδειγμα ένας πληθυσμός για να μπορέσει να αναπαραχθεί μπορεί να παρουσιάζει την ιδιότητα ότι χρειάζεται να βρίσκεται μόνος του σε μια τοποθεσία. Επομένως για να αναπαραχθεί απαιτείται η μετακίνησή του σε τοποθεσία χωρίς άτομα. Επιπρόσθετα με την διαδικασία της μετακίνησης τα άτομα έχουν την ικανότητα να φτάνουν πιο γρήγορα σε περιοχές στις οποίες υπάρχει τροφή και η οποία θα τους βοηθήσει να επιβιώσουν. Ένα άτομο το οποίο έχει την ιδιότητα να μολύνει άλλα άτομα, πρέπει να έχει την ικανότητα να φτάνει με την μετακίνησή του σε τοποθεσίες που υπάρχουν άτομα που μπορεί να μολύνει. Τα πιο πάνω αποτελούν κίνητρα τα οποία με ώθησαν να υλοποιήσω την ενέργεια της μετακίνησης των ατόμων που συμβιώνουν σε ένα κόσμο.

Η γενική ιδέα της υλοποίησης είναι να εκτελείται συνεχώς η μετακίνηση των ατόμων του κόσμου μέχρι να μην μπορούν να δημιουργηθούν άλλες καινούριες καταστάσεις του. Θεωρώ ότι μια εκτέλεση της ενέργειας της μετακίνησης ολοκληρώνεται, μόνο όταν όλα τα άτομα του κόσμου μετακινηθούν μια φορά. Στο σύστημα που υλοποίησα ο κόσμος στον οποίο συνυπάρχουν τα άτομα είναι ένας 3x3 πίνακας και συνεπώς υπάρχουν εννιά διαφορετικές τοποθεσίες, στις οποίες μπορεί ένα άτομο να βρεθεί. Θεωρώ ότι κάθε άτομο έχει την δυνατότητα να μετακινείται στην πάνω, κάτω, δεξιά ή την αριστερή τοποθεσία από την οποία βρίσκεται με ίση πιθανότητα 1/5. Επίσης με πιθανότητα 1/5 έχει την δυνατότητα αντί να μετακινηθεί προς οποιαδήποτε από τις τέσσερεις κατευθύνσεις να παραμείνει στην τοποθεσία που ήδη βρίσκεται. Ο 3x3 κόσμος θεωρώ ότι 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 υλοποίησής μου μπορεί να διαχωριστεί σε δύο σημαντικά μέρη. Στο πρώτο μέρος επιτυγχάνεται το κτίσιμο ενός ενδιάμεσου δέντρου, που απεικονίζει σαν ακραίους κόμβους όλες τις πιθανές καταστάσεις που μπορεί να δημιουργηθούν με την εκτέλεση της ενέργειας της μετακίνησης στα άτομα ενός αρχικοποιημένου κόσμου που ορίζεται σαν η ρίζα του δέντρου. Στο δεύτερο μέρος επιτυγχάνεται μέσω επανάληψης του κτίσματος του ενδιάμεσου δέντρου, η δημιουργία όλων των πιθανών περιπτώσεων που μπορεί να δημιουργηθούν κατά την συνεχή μετακίνηση των ατόμων που συνυπάρχουν στον κόσμο. Στόχος της παραγωγής όλων αυτών των περιπτώσεων είναι η παρατήρηση της πιθανότητας με την οποία μπορεί να προκύψει κάθε διαφορετική περίπτωση.

Στο πιο κάτω πίνακα παρουσιάζονται οι δομές που χρησιμοποίησα κατά την υλοποίηση του συστήματος:

Δομή για αναπαράσταση ενός ατόμου του πληθυσμού: typedef struct individual{

```
    int move;
    int location[2];
    struct individual* next;
}Individual;

Δομή για αναπαράσταση μιας τοποθεσίας του κόσμου
typedef struct location{
        Individual* list;
        int numIndividuals;
}Location;

Δομή για αναπαράσταση κόμβου στο ενδιάμεσο δέντρο κατά την μετακίνηση όλων των
ατόμων από μια φορά, για την εκτέλεση της ενέργειας της μετακκίνησης:
typedef struct middleNode{
        Location world[WORLD];
        int height;
        float pithanotita;
        struct middleNode* next;
}MiddleNode;

Δομή για απεικόνιση του πίνακα κατακερματισμού που δημιουργείται σε κάθε επίπεδο
του δέντρου.
typedef struct hash{
        int size;
        MiddleNode* head;
}Hash;
```

Πίνακας 4.1: Δομές του προγράμματος υλοποίησης

Η δομή Individual χρησιμοποιείται για την απεικόνιση των ατόμων του πληθυσμού. Η μεταβλητή move υποδηλώνει κατά πόσο το άτομο έχει μετακινηθεί ή όχι. Σε περίπτωση που η μεταβλητή είναι ίση με 0 σημαίνει ότι το άτομο δεν έχει ακόμα μετακινηθεί, ενώ όταν είναι ίση με 1 σημαίνει ότι έχει πραγματοποιήσει την ενέργεια της μετακίνησης. Ο

πίνακας μεγέθους δυο ακεραίων με το όνομα location, χρησιμοποιείται για να κρατούνται οι συντεταγμένες του χώρου [x,y] στον οποίο βρίσκεται ένα άτομο. Το x αντιστοιχεί στον αριθμό της γραμμής του δισδιάστατου κόσμου, ενώ το y στον αριθμό της στήλης. Ο δείκτης next αποτελεί ένα δείκτη προς κάποιο άλλο άτομο του πληθυσμού.

Η δομή Location χρησιμοποιείται για την αναπαράσταση μιας τοποθεσίας του κόσμου. Ο κόσμος απεικονίζεται σαν ένας δισδιάστατος πίνακας τύπου Location. Κάθε θέση του πίνακα αποτελεί μια τοποθεσία η οποία μέσω του ακέραιου numIndividuals κρατά τον αριθμό των ατόμων που βρίσκονται στην συγκεκριμένη τοποθεσία, ενώ με τον δείκτη list δείχνει στον κόμβο με το πρώτο άτομο που βρίσκεται στην τοποθεσία.

Για την ολοκλήρωση της ενέργειας της μετακίνησης απαιτείται η μετακίνηση όλων των ατόμων του χώρου από μια φορά. Η δομή MiddleNode χρησιμοποιείται για την αναπαράσταση των κόμβων στο ενδιάμεσο δέντρο κατά την μετακίνηση των ατόμων. Με την βοήθεια του πίνακα world τύπου location κρατείται μια συγκεκριμένη κατάσταση του κόσμου. Η μεταβλητή height κρατά το ύψος στο οποίο βρίσκεται η συγκεκριμένη κατάσταση του κόσμου στο ενδιάμεσο δέντρο. Ο δείκτης next δείχνει σε επόμενο κόμβο με μια άλλη κατάσταση του κόσμου, ενώ η μεταβλητή pithanotita κρατά την πιθανότητα με την οποία μπορεί να παρουσιαστεί η κατάσταση world του κόσμου.

Τέλος η δομή Hash χρησιμοποιείται κατά την χρήση πινάκων κατακερματισμού στο πρόγραμμα, όπου σε κάθε θέση του πίνακα κατακερματισμού υπάρχει η μεταβλητή size που υποδηλώνει τον αριθμό των στοιχείων που είναι αποθηκευμένα στην συγκεκριμένη θέση. Ο δείκτης head δείχνει στο πρώτο στοιχείο το οποίο είναι αποθηκευμένο στην θέση αυτή.

Στον επόμενο πίνακα δίνονται τα πρωτότυπα των συναρτήσεων που χρησιμοποιούνται στο πρόγραμμα:

```
void CreateFinalTreeMovement(Location kosmos[]);
```

```
void CreateMiddleTreeMovement(MiddleNode* node);
MiddleNode* CreateNewSituation(MiddleNode* prokatoxos, Indivitual* tmp, char go,
int thesiMetakAtomou);
void InsertBackOpen(MiddleNode** open, MiddleNode* komvos);
int HashFunction(MiddleNode* newstate, int epipedo);
int InsertToHashTable(MiddleNode* newstate, Hash* hashtable);
void InsertHash(MiddleNode *newstate, Hash *hashtable, int epipedo);
```

Πίνακας 4.2: Συναρτήσεις του προγράμματος υλοποίησης

Οι σημαντικότερες συναρτήσεις στις οποίες βρίσκεται και η κύρια ιδέα υλοποίησης του προγράμματος είναι η CreateFinalTreeMovement και η CreateMiddleTreeMovement. Αμέσως πιο κάτω γίνεται μια πιο λεπτομερής αναφορά στις συναρτήσεις αυτές καθώς επίσης δίνεται σε μορφή ψευδοκώδικα ο τρόπος λειτουργίας τους.

Στην παράγραφο αυτή θα παρουσιαστεί με ψευδοκώδικα ο αλγόριθμος της υλοποίησης του ενδιάμεσου δέντρου (CreateMiddleTreeMovement) για την εκτέλεση μιας ενέργειας μετακίνησης. Καταρχήν δίνεται σαν παράμετρος στην συγκεκριμένη διαδικασία, η αρχική κατάταξη των ατόμων του πληθυσμού που συνυπάρχουν στον 3x3 κόσμο. Ο ψευδοκώδικας βήμα προς βήμα παρουσιάζεται πιο κάτω:

```
void CreateMiddleTreeMovement(MiddleNode* node)
{
```

1) sumIndivituals = Αριθμός των ατόμων που συνυπάρχουν στον κόσμο του node κόμβου .
2) if (sumIndivituals==0) then
   τύπωσε ανάλογο μήνυμα και τερμάτισε.
3) initial = node. Αντιγραφή του κόσμου του node στον κόμβο initial, ο οποίος αρχικοποιείται σαν ο πρώτος κόμβος της λίστας list.
4) for (repeats=0; repeats<sumIndivituals; repeats++)
   {
   4.1) Εντοπισμός του πρώτου ατόμου που συναντούμε στον κόσμο και δεν έχει ακόμα εκτελέσει την διαδικασία της μετακίνησης. Οι πληροφορίες για το

άτομο αυτό δίνονται μέσω της μεταβλητής thesiMetakAtomou και του δείκτη tmp.

4.2) Δέσμευση χώρου και αρχικοποίηση του πίνακα κατακερματισμού hashtable για να κρατά τις καταστάσεις του επόμενου επιπέδου του δέντρου και του οποίου το μέγεθος είναι πενταπλάσιο από το προηγούμενο.

4.3) while(list!=NULL)

{

4.3.1) Δείξε με την βοήθεια κάποιου δείκτη tmp στο άτομο του κόσμου που θα μετακινηθεί.

4.3.2) Δημιούργησε τις πέντε καταστάσεις που προκύπτουν με την μετακίνηση του ατόμου, προς τα πάνω, κάτω, δεξιά, αριστερά ή παραμονή του στην ίδια τοποθεσία με πιθανότητα 1/5. Για κάθε μια από τις περιπτώσεις καλείται η συνάρτηση CreateNewSituation.

4.3.3) Τοποθέτησε τις νέες αυτές καταστάσεις με τις πιθανότητές τους στον πίνακα κατακερματισμού μέσω της κλήσης της InsertHash. Σε περίπτωση που μια κατάσταση είναι ήδη αποθηκευμένη σε αυτόν, τότε απλά προστίθεται η πιθανότητά της χωρίς να ανατοποθετηθείτε στην πίνακα κατακερματισμού.

}

4.4) Αντιγραφή των δεδομένων του πίνακα κατακερματισμού για το παρών επίπεδο του δέντρου στην λίστα list και αποδέσμευση του χώρου του πίνακα κατακερματισμού.

}

}

Από τον πιο πάνω αλγόριθμο προκύπτει ότι το δέντρο σταματά να κτίζεται όταν φτάσει στο επίπεδο που είναι ίσο με τον αριθμό των ατόμων που συνυπάρχουν στον κόσμο, (βλέπε βήμα 4 στον αλγόριθμο). Σε κάθε επίπεδο επιτυγχάνεται η μετακίνηση ενός ατόμου και όπως αναφέρθηκε νωρίτερα η ενέργεια της μετακίνησης ολοκληρώνεται μόνο όταν όλα τα άτομα του κόσμου, εκτελέσουν την ενέργεια της μετακίνησης από μια φορά.

Επιπλέον σε κάθε επανάληψη για δημιουργία ενός νέου επιπέδου του δέντρου ο μέγιστος αριθμός των καταστάσεων που μπορούν να δημιουργηθούν είναι ο πενταπλάσιος του προηγούμενου επιπέδου. Για κάθε κόμβο του προηγούμενου επιπέδου, που είναι αποθηκευμένοι στην συνδεδεμένη λίστα list, υπολογίζονται στην χειρότερη περίπτωση πέντε επιπλέον κόμβοι, λόγο της μετακίνησης ενός ατόμου. Υπάρχει η πιθανότητα δημιουργίας λιγότερων από πέντε καταστάσεων για το λόγο ότι κάποια ή και κάποιες από τις καταστάσεις μπορεί να έχουν δημιουργηθεί προηγούμενος. Σε μια τέτοια περίπτωση δεν αποθηκεύονται ξανά στον πίνακα κατακερματισμού. Για τον λόγο που εξήγησα ακριβώς πιο πάνω κάθε φορά το μέγεθος του πίνακα κατακερματισμού για το επόμενο επίπεδο πενταπλασιάζεται σε σχέση με τον πίνακα κατακερματισμού του προηγούμενου επιπέδου. Επίσης ο λόγος χρήσης του πίνακα κατακερματισμού είναι για να επιτυγχάνεται πολύ πιο αποδοτικά ο έλεγχος κατά πόσο μία κατάσταση είναι ήδη αποθηκευμένη. Η συνάρτηση κατακερματισμού, ο τρόπος δηλαδή που αποφασίζεται σε ποια θέση του πίνακα κατακερματισμού, θα αποθηκευτεί μια νέα κατάσταση, θα περιγραφεί στην συνέχεια.

Με το τέλος της διαδικασίας της δημιουργίας του ενδιάμεσου δέντρου αυτό που χρειάζεται είναι η λίστα με τις καταστάσεις και τις πιθανότητες του τελευταίου επιπέδου του δέντρου. Για τον λόγο αυτό δεν χρειάζεται να κρατώ δεσμευμένο χώρο της μνήμης για τα ενδιάμεσα βήματα παραγωγής των προηγούμενων επιπέδων. Η λίστα με τις καταστάσεις του τελευταίου επιπέδου αποτελεί και τα δεδομένα εξόδου της διαδικασίας.

Στο σημείο 4.3.2 του ψευδοκώδικα γίνεται κλήση της συνάρτησης CreateNewSituation, σύμφωνα με την οποία επιτυγχάνεται η δημιουργία μιας νέας κατάστασης, στην οποία φτάνει ο κόσμος λόγω της μετακίνηση ενός ατόμου. Η κατάσταση που δημιουργείται επιστρέφεται σαν αποτέλεσμα της κλήσης αυτής της συνάρτησης. Το πρότυπο της συνάρτησης όπως παρουσιάζεται στον πίνακα 4.2 είναι:

```
MiddleNode* CreateNewSituation(MiddleNode* prokatoxos, Indivitual* tmp, char go,
                               int thesiMetakAtomou);
```

Ο κόμβος prokatoxos αποτελεί τον κόμβο με την κατάσταση που θα τροποποιηθεί για την πραγματοποίηση της κίνησης ενός ατόμου. Ο tmp δείκτης και η μεταβλητή thesiMetakAtomou κρατούν τις πληροφορίες για εντοπισμό του ατόμου που θα μετακινηθεί στον κόσμο της κατάστασης του προκατόχου. Το go είναι ένας χαρακτήρας που υποδηλώνει το προς τα ποια γειτονική τοποθεσία θα κινηθεί το άτομο.

Αντίστοιχα για κάθε μια από τις πέντε νέες καταστάσεις που δημιουργούνται κατά την μετακίνηση ενός ατόμου, καλείται η συνάρτηση InsertHash (βλέπε βήμα 4.3.3 του αλγορίθμου). Η συνάρτηση αυτή πραγματοποιά την αποθήκευση της κατάστασης του κόσμου στην σωστή θέση ενός πίνακα κατακερματισμού εάν δεν είναι ήδη αποθηκευμένη σε αυτόν. Σε περίπτωση που μια κατάσταση είναι ήδη αποθηκευμένη τότε απλά προστίθεται η πιθανότητα στην μέχρι στιγμής πιθανότητα της κατάστασης αυτής . Το πρώτυπο της συνάρτησης είναι:

```
void InsertHash(MiddleNode *newstate, Hash *hashtable, int epipedo);
```

Η newstate είναι η κατάσταση η οποία θα αποθηκευτεί στον πίνακα κατακερματισμού με το όνομα hashtable. Η μεταβλητή epipedo χρησιμοποιείται στην συνάρτηση κατακερματισμού η οποία θα περιγραφεί στην συνέχεια.

Συνεχίζοντας με την δεύτερη σημαντικότερη συνάρτηση της υλοποίησης, την CreateFinalTreeMovement , στόχος είναι η δημιουργία όλων των πιθανών περιπτώσεων που μπορεί να δημιουργηθούν κατά την συνεχή μετακίνηση των ατόμων που συνυπάρχουν στον κόσμο. Λόγος της παραγωγής όλων αυτών των περιπτώσεων είναι η παρατήρηση της πιθανότητας με την οποία μπορεί να προκύψει κάθε διαφορετική κατάσταση του κόσμου. Αυτό μπορεί να βοηθήσει σε μελέτες ανάλυσης της συμπεριφοράς ενός πληθυσμού. Αμέσως πιο κάτω παρουσιάζεται ο αλγόριθμος της συνάρτησης CreateFinalTreeMovement σε μορφή ψευδοκώδικα:

```
void CreateFinalTreeMovement(Location kosmos[]) {
```

1) Δέσμευση χώρου και αρχικοποίηση του πίνακα κατακερματισμού των τελικών αποτελεσμάτων hashfinal.

2) initial=kosmos. Αρχικοποίηση της αρχικής κατάστασης μέσω του τυχαίου αρχικοποιημενου κόσμου kosmos[].
3) Κλήση της συνάρτησης InsertBackOpen για τοποθέτηση της αρχικής κατάστασης στην συνδεδεμένη λίστα open, με τις ανοικτές καταστάσεις.
4) Κληση της InsertToHashTable για τοποθέτηση της αρχικής κατάστασης στον τελικό πίνακα κατακερματισμού με τα τελικά αποτελέσματα.
5) while(open!=NULL)
   {
   5.1) CreateMiddleTreeMovement(open); Κλήση της συνάρτησης για δημιουργία του ενδιάμεσου δέντρου της κατάστασης που βρίσκεται στον κόμβο open και επιστροφή της λίστας list.
   5.2) while (list!=NULL)
   {
   5.2.1) toptothetisi = InsertToHashTable(list, hashfinal); Τοποθέτησε τον κόμβο της λίστας list στον τελικό πίνακα κατακερματισμού. toptothetisi = 1 εάν γίνει η αποθήκευση. Εάν υπάρχει ήδη αποθηκευμένη η κατάσταση τότε toptothetisi = 0 .
   5.2.2) if (toptothetisi==1) then
   Κλήση της InsertBackOpen για να προστεθεί η κατάσταση στο τέλος της λίστας open και να επεξεργαστεί στην συνέχεια.
   }
   }
6) Τύπωμα των τελικών αποτελεσμάτων που βρίσκονται στο hashfinal.

}

Αποτέλεσμα της πιο πάνω διαδικασίας είναι ο τελικός πίνακας κατακερματισμού, ο οποίος περιέχει όλες τις πιθανές καταστάσεις που μπορούν να δημιουργηθούν σε ένα κόσμο στον οποίο εκτελείται συνεχώς η ενέργεια της μετακίνησης, μέχρι να μην δημιουργούνται άλλες νέες καταστάσεις. Κάθε μια από τις πιθανές αυτές καταστάσεις συνοδεύεται με την πιθανότητά με την οποία μπορεί να προκύψει.

Η λίστα open που χρησιμοποιείται στον αλγόριθμο αποτελεί μια απλά συνδεδεμένη λίστα στην οποία αποθηκεύονται οι ανοικτές καταστάσεις. Οι καταστάσεις δηλαδή για τις οποίες δεν έχει κληθεί ακόμα η συνάρτηση δημιουργίας του ενδιάμεσού τους δέντρου στο βήμα 5.1 του αλγορίθμου. Στο βήμα 3 και στο βήμα 5.2.2 του αλγορίθμου γίνεται κληση της συνάρτησης InsertBackOpen. Το πρώτότυπο της συνάρτησης είναι:

void InsertBackOpen(MiddleNode** open, MiddleNode* komvos)

Στόχος της συνάρτησης αυτής είναι η αποθήκευση του κόμβου komvos στο τέλος της λίστας open ώστε στη συνέχεια ο κόμβος να τύχει επεξεργασίας.

Ο λόγος για τον οποίο ο πιο πάνω αλγόριθμος τερματίζει είναι ότι μετά από κάποιες επαναλήψεις οι περιπτώσεις του κόσμου που θα δημιουργούνται θα είναι ήδη αποθηκευμένες στον τελικό πίνακα κατακερματισμού με αποτέλεσμα να μην προσθέτονται άλλοι κόμβοι στην open. Επομένως αυτό σημαίνει ότι θα αφαιρούμε κόμβους για να τους επεξεργαστούμε χωρίς να προσθέτουμε άλλους, με αποτέλεσμα μετά από κάποιες επαναλήψεις η λίστα open να αδειάσει.

Στο βήμα 4 και στο βήμα 5.2.1 του αλγορίθμου της CreateFinalTreeMovement γίνεται κληση της συνάστησης InsertToHashTable:

int InsertToHashTable(MiddleNode* newstate,Hash* hashtablef)

Η συνάρτηση αυτή προσθέτει την κατάσταση newstate στον πίνακα κατακερματισμού hashtablef εάν δεν υπάρχει ήδη. Η συνάστηση επιστρέφει την τιμή 1 εάν η κατάσταση αποθηκευτεί, ενώ εάν η κατάσταση υπάρχει ήδη στο hashtablef τότε επιστρέφεται η τιμή 0.

Εκτός από την υλοποίηση των πιο πάνω συναρτήσεων γίνεται χρήση και της συνάρτησης:

int HashFunction(MiddleNode* newstate, int epipedo);

Η HashFunction είναι η συνάρτηση κατακερματισμού η οποία χρησιμοποιήθηκε σε όλες τις περιπτώσεις χρήσης πίνακα κατακερματισμού. Σύμφωνα με την συνάρτηση αυτή αποφασίζεται η θέση του πίνακα κατακερματισμού στην οποία θέλουμε να αποθηκεύσουμε μια κατάσταση εάν στην λίστα της συγκεκριμένης θέσης δεν βρίσκεται ήδη αποθηκευμένη. Η συνάρτηση είναι η εξής: Πραγματοποίηση πολλαπλασιασμού των θέσεων κάθε ενός από τα άτομα που υπάρχουν στο χώρο και υπολογισμός του υπολοίπου (mod) σε σχέση με το μέγεθος του πίνακα κατακερματισμού.

4.2 Αποτελέσματα Υλοποίησης

Κατά την έναρξη του προγράμματος που υλοποίησα, ο 3x3 κόσμος αρχικοποιείται με τυχαίο αριθμό ατόμων τα οποία κατανέμονται επίσης τυχαία στις εννιά τοποθεσίες. Όσο πιο μεγάλος είναι ο αριθμός των ατόμων τόσο και πιο πολύ μεγαλώνει το σύνολο των καταστάσεων του κόσμου που λαμβάνονται σαν αποτέλεσμα.

Για λόγους παρουσίασης των αποτελεσμάτων πιο κάτω, θα δώσω τα αποτελέσματα του προγράμματος για κόσμο στον οποίο συνυπάρχουν μόνο δύο άτομα. Καταρχήν το αναμενόμενο αποτέλεσμα με την ύπαρξη μόνο δύο ατόμων είναι η παρουσίαση 45 διαφορετικών καταστάσεων οι οποίες προκύπτουν ως εξής:

Αρχικά σταθεροποιούμε το πρώτο άτομο στην θέση [0,0] του πίνακα και τοποθετούμε το δεύτερο άτομο στις υπόλοιπες τοποθεσίες δημιουργώντας 9 διαφορετικές καταστάσεις. Στην δεύτερη περίπτωση σταθεροποιούμε το πρώτο άτομο στην θέση [0,1]. Ακολούθως τοποθετούμε το δεύτερο άτομο σε όλες τις τοποθεσίες εκτός από την πρώτη. Η περίπτωση αυτή έχει ήδη υπολογιστεί από την περίπτωση που είχαμε σταθεροποιημένο το πρώτο άτομο στην θέση [0,0] και το δεύτερο τοποθετήθηκε στην θέση [0,1]. Αφού θεωρώ ότι τα άτομα δεν ξεχωρίζουν μεταξύ τους, σημαίνει ότι η κατάσταση να βρίσκεται το άτομο A στην [0,0] τοποθεσία και το B στην [0,1], είναι η ίδια κατάσταση με το A να βρίσκεται στην [0,1] και το B στην [0,0] τοποθεσία. Συνοψίζοντας έχουμε 8 διαφορετικές περιπτώσεις όταν το πρώτο άτομο σταθεροποιηθεί στην τοποθεσία [0,1].

Όμοια στο επόμενο βήμα, σταθεροποιούμε το πρώτο άτομο στην τοποθεσία [0,2] και το δεύτερο στις υπόλοιπες τοποθεσίες εκτός από την πρώτη και δεύτερη τοποθεσία, [0,0] και [0,1] αντίστοιχα, οι οποίες περιπτώσεις έχουν ήδη καλυφτεί. Ακολουθώντας τον ίδιο τρόπο σκέψης, σταθεροποιώντας το ένα άτομο σε μια θέση και τοποθετώντας το άλλο στις υπόλοιπες τοποθεσίες του χώρου, καταλήγουμε στο συμπέρασμα ότι το αναμενόμενο αποτέλεσμα είναι 45 διαφορετικές καταστάσεις όπως δημιουργούνται στα εννιά βήματα: 9+8+7+6+5+4+3+2+1=45. Σε κάθε βήμα γίνεται η σταθεροποίηση του πρώτου ατόμου σε διαφορετική θέση κάθε φορά, ενώ το δεύτερο τοποθετείται στις υπόλοιπες τοποθεσίες σύμφωνα με τις οποίες δεν επαναλαμβάνονται καταστάσεις.

Στις επόμενες σελίδες παρουσιάζονται οι 45 καταστάσεις όπως προέκυψαν σαν αποτέλεσμα από το πρόγραμμα, χρησιμοποιώντας σαν αρχική κατάσταση κόσμο με δύο άτομα. Ένα στην θέση [0,1] και ένα στην θέση [1,1]. Με κάθε κατάσταση δίνεται και η πιθανότητα με την οποία είναι δυνατό να προκύψει:

Initial Situation
0	1	0
0	1	0
0	0	0

Created 45 new situations:

2	0	0
0	0	0
0	0	0
probability: 0.001600

0	0	1
0	0	0
1	0	0
probability: 0.001600

1	1	0
0	0	0
0	0	0
probability: 0.040000

0	0	0
1	0	1
0	0	0
probability: 0.003200

1	0	1
0	0	0
0	0	0
probability: 0.001600

0	0	1
0	0	0
0	1	0
probability: 0.040000

0	2	0
0	0	0
0	0	0

probability: 0.040000

0	0	0
0	2	0
0	0	0

probability: 0.040000

1	0	0
1	0	0
0	0	0

probability: 0.040000

0	0	1
0	0	0
0	0	1

probability: 0.001600

1	0	0
0	1	0
0	0	0

probability: 0.040000

0	0	0
1	0	0
1	0	0

probability: 0.001600

1	0	0
0	0	1
0	0	0

probability: 0.040000

0	0	0
0	1	1
0	0	0

probability: 0.040000

0	1	1
0	0	0
0	0	0

probability: 0.040000

0	0	0
1	0	0
0	1	0

probability: 0.040000

1	0	0
0	0	0
1	0	0

probability: 0.001600

0	0	0
0	1	0
1	0	0

probability: 0.001600

0	1	0
1	0	0
0	0	0

probability: 0.040000

0	0	0
0	0	2
0	0	0

probability: 0.001600

1	0	0
0	0	0
0	1	0

probability: 0.040000

0	0	0
1	0	0
0	0	1

probability: 0.001600

0	0	2
0	0	0
0	0	0
probability: 0.001600

0	0	0
0	1	0
0	1	0
probability: 0.080000

1	0	0
0	0	0
0	0	1
probability: 0.001600

0	0	0
0	0	1
1	0	0
probability: 0.001600

0	1	0
0	1	0
0	0	0
probability: 1.000000

0	0	0
0	1	0
0	0	1
probability: 0.001600

0	1	0
0	0	1
0	0	0
probability: 0.040000

0	0	0
0	0	1
0	1	0
probability: 0.040000

0	0	1
1	0	0
0	0	0
probability: 0.040000

0	0	0
0	0	0
2	0	0
probability: 0.001600

0	1	0
0	0	0
1	0	0
probability: 0.001600

0	0	0
0	0	1
0	0	1
probability: 0.001600

0	0	1
0	1	0
0	0	0
probability: 0.040000

0	0	0
0	0	0
1	1	0
probability: 0.001600

0	0	0
2	0	0
0	0	0
probability: 0.001600

0	0	0
0	0	0
1	0	1
probability: 0.001600

0	1	0
0	0	0
0	1	0

probability: 0.080000

0	0	0
0	0	0
0	2	0

probability: 0.040000

0	1	0
0	0	0
0	0	1

probability: 0.003200

0	0	0
0	0	0
0	1	1

probability: 0.001600

0	0	1
0	0	1
0	0	0

probability: 0.040000

0	0	0
0	0	0
0	0	2

probability: 0.001600

0	0	0
1	1	0
0	0	0

probability: 0.040000

Παρατηρώντας τα αποτελέσματα καταλήγουμε στο ότι τα αναμενόμενα αποτελέσματα επαληθεύτηκαν. Δημιουργήθηκαν δηλαδή οι 45 διαφορετικές καταστάσεις με πιθανότητες οι οποίες εάν αθροιστούν θα είναι ίσες με 2. Από την πιθανότητα αυτή αφαιρούμε 1 η οποία είναι η πιθανότητα δημιουργίας της αρχικής κατάστασης, που αρχικοποιήθηκε τυχαία.

Υπάρχει η περίπτωση ελέγχοντας τα αποτελέσματα του προγράμματος με άλλα δεδομένα, με κόσμο που περιλαμβάνει περισσότερα άτομα, να προκύπτει ότι η συνολική πιθανότητα είναι λίγο μικρότερη από 1. Αυτό οφείλεται στο σφάλμα μηχανής, το οποίο έχει σαν αποτέλεσμα να παρουσιάζει για παράδειγμα την πιθανότητα ίση με 0,98.

Κεφάλαιο 5

Συμπεράσματα

5.1 Σημαντικότητα των Πληθυσμιακών Συστημάτων και Αλγεβρών Διεργασιών

Ένα σημαντικό σημείο με το οποίο καταπιάνονται οι επιστήμες της βιολογίας και της οικολογίας είναι η κατανόηση της συμπεριφοράς ενός ή περισσότερων πληθυσμών που συνυπάρχουν στον ίδιο βιότοπο. Στόχος των μελετών γύρω από το θέμα των πληθυσμιακών συστημάτων είναι η κατανόηση του τρόπου που ζουν διάφοροι πληθυσμοί. Αυτό βοηθά στην παραγωγή διαφόρων συμπερασμάτων για την μελλοντική κατάσταση του πληθυσμού, όπως για παράδειγμα το ποσοστό αύξησης ή εξάντλησης ενός πληθυσμού μετά από κάποιο χρονικό διάστημα. Σε κλάδους όπως στην οικολογία όπου πραγματοποιούνται αγώνες προφύλαξης προστατευόμενων πληθυσμών, τέτοιες μελέτες μπορεί να δώσουν σαν αποτέλεσμα τρόπους με τους οποίους ένας τέτοιος πληθυσμός μπορεί να επιβιώσει ή να προφυλαχτεί καλύτερα, ή να δοθούν απαντήσεις σε ερωτήματα του τύπου ποίος είναι ο λόγος εξάντλησης του πληθυσμού, ή σε πόσα χρόνια θα καταφέρει ο πληθυσμός κάτω από συγκεκριμένες συνθήκες να αυξηθεί κατά ένα συγκεκριμένο ποσοστό.

Για την επιτυχία όλων των πιο πάνω, απαιτείται η χρήση τεχνικών μοντελοποίησης και ανάλυσης των πληθυσμών, οι οποίες θα επιτρέπουν στους επιστήμονες την διεξαγωγή πειραμάτων και μετρήσεων, για τον τρόπο που τα άτομα ενός ή περισσότερων πληθυσμών αλληλεπιδρούν μεταξύ τους και επιβιώνουν σε ένα περιβάλλον. Ακριβώς αυτό τον σκοπό μπορούν να εξυπηρετήσουν οι άλγεβρες διεργασιών στο πεδίων των πληθυσμιακών συστημάτων, όπως αποδείχτηκε μέσω των μελετών που έχουν γίνει μέχρις στιγμής στο συγκεκριμένο πεδίο. Μέσα από την εργασία μου κατέληξα στο ότι

οι άλγεβρες διεργασιών και κυρίως η WSCCS στην οποία έδωσα μεγαλύτερη βαρύτητα, αποτελούν ένα αξιόλογο τρόπο μοντελοποίησης πληθυσμών για την διεξαγωγή πειραμάτων που έχουν σχέση με τους συγκεκριμένους πληθυσμούς.

Στην εργασία μου δημιούργησα μια νέα άλγεβρα διεργασιών, στην οποία έχω εντάξει την έννοια του χώρου, επιτυγχάνοντας την μοντελοποίηση των ενεργειών της μετακίνησης, της αναπαραγωγής, της μόλυνσης, και του θανάτου, οι οποίες επιδρούν στην έννοια του χώρου μέσα στον οποίο τα άτομα ενός τουλάχιστον πληθυσμού συνυπάρχουν. Μέσω αυτής της μοντελοποίησης μπορούν να απαντηθούν ερωτήματα του τύπου, με ποια πιθανότητα ή μετά από πόσα χρόνια μπορεί να προκύψει αύξηση ή μείωση του πληθυσμού κατά 30%; Με ποια πιθανότητα ή σε πόσα χρόνια μπορεί να δημιουργηθεί μια συγκεκριμένη κατάταξη των ατόμων στις τοποθεσίες του χώρου; Πώς η ενέργεια της μόλυνσης μπορεί να επηρεάσει τον αριθμό των ατόμων σε μια τοποθεσία του κόσμου ή και σε ολόκληρο τον κόσμο; Η αναγκαιότητα απάντησης σε όλα αυτά τα ερωτήματα μπορεί να οδηγήσει στην δημιουργία μιας εξειδικευμένης λογικής, η οποία να μας δίνει τέτοιου είδους απαντήσεις, μετά την επεξεργασία όλων των πιθανών καταστάσεων στις οποίες μπορεί να οδηγηθεί ένας πληθυσμός. Αυτή είναι και η σημαντικότητα των αποτελεσμάτων του προγράμματος μου κατά την υλοποίηση της ενέργειας της μετακίνησης. Η παραγωγή όλων των πιθανών καταστάσεων του κόσμου του συστήματός που δημιουργούνται με την μετακίνηση, μπορούν να χρησιμοποιηθούν για την εξαγωγή συμπερασμάτων όταν αυτά συνδυαστούν με τις καταστάσεις που παράγονται από την εκτέλεση μιας άλλης ενέργειας.

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.

5.2 Μελλοντικές Επεκτάσεις

Μέχρις στιγμής έχει γίνει αξιόλογη μελέτη γύρω από το θέμα των πληθυσμιακών συστημάτων σε συνδυασμό με τις άλγεβρες διεργασιών. Μεγάλο ρόλο σε όλες αυτές τις μελέτες κατέχει ο Chris Tofts. Παρόλα αυτά, το συγκεκριμένο πεδίο, παρουσιάζει πολλές διαφορετικές πτυχές οι οποίες μπορούν να επεκταθούν στο μέλλον. Καταρχήν όπως οι ίδιοι οι επιστήμονες εκφράζουν χρειάζονται περισσότερα πειράματα για να επιβεβαιωθούν τα μέχρι στιγμής αποτελέσματα και να οδηγηθούμε σε πιο περίπλοκες και ειδικευμένες μεθόδους μοντελοποίησης και ανάλυσης των πληθυσμών[4].

Κατά την μελέτη μου έχω εντοπίσει ότι μέχρις στιγμής έννοιες όπως ο χρόνος, η προτεραιότητα και η πιθανότητα έχουν ήδη ενταχθεί σε άλγεβρες διεργασιών. Στην παρούσα εργασία δημιούργησα μια νέα άλγεβρα διεργασιών, βασισμένη στην σύνταξη και σημασιολογία της WSCCS και στην οποία έχω εντάξει, την έννοια του χώρου, δείχνοντας πως οι ενέργειες των ατόμων συσχετίζονται με αυτόν. 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 ούτος ώστε να μπορεί να πραγματοποιεί τις διάφορες ενέργειες που απαιτούνται για την επιβίωσή του, όπως συλλογή τροφής και φύλαξη της φωλιάς του.

Κατά την υλοποίηση του συστήματος στο οποίο τα άτομα εκτελούσαν συνεχώς την ενέργεια της μετακίνησης αντιμετώπισα πρόβλημα στην μνήμη που χρειαζόταν το πρόγραμμα για να αποθηκευτούν όλες οι πληροφορίες για τις πιθανές καταστάσεις του κόσμου που μπορεί να δημιουργηθούν. Πρόκληση αποτελεί το πρόβλημα αυτό για κάποιον θέλει να μοντελοποιήσει τέτοια συστήματα. Η εύρεση μεθόδων οι οποίες θα βοηθήσουν στην χρήση λιγότερου χώρου μνήμης κατά την αποθήκευση πληροφοριών θα έχει σαν αποτέλεσμα την μελέτη πιο μεγάλων συστημάτων με περισσότερα άτομα.

Επιπλέον επέκταση της παρούσας εργασίας θα μπορούσε να ήταν η υλοποίηση και των υπόλοιπων ενεργειών που παρουσίασα, της αναπαραγωγής, της μόλυνσης και του θανάτου, στην C ή σε άλλη γλώσσα προγραμματισμού. 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. Ένα τέτοιο παράδειγμα θα μπορούσε να είναι: η ενέργεια της μετακίνησης να χρειάζεται λιγότερη κατανάλωση ενέργειας ή χρόνου από κάποιο άτομο για να πραγματοποιηθεί, από ότι η ενέργεια της αναπαραγωγή.

Βιβλιογραφία

[1] A.Sharov. Quantitative Population Ecology. USA:Department of Entomology, Virginia Tech, Blacksburg, USA, August 1997. [On-line Lecture Course]. Available: http://home.comcast.net/~sharov/PopEcol/popecol.html. [Accessed: August 2009].

[2] A.Sharov. "Modelling Forest Insect Dynamics", Caring for the forest: research in a changing world, vol.II, p. 293-303, August 1995. Available: http://home.comcast.net/~sharov/popechome/model/model.html. [Accessed: August 2009].

[3] C.Tofts, M.Hatcher and N.R.Franks. "The Autosynchronization of the Ant Leptothorax acervorum (Fabricius): Theory, Testability and Experiment", Journal of Theoretical Biology, vol.157, no 1, p.71-82, 1992. Available: INIST-CNRS. [Accessed: October 13, 2009].

[4] C.Tofts. "Processes with Probabilities, Priority and Time", Formal Aspects of Computing, vol.6, no 5, p.536-564, 1994. Available: http://www.springerlink.com/content/r3123uq03392w061/. [Accessed: October 2009].

[5] C.Tofts."Task Allocation in Monomorphic Ant Species", Laboratory for foundations of Computer Science, LFCS Report Series, 1991. Available: http://www.lfcs.inf.ed.ac.uk/reports/91/ECS-LFCS-91-144/ECS-LFCS-91-144.pdf. [Accessed: November 4,2009].

[6] D.J.T.Sumpter, G.B.Blanchard and D.S.Broomhead. "Ants and Agents: a Process Algebra Approach to Modelling Ant Colony Behaviour",, Bulletin of Mathematical Biology, no 63, p.951-980, 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=6667AE546A8533D62748D88976A2B1?doi=10.1.1.23.5957&rep=rep1&type=pdf. [Accessed: August 2009].

[7] H.Hansson and B.Jonsson. “A Calculus for Communicating Systems with Time and Probabilities”, Real Time Systems Symposium, vol 11, p.278-287, December 1990. Available: IEEExplore Digital Library. [Accessed: October 9, 2009].

[8] M.Hennessy and J.Riely. “Resource Access Control in Systems of Mobile Agents”, Electronic Notes in Theoretical Computer Science, Presented at MFPS XIV and the 3rd International Workshop on High-Level Concurrent Languages, vol.16, no 3, February 1998. Available: http://www.informatics.sussex.ac.uk/users/matthewh.

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

[10]Άννα Φιλίππου, Σημειώσεις του μαθήματος ΕΠΛ664 Ανάλυση και Επαλήθευση Συστημάτων, 2009

Παράρτημα

Πιο κάτω παρουσιάζεται ο κώδικας υλοποίησης του συστήματος στην γλώσσα προγραμματισμού C:

```
//Enswmatwsi vivliothikwn
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

//Dilwsi statherwn
#define WORLD_X 3       //arithmos twn grammwn tou kosmou
#define WORLD_Y 3       //arithmos twn stilwn tou kosmou
#define WORLD 9         //arithmos twn diaforetikwn topothesiwn tou kosmou
#define MOVEMENTS 5 //arithmos twn kinisewn pou pragmatopoia atomo
                        //(panw,katw,dexia,aristera,paramoni)

//Domi gia anaparastasi enos atomou
typedef struct indivitual{
    int move;           //flag gia elegxo ean to atomo metakinithtike
    int location[2];    //sintetagmenes tis topothesias pou vrisketai[grammi,stili] tou disdiastatou
                        //kosmou
    struct indivitual* next;//deiktis pros to epomeno atomo tis idias topothesias
}Indivitual;

//Domi gia anaparastasi mias topothesias
typedef struct location{
        Indivitual* list;       //deiktis pros to prwto atomo pou vrisketai stin topothesia
        int numIndividuals; //arithmos twn atomwn stin sigkekrimeni topothesia
}Location;

//Domi gia anaparastasi komvou sto endiameso dentro kata tin metakinisi olwn twn atomwn mia fora
typedef struct middleNode{
        Location world[WORLD];  //pinakas tipou location pou krata tin katastasi tou kosmou
        int height;             //to height einai iso me to ipsos enos komvou sto dentro
        float pithanotita;      //h pithanotita me tin opoia mporei na prokipsei i katastasi
        struct middleNode* next;   //deiktis pros ton epomeno komvo
}MiddleNode;

//Domi gia apeikonisi tou hash table pou tha dimiourgeitai se kathe epipedo tou dentrou
typedef struct hash{
        int size;               //Arithmos thesewn tou hash table
        MiddleNode* head;       //Deiktis pros tin lista me ta stoixeia pou einai apothikevmena se
                                //sigkekrimeni thesi sto hashtable
}Hash;

MiddleNode* lista;

//Prototypa Synartisewn
void CreateFinalTreeMovement(Location kosmos[]);
void CreateMiddleTreeMovement(MiddleNode* node);
MiddleNode* CreateNewSituation(MiddleNode* prokatoxos, Indivitual* tmp, char go, int
thesiMetakAtomou);
void InsertBackOpen(MiddleNode** open, MiddleNode* komvos);
int HashFunction(MiddleNode* newstate, int epipedo);
int InsertToHashTable(MiddleNode* newstate,Hash* hashtable);
void InsertHash(MiddleNode *newstate,Hash *hashtable,int epipedo);
```

```
/*********************************************************************/
/*****                          MAIN                             *****/
/*********************************************************************/
int main(){
	int i, j, numIndiv, x, y, eidos;
	int numAtomwn=0, topothesia=0;
	Location world[WORLD]; //Pinakas pou antistoixei stin apeikonisi tou xwrou.

	//Arxikopoiisi tis listas twn atomwn kathe xwrou me null.
	for (i=0; i<WORLD; i++){
		world[i].numIndividuals=0;
		world[i].list=NULL;
	}

	//Tyxaia arxikopoiisi tou kosmou
	srand(time(0));	//Entoli wste kathe fora pou trexei to programma na arxikopoieitai me
				//diaforetikous arithmous o kosmos
	//Logo mnimis ston komso mporoun na siniparxoun mexri kai 5 atoma
	numAtomwn=rand()%5;
	//Epanelave gia kathe ena apo ta atoma
	for(i=0; i<numAtomwn; i++){
		//Tyxaia epilogi gia topothetisi tou atomou i
		topothesia=rand()%8;
		//Afxisi twn atomwn toy sigkekrimenou xwrou kata ena
		world[topothesia].numIndividuals++;

		//Desmefsi xwrou gia to atomo kai arxikopoiisi tou
		Indivitual* newnode;
		newnode=(Indivitual*)malloc(sizeof(Indivitual));
		newnode->next=world[topothesia].list;
		world[topothesia].list=newnode;
		newnode->move=0;
		//Ypologismos tis thesis stin opoia vrisketai to atomo se grammi kai stili
		x=(int)(topothesia/WORLD_X);
		y=topothesia % WORLD_Y;
		newnode->location[0]=x;
		newnode->location[1]=y;
	}

	//Typwma tis arxikis katastasis tou kosmou
	printf("\tInitial Situation\n");
	printf("\t-------------\n");
	printf("\t|	%d	|	%d	|	%d
|\n",world[0].numIndividuals,world[1].numIndividuals,world[2].numIndividuals );
	printf("\t|	%d	|	%d	|	%d
|\n",world[3].numIndividuals,world[4].numIndividuals,world[5].numIndividuals );
	printf("\t|	%d	|	%d	|	%d
|\n",world[6].numIndividuals,world[7].numIndividuals,world[8].numIndividuals );
	printf("\t-------------\n");

	//Klisi tis sinartisis CreateFinalTreeMovement gia ipologismo twn pithanwn katastasewn tou
	//xwrou. San parametros dinetai o tixaios arxikopoiimenos kosmos
	CreateFinalTreeMovement(world);

}
```

```
/*********************************************************************/
/***                    CreateFinalTreeMovement                    ***/
/*********************************************************************/
/*** Sinartisi gia ipologismo olwn twn pithanonwn katastasewn pou dimiourgoyntai ***/
/*** kata thn metakinisi twn atomwn ston kosmo.                      ***/
/*********************************************************************/
void CreateFinalTreeMovement(Location kosmos[]){
        Hash* hashfinal;                                    //Hashtable me ta telika apotelesmata
        MiddleNode *initial=NULL;                           //Arxiki katastasi tou kosmou
        MiddleNode *tmp=NULL, *pp=NULL, *f=NULL;   //Voithitikos deiktis typou MiddleNode
        int i,j,epipedo=0,topothetisi=-1;                   //Metrites
        Indivitual *newnode, *t;                            //Voithitikoi komvoi typou Indivitual
        MiddleNode *open=NULL;                              //Deiktis pros tin lista open me tis
                                                            //katastaseis pou xreiazontai epexergasia

        //Desmefsi xwrou gia hashtable megethous 100 kai arxikopoiisi tou
        //Sto hashfinal tha apothikeftoun oi komvoi tou telikou dentrou
        hashfinal=(Hash*)malloc(sizeof(Hash)*100);
        for(i=0; i<100; i++){
                hashfinal[i].size=0;
                hashfinal[i].head=NULL;
        }

        //Desmefsi xwrou kai arxikopoiisi tis arxikis katastasis
        initial=(MiddleNode*)malloc(sizeof(MiddleNode));
        initial->height=0;
        initial->pithanotita=1;
        initial->next=NULL;
        for (i=0; i<WORLD; i++){
                initial->world[i].list=NULL;
                initial->world[i].numIndividuals=0;
        }

        //Antigrafi tou kosmou pou dimiourgisame tixaia san kosmos tis katastasis initial
        for (i=0; i<WORLD; i++){
                initial->world[i].numIndividuals=kosmos[i].numIndividuals;
                t=kosmos[i].list;
                while(t!=NULL){
                        newnode=(Indivitual*)malloc(sizeof(Indivitual));
                        newnode->next=initial->world[i].list;
                        initial->world[i].list=newnode;
                        newnode->move=t->move;
                        newnode->location[0]=t->location[0];
                        newnode->location[1]=t->location[1];
                        t=t->next;
                }
        }

        //Klisi tis InsertBackOpen gia topothetisi tis arxikis katastasis sto telos tis listas open
        InsertBackOpen(&open, initial);
        //Apothikefsi sto hashfinal tis arxikis katastasis ean den exei idi apothikeftei
        topothetisi=InsertToHashTable(initial,hashfinal);

        //Enoso h open periexei stoixeia twn opoiwn prepei na vrethoun ta 'paidia' tous epanelave
        while(open!=NULL){
                initial=open;       //o deiktis initial deixnei sto prwto komvo tis listas open
                lista=NULL;

                //Klisi tis CreateMiddleTreeMovement gia dimiourgia tou endiamesou dentrou pou tha
```

```
		//dwsei tin lista me ta paidia tis katastasis initial.
		CreateMiddleTreeMovement(open);

		//O deiktis tmp deixnei stin lista pou epestrepse i klisi tis CreateMiddleTreeMovement
		//kai i lista deixnei sto null
		tmp=NULL;
		tmp=lista;
		lista=NULL;

		//Epanelave gia ola ta stoixeia tis listas
		while (tmp!=NULL) {
			f=NULL; t=NULL;
			topothetisi=-1;

			//Apothikefsi sto hashfinal tis tmp katastasis ean den exei idi apothikeftei
			topothetisi=InsertToHashTable(tmp,hashfinal);

			//Ean h katastasi apothikeftei sto hashfinal
			if (topothetisi==1){
				//Prin topothetithei o komvos stin open ginetai to move kathe
				//atomou iso me 0 gia na mporei na xanametakinithei
				for(i=0; i<WORLD; i++){
					if(tmp->world[i].numIndividuals!=0){
						t=tmp->world[i].list;
						while(t!=NULL){
							t->move=0;
							t=t->next;
						}
					}
				}

				 //Klisi tis InsertBackOpen gia topothetisi tis tmp katastasis stin open
				f=tmp->next;
				InsertBackOpen(&open, tmp);
				tmp=f;

			}
			//Ean idi iparxei sto hashfinal o komvos tote apodesmevw tin mnimi tou kai
			//proxwrw ston epomeno
			else{
				pp=tmp->next;
				free(tmp);
				tmp=pp;
			}
		}

//Metakinisi ston epomeno komvo tis lista open kai apodesmefsi tou komvou pou epexergastike
//pio panw
open=open->next;
free(initial);
}


//Typwma twn telikwn apotelesmatwn poy vriskontai apothikevmena sto hashfinal
int situations=0;
for(j=0; j<100; j++)
	situations=situations+hashfinal[j].size;
printf("\n\n Created %d new situations:\n\n",situations);
```

```
	for(j=0; j<100; j++){
		MiddleNode *u;
		u=hashfinal[j].head;
		while (u!=NULL){
			Indivitual* tep;
			tep=u->world[i].list;
			printf("\t-------------\n");
			printf("\t|	%d	|	%d	|	%d	|\n",u->world[0].numIndividuals,u-
>world[1].numIndividuals,u->world[2].numIndividuals );
			printf("\t|	%d	|	%d	|	%d	|\n",u->world[3].numIndividuals,u-
>world[4].numIndividuals,u->world[5].numIndividuals );
			printf("\t|	%d	|	%d	|	%d	|\n",u->world[6].numIndividuals,u-
>world[7].numIndividuals,u->world[8].numIndividuals );
			printf("\t-------------\n");
			printf("   percentage: %f\n\n", u->pithanotita);

			u=u->next;
		}
	}
}



/*********************************************************************/
/***                          InsertToHashTable                    ***/
/*********************************************************************/
/*** H sinartisi InsertToHashTable prosthetei tin katastasi newstate sto hashtable ***/
/***ean den xanayparxei. Epistrefei tin timi 1 ean h katastasi apothikeftei, enw    ***/
/***ean i katastasi iparxei idi sto hashtable epistrefei tin timi 0                 ***/
/*********************************************************************/
//H prwti parametros anaferetai stin katastasi pou tha prostethei sto hashtable kai
//h defteri metavliti sto hashtable sto opoio tha apothikeftei
int InsertToHashTable(MiddleNode* newstate,Hash* hashtablef){
	int thesiHash=0;
	int counter=0, i, num1=0, num2=0, k, j;
	MiddleNode* temp;

	//Klisi tis sinartisis HashFunction me tin voithia tis opoias ipologizetai i thesi tou hashtable stin
	// opoia tha apothikeftei o komvos newstate
	thesiHash=HashFunction(newstate,-1);

	//O deiktis temp deixnei stin lista tou hashtable stin sigkekrimeni thesi apothikefsis
	temp=hashtablef[thesiHash].head;

	//Diatrexw tin lista twn komvwn tis sigkekrimenis thesis tou hashtable gia na exakrivwsw kata
	//poso iparxei xana apothikevmeni i katastasi tou kosmou tis newstate
	while(temp!=NULL){
		counter=0;
		for(i=0; i<WORLD; i++){
			num1=temp->world[i].numIndividuals;
			num2=newstate->world[i].numIndividuals;
			if (num1==num2){
				counter++;
			}
			//Se periptwsi pou i katastasi yparxei idi stin lista tou hashtable termatizei.
			if (counter==WORLD){
				return 0;  //epistrefetai to 0 afou idi iparxei o komvos sto hashtable
			}
		}
```

```
                temp=temp->next;
        }


        //Periptwsi pou i katastasi den vrethike stin lista me tis pithanes periptwseis. Desmevetai xwros
        //gia thn nea katastasi kai prostithetai stin arxi tis listas stin swsti thesi tou hashtable.
        MiddleNode* new;
        new=(MiddleNode*)malloc(sizeof(MiddleNode));
        new->next=hashtablef[thesiHash].head;
        hashtablef[thesiHash].head=new;
        new->height=newstate->height;
        new->pithanotita=newstate->pithanotita;
        for (k=0; k<WORLD; k++){
                new->world[k].list=NULL;
                new->world[k].numIndividuals=0;
        }
        Indivitual* tt,*newnode;
        //Antigrafi tou kosmou tou hashtable ston komvo new tis listas. Gia exikonomisi tis mnimis
        //apothikevontai mono o arithmos twn atomwn kathe topothesias xwris na apothikevetai kathe
        //atomo san komvos.
        for (j=0; j<WORLD; j++){
                new->world[j].numIndividuals=newstate->world[j].numIndividuals;
                new->world[j].list=NULL;
        }

        hashtablef[thesiHash].size++;// Afxisi tou size kata ena stin thesi pou egine apothikefsi
        return 1;//Afou egine topothetisi tou komvou sto hashtable epistrefetai i timi 1
}


/*************************************************************/
/***                    InsertBackOpen                     ***/
/*************************************************************/
/***  H sinartisi Prosthetei sto telos tis listas open tin korifi komvos  ***/
/*************************************************************/
void InsertBackOpen(MiddleNode** open, MiddleNode* komvos){
        MiddleNode* t=NULL;
        komvos->next=NULL;int counter=0;
        t=*open;

        //Ean i lista open einai adeia tote prosthese san prwti korifi ton komvo
        if (*open==NULL){
                *open=komvos;
                komvos->next=NULL;
        }
        //Ean i open den einai adeia tote diatrexe tin lista mexri kai tin teleftaia tis korifi kai prosthese
        //ton komvo sto telos tis listas
        else{
                while(t->next!=NULL){
                        t=t->next;}
                komvos->next=NULL;
                t->next=komvos;
        }
}
```

```
/*********************************************************************/
/***                        CreateMiddleTreeMovement                 ***/
/*********************************************************************/
/*** Sinartisi gia dimiourgia tou endiamesou dentrou sto opoio kathe atomo ston  ***/
/*** kosmo metakineitai apo mia fora.H parametros einai i arxiki katastasi i     ***/
/*** opoia tha apotelesei tin riza tou dentrou                                   ***/
/*********************************************************************/
void CreateMiddleTreeMovement(MiddleNode* node){
        MiddleNode* initial;
        int sumIndividuals=0; //Metavliti pou krata ton arithmo twn atomwn tou kosmou
        int repeats=0;        //Metritis pou krata ton arithmo twn epenalipsewn pou tha pragmatopoiithoun
        Indivitual* tmp=NULL, *newnode=NULL,*t=NULL;       //Deiktes typou Individuals
        MiddleNode *goUp, *goDown, *goRight, *goLeft, *stay; //Nees katastaseis pou dimiourgountai
        MiddleNode *l=NULL, *p=NULL;                       //Deiktes tupou MiddleNode
        int thesiMetakAtomou=0;  //Krata tin thesi tis topothesias stin opoia vrisketai to atomo pou tha
                                 //metakinithei
        int i,j,k, number, flag=0;   //Metrites typou integer
        int size=1, size2=0;         //Metavlites tipou int ises me to size twn hashtable pou dimiourgountai


        //Evresi tou arithmou twn atomwn pou sunuparxoun ston kosmo
        for(i=0; i<WORLD;i++){
                sumIndividuals=sumIndividuals+(node->world[i].numIndividuals);
        }

        //Ean o kathe xwros tou kosmou arxikopoiithei me 0 atoma tote den yparxoun atoma, ara
        //termatizei.
        if (sumIndividuals==0){
                printf("\nH arxiki katastasi tou kosmou den periexei atoma!To programma
termatizei\n");
                exit(-1);
      }

        //Arxikopoisi tis listas me Null kai tou size me 1
        lista=NULL;
        size=1;

        //Desmefsi xwrou kai arxikopoiisi tis arxikis katastasis
        initial=(MiddleNode*)malloc(sizeof(MiddleNode));
        initial->height=node->height;
        initial->pithanotita=node->pithanotita;
        initial->next=NULL;
        for (i=0; i<WORLD; i++){
                initial->world[i].list=NULL;
                initial->world[i].numIndividuals=0;
        }

        //Antigrafi tou kosmou pou dimiourgisame tixaia san kosmos tis katastasis initial
        for (i=0; i<WORLD; i++){
                initial->world[i].numIndividuals=node->world[i].numIndividuals;
                t=node->world[i].list;
                while(t!=NULL){
                        newnode=(Indivitual*)malloc(sizeof(Indivitual));
                        newnode->next=initial->world[i].list;
                        initial->world[i].list=newnode;
                        newnode->move=t->move;
                        newnode->location[0]=t->location[0];
                        newnode->location[1]=t->location[1];
                        t=t->next;
```

```
		}
}


//O deiktis lista deixnei ston komvo me tin arxiki katastasi tou kosmou
lista=initial;

//Epanelave gia osa epipeda tou dentrou theleis na dimiourgithoun = osos kai o arithmos twn
//atomwn pou siniparxoun sto kosmo
for(repeats=0; repeats<sumIndividuals; repeats++){
		number=0;
		if (lista!=NULL){
			flag=0;
			//Diatrexe ton komvo temp kai entompise to atomo to opoio den exei
			//metakinithei
			for(i=0; i<WORLD; i++){
				thesiMetakAtomou=0;
				tmp=lista->world[i].list;
				while(tmp!=NULL){
					thesiMetakAtomou++;
					if (tmp->move==0){
						flag=1;
						number=i;
						break;
					}
					tmp=tmp->next;
				}
				if (flag==1)
					break;
			}

		}


		//Ypologismos tou size2 pou einai o megistos arithmos twn katastasewn pou tha
		//dimiourgithoun sto epipedo repeats tou dentrou
		size2=size*MOVEMENTS;

		//Desmefsi tou hashtable to opoio tha apothikefsei tis katastaseis tou epipedou tou
		//endiamesou dentrou sto opoio eimaste.
		Hash hashtable[size2];

		//Arxikopoiisi twn timwn tou hashtable
		for (j=0; j<size2; j++){
			hashtable[j].size=0;
			hashtable[j].head=NULL;
		}

		//To l deixnei ston prwto komvo tis listas me tis katastaseis tou proigoumenou epipedou
		l=lista;

		//Gia kathe komvo tou proigoumenou epipedou (tis listas) dimiourgise tous diadoxous
		//kai apothikefse tous sto hashtable wste na min apothikevetai defteri fora mia katastasi
		//tou kosmou pou idi iparxei.
		while(l!=NULL){
			Indivitual* tmp2;
			tmp2=l->world[number].list;
			thesiMetakAtomou=0;
			while(tmp2!=NULL){
```

```
                thesiMetakAtomou++;
                if (tmp2->move==0){
                        flag=1;
                        break;
                }
                tmp2=tmp2->next;
        }

        //Apo kathe komvo prokeiptoun 5 diadoxoi.Klisi sinartisis gia dimiourgia twn
        //5 diadoxwn
        goUp=CreateNewSituation(lista,tmp2,'U',thesiMetakAtomou);
        goDown=CreateNewSituation(lista,tmp2,'D',thesiMetakAtomou);
        goRight=CreateNewSituation(lista,tmp2,'R',thesiMetakAtomou);
        goLeft=CreateNewSituation(lista,tmp2,'L',thesiMetakAtomou);
        stay=CreateNewSituation(lista,tmp2,'S',thesiMetakAtomou);

        //Apothikefsi twn 5 diadoxwn kathe komvou sto hashtable
        InsertHash(goUp, hashtable, repeats);
        InsertHash(goDown, hashtable, repeats);
        InsertHash(goRight, hashtable, repeats);
        InsertHash(goLeft, hashtable, repeats);
        InsertHash(stay, hashtable, repeats);


        p=l;            //O deiktis p deixnei ston deikti l stin arxi tis listas
        l=l->next;   //Kinoumaste ston epomeno komvo tis  listas
        free(p);      //Apodesmefsi mnimis tou komvou tou opoiou vrikame tous
                       //diadoxous
        lista=l;       //H lista deixnei sto epomeno prwto stoixeio tis listas

        }


MiddleNode *t=NULL;
//Sto telos kathe epanalipsis ginetai antigrafi twn komvwn tou hashtable stin lista
for(i=0; i<size2; i++){
        t=hashtable[i].head;
        while (t!=NULL){
                MiddleNode* new;
                new=(MiddleNode*)malloc(sizeof(MiddleNode));
                new->next=lista;
                lista=new;
                new->height=t->height;
                new->pithanotita=t->pithanotita;
                for (k=0; k<WORLD; k++){
                        new->world[k].list=NULL;
                        new->world[k].numIndividuals=0;
                }

                Indivitual* tt;
                //Antigrafi tou kosmou tou hashtable ston komvo new tis listas
                for (j=0; j<WORLD; j++){
                        new->world[j].numIndividuals=t->world[j].numIndividuals;
                        tt=t->world[j].list;
                        while(tt!=NULL){
                                newnode=(Indivitual*)malloc(sizeof(Indivitual));
                                newnode->next=new->world[j].list;
                                new->world[j].list=newnode;
                                newnode->move=tt->move;
```

```
                                        newnode->location[0]=tt->location[0];
                                        newnode->location[1]=tt->location[1];
                                        tt=tt->next;
                                    }
                                }
                                t=t->next;
                            }
                        }

                        //Apodesmefsi tis mnimis tou hashtable afou ta stoixeia tou exoun antigrafei stin lista.
                        MiddleNode* k;
                        for (i=0; i<size2; i++){
                            MiddleNode* k;
                            t=hashtable[i].head;
                            while (t!=NULL){
                                k=t;
                                t=t->next;
                                free(k);
                            }
                        }
                        size=size2;         //Sthn epomeni epanalipsi to size tha einai iso me to size2
                }
}



/*****************************************************************************/
/***                                InsertHash                               ***/
/*****************************************************************************/
/*** Sinartisi apothikefsis mias neas katastasis tou kosmou stin swsti thesi sto ***/
/*** hashtable pou antistoixei sto epipedo tou endiamesou dentrou sto opoio vriskomaste. ***/
/*****************************************************************************/
void InsertHash(MiddleNode *newstate,Hash *hashtable,int epipedo){
        int thesiHash=0;
        int counter=0, i, num1, num2;
        MiddleNode* temp;

        //Klisi tis sinartisis HashFunction me tin voithia tis opoias ipologizetai i thesi tou hashtable stin
        //opoia tha apothikeftei o komvos newstate
        thesiHash=HashFunction(newstate, epipedo);
        //O deiktis temp deixnei stin lista tou hashtable stin sigkekrimeni thesi
        temp=hashtable[thesiHash].head;

        //Diatrexw tin lista twn komvwn tis sigkekrimenis thesis tou hashtable gia na exakrivwsw kata
        //poso iparxei xana apothikevmeni i katastasi tou kosmou tis newstate
        while(temp!=NULL){
                counter=0;
                for(i=0; i<WORLD; i++){
                        num1=temp->world[i].numIndividuals;
                        num2=newstate->world[i].numIndividuals;
                        if (num1==num2){
                                counter++;
                        }
                        //Se periptwsi pou i katastasi yparxei idi stin lista tou hashtable prostithetai
                        //ston omoio komvo i pithanotita tis newstate stin pithanotita tou omoiou
                        //komvou kai termatizei.
                        if (counter==WORLD){
                                temp->pithanotita=(temp->pithanotita) + (newstate->pithanotita);
                                return;
```

```
                        }
                }
                temp=temp->next;
        }

        //Periptwsi pou i katastasi den vrethike stin lista me tis pithanes periptwseis. H nea katastasi
        //prostithetai stin arxi tis listas stin swsti thesi tou hashtable.
        newstate->next=hashtable[thesiHash].head;
        hashtable[thesiHash].head=newstate;
        hashtable[thesiHash].size++;

}



/*****************************************************************/
/***                      HashFunction                         ***/
/*****************************************************************/
/*** Sinartisi pou ipologizei tin thesi tou hashtable stin opoiia tha apo8ikeftei ***/
/*** to neo stoixeio newstate                                    ***/
/*****************************************************************/
//Analoga me to apo poia sinartisi kaleitai i HashFunction ektalountai kapws diaforetika ta vimata
//Afto elegxetai apo tin timi tis defteris parametrou epipedo
int HashFunction(MiddleNode* newstate, int epipedo){
        int i, j, ginomeno=1, result=0, ginom=1, num=0;
        MiddleNode* temp;

        //Diatrexw tin katastasi tou komvou kai pollaplasiazw tin thesi pou vrisketai kathe atomo ston
kosmo
        for(i=0; i<WORLD; i++){
                ginom=1;
                num=newstate->world[i].numIndividuals;
                for(j=0; j<num; j++){
                        ginom=ginom*(i+1);
                }
                ginomeno=ginomeno*ginom;
        }

        //Periptwsi pou i klisi tis HashFunction kaleitai gia skopous tis CreateMiddleTreeMovement
        if (epipedo!=-1){
                j=1;
                for (i=0; i<=epipedo; i++)
                        j=j*MOVEMENTS;
                //H thesi einai isi me to (ginomeno twn thesewn twn atomwn po siniparxoun ston
                //kosmo) mod (to size tou
                //hashtable  gia to sigkekrimeno epipedo sto opoio vriskomaste)
                result=ginomeno%(j);
        }
        //Periptwsi pou i klisi tis HashFunction kaleitai gia skopous tis CreateFinalTreeMovement
        else
                result=ginomeno%(100);
        return result;
}
```

```
/*****************************************************************************/
/***                          CreateNewSituation                           ***/
/*****************************************************************************/
/*** Sinartisi gia dimiourgia mias neas katastasis stin opoia ftanei o kosmos ***/
/*** se periptwsi metakinisis enos atomou. O komvos prokatoxos einai o komvos ***/
/*** me tin katastasi pou tha tropopoiisoume gia na pragmatopoiisoume tin kinisi ***/
/*** enos atomou. To tmp kai to thesiMetakAtomou krata tis plirofories gia to pou ***/
/*** vrisketai to atomo pou tha metakinisoume. To go einai enas xaraktiras pou ***/
/*** ipodilwnei to pros ta pou tha metakinithei to atomo.                      ***/
/*****************************************************************************/
MiddleNode* CreateNewSituation(MiddleNode* prokatoxos, Indivitual* tmp, char go, int thesiMetakAtomou){
        int x, y, i;
        Indivitual* temp, *newnode;
        Indivitual* p,*move1;
        int count=0, thesi=0 ,newthesi=0, coun=0;

        //Vriskw se poia thesi tou disdiastatou pinaka vrisketai to atomo pou tha metakinithei
        x=tmp->location[0];
        y=tmp->location[1];
        thesi=(x*3)+y;

        //Desmefsi xwrou gia tin nea pithani katastasi pou tha dimiourgithei.
        MiddleNode* new;
        new=(MiddleNode*)malloc(sizeof(MiddleNode));
        new->height=(prokatoxos->height)+1;
        //Pros opoiadipote katefthinsi metakinithei to atomo, i pithanotita einai 1/5=0.2
        new->pithanotita=(prokatoxos->pithanotita)*(0.2);
        new->next=NULL;

        //Arxikopoiisi tis listas me ta atoma kathe topothesias me NULL
        for(i=0; i<WORLD; i++){
                new->world[i].list=NULL;
                new->world[i].numIndividuals=0;
        }
        //Diatrexontas tin katastasi tou prokatoxou tin antigrafoume stin nea katastasi
        for (i=0; i<WORLD; i++){
                coun=0;
                new->world[i].numIndividuals=prokatoxos->world[i].numIndividuals;
                temp=prokatoxos->world[i].list;
                while(temp!=NULL){
                        coun++;
                        newnode=(Indivitual*)malloc(sizeof(Indivitual));
                        //Ean antigrafetai to prwto stoixeio tis listas i tote prosthetw ton komvo prwto
                        //diaforetika ton prosthetw teleftaio stin lista.
                        Indivitual * f;
                        f=new->world[i].list;
                        newnode->next=NULL;
                        newnode->move=0;
                        if (f==NULL)
                                new->world[i].list=newnode;
                        else{
                                while(f->next!=NULL)
                                        f=f->next;
                                f->next=newnode;
                        }

                        //Se periptwsi pou eimaste sto simeio me to atomo pou metakineitai tote ston
                        //sigkekrimeno komvo i timi tou move afxanetai kata 1
```

```
                    if((i==thesi) && (coun==thesiMetakAtomou))
                            newnode->move=(temp->move)+1;
                    //diaforetika pairnei tin idia timi me tou prokatoxou
                    else
                            newnode->move=temp->move;
                    newnode->location[0]=temp->location[0];
                    newnode->location[1]=temp->location[1];
                    temp=temp->next;
            }
    }

    //Se periptwsi pou to atomo epilexei na meinei stin idia topothesia tote exei idi dimiourgithei i
    //nea katastasi i opoia epistrefetai
    if (go=='S')
            return(new);

    //Diaforetika se periptwsi pou to atomo epilexei mia apo tis katefthinseis
     //panw,katw,dexia,aristera ektelountai ta pio katw:
    new->world[thesi].numIndividuals--; //Afou to atomo metakineitai apo tin topothesia thesi
                                        //meiwnoume to sinolo twn atomwn aftis tis topothesias
                                        //kata ena.
    p=new->world[thesi].list;     //O deiktis p deixnei stin lista me to atomo pou metakinithike

    //Se periptwsi pou to atomo pou metakinithike vrisketai apothikevmeno prwto stin lista tis
    //topothesia tou tote o deiktis move1 deixnei stin lista tis sigkekrimenis topothesias kai stin nea
    //katastasi i lista afti den simperilamvanei ton komvo tou atomou pou metakinithike alla deixnei
    //ston epomeno komvo apo auton tou atomo pou metakinithike.
    if (thesiMetakAtomou==1){
            move1=p;
            new->world[thesi].list=p->next;
    }
    //Ean to atomo pou metakinithike den einai prwto stin lista tis topothesias tou proxwra mexri na
    //to entopisei
    else{
            for(count=1; count<thesiMetakAtomou-1; count++)
                    p=p->next;
           //Otan entopisei to atomo deixnei se afto me ton deikti move1 kai enwnei ton
            //proigoumeno komvo tou atomou pou metakinithike me ton epomeno tou.
            move1=p->next;
            p->next=move1->next;
            move1->next=NULL;
    }


   //Analogo me to pou metakinithike to atomo kathorizontai swsta oi nees sintetagmenes tou, pou
   //antistoixoun stin nea tou toothesia. O kosmos thewreitai kiklikos kai ara opoiodipote atomo apo
   //opoiadipote topothesia mporei na metakinithei kai pros tis 5 topothesies.
    if (go=='U'){
            if(x==0)
                    x=2;
            else
                    x=x-1;

    }
    else if (go=='D'){
            if(x==2)
                    x=0;
            else
                    x=x+1;
```

```
	}
	else if (go=='R'){
		if(y==2)
			y=0;
		else
			y=y+1;

	}
	else{
		if(y==0)
			y=2;
		else
			y=y-1;
	}

	//Dinontai ston komvo pou apeikonizei to atomo pou metakinithike oi nees sintetagmenes tou
	move1->location[0]=x;
	move1->location[1]=y;
	//Ypologismos tis neas thesis tou atomou
	newthesi= (3*x)+y;
	//O deiktis p deixnei stin lista twn atomwn tis topothesias stin opoia eftase to metakinoumeno
	//atomo
	p=new->world[newthesi].list;
	//To metakinoumeno atomo topotheteitai prwto stin lista twn atomwn tis neas tou topothesias
	move1->next=new->world[newthesi].list;
	new->world[newthesi].list=move1;
	//Stin sigkekrimeni topothesia o arithmos ton atomwn afxanetai kata ena afou to metakinoumeno
atomo eftase
	new->world[newthesi].numIndividuals++;

	return (new);	//Epistrofi tou neou komvou me tin nea katastasi

}
```