Ατομική Διπλωματική Εργασία

ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΜΟΡΙΑΚΗΣ ΒΙΟΛΟΓΙΑΣ ΜΕ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΔΡΑΣΗΣ.

Μαρία Κουμή

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2008

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΜΟΡΙΑΚΗΣ
ΒΙΟΛΟΓΙΑΣ
ΜΕ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΔΡΑΣΗΣ

Κουμή Μαρία

Επιβλέπων Καθηγητής

Γιάννης Δημόπουλος

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

Μάιος 2008

Περίληψη

Σε αυτή την διπλωματική εργασία έγινε μελέτη και η μοντελοποίηση προβλημάτων της Μοριακής Βιολογίας με προγραμματισμό δράσης (planning). Έχουν μελετηθεί και στην συνέχεια μοντελοποιηθεί τρία προβλήματα μοριακής βιολογίας: το πρόβλημα των βιοχημικών μονοπατιών, πρόβλημα της αναδιοργάνωσης γονιδιώματος και το πρόβλημα της δυαδικής ευθυγράμμισης ακολουθίας.

Στο κεφάλαιο 2 παρουσιάζεται το πρόβλημα των βιοχημικών μονοπατιών και η μοντελοποίηση του με προγραμματισμό δράσης. Από πλευράς βιοχημείας,ένα βιοχημικό μονοπάτι είναι μια ακολουθία από χημικές αντιδράσεις σε ένα οργανισμό. Από πλευράς προγραμματισμού δράσης, το πρόβλημα των βιοχημικών μονοπατιών είναι, δεδομένου κάποιων αρχικών βιοχημικών ουσιών, να εξευρεθεί η σειρά με την οποία θα πρέπει να εκτελεστούν οι διάφορες βιοχημικές αντίδρασης σε ένα οργανισμό, ούτως ώστε να παραχθούν κάποιες άλλες βιοχημικές ουσίες - στόχοι.

Στο κεφάλαιο 3 παρουσιάζεται το πρόβλημα της αναδιοργάνωσης γονιδιώματος είναι το εξής: Δεδομένου δύο γονιδιωμάτων και ενός θετικού ακεραίου αριθμού k, πρέπει να βρεθεί μια ακολουθία από το πολύ k γεγονότα τα οποία μετατρέπουν το ένα γονιδίωμα στο άλλο. Στην μοντελοποίηση που κάναμε, εμείς επικεντρωθήκαμε απλά στην ανεύρεση της κατάλληλης σειράς εκτέλεσης των γεγονόντων – δράσεων.

Στο 4 κεφάλαιο παρουσιάζεται το πρόβλημα της δυαδικής ευθυγράμμισης ακολουθίας το οποίο είναι, δεδομένου δύο ακολουθιών s και t οι οποίες έχουν το ίδιο μήκος, να διαπιστωθεί πόσο ‘κοντά’ είναι οι δυο ακολουθίες ως προς την ομοιότητα των χαρακτήρων ανά θέση και με ανακατατάξεις που θα γίνουν στην μια ακολουθία s να ταυτοποιηθεί στο τέλος με την ακολουθία στόχος t.

Στην συνέχεια στο κεφάλαιο 5 ακολουθούν τα συμπεράσματα.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Σκοπός αυτής της διπλωματικής εργασίας είναι η μελέτη και η μοντελοποίηση προβλημάτων από την Μοριακή Βιολογία με προγραμματισμό δράσης (planning). Έχουν μελετηθεί και στην συνέχεια μοντελοποιηθεί τρία προβλήματα μοριακής βιολογίας: το πρόβλημα των βιοχημικών μονοπατιών πρόβλημα της αναδιοργάνωσης γονιδιώματος και το πρόβλημα της δυαδικής ευθυγράμμισης ακολουθίας.

Αρχικά, όταν αναφερόμαστε σε προγραμματισμό δράσης,εννοούμε ότι για την επίλυση ενός συγκεκριμένου προβλήματος θα πρέπει να υλοποιήσουμε δράσεις οι οποίες με την εκτέλεση τους στην κατάλληλη σειρά θα έχουν σαν αποτέλεσμα την επίτευξη του στόχου που τέθηκε από το πρόβλημα.

Ακολουθεί μια σύντομη περιγραφή των προβλημάτων τα οποία έχουν μελετηθεί και μοντελοποιηθεί:

- Πρόβλημα των βιοχημικών μονοπατιών:
  Από πλευράς βιοχημείας,ένα βιοχημικό μονοπάτι είναι μια ακολουθία από χημικές αντιδράσεις σε ένα οργανισμό.Κατά Thagard 2003, πολλές ασθένειες μπορούν να εξηγηθούν σαν ατέλειες, στο βιοχημικό μονοπάτι και η θεραπεία τους περιλαμβάνει την ανακάλυψη φαρμάκων τα οποία να διορθώνουν αυτό το ελάττωμα.Από πλευράς προγραμματισμού δράσης, το πρόβλημα των βιοχημικών μονοπατιών είναι, δεδομένου κάποιων αρχικών

βιοχημικών ουσιών, να εξευρεθεί η σειρά με την οποία θα πρέπει να εκτελεστούν οι διάφορες βιοχημικές αντιδράσεις σε ένα οργανισμό, ούτως ώστε να παραχθούν κάποιες άλλες βιοχημικές ουσίες - στόχοι.

- Πρόβλημα της αναδιοργάνωσης γονιδιώματος:
  Η αναδιοργάνωση γονιδιώματος, αναφέρεται σε κάποια «γεγονότα», μέσον των οποίων αλλάζει η σειρά των γονιδίων στο γονιδίωμα. Μέσο των γεγονότων αναδιοργάνωσης, μπορούμε να οδηγηθούμε από ένα γονιδίωμα ενός οργανισμού, στο γονιδίωμα κάποιου άλλου οργανισμού. Όσο πιο κοντά γενετικά βρίσκονται (φυλογενετικά δέντρα) δύο οργανισμοί, τόσο λιγότερα γεγονότα αναδιοργάνωσης γονιδιώματος, χρειάζονται για να μετατραπεί το γονιδίωμα του ενός οργανισμού στου άλλου.Το πρόβλημα της γονιδιακής αναδιοργάνωσης αναφέρεται στην εύρεση του ελαχίστου αριθμού γεγονότων που απαιτούνται για την μετατροπή γονιδιώματος ενός οργανισμού σε κάποιου άλλου. Από πλευράς προγραμματισμού δράσης, το πρόβλημα μεταφέρεται ως εξής: Δεδομένου δύο γονιδιώματα και ενός θετικού ακεραίου αριθμού k, πρέπει να βρεθεί μια ακολουθία από το πολύ k γεγονότα τα οποία μετατρέπουν το ένα γονιδίωμα στο άλλο. Στην μοντελοποίηση που κάναμε εμείς επικεντρωθήκαμε απλά στην ανεύρεση της κατάλληλης σειράς εκτέλεσης των γεγονότων – δράσεων.

- Πρόβλημα της δυαδικής ευθυγράμμισης ακολουθίας:
  Το πρόβλημα της δυαδικής ευθυγράμμισης ακολουθίας είναι, δεδομένου δύο ακολουθιών s και t οι οποίες έχουν το ίδιο μήκος, να διαπιστωθεί πόσο ‘κοντά’ είναι οι δυο ακολουθίες ως προς την ομοιότητα των χαρακτήρων ανά θέση και με ανακατατάξεις που θα γίνουν στην μια ακολουθία s να ταυτοποιηθεί στο τέλος με την ακολουθία στόχος t.Οι δύο αυτές ακολουθίες μπορούν να είναι μονόκλωνες αλυσίδες DNA αλλά και πρωτεϊνικές αλυσίδες οι οποίες αποτελούνται από αμινοξέα. Βέβαια σκοπός και σε αυτό το πρόβλημα είναι η λύση η οποία θα δοθεί να είναι και η ποιό σύντομη.Εμείς ωστόσο έχουμε ασχοληθεί απλά μόνο με την

επίλυση του προβλήματος χωρίς να επιζητούμε την παραγωγή της πιο σύντομης λύσης.

1.1 **Προγραμματισμός Δράσης**

Στην καθημερινή μας ζωή, για να επιτύχουμε ένα συγκεκριμένο στόχο που τέθηκε, ακολουθούμε μια σειρά από ενέργειες που με την εκτέλεση τους θα περατωθεί ο στόχος, δηλαδή "προγραμματίζουμε" με ποια σειρά θα εκτελέσουμε κάποιες "δράσεις" που σαν τελικό αποτέλεσμα να έχουν την εκπλήρωση του στόχου που τέθηκε. Αντίστοιχα στην επιστήμη της Πληροφορικής ορίζουμε σαν Προγραμματισμό Δράσης (planning), την εύρεση μιας σειράς από δράσεις οι οποίες με την εκτέλεση τους θα έχουν ως αποτέλεσμα την επίτευξη κάποιου συγκεκριμένου στόχου που τέθηκε.
Η ακολουθία από δράσεις μέσω της οποίας οδηγούμαστε σε κάποιο συγκεκριμένο στόχο ορίζεται σαν σχέδιο δράσης (plan).
Γενικότερα:

- Όταν επιθυμούμε να παράξουμε κάποιο πρόγραμμα/πλάνο δράσης (plan), έχουμε ως δεδομένα:
  - Την περιγραφή του κόσμου (domain) στον οποίο υπάρχει το πρόβλημα που θέλουμε να επιλύσουμε.
  - Ένα αρχικό στάδιο στο οποίο βρισκόμαστε.
  - Μία περιγραφή του στόχου (goal) τον οποίο επιθυμούμε να επιτύχουμε.
  - Ένα σύνολο από πιθανές δράσεις (actions) για τον συγκεκριμένο κόσμο, μέσον των οποίων πιθανών να οδηγηθούμε στην εκπλήρωση του συγκεκριμένου στόχου.

- Αυτό που αναζητούμε είναι:
  - Μία σειρά από δράσεις (από το σύνολο των δράσεων για τον συγκεκριμένο κόσμο που μας δόθηκε), με την εκτέλεση των οποίας θα μεταφερθούμε από το αρχικό στάδιο όπου βρισκόμαστε σε κάποιο άλλο το οποίο ικανοποιεί τον στόχο που τέθηκε.

Οι προγραμματιστές δράσεις (planners), είναι υπεύθυνοι ανάλογα με τα δεδομένα που τους δίδονται να παράξουν το κατάλληλο σχέδιο δράσης (plan).

Το νόημα της επίλυσης προβλημάτων μέσω του προγραμματισμού δράσης μπορεί να αποδοθεί καλύτερα μέσω των μοντέλων καταστάσεων [[3] p 5] (state – space model). Συγκεκριμένα ένα μοντέλο καταστάσεων είναι μια πεντάδα <S,s_0,S_G,A, next> όπου:

1. Το S είναι ένα πεπερασμένο σύνολο καταστάσεων
2. Το s_0 ανήκει στο σύνολο S και είναι η αρχική κατάσταση
3. S_G υποσύνολο του S είναι ένα μη κενό σύνολο από καταστάσεις-στόχους
4. A(s) είναι ένα σύνολο από δράσεις α οι οποίες είναι εφαρμόσιμες στην κατάσταση s
5. next είναι η συνάρτηση μετάβασης από το s σε μια διαδοχική κατάσταση s_a = next(a, s) για κάποια δράση a η οποία ανήκει στο A(s)

Η λύση κάποιου προβλήματος στο μοντέλο αυτό, είναι μια πεπερασμένη σειρά από εφαρμόσιμες δράσεις a_0, a_1,….., a_n, μέσον των οποίων οδηγούμαστε από την αρχική κατάσταση s_0 σε κατάσταση-στόχου s η οποία ανήκει στο S_G σύνολο.

1.2 Γλώσσες προγραμματισμού δράσης

Για την επίλυση προβλημάτων προγραμματισμού δράσης απαιτούνται γλώσσες οι οποίες να είναι αρκετά εκφραστικές ούτως ώστε να μπορούν περιγράψουν ένα ευρύ φάσμα προβλημάτων του εκάστοτε κόσμου, αλλά και αρκετά αυστηρές ώστε να επιτρέψουν στους διάφορους αλγορίθμους να δράσουν πάνω τους. Αρχίζουμε με την περιγραφή της γλώσσας STRIPS η οποία είναι πιο περιοριστική σε σχέση με την PDDL γλώσσα (η οποία είναι πιο εκφραστική), γι’ αυτό και η STRIPS θεωρείτε γλώσσα του κλασσικού προγραμματισμού δράσης.

1.2.1 Γλώσσα STRIPS

Πιο κάτω θα παρουσιαστεί η γλώσσα STRIPS (Stanford Research Institute Problem Solver), η οποία δημιουργήθηκε το 1971 από τους Fikes και Nilsson, με μια πιο θεωρητική προσέγγιση (η προγραμματιστική σύνταξη θα παρουσιαστεί πιο κάτω).

Για το σωστό χειρισμό της γλώσσας STRIPS ο αναγνώστης θα πρέπει να γνωρίζει τις ακόλουθες έννοιες:

- Κόσμος (Domain): Περιέχει κατηγορήματα και δράσεις που πιθανόν να εφαρμοστούν πάνω στα κατηγορήματα και να αλλάξουν με τα αποτελέσματα τους, την κατάσταση που βρίσκεται ο κόσμος.

- Καταστάσεις (States): Μπορούμε να σκεφτούμε μια κατάσταση στην οποία βρισκόμαστε, σαν ένα στιγμιότυπο του κόσμου (domain), στον οποίο λαμβάνει χώρα το πρόβλημα για το οποίο προσπαθούμε να βρούμε κάποιο σχέδιο δράσης, ούτως ώστε να φτάσουμε τον στόχο [[7] p 377]. Δηλαδή σε κάθε κατάσταση που βρισκόμαστε, ισχύουν διαφορετικά “στοιχεία” από τα οποία αποτελείτε ο κόσμος του προβλήματος που τέθηκε. Με την εκτέλεση κάποιας δράσης μεταβαίνουμε από μια κατάσταση σε κάποια άλλη ή μπορούμε να παραμείνουμε στην ίδια. Μια κατάσταση αναπαρίσταται σαν μια σύζευξη από αληθή κατηγορήματα.Επίσης, θεωρούμε ότι ισχύει το closed-world-assumption, δηλαδή τα κατηγορήματα τα οποία υπάρχουν στον κόσμο (domain), αλλά δεν γίνεται αναφορά σε αυτά στην αναπαράσταση κάποιας κατάστασης, θεωρούνται ότι είναι ψευδή για την συγκεκριμένη κατάσταση.

- Στόχοι (Goals): Οι στόχοι θεωρούνται σαν μερικές καταστάσεις. Αναπαρίστανται σαν μια σύζευξη από αληθή κατηγορήματα. Μια κατάσταση s ικανοποιεί κάποιο στόχο g,εάν η s περιέχει όλα τα μέλη του κατηγορήματος (και πιθανόν άλλα), τα οποία ανήκουν στην g. Παραδείγματος χάριν, η κατάσταση που αναπαρίσταται με το κατηγόρημα Πλούσιος $\wedge$ Διάσημος $\wedge$ Επιτυχημένος ικανοποιεί τον στόχο Επιτυχημένος $\wedge$ Διάσημος [[7] p 377].

- Δράσεις (Actions): Μια δράση χωρίζεται σε δύο μέρη: τις συνθήκες οι οποίες πρέπει να ισχύουν ούτως ώστε η δράση να μπορεί να εκτελεστεί (preconditions) και τα αποτελέσματα (effects) της εκτέλεσης της συγκεκριμένης δράσης. Για παράδειγμα, για να εκφράσουμε την δράση του ταξιδιού με ένα αυτοκίνητο από ένα μέρος σε ένα άλλο χρησιμοποιούμε την ακόλουθη δράση – Travel (Σχήμα 1.1), η οποία έχει σαν προαπαίτηση (precondition) τα κατηγορήματα: At(c, from) το οποίο αντιπροσωπεύει την ύπαρξη του αυτοκινήτου c στο μέρος εκκίνησης from, Car (c) το οποίο αντιπροσωπεύει την ύπαρξη του αυτοκινήτου c, Begin_place (from) το οποίο αντιπροσωπεύει την ύπαρξη του μέρους εκκίνησης c και Destination_place (to) το οποίο αντιπροσωπεύει την ύπαρξη του μέρους προορισμού.

Action (Travel(c, from, to),
Precond: At(c, from) $\wedge$ Car (c) $\wedge$ Begin_place (from) $\wedge$ Destination_place (to)
Effect: $\neg$At(c, from) $\wedge$ At (c, to))

Σχήμα 1.1:Σχήμα δράσης για την δράση Travel

Η πιο πάνω αναπαράσταση της δράσης είναι κατ' ακρίβεια ένα σχήμα δράσης (action schema) νοουμένου ότι αναπαριστά πολλές δράσεις που προκύπτουν από ανάθεση διαφόρων τιμών στις μεταβλητές c, from και to[[7] p 377]. Γενικά, ένα σχήμα δράσης αποτελείται από τρία μέρη [[7] p 377]:

1. το όνομα της δράσης και την λίστα παραμέτρων, τα οποία χρησιμεύουν ως αναγνωριστικά για την δράση.

2. τις προαπαιτούμενες συνθήκες (preconditions),οι οποίες αναπαρίστανται με την μορφή συζεύξεως από θετικά κατηγορήματα τα οποία καθορίζουν τι πρέπει να ισχύει σε κάποια κατάσταση πριν την εκτέλεση κάποιας δράσης. Μεταβλητές οι οποίες εμφανίζονται στις προαπαιτούμενες συνθήκες θα πρέπει να προστεθούν στην λίστα παραμέτρων της δράσης.

3. τα αποτελέσματα (effects) της εκτέλεσης της συγκεκριμένης δράσης, τα οποία αναπαρίστανται με την μορφή συζεύξεως από θετικά κατηγορήματα, περιγράφοντας έτσι το πώς περνάμε από μια κατάσταση σε κάποια άλλη μέσο της δράσης. Μεταβλητές οι οποίες εμφανίζονται στα αποτελέσματα της δράσης θα πρέπει να προστεθούν στην λίστα παραμέτρων της δράσης. Ένα θετικό κατηγόρημα P το οποίο εμφανίζεται στο αποτέλεσμα της δράσης θα ισχύει και στην κατάσταση στην οποία μεταβαίνουμε με την εκτέλεση της δράσης. Ένα ψευδές κατηγόρημα ¬P το οποίο εμφανίζεται στο αποτέλεσμα της δράσης θα είναι ψευδές και στην κατάσταση στην οποία μεταβαίνουμε με την εκτέλεση της δράσης.

- Μια δράση είναι εφαρμόσιμη σε κάποια κατάσταση εάν ικανοποιούνται οι προαπαιτούμενες συνθήκες, αλλιώς η δράση δεν έχει αποτελέσματα[[7] p 378].

- Υπόθεση STRIPS (STRIPS assumption): κάθε κατηγόρημα το οποίο δεν αναφέρεται στα αποτελέσματα δεν μεταβάλλετε μετά το πέρας της δράσης [[7] p 378].

- Λύση (solution): Σειρά από δράσεις οι οποίες όταν εκτελεστούν στην αρχική κατάσταση καταλήγουμε σε μια κατάσταση η οποία ικανοποιεί τον στόχο.

- Add list – Delete list: Για σκοπούς βελτίωσης της ανάγνωσης κάποια συστήματα διαχωρίζουν τα αποτελέσματα των δράσεων σε Add list για αληθή κατηγορήματα και σε Delete list για ψευδή κατηγορήματα. Αρχίζοντας από μια κατάσταση s με την εκτέλεση μιας εφαρμόσιμης δράσης α, μεταβαίνουμε στην κατάσταση s',η οποία είναι ίδια με την s εκτός από το ότι όσα κατηγορήματα είναι αληθή στο αποτέλεσμα της α προστίθενται στην κατάσταση s'και όποιο κατηγόρημα είναι ψευδής αφαιρείται από την s'[[7] p 378].

Ένα πρόβλημα προγραμματισμού σε STRIPS $P = \langle L_s, O_s, I_s, G_s \rangle$ μπορεί να αποδοθεί έχοντας υπόψη το μοντέλο καταστάσεων που περιγράφηκε πιο πάνω

$S(P) = <S,s_0,S_G,A, next>$ [[3], p 5] όπου:

1. Οι καταστάσεις S είναι σύνολα από κατηγορήματα L_s.
2. Η αρχική κατάσταση s_0 είναι η I_s (αρχική κατάσταση που περιγράφεται στο πρόβλημα).
3. Οι καταστάσεις στόχοι είναι οι καταστάσεις S για τις οποίες ισχύει ότι G_s είναι υποσύνολο του S.
4. A(s) είναι το υποσύνολο από δράσεις οι οποίες ανήκουν στο σύνολο δράσεων O_s ούτως ώστε το οι προαπαιτήσεις να ανήκουν στο S.
5. next είναι η συνάρτηση μετάβασης next(a, s)= s + Add(a) - Del(a) για a η οποία ανήκει στο A(s).

Παραδείγματα :

1.The spare tire problem (Πρόβλημα αλλαγής ελαστικού αυτοκινήτου)

Το πρόβλημα αυτό αφορά την αλλαγή ενός άδειου λάστιχου αυτοκινήτου. Ο στόχος είναι να εφαρμόσει ο νέος τροχός στον άξονα του αυτοκινήτου. Αρχικά κάποιο λάστιχο είναι άδειο και η ρεζέρβα είναι στον αποθηκευτικό χώρο του αυτοκινήτου. Υποθέτουμε ότι το αυτοκίνητο βρίσκεται σε κάποια κακόφημη συνοικία και αν παραμείνει εκεί για ένα βράδυ θα έχει σαν αποτέλεσμα να κλαπούν οι τροχοί[[7] p 381].

Πιο κάτω, στο Σχήμα1.1 παρουσιάζεται η περιγραφή του προβλήματος στην γλώσσα STRIPS. Σύμφωνα με την περιγραφή του προβλήματος, στην αρχική κατάσταση κάποιο λάστιχο του αυτοκινήτου είναι άδειο και η ρεζέρβα βρίσκεται στον αποθηκευτικό χώρο (κατηγορήματα At(Flat,Axle), At (Spare, Trunk) αντίστοιχα).Στόχος είναι να αλλαχθεί το άδειο λάστιχο του αυτοκινήτου με την ρεζέρβα-να εφαρμόσει η ρεζέρβα στον άξονα(At (Spare, Axle)).

Σκοπός της δράσης Remove_Spare, η οποία παίρνει σαν παράμετρο την ρεζέρβα, τον αποθηκευτικό χώρο του αυτοκινήτου και το έδαφος, είναι να πάρουμε την ρεζέρβα από τον αποθηκευτικό χώρο και να την αφήσουμε στο

έδαφος. Σαν προαπαιτήση, έχει η ρεζέρβα να βρίσκεται στον αποθηκευτικό χώρο του αυτοκινήτου(At (Spare, Trunk)). Με την εκτέλεση της, η δράση σαν αποτέλεσμα θα έχει η ρεζέρβα να μην βρίσκεται στον αποθηκευτικό χώρο πλέον αλλά στο έδαφος (κατηγορήματα ¬ At (Spare, Trunk) , At (Spare, Ground) αντίστοιχα).

Η δράση Remove_Flat,παίρνει παραμέτρους το άδειο λάστιχο του αυτοκινήτου, τον άξονα και το έδαφος. Σκοπός της, είναι η αφαίρεση του άδειου λάστιχου από τον άξονα του αυτοκινήτου και η τοποθέτηση του, στο έδαφος. Προαπαίτηση της είναι, το άδειο λάστιχο να είναι στον άξονα (At (Flat, Axle)).Σαν αποτέλεσμα της δράσης το άδειο λάστιχο βρίσκεται στο έδαφος και όχι στον άξονα(At (Flat, Ground),¬At (Flat, Axle)).

Η δράση PutOn,παίρνει παραμέτρους την ρεζέρβα, τον άξονα και το έδαφος. Σκοπός της είναι, η ρεζέρβα να εφαρμόσει στον άξονα του αυτοκινήτου. Προαπαίτηση της είναι, η ρεζέρβα να είναι στο έδαφος και στον άξονα να μην υπάρχει άλλο λάστιχο(κατηγορήματα At (Spare, Ground) ,Clear (Axle) αντίστοιχα). Αποτέλεσμα της είναι, η ρεζέρβα να εφαρμόσει στον άξονα και να μην είναι πλέον στο έδαφος (κατηγορήματα ¬ At (Spare, Ground), ¬At (Spare, Axle) αντίστοιχα).

Η δράση LeaveOvernight παίρνει παραμέτρους την ρεζέρβα, το έδαφος, τον άξονα και το άδειο λάστιχο. Σκοπός της είναι, να περιγράψει το τι θα συμβεί αν το αυτοκίνητο μείνει την νύχτα σε κάποια κακόφημη συνοικία. Δεν υπάρχουν προαπαιτήσεις. Αποτέλεσμα της είναι η ρεζέρβα να μην βρίσκεται στο έδαφος (¬ At (Spare, Ground)), το άδειο λάστιχο να μην είναι στον άξονα(¬At (Spare, Axle)) , η ρεζέρβα να μην βρίσκεται στον αποθηκευτικό χώρο (¬At (Spare, Trunk)), το άδειο λάστιχο να μην βρίσκεται στο έδαφος (¬At (Flat, Ground)) αλλά ούτε και στον άξονα(¬At (Flat, Axle)).

Init (At (Flat, Axle) $\wedge$ At (Spare, Trunk))
Goal (At (Spare, Axle))

Action (Remove_Spare (Spare, Trunk, Ground),

PRECOND: At (Spare, Trunk)

EFFECT: ¬ At (Spare, Trunk) ∧ At (Spare, Ground))

Action (Remove_Flat (Flat, Axle, Ground),

PRECOND: At (Flat, Axle)

EFFECT: ¬At (Flat, Axle) ∧At (Flat, Ground))

Action (PutOn (Spare, Axle, Ground),

PRECOND: At (Spare, Ground) ∧ Clear (Axle)

EFFECT: ¬ At (Spare, Ground) ∧ At (Spare, Axle))

Action (LeaveOvernight (Spare,Ground, Axle, Flat),

PRECOND:

EFFECT: ¬ At (Spare, Ground) ∧ ¬At (Spare, Axle) ∧ ¬At (Spare, Trunk) ∧ ¬At (Flat, Ground) ∧ ¬At (Flat, Axle))

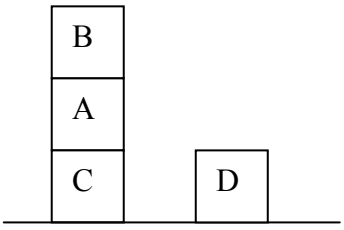
Σχήμα 1.1:Περιγραφή του spare tire problem σε STRIPS

2. The blocks world (Κόσμος των κύβων)

Ένας από τους πιο δημοφιλής κόσμους προγραμματισμού δράσης είναι το block domain,το οποίο αποτελείται από κύβους που βρίσκονται πάνω σε ένα τραπέζι. Οι κύβοι μπορούν να βρίσκονται σε στοίβα αλλά μόνο ένας κύβος εφαρμόζει κατευθείαν πάνω σε ένα άλλο). Το ρομποτικό χέρι μπορεί να πάρει μόνο ένα κύβο την φορά(δηλαδή ο κύβος που θα πάρει δεν μπορεί να έχει άλλο κύβο από πάνω του) και να τον τοποθετήσει σε μια άλλη θέση είτε πάνω σε ένα άλλο κύβο είτε απευθείας πάνω στο τραπέζι. Στόχος μας, είναι να δημιουργήσουμε μια ή περισσότερες στοίβες από κύβους αφού προκαθορίσουμε την σειρά των κύβων στην στοίβα (ποιος κύβος πρέπει να είναι πάνω από ποιό.

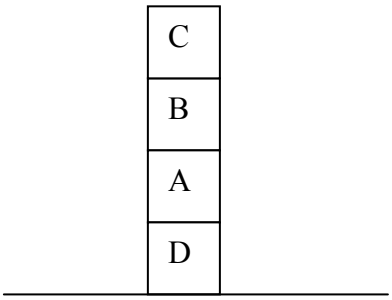
Στο παράδειγμα μας, θεωρούμε ότι αρχικά ο κύβος Β βρίσκεται τοποθετημένος πάνω από τον Α και ο Α πάνω από τον C.

Οι κύβοι C και D είναι απευθείας τοποθετημένοι πάνω στο τραπέζι(Σχήμα 1.2α). Σαν στόχο έχουμε θέσει να δημιουργηθεί μια στοίβα στην οποία ο C θα βρίσκεται στην κορυφή και πάνω από τον Β.Ο Β θα βρίσκεται πάνω από τον Α και ο Α πάνω από τον D.Ο D θα βρίσκεται απευθείας τοποθετημένος πάνω στο τραπέζι (Σχήμα 1.2β).



Initial state

Σχήμα 1.2α



Goal state

Σχήμα 1.2β

Σχήμα 1.2:Διαταξη των κύβων στην αρχική κατάσταση (Σχήμα 1.2α) και τελική διάταξη των κύβων με την επίτευξη του στόχου(Σχήμα 1.2β).

Πιο κάτω, στο Σχήμα1.3 παρουσιάζεται η περιγραφή του προβλήματος στην γλώσσα STRIPS. Σύμφωνα με την περιγραφή του προβλήματος, στην αρχική κατάσταση ο κύβος Β βρίσκεται απευθείας πάνω από τον Α(κατηγόρημα On(B,A)).Ο Α βρίσκεται απευθείας πάνω από τον C (On(A, C)) και ο C, D βρίσκονται πάνω στο απευθείας πάνω στο τραπέζι (κατηγορήματα On(C, Table),On(D, Table)).Επίσης, δεν υπάρχει άλλος κύβος πάνω από τον Β και D (κατηγορήματα Clear(B), Clear(D)).

Στόχος είναι, ο C κύβος να βρίσκεται απευθείας πάνω από τον Β(On(C,B)),ο Β απευθείας πάνω από τον Α (On(B,A)) και ο Α απευθείας πάνω από τον D (On(A,D)).

Η δράση put-on παίρνει σαν παραμέτρους x, y, z κύβους. Σκοπός της, είναι η τοποθέτηση ενός κύβου πάνω σε κάποιο άλλο. Προαπαίτηση της είναι, να μην υπάρχει άλλος κύβος πάνω από τον x (Clear(x)), να μην υπάρχει άλλος κύβος πάνω από τον y (Clear(y)),το x , y , z να είναι κύβοι (Block(x),Block(y), Block(z)) και ο x κύβος να είναι απευθείας πάνω από τον z (On (x, z)). Σαν αποτέλεσμα της δράσης, ο x είναι απευθείας πάνω από τον y(On(x, y)),δεν υπάρχει άλλος κύβος πάνω από τον z (Clear (z)),αλλά υπάρχει κύβος πάνω από τον y (¬ Clear(y)) , ενώ ο x δεν είναι πλέον απευθείας πάνω από τον z (¬On(x, z)).
Η δράση put-table παίρνει σαν παραμέτρους x, z κύβους και το τραπέζι Table. Σκοπός της, είναι η τοποθέτηση κάποιου κύβου απευθείας πάνω στο τραπέζι. Προαπαίτηση της είναι, να μην υπάρχει άλλος κύβος πάνω από τον x (Clear(x)),ο x να είναι απευθείας πάνω από τον z (On(x,z)) και οι x,z να είναι κύβοι(Block(x),Block(z)). Σαν αποτέλεσμα της δράσης ο x κύβος τοποθετείτε απευθείας πάνω στο τραπέζι (On(x, Table)), δεν υπάρχει άλλος κύβος πάνω από τον z (Clear (z)) και ο x δεν βρίσκεται πλέον πάνω από τον z (¬On(x,z)).

Προσέξτε ότι θέσαμε σαν προϋπόθεση οι δράσεις να εκτελούνται πάνω σε κύβους για να είναι ξεκάθαρη η λειτουργία της κάθε μιας δράσης. Θα μπορούσε να χρησιμοποιηθεί η put-on δράση αλλά σαν ορίσματα να περνούσαμε το τραπέζι και ένα κύβο και έτσι θα τοποθετηθώ ο κύβος πάνω στο τραπέζι μέσο της put-on δράσης και όχι με την put-table η οποία δημιουργήθηκε για αυτό τον σκοπό.

Init (On (B, A) $\wedge$ On (A, C) $\wedge$ On(C, Table) $\wedge$ On (D, Table) $\wedge$ Clear (B) $\wedge$
Clear (D))

Goal (On(C, B) $\wedge$ On (B, A) $\wedge$ On (A, D))
Action (put-on(x, y, z),
PRECOND: Clear(x) $\wedge$Clear(y) $\wedge$ On (x, z) $\wedge$Block(x) $\wedge$ Block(y)
$\wedge$ Block(z)
EFFECT: On(x, y) $\wedge$ Clear (z) $\wedge$ ¬ Clear(y) $\wedge$¬On(x, z))

Action (put-table(x, z, Table),

PRECOND: Clear(x) $\wedge$ On(x, z) $\wedge$ Block(x) $\wedge$ Block (z)

EFFECT: On(x, Table) $\wedge$ Clear (z) $\wedge$ ¬On(x, z))

Σχήμα 1.3: Περιγραφή του blocks world problem σε STRIPS

3. Air Cargo Transport

Στο πρόβλημα αυτό, μελετάτε η φόρτωση και η εκφόρτωση αποσκευών σε αεροπλάνα και τα δρομολόγια τους [[7] p380]. Πιο κάτω, στο Σχήμα 1.4 παρουσιάζεται η περιγραφή του προβλήματος στην γλώσσα STRIPS.

Σύμφωνα με την περιγραφή του προβλήματος, στην αρχική κατάσταση ισχύει ότι το C1 φορτίο βρίσκεται στο αεροδρόμιο SFO (κατηγόρημα At(C1,SFO)),το C2 φορτίο στο JFK (At(C2,JFK)),το αεροπλάνο P1 βρίσκεται στο SFO (At(P1,SFO)) και το P2 βρίσκεται στο JFK (At(P2,JFK)).Επίσης, μέσον των κατηγορημάτων Airport(a), Plane(p) και Cargo(c) επαληθεύεται η υπάρξει των αεροδρομίων, αεροπλάνων και φορτίων που χρησιμοποιούνται Cargo(C1),Cargo(C2)Plane(P1),Plane(P2), Airport(JFK), Airport(SFO)). Στόχος είναι το φορτίο C1 να βρεθεί στο αεροδρόμιο JFK (At (C1, JFK)) και το φορτίο C2 στο SFO αεροδρόμιο (At (C2, SFO)).

Η δράση Load, παίρνει σαν παραμέτρους ένα φορτίο, ένα αεροπλάνο και ένα αεροδρόμιο (c, p, a αντίστοιχα). Σκοπός της είναι να γίνει η φόρτωση του φορτίου c στο p αεροπλάνο. Προαπαίτηση της είναι το φορτίο c να βρίσκεται στο αεροδρόμιο a (At(c,a)), το αεροπλάνο p να βρίσκεται και αυτό στο αεροδρόμιο a (At(p,a)) και να υπάρχουν το αεροπλάνο, το φορτίο και το αεροδρόμιο(Cargo(c),Plane(p),Airport(a)). Αποτέλεσμα της είναι, το φορτίο c να μην βρίσκετε πλέον στο αεροδρόμιο a αλλά μέσα στο αεροπλάνο p (¬At(c,a) , In(c,p)).

Η δράση Unload παίρνει σαν παραμέτρους ένα φορτίο, ένα αεροπλάνο και ένα αεροδρόμιο (c, p, a αντίστοιχα). Σκοπός της είναι να γίνει η εκφόρτωση από το

αεροπλάνο p στο αεροδρόμιο a. Προαπαίτηση της είναι το φορτίο c να βρίσκεται μέσα στο αεροπλάνο p, το οποίο βρίσκεται στο αεροδρόμιο a ((c,p) ,At(p,a)) και να υπάρχουν το αεροπλάνο, το φορτίο και το αεροδρόμιο(Cargo(c),Plane(p),Airport(a)). Αποτέλεσμα της είναι, το φορτίο c να βρεθεί στο αεροδρόμιο a και να μην βρίσκεται πλέον μέσα στο αεροπλάνο p (At(c, a) ,¬ In(c, p)).

Η δράση Fly παίρνει σαν παραμέτρους ένα αεροπλάνο και δύο αεροδρόμια (p, from, to αντίστοιχα). Σκοπός της είναι να γίνει η πτήση από ένα αεροδρόμιο σε ένα άλλο. Προαπαίτηση της είναι να υπάρχει το αεροπλάνο p και να είναι στο αεροδρόμιο αναχώρησης (At (p, from),Plane (p)).Επίσης , πρέπει να υπάρχει το αεροδρόμιο προορισμού και αναχώρησης(Airport (from) ,Airport(to)). Αποτέλεσμα της είναι, το αεροπλάνο p να μην βρίσκεται στο αεροδρόμιο αναχώρησης αλλά στο προορισμού (¬ At (p, from),At(p,to)).

Init(At(C1,SFO) ∧At(C2,JFK) ∧At(P1,SFO) ∧
At (P2,JFK) ∧ Cargo(C1) ∧Cargo(C2) ∧Plane (P1) ∧
Plane (P2) ∧ Airport (JFK)∧Airport (SFO))

Goal (At (C1, JFK) ∧At (C2, SFO))

Action (Load(c, p, a),
PRECOND: At(c,a) ∧At(p,a) ∧Cargo(c) ∧Plane(p) ∧Airport(a)
EFFECT: ¬At(c, a) ∧In(c,p))

Action (Unload(c,p,a),
PRECOND: In(c,p) ∧At(p,a) ∧Cargo(c) ∧ Plane(p) ∧Airport(a)
EFFECT: At(c, a) ∧ ¬ In(c, p))

Action (Fly (p, from, to),

PRECOND: At (p, from) ∧Plane (p) ∧Airport (from) ∧Airport(to)

EFFECT: ¬ At (p, from) ∧ At(p,to))

Σχήμα 1.4: Περιγραφή του Air Cargo Transport problem σε STRIPS

1.2.2 Γλώσσα PDDL

Η γλώσσα PDDL αναπτύχθηκε το 1998 από τον Drew Mc Dermott.Τα αρχικά της αναπαριστούν: Planning Domain Description Language.Έκτοτε, η γλώσσα αυτή είναι πρότυπο για την αναπαράσταση των μοντέλων του κόσμου προγραμματισμού δράσης (planning domain models).

Μέσον της γλώσσας αυτής θα αναπαραστήσουμε τον κόσμο του προβλήματος και το πρόβλημα. Αρχή είναι, ο κάθε κόσμος και το κάθε πρόβλημα, να αναπαρίσταται το κάθε ένα σε ένα αρχείο ξεχωριστά.

Με στόχο τον χειρισμό της γλώσσας από διάφορους προγραμματιστές δράσεις (planners), έχουν δημιουργηθεί υποσύνολα με χαρακτηριστικά της γλώσσας τα οποία ονομάζονται απαιτήσεις (requirements) και δηλώνονται στον ορισμό του κόσμου για να μπορεί ο εκάστοτε προγραμματιστής δράσεων να καθορίσει αν μπορεί να χειριστεί το υποσύνολο της γλώσσας που του δόθηκε. Η γλώσσα STRIPS είναι ένα υποσύνολο της PDDL.

Για να δώσουμε μια πρώτη “γεύση” της γλώσσας, θα εξετάσουμε το παράδειγμα του Pendnault [[4] p 2],το οποίο περιλαμβάνει την μεταφορά αντικειμένων μεταξύ του σπιτιού και του χώρου εργασίας με την χρήση χαρτοφύλακα. (Η πλήρης σύνταξη και σημασιολογία της γλώσσας θα δοθεί πιο κάτω).

Στο Σχήμα 1.5 παρουσιάζεται το πρώτο μέρος του ορισμού του κόσμου του χαρτοφύλακα, το οποίο περιλαμβάνει τις απαιτήσεις, τους τύπους, τις σταθερές

και τα κατηγορήματα που υπάρχουν σε αυτό τον κόσμο. Μέσον των απαιτήσεων του κόσμου, ο προγραμματιστής δράσης μπορεί να καθορίσει αν μπορεί να χειριστεί τον συγκεκριμένο κόσμο. Στο παράδειγμα μας, στα αποτελέσματα δράσεων όπως θα δούμε αργότερα χρησιμοποιούνται σχεσιακοί τελεστές οπότε κάποιος προγραμματιστής δράσης ο οποίος χειρίζεται μόνο STRIPS δεν θα μπορούσε να χειριστεί το συγκεκριμένο κόσμο και πρόβλημα. Παρατηρούμε επίσης, την χρήση δεσμευμένων λέξεων στην δήλωση των απαιτήσεων αλλά και των τύπων και των σταθερών. Η μορφή μιας δεσμευμένης λέξης είναι :<όνομα δεσμευμένης λέξης>. Στο παράδειγμα μας, η :requirements υποδηλώνει τις απαιτήσεις του κόσμου, η :strips υποδηλώνει ότι ο συγκεκριμένος κόσμος εκφράζεται με την χρήση της γλώσσας STRIPS,η :equality υποδηλώνει ότι γίνεται χρήση του σχεσιακού τελεστή της ισότητας, η :typing ότι γίνετε χρήση τύπων, η :conditional-effects (αποτελέσματα υπό συνθήκη), ότι στα αποτελέσματα δράσεων θα γίνει χρήση σχεσιακών τελεστών, η :types για δήλωση των τύπων που υπάρχουν στον κόσμο, η :constants για τις σταθερές και η :predicates για τα κατηγορήματα).Μια δεσμευμένη λέξη που χρησιμοποιείται στις απαιτήσεις (:requirements) ονομάζεται requirement flag.Στον κάθε κόσμο (domain) πρέπει να δηλώνονται τα requirement flags του.

Όλοι οι κόσμοι περιλαμβάνουν κάποιους βασικούς τύπους. Ένας από αυτούς είναι ο object .Στο παράδειγμα μας ορίζονται δύο επιπλέον τύποι ο location ο οποίος υποδεικνύει τοποθεσία και ο physob «φυσικό αντικείμενο».

Μια σταθερά (constant), είναι ένα «σύμβολο» το οποίο θα έχει το ίδιο νόημα για όλα τα προβλήματα του συγκεκριμένου κόσμου. Στο παραδειγμά μας, ο χαρτοφύλακας Β είναι μια σταθερά. Προσέξτε ότι ενώ θα μπορούσαμε να έχουμε σαν τύπο τον ίδιο τον χαρτοφύλακα αυτό δεν γίνεται διότι μόνο ένας υπάρχει.
Έπειτα γίνεται η δήλωση των κατηγορημάτων που υπάρχουν στον κόσμο (:predicates). Το κατηγόρημα at ?x – physob ?l - location αναφέρεται στο ότι το φυσικό αντικείμενο x βρίσκεται στην τοποθεσία l. Το in ?x ?y – physob αναφέρεται στο ότι το φυσικό αντικείμενο x βρίσκεται μέσα στο y.

```
(define (domain briefcase-world)
(: requirements :strips :equality :typing :conditional-effects)
(:types location physob)
(:constants (B - physob))
(:predicates (at ?x - physob ?l - location)

  (in ?x ?y - physob))
...
```

Σχήμα 1.5:Πρώτο μέρος περιγραφής του κόσμου του χαρτοφύλακα σε PDDL

Στο δεύτερο μέρος ορισμού του κόσμου ορίζονται οι δράσεις. Γενικά, ο ορισμός μιας δράσης αποτελείτε από τέσσερα μέρη: το όνομα της δράσης, τις παραμέτρους που δέχεται η δράση, τις συνθήκες οι οποίες πρέπει να ισχύουν ούτως ώστε η δράση να μπορεί να εκτελεστεί (preconditions) και τα αποτελέσματα (effects) της εκτέλεσης της συγκεκριμένης δράσης.

Στο Σχήμα 1.6 παρουσιάζεται ο ορισμός της δράσης mov-b μέσον της οποίας ο χαρτοφύλακας μπορεί να μεταφερθεί από την τοποθεσία m στην τοποθεσία l.Οι δύο αυτές τοποθεσίες δίδονται σαν παράμετροι της δράσεις. Μια μεταβλητή δηλώνεται ως εξής: ?<όνομα μεταβλητής> - <τύπος μεταβλητής>. Προαπαίτηση της δράσης είναι, ο χαρτοφύλακας να βρίσκεται αρχικά στην τοποθεσία m και η τοποθεσία m που είναι τώρα ο χαρτοφύλακας να μην είναι η ίδια με την τοποθεσία που θέλουμε να τον μεταφέρουμε. Δηλαδή δεν μπορούμε να χρησιμοποιήσουμε αυτή την δράση και ο χαρτοφύλακας να παραμείνει στην ίδια θέση. Σαν αποτέλεσμα της δράσης ο χαρτοφύλακας μεταφέρεται στον προορισμό του, δεν βρίσκεται πλέον εκεί που ήταν αρχικά (κατηγορήματα (at B ?l) και (not (at B ?m) αντίστοιχα) και βέβαια οτιδήποτε βρίσκεται μέσα στον χαρτοφύλακα έχει μετακινηθεί και αυτό ((forall (?z) when (and (in ?z) (not (= ?z B))) (and (at ?z ?l) (not (at ?z ?m))))))). Παρατηρούμε ότι στα αποτελέσματα της δράσης έγινε χρήση των υπό συνθήκη αποτελεσμάτων (conditional effects) που έχουν δηλωθεί σαν απαιτήσεις του κόσμου του χαρτοφύλακα στο πρώτο μέρος του ορισμού του κόσμου.

```
(:action mov-b
        :parameters (?m ?l - location)
        :precondition (and (at B ?m) (not (= ?m  ?l)))
        :effect (and (at B ?l) (not (at B ?m))
                        (forall (?z)
                        (when (and (in ?z) (not (= ?z B)))
                                (and (at ?z ?l) (not (at ?z ?m)))))) )
```

Σχήμα 1.6:Ορισμός δράσης mov-b του κόσμου του χαρτοφύλακα σε PDDL.

Ακολούθως, παρουσιάζεται στο Σχήμα 1.7 ο ορισμός της δράσης put-in μέσον της οποίας τοποθετούμε αντικείμενα μέσα στον χαρτοφύλακα. Η δράση αυτή, δέχεται σαν παραμέτρους ένα αντικείμενο (μεταβλητή x τύπου physob) και μια τοποθεσία (μεταβλητή l τύπου location).Προαπαίτηση της είναι το αντικείμενο να μην είναι ο ίδιος ο χαρτοφύλακας (not (= ?x B)). Αποτέλεσμα της είναι όταν το αντικείμενο να βρεθεί στην ίδια τοποθεσία με τον χαρτοφύλακα να είναι μέσα στον χαρτοφύλακα. Παρατηρούμε ότι και εδώ έχουμε υπό συνθήκη αποτελέσματα (conditional effects) , δηλαδή αν δράση εκτελεστή όταν το αντικείμενο δεν βρίσκεται στην ίδια τοποθεσία με τον χαρτοφύλακα τότε η δράση δεν θα έχει αποτελέσματα.

```
(:action put-in
        :parameters (?x - physob ?l - location)
        :precondition (not (= ?x B))
        :effect (when (and (at ?x ?l) (at B ?l))
                        (in ?x)) )
```

Σχήμα 1.7: Ορισμός δράσης put-in του κόσμου του χαρτοφύλακα σε PDDL.

Η τελευταία δράση του κόσμου του χαρτοφύλακα είναι η take-out και ορίζεται στο Σχήμα 1.8. Μέσον αυτής της δράσης, μπορούμε να πάρουμε κάποιο αντικείμενο από τον χαρτοφύλακα. Παράμετρος της είναι, το αντικείμενο που θέλουμε να πάρουμε (μεταβλητή x τύπου physob). Προαπαίτηση της δράσης

είναι, το αντικείμενο που θέλουμε να αφαιρέσουμε, να μην είναι ο ίδιος ο χαρτοφύλακας B (not (= ?x B)). Αποτέλεσμα της είναι, το αντικείμενο x να μην βρίσκεται πλέον μέσα στον χαρτοφύλακα B. Αυτή είναι η τελευταία δράση του κόσμου του χαρτοφύλακα και με αυτήν ολοκληρώνεται ο ορισμός του κόσμου.

```
(:action take-out)
            :parameters (?x - physob)
            :precondition (not (= ?x B))
            :effect (not (in ?x)) )
```

Σχήμα 1.8: Ορισμός δράσης take-out του κόσμου του χαρτοφύλακα σε PDDL.

Το πρόβλημα για το παράδειγμα μας είναι το εξής: στο σπίτι έχουμε ένα λεξικό και ένα χαρτοφύλακα με μια επιταγή μέσα. Στόχος μας είναι να μεταφέρουμε το λεξικό και τον χαρτοφύλακα στον χώρο εργασίας μας αλλά θέλουμε να βγάλουμε την επιταγή από τον χαρτοφύλακα και την αφήσουμε έξω. Στο Σχήμα 1.9 παρουσιάζεται ο ορισμός του προβλήματος. Αρχικά δηλώνουμε το όνομα του προβλήματος (get-paid).Στην συνέχεια, καθορίζουμε τον κόσμο στον οποίο αναφέρετε το πρόβλημα. Στην περίπτωση μας είναι ο briefcase-world. Έπειτα ορίζεται η αρχική κατάσταση του κόσμου για το συγκεκριμένο πρόβλημα. Στην αρχική κατάσταση του προβλήματος μας, υπάρχουν το σπίτι μας ο χώρος εργασίας μας , η επιταγή το λεξικό και ο χαρτοφύλακας (κατηγορήματα (place home), (place office), (object p), (object d), (object b)). Το λεξικό και ο χαρτοφύλακας με την επιταγή μέσα βρίσκονται στο σπίτι μας (κατηγορήματα (at B home) (at P home) (at D home) (in P)). Στόχος είναι ο χαρτοφύλακας και το λεξικό να βρεθούν στον χώρο εργασίας μας (κατηγορήματα (at B office) ,(at D office)), ενώ η επιταγή πρέπει να παραμείνει στο σπίτι(κατηγορήματα(at P home)).

```
(define (problem get-paid)
        (:domain briefcase-world)
```

```
(:init (place home) (place office)
       (object p) (object d) (object b)
       (at B home) (at P home) (at D home) (in P))
(:goal (and (at B office) (at D office) (at P home))))
```

Σχήμα 1.9: Ορισμός δράσης get-paid του κόσμου του χαρτοφύλακα σε PDDL.

Σύνταξή γλώσσας PDDL

Για την σύνταξη της γλώσσας χρησιμοποιούνται οι ακόλουθες συμβάσεις [[4] p 3]:

- Κάθε κανόνας είναι της μορφής <συντακτικό στοιχείο> ::= ορισμός. Το πρώτο μέρος του κανόνα (<συντακτικό στοιχείο> ::=) είναι για δική μας χρήση για κατανόηση του τι ορίζει το δεύτερο μέρος του κανόνα, δεν χρησιμοποιείται στον κώδικα.
- Οι < > προσδιορίζουν ονόματα και συντακτικά στοιχεία.
- Οι [ ] περικλείουν προαιρετικά στοιχεία. Όταν έχουμε την μορφή [(:types ...)](:typing) όπου οι [ ] έχουν σαν εκθέτη requirement flag, τότε το περιεχόμενο των [ ] θα συμπεριληφθεί, αν στον ορισμό του κόσμου έχει δηλωθεί σαν απαίτηση (requirement) το requirement flag.
- Παρόμοια στο σύμβολο ::= μπορούμε να χρησιμοποιήσουμε σαν εκθέτη κάποιο requirement flag,το οποίο υποδεικνύει ότι ο ορισμός είναι πιθανός, αν έχει δηλωθεί στον κόσμο το συγκεκριμένο requirement flag.
- Ο αστερίσκος (*) σημαίνει μηδέν ή περισσότερο και το (+) σημαίνει ένα ή περισσότερο.
- Μερικά συντακτικά στοιχεία τοποθετούνται σε παρένθεση. Π.χ. το <list (symbol)> μπορεί να σημαίνει ότι έχουμε μια λίστα από σύμβολα. Μπορεί να υπάρχει ένας συντακτικός ορισμός για το <list x> και ακόμα ένας για το <symbol>. Π.χ συντακτικός ορισμός για <list x> ::= (x*) οπότε μια λίστα από σύμβολα μπορεί να συμβολίζεται απλούστερα σαν (<symbol>*).

- Οι παρενθέσεις για την γλώσσα χρησιμοποιούνται με την προκαθορισμένη – κοινοί τους σημασία και δεν έχουν κάποιο άλλο εξειδικευμένο νόημα.

Κόσμος – Domain

Στο Σχήμα 1.10 [[5] p1] που ακολουθεί, παρουσιάζονται οι συντακτική κανόνες της γλώσσας PDDL για τον ορισμό κόσμου – domain. Ο ορισμός του κόσμου αποτελείται από δύο μέρη: το πρώτο είναι ο ορισμός των στοιχείων του κόσμου (εκτός των δράσεων) και το δεύτερο από τον ορισμό των δράσεων. Στο πρώτο μέρος, αρχικά δηλώνεται το όνομα του domain (Τα διάφορα ονόματα που χρησιμοποιούνται (<name>) είναι σειρές από χαρακτήρες οι οποίες αρχίζουν με γράμμα , περιέχουν γράμματα, ψηφία, "-" και " _" .Η θήκη δεν έχει σημασία). Ακολούθως, θα πρέπει να οριστούν οι διάφορες απαιτήσεις ([<require-def>]), οι τύποι([<types-def>] $^{:typing}$), οι σταθερές ([<constants-def>]),τα κατηγορήματα ([<predicates-def>]), οι συναρτήσεις([<functions-def>]$^{:fluents}$) και οι περιορισμοί ([<constraints>]). Όλα αυτά τα πεδία είναι προαιρετικά για τον ορισμό του κόσμου. Στο δεύτερο μέρος, υποχρεωτικά θα πρέπει να οριστούν οι δράσεις του κόσμου (<structure-def>*).

Αναλυτικότερα, όσον αφορά τον ορισμό των απαιτήσεων (<require-def>) θα δοθεί πιο κάτω μια λίστα με τις απαιτήσεις που μπορούν να χρησιμοποιηθούν. Αν δεν δηλωθεί κάποια απαίτηση θεωρούμε ότι η προκαθορισμένη απαίτηση είναι η γλώσσα STRIPS (:strips).

Όσον αφορά τον ορισμό τύπων, αν ο προγραμματιστής δράσης που χρησιμοποιούμε μπορεί να χειριστεί ονόματα τύπων όταν δηλώνουμε μεταβλητές (στα :requirements πρέπει να δηλωθεί η δεσμευμένη λέξη :typing για να επιτραπούν οι τύποι μεταβλητών) τότε μπορούμε να δηλώσουμε τύπους(<types-def>). Στον ορισμό των τύπων γίνεται χρήση μιας λίστας τύπων μέσον της οποίας δηλώνουμε τους τύπους μιας λίστας αντικειμένων. Πριν από τους τύπους προηγείται «-« και ακολούθως τα αντικείμενα. Το κάθε αντικείμενο ανήκει στον πρώτο τύπο που ακολουθεί μετά την «-«. Ένα παράδειγμα λίστας τύπων <typed list(name)> είναι car truck - vehicle. Αν αυτή η λίστα τύπων δηλωθεί στους τύπους (:types) του κόσμου τότε έχουμε δηλώσει δύο νέους τύπους τον truck(φορτηγό) και car (αυτοκίνητο) οι οποίοι είναι υποκλάσεις του τύπου vehicle (οχήματα) .Έτσι, κάθε αυτοκίνητο και κάθε φορτηγό είναι

οχήματα. Εκτός από απλούς τύπους υπάρχουν και σύνθετοι τύποι. Στον συντακτικό κανόνα: <type> ::= (either <primitive-type>$^{+}$) υποδηλώνεται η ένωση τύπων (π.χ either $t_1 \ldots t_k$ είναι η ένωση των τύπων $t_1 \ldots t_k$).

Για τον ορισμό σταθερών, χρησιμοποιείται λίστα τύπων όπως και στον ορισμό τύπων. Πρώτα μπαίνει η λίστα με τις σταθερές, μετά το «-« και μετά ο τύπος των σταθερών. Π.χ. :constants sahara – desert division1 division2 – division, όπου η σταθερά sahara υποδηλώνει desert και οι division1, division2 σταθερές υποδηλώνουν division.

Η δήλωση των κατηγορημάτων του κόσμου γίνεται με λίστα δήλωσης κατηγορημάτων , η οποία συντάσσεται όπως την λίστα τύπων.

Συναρτήσεις μπορούν να οριστούν αν στην δήλωση των απαιτήσεων του κόσμου δηλωθεί η δεσμευμένη λέξη :fluents με την οποία επιτρέπονται οι ορισμοί συναρτήσεων και η χρήση αποτελεσμάτων (effects) με τελεστές ανάθεσης και αριθμητικές συνθήκες. Οι συναρτήσεις μπορούν να επιστρέψουν μόνο αριθμό.

<domain> ::= (define (domain <name>)
[<require-def>]
[<types-def>] :typing
[<constants-def>]
[<predicates-def>]
[<functions-def>]:fluents
[<constraints>]
<structure-def>*)
<require-def> ::= (:requirements <require-key>$^{+}$)
<require-key> ::= βλέπε απαιτήσεις - requirements
<types-def> ::= (:types <typed list (name)>)
<constants-def> ::= (:constants <.typed list (name)>)
<predicates-def> ::= (:predicates <atomic formula skeleton>$^{+}$)
<atomic formula skeleton> ::= (<predicate> <typed list (variable)>)

<predicate> ::= <name>

<variable> ::= ?<name>

<atomic function skeleton> ::= (<function-symbol> <typed list (variable)>)

<function-symbol> ::= <name>

<functions-def> ::=$^{:fluents}$ (:functions <function typed list

(atomic function skeleton)>)

<constraints> ::=:constraints (:constraints <con-GD>)

<structure-def> ::= <action-def>

<structure-def> ::=:durative−actions <durative-action-def>

<structure-def> ::=:derived−predicates <derived-def>

<typed list (x)> ::= x*

<typed list (x)> ::=:typing x^{+}- <type> <typed list(x)>

<primitive-type> ::= <name>

<type> ::= (either <primitive-type>$^{+}$)

<type> ::= <primitive-type>

<function typed list (x)> ::= x*

<function typed list (x)> ::=:typing x^{+}- <function type>

<function typed list(x)>

<function type> ::= number

<emptyOr (x)> ::= ()

<emptyOr (x)> ::= x

Σχήμα 1.10:Σύντακτικοί κανόνες ορισμού στοιχείων του κόσμου σε PDDL.

Δράσεις – Actions

Για να ολοκληρωθεί ο ορισμός του κόσμου , θα πρέπει να οριστούν οι δράσεις που υπάρχουν. Στο Σχήμα 1.11 [[5] p2] που ακολουθεί, παρουσιάζονται οι συντακτική κανόνες της γλώσσας PDDL για τον ορισμό των δράσεων (actions) του κόσμου (domain).Ο ορισμός μιας δράσης αποτελείται από δύο μέρη: στο πρώτο μέρος ορίζουμε το όνομα της δράσης και τις παραμέτρους που δέχεται.

Στο δεύτερο, ορίζουμε το σώμα της δράσης δηλώνοντας τα προαπαιτούμενα κατηγορήματα τα οποία θα πρέπει να ισχύουν για να μπορεί να εκτελεστεί η δράση και τα αποτελέσματα μετά την εκτέλεση της δράσης. Παρατηρούμε ότι, πριν την δήλωση του ονόματος της δράσης προηγείται η δεσμευμένη λέξη :action και πριν την δήλωση παραμέτρων η :parameters δεσμευμένη λέξη. Η δήλωση των παραμέτρων της δράσης γίνεται με την σύνταξη λίστας τύπων. Η δράση έχει σαν προαπαιτήσεις, την περιγραφή στόχων οι οποίο πρέπει να εκπληρωθούν για να εκτελεστεί η δράση(<emptyOr (pre-GD)>). Μια δράση η οποία δεν έχει προαπαιτήσεις είναι πάντα εκτελέσιμη. Ακολουθούν τα αποτελέσματα μετά την εκτέλεση της δράσης.

<action-def> ::= (:action <action-symbol>
:parameters (<typed list (variable)>)
<action-def body>)
<action-symbol> ::= <name>
<action-def body> ::= [:precondition <emptyOr (pre-GD)>]
[:effect <emptyOr (effect)>]

Σχήμα 1.11:Συντάκτικοί κανόνες PDDL για ορισμό δράσεων.

Στο Σχήμα 1.12, παρουσιάζονται οι συντακτικοί κανόνες για τις περιγραφές στόχων[[5],p2],(δηλαδή τις προαπαιτήσεις για την εκτέλεση της δράσης) και κανόνες για συναρτήσεις σε μια δράση.

Οι προαπαιτήσεις μπορούν να εκφραστούν σαν μια σύζευξη πολλών περιγραφών στόχων. Ακόμη, μπορούν να εκφραστούν μέσον του τελεστή forall. Η χρήση του συγκεκριμένου τελεστή θα πρέπει να δηλωθεί στις προαπαιτήσεις του κόσμου(πρέπει να δηλωθεί :universal−preconditions σαν requirements flag) . Τον καθολικό τελεστή forall ακολουθεί μια λίστα τύπων μεταβλητών. Ο τελεστής θα επιστρέψει αληθή τιμή αν όλες οι μεταβλητές στην λίστα τύπων που του δόθηκε ως όρισμα είναι αληθείς . Την λίστα τύπων μεταβλητών ακολουθεί η περιγραφή στόχου.

Επίσης, η περιγραφή στόχων μπορεί να γίνει και μέσον του τελεστή προτιμήσεων preference. Με αυτόν τον τελεστή, δηλώνουμε ποιές περιγραφές στόχων θα προτιμούσαμε εμείς ως χρήστες να ισχύουν για να εκτελεστεί η δράση αλλά και αν ακόμα αυτές οι περιγραφές στόχων δεν εκπληρωθούν η δράση πάλι θα μπορεί να εκτελεστεί. Θεωρούμε ως παραβίαση της σύνταξης της γλώσσας αν υπάρχει κάποια προτίμηση σαν μέρος της συνθήκης κάποιας δράσης με αποτελέσματα υπό συνθήκη (conditional effects).Βέβαια, για την χρήση του τελεστή προτιμήσεων, απαραίτητη είναι η δήλωση του στις προαπαιτήσεις του κόσμου (σαν :requirements flag δηλώνουμε :preferences). Πάντα θεωρούμε τις προτιμήσεις ως αληθείς για τον λόγο ότι η αποτυχία να τις ικανοποιήσουμε δεν καθιστά τις περιγραφές στόχου ψευδής[[6] p 5].

Στην περιγραφή στόχων μπορούν να χρησιμοποιηθούν και ψευδή κατηγορήματα (κανόνας <GD> ::=$^{:negative-preconditions}$ <literal(term)>) αφού δηλωθεί στις προαπαιτήσεις του κόσμου το ανάλογο requirement flag (:negative−preconditions).

Επιπλέον, μπορούν να χρησιμοποιηθούν οι τελεστές exist, imply, or, and και not για περιγραφή του στόχου. Για να επιστρέψει αληθή τιμή ο τελεστής exists θα πρέπει έστω μία από τις μεταβλητές που δέχεται σαν όρισμα να είναι αληθείς. Για την χρήση των τελεστών θα πρέπει να γίνει η δήλωση των ανάλογων requirement flags στις προαπαιτήσεις του κόσμου.

<pre-GD> ::= <pref-GD>
<pre-GD> ::= (and <pre-GD>*)
<pre-GD> ::=$^{:universal-preconditions}$(forall (<typed list(variable)>) <pre-GD>)
<pref-GD> ::=$^{:preferences}$ (preference [<pref-name>] <GD>)
<pref-GD> ::= <GD>
<pref-name> ::= <name>
<GD> ::= <atomic formula(term)>
<GD> ::=$^{:negative-preconditions}$ <literal(term)>
<GD> ::= (and <GD>*)
<GD> ::=$^{:disjunctive-preconditions}$ (or <GD>*)

<GD> ::=:disjunctive−preconditions (not <GD>)

<GD> ::=:disjunctive−preconditions (imply <GD> <GD>)

<GD> ::=:existential−preconditions (exists (<typed list(variable)>) <GD>)

<GD> ::=:universal−preconditions (forall (<typed list(variable)>) <GD>)

<GD> ::=:fluents <f-comp>

<f-comp> ::= (<binary-comp> <f-exp> <f-exp>)

<literal(t)> ::= <atomic formula(t)>

<literal(t)> ::= (not <atomic formula(t)>)

<atomic formula(t)> ::= (<predicate> t*)

<term> ::= <name>

<term> ::= <variable>

<f-exp> ::= <number>

<f-exp> ::= (<binary-op> <f-exp> <f-exp>)

<f-exp> ::= (- <f-exp>)

<f-exp> ::= <f-head>

<f-head> ::= (<function-symbol> <term>*)

<f-head> ::= <function-symbol>

<binary-op> ::= <multi-op>

<binary-op> ::= − <binary-op> ::= /

<multi-op> ::= _ <multi-op> ::= +

<binary-comp> ::= >

<binary-comp> ::= <

<binary-comp> ::= =

<binary-comp> ::= >=

<binary-comp> ::= <=

<number> ::= Any numeric literal (integers and floats of form n.n).

Σχήμα 1.12: Συντακτικοί κανόνες για τις περιγραφές στόχων σε PDDL.

Οι συντακτικοί κανόνες για τα αποτελέσματα από την εκτέλεση μιας δράσης[6] παρουσιάζονται στο Σχήμα 1.13. Τα αποτελέσματα μπορούν να παρουσιαστούν σαν μια σύζευξη αποτελεσμάτων. Επίσης στην σύνταξη

αποτελεσμάτων είναι επιτρεπτό από την γλώσσα η χρήση αποτελεσμάτων υπό συνθήκη (conditional effects),μετά από την ανάλογη δήλωση τους (:conditional−effects) σαν requirement flags στις απαιτήσεις του κόσμου.
Στα υπό συνθήκη αποτελέσματα γίνεται χρήση των τελεστών forall και when.Ο τελεστής when(P E) θα επιστρέψει αληθή τιμή αν η συνθήκη P του τελεστή είναι αληθείς πριν την εκτέλεση της δράσης και το αποτέλεσμα E θα εκτελεστεί μετά. Η συνθήκη P είναι δευτερεύουσα προαπαίτηση οπότε ακόμα και αν δεν ικανοποιηθεί, η δράση θα εκτελεστεί αλλά το αποτέλεσμα E δεν θα είναι αληθές. Το αποτέλεσμα E θα συμβεί μόνο αν η συνθήκη P είναι αληθείς.
Ακόμα ένας τρόπος έκφρασης των αποτελεσμάτων είναι με την χρήση συναρτήσεων.

<effect> ::= (and <c-effect>*)
<effect> ::= <c-effect>
<c-effect> ::=$^{:conditional-effects}$ (forall (<typed list (variable)>*) <effect>)
<c-effect> ::=$^{:conditional-effects}$ (when <GD> <cond-effect>)
<c-effect> ::= <p-effect>
<p-effect> ::= (<assign-op> <f-head> <f-exp>)
<p-effect> ::= (not <atomic formula(term)>)
<p-effect> ::= <atomic formula(term)>
<p-effect> ::=$^{:fluents}$ (<assign-op> <f-head> <f-exp>)
<cond-effect> ::= (and <p-effect>*)
<cond-effect> ::= <p-effect>
<assign-op> ::= assign
<assign-op> ::= scale-up
<assign-op> ::= scale-down
<assign-op> ::= increase
<assign-op> ::= decrease

Σχήμα 1.13: Συντακτικοί κανόνες για τα αποτελέσματα από την εκτέλεση μιας δράσης σε PDDL.

Durative Actions

Στις Durative Actions μας ενδιαφέρει η διάρκεια της δράσης γι αυτό και επεκτείνεται η σύνταξη των απλών δράσεων ώστε να πληρούνται οι ανάγκες των δράσεων Durative. Ο ορισμός μιας Durative δράσης αποτελείται από δύο μέρη: στο πρώτο μέρος ορίζουμε το όνομα της δράσης και τις παραμέτρους που δέχεται(παρόμοια με τις απλές δράσεις). Στο δεύτερο, ορίζουμε το σώμα της δράσης δηλώνοντας την διάρκεια της δράσης, τα προαπαιτούμενα κατηγορήματα τα οποία θα πρέπει να ισχύουν για να μπορεί να εκτελεστεί η δράση και τα αποτελέσματα μετά την εκτέλεση της δράσης. Η δήλωση των παραμέτρων της δράσης γίνεται με την σύνταξη λίστας τύπων. Οι προαπαιτούμενες συνθήκες μιας durative-action έχουν περισσότερη πολυπλοκότητα από τις απλές δράσεις. Αυτό,οφείλεται στο γεγονός ότι δεν πρέπει μόνο να καθοριστούν οι προαπαιτήσεις για την εκτέλεση μιας durative-δράσης αλλά και οι συνθήκες- conditions που πρέπει να ισχύουν κατά την διάρκεια της και στον τερματισμό της. Κάποιοι από αυτούς τους περιορισμούς μπορούν να καθορίσουν το τι πρέπει να ισχύει και αφού έχει αρχίσει η δράση. Στη συνέχεια, ακολουθούν τα αποτελέσματα μετά την εκτέλεση της durative –δράσης, η σύνταξη των οποίων είναι παρόμοια με τις απλές δράσεις με το επιπλέον ότι λαμβάνουμε υπόψη το χρόνο και την διάρκεια των δράσεων.Στα προβλήματα που μελετήθηκαν και στις μοντελοποιήσεις που έγιναν, δεν έγινε χρήση αυτού του χαρακτηριστικού της γλώσσας,οπότε δεν δίνεται αναλυτική περιγραφή της σύνταξης αυτών των δράσεων.

Problems – Προβλήματα

Το πρόβλημα, είναι αυτό που πρέπει να επιλυθεί από τον προγραμματιστή δράσης και αναφέρεται σε κάποιο κόσμο. Γενικά ένα πρόβλημα αποτελείται από δύο μέρη: την αρχική κατάσταση και τον στόχο που πρέπει να επιτευχθεί. Στο Σχήμα 1.15 παρουσιάζονται οι συντακτικοί κανόνες για τον ορισμό προβλήματος[[5],p5]. Αρχικά δηλώνεται το όνομα του προβλήματος .Στην

συνέχεια, το όνομα του κόσμου (:domain <name>) , στον οποίο αναφέρεται. Μετά δηλώνονται τα requirement flags ([<require-def>]), τα αντικείμενα τα οποία υπάρχουν στον ορισμό του συγκεκριμένου προβλήματος ([<object declaration>]) ή στην αρχική κατάσταση αλλά δεν έχουν δηλωθεί ως σταθερές στον ορισμό του κόσμου στον οποίο αναφέρεται το πρόβλημα, η αρχική κατάσταση (<init>), ο στόχος (<goal>) , οι περιορισμοί ([<constraints>]), metric-spec ([<metric-spec>])και length-spec ([<length-spec>]) μέσον του οποίου δηλώνουμε ότι είναι γνωστό ότι υπάρχει λύση για το συγκεκριμένο πρόβλημα με το συγκεκριμένο length.Αυτή η πληροφορία είναι χρήσιμη σε προγραμματιστές δράσης οι οποίοι αναζητούν λύσεις σε προβλήματα μα βάση το length.

```
<problem> ::= (define (problem <name>)
                        (:domain <name>)
                        [<require-def>]
                        [<object declaration> ]
                        <init>
                        <goal>
                       [<constraints>]
                       [<metric-spec>]
                        [<length-spec> ])
```

Σχήμα 1.15: Συντακτικοί κανόνες για ορισμό προβλήματος σε PDDL.

Το Σχήμα 1.16 παρουσιάζει τους συντακτικούς κανόνες για την δήλωση αντικειμένων και για τον ορισμό της αρχικής κατάστασης[[5],p5]. Τα αντικείμενα δηλώνονται με την σύνταξη λίστας τύπων. Η αρχική κατάσταση αναπαρίσταται με λογικές και συναρτησιακές εκφράσεις. Επίσης στην αρχική κατάσταση μπορούμε να καθορίσουμε κατηγορήματα τα οποία θα γίνουν αληθή σε κάποιο καθορισμένο χρονικό σημείο (durative actions).

```
<object declaration> ::= (:objects <typed list (name)>)
<init> ::= (:init <init-el>*)
<init-el> ::= <literal(name)>
```

<init-el> ::=:fluents (= <f-head> <number>)
<init-el> ::=:timed−initial−literals (at <number> <literal(name)>)

Σχήμα 1.16: Συντακτικοί κανόνες για δήλωση αντικειμένων και για ορισμό αρχικής κατάστασης σε PDDL.

Στο Σχήμα 1.17 παρουσιάζονται οι συντακτικοί κανόνες για τις περιγραφές στόχων των δράσεων[[5],p5]. Παρατηρούμε ότι είναι παρόμοιοι με αυτούς των απλών δράσεων μόνο που εδώ εισάγεται και η έννοια του χρόνου(μέσο των τελεστών always, sometime-before, sometime-after, at-most-once, within, sometime, at-most-once, hold-during, hold-after).Δηλαδή με την χρήση αυτών των τελεστών καθορίζουμε την χρονική περίοδο μέσα στην οποία θα ισχύουν τα αποτελέσματα μιας δράσης. Ο τελεστής hold-during(t1,t2,g) καθορίζει ότι το g πρέπει να είναι αληθές μέσα στο χρονικό διάστημα t1- t2,ενώ ο hold-after(t1, g) ότι το g πρέπει να είναι αληθείς μετά το χρονικό διάστημα t1.

Ακολούθως, δίνεται η σύνταξη του metric-spec το οποίο μπορεί να χρησιμοποιηθεί για να μας δώσει μια έκφραση η οποία πρέπει να βελτιστοποιηθεί στην κατασκευή του προγραμματισμού δράσης. Η βελτιστοποίηση που μπορεί να γίνει είναι είτε μεγιστοποίηση είτε ελαχιστοποιήσει. Βέβαια ο προγραμματιστής δράσης μπορεί να αγνοήσει τις μετρικές και να θεωρήσει ότι προγραμματισμοί δράσης με λιγότερα βήματα είναι καλύτεροι. Η μεταβλητή total-time παίρνει την τιμή της διάρκειας της εκτέλεσης ενός προγραμματισμού δράσης. Με αυτό τον τρόπο, είναι πιο βολικό για εμάς, να εκφράσουμε την πρόθεση μας για ελαχιστοποίηση συνολικού χρόνου εκτέλεσης του προγραμματισμού δράσης.

<goal> ::= (:goal <pre-GD>)
<constraints> ::=:constraints (:constraints <pref-con-GD>)
<pref-con-GD> ::= (and <pref-con-GD>_)
<pref-con-GD> ::=:universal−preconditions
(forall (<typed list (variable)>) <pref-con- GD>)
<pref-con-GD> ::=:preferences (preference [<pref-name>] <con-GD>)

<pref-con-GD> :: <con-GD>
<con-GD> ::= (and <con-GD>*)
<con-GD> ::= (forall (<typed list (variable)>) <con-GD>)
<con-GD> ::= (at end <GD>)
<con-GD> ::= (always <GD>)
<con-GD> ::= (sometime <GD>)
<con-GD> ::= (within <number> <GD>)
<con-GD> ::= (at-most-once <GD>)
<con-GD> ::= (sometime-after <GD> <GD>)
<con-GD> ::= (sometime-before <GD> <GD>)
<con-GD> ::= (always-within <number> <GD> <GD>)
<con-GD> ::= (hold-during <number> <number> <GD>)
<con-GD> ::= (hold-after <number> <GD>)
<metric-spec> ::= (:metric <optimization> <metric-f-exp>)
<optimization> ::= minimize
<optimization> ::= maximize
<metric-f-exp> ::= (<binary-op> <metric-f-exp> <metric-f-exp>)
<metric-f-exp> ::= (<multi-op> <metric-f-exp> <metric-f-exp>$^{+}$)
<metric-f-exp> ::= (- <metric-f-exp>)
<metric-f-exp> ::= <number>
<metric-f-exp> ::= (<function-symbol> <name>*)
<metric-f-exp> ::= <function-symbol>
<metric-f-exp> ::= total-time
<metric-f-exp> ::= (is-violated <pref-name>)

Σχήμα 1.17: Συντακτικοί κανόνες για περιγραφές στόχων των δράσεων σε PDDL.

Απαιτήσεις – Requirements

Πιο κάτω, ακολουθεί (Σχήμα 1.18) μια λίστα με τις διάφορες απαιτήσεις που μπορούν να υπάρξουν κατά την δήλωση απαιτήσεων στον ορισμό του κόσμου[[5] p 6]. Υπενθυμίζουμε ότι, αν δεν δηλωθούν απαιτήσεις τότε θεωρείται ως απαίτηση η γλώσσα STRIPS.

Απαίτηση -Requirement	**Περιγραφή**
:strips	STRIPS υποσύνολο της PDDL
:typing	Επιτρέπονται ονόματα τύπων (type names) στην δήλωση μεταβλητών
:negative-preconditions	Δεν επιτρέπονται αρνητικές προαπαιτούμενες συνθήκες στον ορισμό των περιγραφών στόχων.
:disjunctive-preconditions	Επιτρέπονται σε διαζευτική (disjunctive) μορφή, προαπαιτούμενες συνθήκες ή περιγραφές στόχων (goal descriptions).
:equality	Υποστηρίζεται η ισότητα σαν κατηγόρημα από την γλώσσα predicate
:existential-preconditions	Επιτρέπεται η χρήση του τελεστή exists στην περιγραφή στόχων.
:universal-preconditions	Επιτρέπεται η χρήση του τελεστή forall στην περιγραφή στόχων.
:quantified-preconditions	= :existential-preconditions + :universal-preconditions
:conditional-effects	Επιτρέπεται η χρήση του τελεστή when στα αποτελέσματα δράσης.
:fluents	Επιτρέπεται ο ορισμός συναρτήσεων και η χρήση αποτελεσμάτων στα οποία

	χρησιμοποιούνται τελεστές ανάθεσης και αριθμητικές συνθήκες .
:adl	= :strips + :typing + :negative-preconditions + :disjunctive-preconditions + :equality + :quantified-preconditions + :conditional-effects
:durative-actions	Επιτρέπονται οι durative actions. Σημειώστε ότι αυτό δεν υπονοεί :fluents.
:derived-predicates	Επιτρέπει κατηγορήματα των οποίων η αληθείς τιμή τους καθορίζεται από φόρμουλα.
:timed-initial-literals	Επιτρέπεται στην αρχική κατάσταση να προσδιοριστούν κατηγορήματα τα οποία θα γίνουν αληθή σε κάποια συγκεκριμένη χρονική στιγμή. Υπονοούνται οι durative actions.
:preferences	Επιτρέπει την χρήση προτιμήσεων στις προαπαιτήσεις δράσης και στους στόχους.
:constraints	Επιτρέπει την χρήση περιορισμών στα αρχεία του κόσμου και του προβλήματος.

Σχήμα 1. 18: Λίστα με τις διάφορες απαιτήσεις που μπορούν να υπάρξουν κατά την δήλωση απαιτήσεων στον ορισμό του κόσμου στην PDDL.

Παραδείγματα

Στην μελέτη μας αυτή θα επικεντρωθούμε στο strips υποσύνολο της PDDL και γι αυτό τον λόγο τα διάφορα παραδείγματα αφορούν την strips.

Παράδειγμα 1 Blocks World:

Σε αυτό το παράδειγμα παρουσιάζουμε μια παραλλαγή του γνωστού μας κόσμου των κύβων, υλοποιημένου με την χρήση του υποσυνόλου strips της pddl[9]. Η όλη φιλοσοφία του κόσμου παραμένει ίδια, όπως παρουσιάστηκε νωρίτερα αλλά παρουσιάζονται κάποιες μικρές διαφοροποιήσεις στις δράσεις. Στο Σχήμα 1.19 παρουσιάζεται ο ορισμός του κόσμου των κύβων. Σαν προαπαιτήση του κόσμου είναι η γλώσσα strips και γι‘ αυτό τον λόγο δεν γίνεται η χρήση τύπων. Ο κόσμος έχει τα εξής κατηγορήματα on-table ?x το οποίο αναφέρεται στο ότι το x βρίσκεται απευθείας πάνω στο τραπέζι , το on ?x ?y αναφέρεται στο ότι το x αντικείμενο είναι πάνω από το y και το clear ?x
Ότι πάω από το x δεν υπάρχει αντικείμενο. Τα κατηγορήματα (block ?x) ,(block ?y) ,(block ?z) χρησιμοποιούνται για να δηλώσουμε με έμμεσο τρόπο ότι τα κατηγορήματα «εφαρμόζονται» πάνω σε κύβους.

Η δράση MoveToTable χρησιμοποιείται για την μετακίνηση κάποιου κύβου x ο οποίος βρίσκεται πάνω από ένα κύβο y απ’ ευθείας πάνω στο τραπέζι.

Η MoveToBlock1 δράση χρησιμοποιείται για να μετακινηθεί ένας ελεύθερος κύβος από πάνω σε κάποιο άλλο ελεύθερο κύβο.

Μέσον της ToBlock2 μεταφέρεται ένας κύβος ο οποίος βρίσκεται απευθείας πάνω στο τραπέζι πάνω σε κάποιο άλλο κύβο ο οποίος είναι ελεύθερος.

```
(define (domain blocks_world)
  (:requirements :strips)
  (:predicates (on-table ?x) (on ?x ?y) (clear ?x)(block ?x))

  (:action MoveToTable
    :parameters (?x ?y)
```

```
   :precondition (and (clear ?x) (on ?x ?y)
                              (block ?x) (block ?y))
   :effect (and (clear ?y) (on-table ?x) (not (on ?x ?y))))


 (:action MoveToBlock1
   :parameters (?x ?y ?z)
   :precondition (and (clear ?x) (clear ?z)
                              (on ?x ?y)(block ?x)
                              (block ?y) (block     ?z))
   :effect (and (clear ?y) (on ?x ?z) (not (clear ?z)) (not (on ?x ?y))))


 (:action MoveToBlock2
   :parameters (?x ?y)
   :precondition (and (clear ?x) (clear ?y)
                             (on-table ?x)(block ?x) (block ?y))
   :effect (and (on ?x ?y) (not (clear ?y)) (not (on-table ?x))))
 )
```

Σχήμα 1.19: Ορισμός του κόσμου των κύβων σε STRIPS.

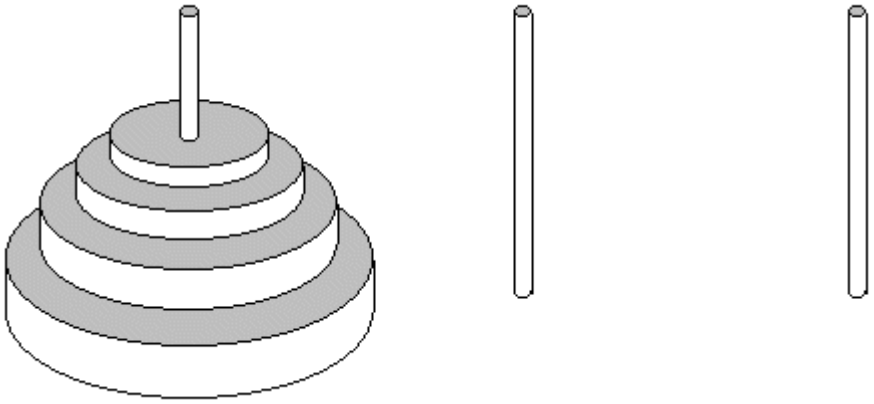
Πιο κάτω στο Σχήμα 1.20, παρουσιάζεται η περιγραφή του προβλήματος το οποίο αναφέρεται στον κόσμο των κύβων[9]. Το πρόβλημα αυτό έχει ως εξής: υπάρχουν οι κύβοι A, B,C,D,E,F,G. Αρχικά ο κύβος A και B είναι ελεύθερος και είναι απευθείας πάνω στο τραπέζι. Ο G βρίσκεται απευθείας πάνω στο τραπέζι**,** ο F , ο E, ο D και ο C είναι πάνω του. Ο C βρίσκεται στην κορυφή και είναι ελεύθερος. Ο στόχος που πρέπει να επιτευχθεί, είναι να δημιουργηθούν στοίβες από κύβους. Ο A πρέπει να είναι απευθείας πάνω στο τραπέζι , και πάνω του ο F.Επίσης ο B πρέπει να είναι πάνω στον C και ο D πάνω στον C.

```
(define (problem bw-world)
  (:domain blocks_world)
  (:objects block_A block_B block_C block_D block_E block_F block_G)

  (:init (on-table block_A) (clear block_A)
         (on-table block_B) (clear block_B)
         (on-table block_G)  (on block_F block_G)
         (on block_E block_F) (on block_D block_E)
         (on block_C block_D) (clear block_C)
         (block block_A)(block block_B)
         (block block_C)(block block_D)
         (block block_E)(block block_F) (block block_G))

  (:goal (and (on block_B block_C) (on-table block_A)
    (on block_F block_A) (on block_C block_D)))
  )
```

Σχήμα 1.20: Ορισμός προβλήματος το οποίο αναφέρεται στον κόσμο των κύβων σε STRIPS.

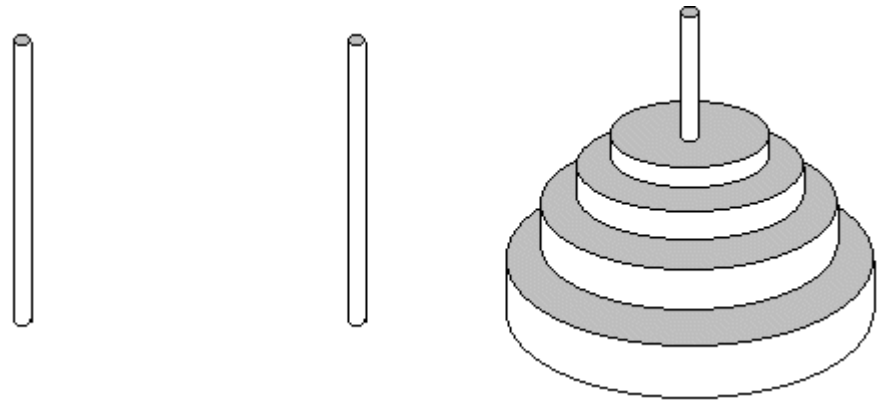
Παράδειγμα 2 Towers of Hanoi:

Ζητείται να μεταφερθούν n δακτύλιοι από τον πρώτο στον τρίτο στύλο, χρησιμοποιώντας το δεύτερο στύλο ως βοηθητικό χώρο. Οι δακτύλιοι είναι τοποθετημένοι κατά σειρά μεγέθους. Η αρχική διάταξη των δακτυλίων φαίνεται στο Σχήμα 1.21.



Σχήμα 1.21: Αρχική διάταξη δακτυλίων (Towers of Hanoi).

Τελική διάταξη των δακτυλίων με την λύση του προβλήματος:



Σχήμα 1.22: Τελική διάταξη των δακτυλίων με την λύση του προβλήματος(Towers of Hanoi).

Κατά τη μεταφορά των δακτυλίων πρέπει να τηρούνται οι παρακάτω κανόνες:

1. Κάθε φορά πρέπει να μετακινείται ένας μόνο δακτύλιος.
2. Δεν μπορεί να τοποθετηθεί μεγαλύτερος δακτύλιος πάνω από μικρότερο.

Στο Σχήμα 1.23 παρουσιάζεται ο ορισμός του κόσμου των πύργων του Hanoi[9]. Η περιγραφή του κόσμου είναι σε strips (τίθεται σαν προαπαίτηση)

οπότε και σε αυτό το παράδειγμα δεν γίνεται η χρήση τύπων αντικειμένων αλλά δηλώνονται έμμεσα μέσων των κατηγορημάτων.

Στον κόσμο υπάρχουν τα εξής κατηγορήματα: clear ?x το οποίο αναφέρεται στο ότι το x είναι ελεύθερο (αν το x είναι δακτύλιος τότε δεν υπάρχει άλλος δακτύλιος πάνω του, αν είναι στύλος τότε δεν έχει δακτυλίους). Το on ?x ?y κατηγόρημα αναφέρεται στο ότι το x είναι πάνω από στο y. Το smaller ?x ?y αναφέρετε στο ότι δεν μπορεί να τοποθετηθεί μεγαλύτερος δακτύλιος πάνω από μικρότερο. Το κατηγόρημα disk ?x είναι αληθές όταν το x είναι δακτύλιος.

Αυτός ο κόσμος διαθέτει μόνο μια δράση την move μέσον της οποίας γίνεται η μετακίνηση των δακτυλίων στους στύλους. Προαπαιτήσεις της δράσης είναι, ο δακτύλιος που θα μετατοπιστεί να βρίσκεται σε κάποιο στύλο, να είναι ελεύθερος, να είναι μικρότερος από τους δακτυλίους του προορισμού και ο στύλος ή ο δακτύλιος προορισμού να είναι ελεύθερος.

```
(define (domain hanoi)
  (:requirements :strips)
  (:predicates (clear ?x) (on ?x ?y) (smaller ?x ?y)(disc ?x)(peg ?x))

  (:action move
    :parameters (?dis ?from ?to)

    :precondition (and (smaller ?to ?dis) (on ?dis ?from)
                       (clear ?dis) (clear ?to) (disc ?dis))

    :effect  (and (clear ?from) (on ?dis ?to) (not (on ?dis ?from))
                  (not (clear ?to))))  )
```

Σχήμα 1.23: Ορισμός κόσμο πύργων του Hanoi σε STRIPS.

Το πρόβλημα που παρουσιάζεται στο Σχήμα 1.24 έχουμε σαν αντικείμενα τρείς δακτυλίους και τρείς στύλους. Στην αρχική κατάσταση ο στύλος δύο και τρία

είναι ελεύθεροι , οι δακτύλιοι d3,d2,d1 βρίσκονται στον στύλο ένα, με τον δακτύλιο ένα να είναι ελεύθερος.

Στόχος είναι να μετακινηθούν οι δακτύλιοι στον στύλο τρία και να βρίσκεται στην βάση ο d3 ακολούθως ο d2 και ο d1.

```
(define (problem hanoi3)
  (:domain hanoi)
  (:objects peg1 peg2 peg3 d1 d2 d3)

  (:init
        (smaller peg1 d1) (smaller peg1 d2) (smaller peg1 d3)
        (smaller peg2 d1) (smaller peg2 d2) (smaller peg2 d3)
        (smaller peg3 d1) (smaller peg3 d2) (smaller peg3 d3)
        (smaller d2 d1) (smaller d3 d1) (smaller d3 d2)
        (clear peg2) (clear peg3) (clear d1)
        (on d3 peg1) (on d2 d3) (on d1 d2) (disc d1) (disc d2)
        (disc d3) (peg peg1) (peg  peg2) (peg peg3))

  (:goal (and (on d3 peg3) (on d2 d3) (on d1 d2)))  )
```

Σχήμα 1.24: Ορισμός κόσμου πύργων του Hanoi σε STRIPS.

Παράδειγμα 3 ταξίδια (travel):

Στο Σχήμα 1.25 παρουσιάζεται ο κόσμος των ταξιδιών[9] στην γλώσσα strips. Σε αυτό τον κόσμο υπάρχουν οχήματα, τοποθεσίες, γέφυρες, δρόμοι στους οποίους ταξιδεύουν τα οχήματα για να πάν από κάποιο προορισμό σε ένα άλλο.

Κατηγορήματα του κόσμου είναι: at ?v ?l το οποίο αναφέρεται σε κάποιο όχημα το οποίο βρίσκεται σε κάποια τοποθεσία. Το road ?l1 ?l2 αναφέρεται σε κάποιο δρόμο που ενώνει τα μέρη l1 και l2.Το bridge ?l1 ?l2 αναφέρεται σε κάποια

γέφυρα που συνδέει δύο τοποθεσίες μέρη l1 και l2. Μέσον των κατηγορημάτων place ?l και vehicle ?v υποδηλώνονται το μέρος l και το όχημα v αντίστοιχα.

Υπάρχουν δύο δράσης η drive και η cross.Μέσον της drive μπορούμε να ταξιδέψουμε από μια τοποθεσία σε κάποια άλλη με την προϋπόθεση ότι υπάρχει κάποιος δρόμος που να ενώνει τις δύο περιοχές και εμείς βρισκόμαστε μέσα στο όχημα μας στην περιοχή αφετηρία. Σαν αποτέλεσμα της δράσης βρισκόμαστε στο όχημα μας στην περιοχή προορισμός και όχι στην αφετηρία.

Η δράση cross είναι παρόμοια με την drive μόνο που τώρα θα πρέπει για να φτάσουμε στην περιοχή προορισμού να διασχίσουμε κάποια γέφυρα η οποία ενώνει τις δύο τοποθεσίες.

```
(define (domain travel)

 (:requirements :strips)

 (:predicates (at ?v ?l)
              (road ?l1 ?l2)
              (bridge ?l1 ?l2)
              (place ?l)
              (vehicle ?v)
             (location ?l))

 (:action drive
   :parameters (?vehic ?loc1 ?loc2)
   :precondition (and (at ?vehic ?loc1)(road ?loc1 ?loc2)
                      (vehicle ?vehic)(location ?loc1)(location ?loc2))
   :effect
   (and (at ?vehic ?loc2) (not (at ?vehic ?loc1))))

 (:action cross
   :parameters (?vehic ?loc1 ?loc2)
   :precondition (and (at ?vehic ?loc1)(bridge ?loc1 ?loc2)
```

```
                    (vehicle ?vehic)(location ?loc1)(location ?loc2))
  :effect
  (and (at ?vehic ?loc2)
       (not (at ?vehic ?loc1))))    )
```

Σχήμα 1.25: Ορισμός κόσμου ταξιδιών στην γλώσσα strips.

Ένα πρόβλημα αυτού του κόσμου είναι, αυτό που παρουσιάζετε στο Σχήμα 1.26.Υπάρχουν πέντε αντικείμενα, τα πρώτα τρία είναι τοποθεσίες a,d, g και τα άλλα δύο οχήματα car και bulldozer. Αρχικά τα δύο οχήματα βρίσκονται στην τοποθεσία a και υπάρχει ο δρόμος που ενώνει τις τοποθεσίες d g και g d. Επίσης υπάρχει γέφυρα που ενώνει a d και d a. Στόχος μας είναι τα δύο οχήματα να ταξιδέψουν και να παν στην τοποθεσία g.

```
(define (problem road-test)
 (:domain travel)
 (:objects a d g car bulldozer)

 (:init
          (vehicle car) (vehicle bulldozer)
          (location a) (location d) (location g)
          (at car a) (at bulldozer a)
          (road d g) (road g d)
          (bridge a d) (bridge d a))

 (:goal (and (at car g) (at bulldozer g)))

 )
```

Σχήμα 1.26: Ορισμός προβλήματος κόσμου ταξιδιών στην γλώσσα strips.

Κεφάλαιο 2

Πρόβλημα Βιοχημικών Μονοπατιών

(The Pathways Domain)

2.1 Περιγραφή προβλήματος

Αυτός ο κόσμος είναι εμπνευσμένος από την μοριακή βιολογία και συγκεκριμένα από τα βιοχημικά μονοπάτια (biochemical pathways).”Ένα βιοχημικό μονοπάτι είναι μια ακολουθία από χημικές αντιδράσεις σε ένα οργανισμό. Με τα βιοχημικά μονοπάτια αποκαλύπτονται οι διάφοροι μηχανισμοί μέσον των οποίων τα κύτταρα εκτελούν κύριες λειτουργίες τους σε μοριακό επίπεδο βιοχημικών αντιδράσεων που παράγουν κανονικές αλλαγές. Πολλές ασθένειες μπορούν να εξηγηθούν σαν ατέλειες, στο βιοχημικό μονοπάτι και η θεραπεία τους περιλαμβάνει την ανακάλυψη φαρμάκων τα οποία να διορθώνουν αυτό το ελάττωμα” (Thagard 2003). Η προσπάθεια μας είναι να μοντελοποιηθούν μέρη της λειτουργικότητας των βιολογικών μονοπατιών μέσον του προγραμματισμού δράσης. Δηλαδή οι διάφορες βιοχημικές αντιδράσεις που εκτελούνται στα βιοχημικά μονοπάτια να αναπαρασταθούν σαν δράσεις και σαν πρόβλημα, να έχουμε κάποιο περιορισμένο αριθμό βιοχημικών ουσιών για να παράξουμε κάποιες άλλες βιοχημικές ουσίες(στόχος), μέσω της εκτέλεσης μιας σειράς αντιδράσεων. [[1] p 5] Ο προγραμματιστής δράσης είναι μερικός ελεύθερος να επιλέξει ποιες από τις ουσίες που υπάρχουν αρχικά θα χρησιμοποιήσει (υπάρχει δράση μέσον της οποίας θα γίνει η επιλογή).

Το παράδειγμα που θα εξεταστεί πάρθηκε από τον 5^{ο} Διεθνή Διαγωνισμό Προγραμματισμού Δράσης. Οι κόσμοι βασίζονται στο βιοχημικό μονοπάτι του κύκλου ζωής κυττάρων θηλαστικών (Mammalian Cell Cycle Control) όπως περιγράφεται από τον (Kohn 1999) και μοντελοποιείτε από τον (Chabrier 2003).

Προτού μελετήσουμε αναλυτικά τα παραδείγματα αξίζει να σημειωθεί ότι για την περιγραφή των βιοχημικών αντιδράσεων χρησιμοποιείται η γλώσσα ADL (Action Description Language) η οποία είναι υποσύνολο της PDDL. Σαν γλώσσα είναι περισσότερο εκφραστική από την STRIPS αφού περιέχει στοιχεία της γλώσσας STRIPS αλλά με κάποιες προσθήκες όπως το να υπάρχουν ψευδή κατηγορήματα στις προαπαιτήσεις δράσεων (negative - preconditions), χρήση διαζευκτικής κανονικής μορφής στις προαπαιτήσεις (disjunctive - preconditions), χρήση του τελεστή ισότητας, χρήση του υπαρξιακού τελεστή στις προαπαιτήσεις δράσης(quantified-preconditions) και χρήση συνθηκών στα αποτελέσματα δράσεων (conditional-effects). Επίσης γίνεται χρήση τύπων μεταβλητών.

Αφού μελετήθηκαν παραδείγματα που ήταν αρχικά γραμμένα στην adl έγινε μετατροπή τους στην γλώσσα STRIPS και στην συνέχεια παράχθηκαν προγραμματισμοί δράσεις μέσο του προγραμματιστή δράσης SatPlan. Στον προγραμματιστή δράσης δόθηκαν σαν είσοδος ένα αρχείο με την περιγραφή του κόσμου και ένα αρχείο με τη περιγραφή του προβλήματος.

Στο παράδειγμα που παρουσιάζεται ενδεικτικά, έχοντας ως πρώτη ύλη κάποιες μοριακές – χημικές ουσίες προσπαθούμε να παράξουμε κάποιες άλλες(στόχος), μέσον των βιοχημικών αντιδράσεων (δράσεις) που υπάρχουν στους κόσμους. Ο προγραμματιστής δράσης καλείται να βρει μια ακολουθία με την οποία θα πρέπει να εκτελεστούν οι διάφορες βιοχημικές αντιδράσεις ούτως ώστε να παραχθούν οι ουσίες- στόχοι. Η ακολουθία αυτή των βιοχημικών αντιδράσεων αποτελεί το βιοχημικό μονοπάτι.

2.1 Παράδειγμα: Βιοχημικά Μονοπάτια - Pathways Propositional

Στο Σχήμα 2.1 παρουσιάζεται ο κώδικας για τον ορισμό του κόσμου Pathways-Propositional στην γλώσσα adl.

Οι δεσμευμένες λέξεις define domain υποδηλώνουν ότι ακολουθεί ο ορισμός του κόσμου. Ακολουθώντας τους κανόνες σύνταξης που αναφέρθηκαν πιο πάνω στην συνέχεια δηλώνουμε σαν requirement flags τη γλώσσα adl και την χρήση τύπων στην δήλωση μεταβλητών.

Συνεχίζουμε με τον ορισμό των τύπων του κόσμου. Υπάρχουν οντότητες με τύπους level, molecule, simple και complex. Οι level, molecule τύποι προέρχονται από την κλάση object (η οποία είναι η προκαθορισμένη κλάση τύπων), ενώ οι simple και complex από την molecule. Ουσιαστικά, σε αυτό τον κόσμο οι τύποι των οντοτήτων - χημικών ουσιών που μπορούμε να έχουμε, είναι μόρια (molecule) ή επίπεδα συγκέντρωσης ουσιών (level). Τα μόρια διαχωρίζονται σε δύο τύπους τα απλά(simple) και τα πολύπλοκα(complex).

Στην συνέχεια δηλώνουμε τις σταθερές, οι οποίες στην περίπτωση μας είναι οι c-Myc-Max,cycDp1,p107-E2F4-DP12p1 και p107-E2F4-DP12p1-gE2 οι οποίες είναι πολύπλοκα μόρια. Έπειτα γίνεται η δήλωση των κατηγορημάτων (predicates) που υπάρχουν στον κόσμο. Τα κατηγορήματα αφορούν διάφορους τύπους βιοχημικών αντιδράσεων και την κατάσταση στην οποία βρίσκονται διάφορες ουσίες. Το κατηγόρημα association-reaction αναφέρεται στην αντίδραση με την οποία δύο μόρια συσχετίζονται για να παραχθεί ένα πιο πολύπλοκο μόριο (τότε το κατηγόρημα θα είναι αληθές). Το κατηγόρημα catalyzed-association-reaction αναφέρεται στην αντίδραση με την οποία δύο μόρια συσχετίζονται με την βοήθεια καταλύτη, για να παραχθεί ένα πιο πολύπλοκο μόριο (τότε το κατηγόρημα θα είναι αληθές). Το synthesis-reaction κατηγόρημα αναφέρεται στην αντίδραση σύνθεσης. Το κατηγόρημα possible αναφέρεται στο αν ένα μόριο είναι πιθανόν να επιλεχθεί για να χρησιμοποιηθεί σε κάποια από τις αντιδράσεις. Το κατηγόρημα available αναφέρεται στο αν

κάποιο μόριο είναι διαθέσιμο να χρησιμοποιηθεί σε κάποια αντίδραση. Το chosen κατηγόρημα αναφέρεται στο αν έχει επιλεχθεί κάποιο απλό μόριο για να συμμετέχει σε κάποια αντίδραση. Next αναφέρεται στο επόμενο επίπεδο συγκέντρωσης που μπορεί να φτάσει μια ουσία ,num-subs αναφέρεται στο επίπεδο συγκέντρωσης κάποιας ουσίας, τα goal1και goal2 αναφέρονται στους στόχους που τέθηκαν.

Σημείωση: Το «?» μπροστά από <name> συμβολίζει μεταβλητή και η σύνταξη των σχολίων είναι ;<σχόλια>.

Ακολουθούν οι ορισμοί των δράσεων του κόσμου. Στον υπό μελέτη κόσμο υπάρχουν έξη δράσεις: choose, initialize, associate, associate-with-catalyze, synthesize DUMMY-ACTION-1και DUMMY-ACTION-2. Σκοπός της δράσης choose είναι να επιλεχθούν οι αρχικές ουσίες. Για να μπορέσει να επιλεχθεί μια ουσία η συγκέντρωση της πρέπει να βρίσκεται σε κάποιο επίπεδο. Παράμετροι αυτής της δράσης είναι ένα απλό μόριο x και δύο μεταβλητές τύπου level, που καθορίζουν τα επίπεδο ουσιών l1 ,l2.Προαπαιτήσεις της, είναι το απλό μόριο x να μην έχει ήδη επιλεχθεί αλλά να είναι πιθανόν να επιλεγχθεί. Επίσης το επίπεδο συγκέντρωσης του να είναι l2 και το επόμενο επίπεδο από το l2 είναι το l1.Σαν αποτέλεσμα της δράσης επιλέγεται το x του οποίου η συγκέντρωση του δεν είναι πλέον l2 αλλά l1.

Σκοπός της δράσης initialize είναι η αρχικοποίηση ουσιών (δηλαδή μέσον αυτής της δράσης αυξάνεται η ποσότητα μιας συγκεκριμένης ουσίας, η οποία έχει επιλεχθεί για να συμμετέχει στις βιοχημικές αντιδράσεις, ώστε να είναι διαθέσιμη). Παράμετροι της είναι ένα απλό μόριο x. Προαπαίτηση της είναι να έχει επιλεχθεί το x και αποτέλεσμα της είναι η ουσία x να είναι διαθέσιμη για χρησιμοποίηση.

Σκοπός της associate δράσης είναι ο συσχετισμός δύο μορίων x1 και x2 και ενός πολύπλοκού μορίου x3 (παράμετροι). Προαπαιτήσεις της είναι να είναι διαθέσιμες οι ουσίες x1 και x2 και να έχουν συσχετιστεί οι x1, x2 με την x3

μέσον κάποιας αντίδρασης (παράγεται η x 3 από την αντίδραση). Αποτέλεσμα της είναι οι x1 και x2 να μην είναι πλέον διαθέσιμες και να είναι διαθέσιμη η x3.

Η associate-with-catalyze δρα παρόμοια με την associate μόνο που εδώ η αντίδραση χρειάζεται την βοήθεια καταλύτη. Παρόμοια με την associate δέχεται σαν παραμέτρους δύο μόρια x1 και x2 και ένα πολύπλοκο μόριο x3.Προαπαιτήσεις της είναι τα x1 και x2 μόρια να είναι διαθέσιμα και να έχουν συσχετιστεί τα x1 και x2 με το x3 μέσον μιας αντίδρασης συσχέτισης με καταλύτη. Σαν αποτέλεσμα της δράσης η x1 δεν είναι διαθέσιμη ενώ η x3 είναι.

Σκοπός της synthesize δράσης είναι δεδομένου δύο μορίων x1 , x2 (παράμετροι), από το ένα να παράξουμε το άλλο μέσον της σύνθεσης. Προαπαιτήσεις της, είναι να είναι διαθέσιμο το x1 και έχει γίνει η σύνθεση της x2 από την x1.Αποτελέσματα της δράσης είναι, η x2 να είναι διαθέσιμη.

Σκοπός της DUMMY-ACTION-1 δράσης είναι η επίτευξη του στόχου που αντιπροσωπεύεται από το κατηγόρημα goal1. Προαπαιτήσεις της είναι είτε η ουσία p107-E2F4-DP12p1-gE2 είτε η p107-E2F4-DP12p1 είτε και οι δύο να είναι διαθέσιμες. Αποτέλεσμα της είναι, να επιτευχθεί ο goal1.

Σκοπός της DUMMY-ACTION-2 δράσης είναι η επίτευξη του στόχου που αντιπροσωπεύεται από το κατηγόρημα goal2. Προαπαιτήσεις της είναι είτε η ουσία cycDp1 είτε η c-Myc-Max είτε και οι δύο να είναι διαθέσιμες. Αποτέλεσμα της είναι, να επιτευχθεί ο goal2.

```
(define (domain Pathways-Propositional)
(:requirements :typing :adl)

(:types level molecule - object
  simple complex -molecule)

(:constants c-Myc-Max cycDp1 p107-E2F4-DP12p1 p107-E2F4-DP12p1-gE2 -
complex)
```

```
(:predicates
        (association-reaction ?x1 ?x2 -molecule ?x3 -complex)
        (catalyzed-association-reaction ?x1 ?x2 -molecule ?x3 -complex)
        (synthesis-reaction ?x1 ?x2 - molecule)
        (possible ?x - molecule)
        (available ?x - molecule)
        (chosen ?s - simple)
        (next ?l1 ?l2 - level)
        (num-subs ?l1 -level)
        (goal1)
        (goal2))

  (:action choose
    :parameters (?x - simple ?l1 ?l2 - level)
    :precondition (and (possible ?x) (not (chosen ?x))
                       (num-subs ?l2) (next ?l1 ?l2))
    :effect (and (chosen ?x) (not (num-subs ?l2)) (num-subs ?l1)))

(:action initialize
   :parameters (?x - simple)
   :precondition (and (chosen ?x))
   :effect (and (available ?x)))

     (:action associate
       :parameters  (?x1 ?x2 - molecule ?x3 - complex)
      :precondition (and (association-reaction ?x1 ?x2 ?x3)
                         (available ?x1) (available ?x2))
      :effect (and (not (available ?x1)) (not (available ?x2)) (available ?x3)))


   (:action associate-with-catalyze
    :parameters  (?x1 ?x2 - molecule ?x3 - complex)
    :precondition (and (catalyzed-association-reaction ?x1 ?x2 ?x3)
```

```
                    (available ?x1) (available ?x2))
    :effect (and (not (available ?x1)) (available ?x3)))


 (:action synthesize
  :parameters (?x1 ?x2 - molecule)
  :precondition (and (synthesis-reaction ?x1 ?x2) (available ?x1))
  :effect (and (available ?x2)))


(:action DUMMY-ACTION-1
 :parameters ()
 :precondition (or (available p107-E2F4-DP12p1-gE2)
                   (available p107-E2F4-DP12p1))
 :effect (and (goal1)))
 (:action DUMMY-ACTION-2
 :parameters ()
 :precondition (or (available cycDp1)
                   (available c-Myc-Max))
 :effect(and (goal2)))
)
```

Σχήμα 2.1:Ορισμός κόσμου Pathways-Propositional στην γλώσσα adl.

Αφού μελετήθηκε ο ορισμός του κόσμου (Σχήμα 2.1) μετατράπηκε σε strips. Στο Σχήμα 2.2 παρουσιάζεται η strips εκδοχή του ορισμού του κόσμου Pathways Propositional. Σαν requirement flag του κόσμου είναι η strips. Λόγω του ότι η γλώσσα δεν υποστηρίζει τύπους, αυτοί έχουν αφαιρεθεί και έχουν υπάρχουν με έμμεσο τρόπο αφού έχουν προστεθεί σαν κατηγορήματα. Τα υπόλοιπα κατηγορήματα του κόσμου παραμένουν οι ίδια με την adl εκδοχή.

Η ουσιαστική διαφοροποίηση των δύο εκδοχών του κόσμου βρίσκεται στον ορισμό των δράσεων. Γενικά σε όλες τις δράσεις που δέχονταν παραμέτρους,

έχουν προστεθεί στις προαπαιτήσεις τους τα αντίστοιχα κατηγορήματα με τους τύπους των παραμέτρων που δέχονταν, ώστε να διασφαλιστεί ότι οι παραμέτροι που δέχονται είναι οι δράσεις είναι του ιδίου τύπου όπως και στην adl εκδοχή.

Ο σκοπός και τα αποτελέσματα της δράσης choose παραμένουν ίδια με πριν. Η σύνταξη των προαπαιτήσεων παραβιάζει τους κανόνες της γλώσσας strips λόγω του ότι γίνεται χρήση του τελεστή άρνησης στο κατηγόρημα chosen. Γι‘ αυτό τον λόγο στην strips εκδοχή του κόσμου εκμεταλλευόμαστε το closed world assumption (κατηγορήματα τα οποία υπάρχουν στον κόσμο (domain), αλλά δεν γίνεται αναφορά σε αυτά στην αναπαράσταση κάποιας κατάστασης, θεωρούνται ότι είναι ψευδή για την συγκεκριμένη κατάσταση) και παραλείπουμε το κατηγόρημα chosen από τις προαπαιτήσεις της δράσης (αφού είναι ψευδής). Στις παραμέτρους που δέχεται η δράση δεν αναγράφεται ο τύπος της κάθε παραμέτρου λόγω αυτό δεν υποστηρίζεται από την γλώσσα. Ακόμα, έχουν προστεθεί τα αντίστοιχα κατηγορήματα των τύπων των παραμέτρων, στις προαπαιτήσεις της δράσης.

Οι initialize, associate, associate-with-catalyze και synthesize δράσεις παραμένουν ουσιαστικά ίδιες ως προς τον σκοπό και τα αποτελέσματα με πριν. Στις παραμέτρους που δέχονται οι δράσεις δεν αναγράφεται ο τύπος της κάθε παραμέτρου λόγω του ότι, αυτό δεν υποστηρίζεται από την γλώσσα. Επίσης, έχουν προστεθεί τα αντίστοιχα κατηγορήματα των τύπων των παραμέτρων, στις προαπαιτήσεις των δράσεων.

Όσον αφορά την δράση DUMMY-ACTION-1 οι παραμέτροι ,ο σκοπός και τα αποτελέσματα παραμένουν ίδια με πριν. Ο τελεστής or ο οποίος υπάρχει στις προαπαιτήσεις της δράσης παραβιάζει τους κανόνες σύνταξης της strips.Γι’ αυτό, εκμεταλλευόμαστε την λειτουργία του τελεστή or(A,B),δηλαδή για έχει αληθή τιμή ο τελεστής θα πρέπει είτε το A να είναι αληθείς είτε το B να είναι αληθείς, είτε και το A και το B να είναι αληθείς και γι’ αυτό σπάζουμε την δράση σε τρείς (DUMMY-ACTION-1a, DUMMY-ACTION-1b ,DUMMY-ACTION-1c)

Διατηρώντας τα λοιπά μέρη της δράσης ίδια, στην DUMMY-ACTION-1a έχουμε την σαν προαπαίτηση τις p107-E2F4-DP12p1-gE2 και p107-E2F4-DP12p1 ουσίες να είναι διαθέσιμες, στην DUMMY-ACTION-1b την p107-E2F4-DP12p1-gE2 και στην DUMMY-ACTION-1c η p107-E2F4-DP12p1.

Για την DUMMY-ACTION-2 ,εφαρμόσαμε τον ίδιο τρόπο σκέψης με την DUMMY-ACTION-1 και την σπάσαμε σε τρείς δράσεις DUMMY-ACTION-2a, DUMMY-ACTION-2b, DUMMY-ACTION-2c.

Pathways Propositional - Ορισμός κόσμου στο strips υποσύνολο της pddl

```
(define (domain Pathways-Propositional-2)
        (:requirements :strips)
        (:predicates
                (level ?l)
                (molecule ?m)
                 (simple ?s)
                    (complex ?c)
                    (association-reaction ?x1_m ?x2_m ?x3_c)
(catalyzed-association-reaction ?x1_m ?x2_m ?x3_c)
(synthesis-reaction ?x1_m ?x2_m)
                     (possible ?x_m)
                     (available ?x_m)
                     (chosen ?s_s)
                     (next ?l1_l ?l2_l)
                      (num-subs ?l1_l)
                     (goal1)
                     (goal2))

(:action choose
 :parameters (?x_s ?l1_l ?l2_l)
:precondition (and (possible ?x_s)(simple ?x_s)(level ?l1_l)
                    (level ?l2_l)(num-subs ?l2_l)
```

```
                    (num-subs ?l2_l) (next ?l1_l ?l2_l))
:effect (and (chosen ?x_s) (not (num-subs ?l2_l)) (num-subs ?l1_l)))


(:action initialize
:parameters (?x_s)
:precondition (and (chosen ?x_s)(simple ?x_s))
 :effect (and (available ?x_s)))
(:action associate
:parameters  (?x1_m ?x2_m ?x3_c)
:precondition (and (association-reaction ?x1_m ?x2_m ?x3_c)
                    (molecule ?x1_m)(molecule ?x2_m)(complex ?x3_c)
                    (available ?x1_m) (available ?x2_m))
:effect (and (not (available ?x1_m)) (not (available ?x2_m)) (available ?x3_c)))


(:action associate-with-catalyze
:parameters  (?x1_m ?x2_m ?x3_c)
:precondition (and (catalyzed-association-reaction ?x1_m ?x2_m ?x3_c)
                    (molecule ?x1_m)(molecule ?x2_m)(complex ?x3_c)
                    (available ?x1_m) (available ?x2_m))
 :effect (and (not (available ?x1_m)) (available ?x3_c)))


 (:action synthesize
 :parameters (?x1_m ?x2_m)
 :precondition (and (molecule ?x1_m)(molecule ?x2_m)
                     (synthesis-reaction ?x1_m ?x2_m)(available ?x1_m))
:effect (and (available ?x2_m)))
```

```
(:action DUMMY-ACTION-1a
 :parameters ()
 :precondition (and(available p107-E2F4-DP12p1-gE2)
                   (available p107-E2F4-DP12p1))
 :effect (and (goal1)))


(:action DUMMY-ACTION-1b
 :parameters ()
 :precondition (available p107-E2F4-DP12p1-gE2)
 :effect (and (goal1)))


 (:action DUMMY-ACTION-1c
 :parameters ()
 :precondition (available p107-E2F4-DP12p1)
 :effect (and (goal1)))

(:action DUMMY-ACTION-2a
 :parameters ()
 :precondition (and (available cycDp1)
                    (available c-Myc-Max))
 :effect(and (goal2)))


(:action DUMMY-ACTION-2b
 :parameters ()
 :precondition (available cycDp1)
 :effect(and (goal2)))
```

```
(:action DUMMY-ACTION-2c
 :parameters ()
 :precondition (available c-Myc-Max)
 :effect(and (goal2)))    )
```

Σχήμα 2.2: STRIPS εκδοχή ορισμού του κόσμου Pathways Propositional.

Στο Σχήμα 2.3 παρουσιάζεται ο ορισμός του προβλήματος που αναφέρεται στον Pathways Propositional κόσμο. Αρχικά, καθορίζουμε σε ποιό κόσμο αναφέρεται το πρόβλημα που θα ορίσουμε. Στην συνέχεια δηλώνουμε επιπλέον αντικείμενα – ουσίες που υπάρχουν στον κόσμο και δεν είχαν δηλωθεί στον ορισμό του. Μετά ορίζουμε την αρχική κατάσταση (τα κατηγορήματα που ισχύουν σε αυτή) και τον στόχο που πρέπει να εκπληρωθεί. Στο Σχήμα παρουσιάζονται αποσπάσματα του κώδικα,ολόκληρος ο κώδικας βρίσκεται στο Παράρτημα Α.

Pathways Propositional Problem - Ορισμός προβλήματος Pathways σε pddl

```
(define (problem Pathways)
(:domain IPC5b)
(:objects
      p53p1 - simple
      p130 - simple
      Max - simple
      HDAC1-pRbp1-E2F4-DP12 - simple
      HDAC1-pRbp1-E2F13-DP12 - simple
        …….
        E2F6-DP12p1 - simple
        E2F4-DP12p1 - simple
        E2F13p1-DP12 - simple
        cdk1p1p2 - simple
        cdk1p1p2-Gadd45 - complex
```

c-Myc-Max - complex
E2F13p1-DP12-gE2 - complex
E2F6-DP12p1-gE2 - complex
HBP1-p130 - complex
…..
10 -level
11 -level
12 -level
13 -level)

(:init
(possible p53p1)
(possible p130)
(possible Max)
(possible HDAC1-pRbp1-E2F4-DP12)
….
(association-reaction cdk1p1p2 Gadd45 cdk1p1p2-Gadd45)
(association-reaction c-Myc Max c-Myc-Max)
…
(synthesis-reaction E2F13p1-DP12-gE2 cycD)
(synthesis-reaction E2F13p1-DP12-gE2 cycDp1)
. . .
(association-reaction E2F13p1-DP12 gE2 E2F13p1-DP12-gE2)
(synthesis-reaction E2F13p1-DP12-gE2 p107)
(synthesis-reaction E2F13p1-DP12-gE2 p107p1)
. . .
(association-reaction E2F6-DP12p1 gE2 E2F6-DP12p1-gE2)
(association-reaction HBP1 p130 HBP1-p130)
. . .
(synthesis-reaction p53p1 c-Fos)
(synthesis-reaction p53p1 Gadd45)
(synthesis-reaction p53p1 Mdm2)
(synthesis-reaction p53p1 p21)

```
  (num-subs 10)
   (next 11 10)
   (next 12 11)
   (next 13 12))

(:goal
    (and
    (goal1)
    (goal2)))
)
```

Σχήμα 2.3: Ορισμός προβλήματος που αναφέρεται στον Pathways Propositional κόσμο.

Λαμβάνοντας υπόψη μας, τους περιορισμούς της γλώσσας strips έχουμε κάνει ορισμένες μετατροπές στην σύνταξη του προβλήματος ώστε να είναι σύμφωνο με την γλώσσα (Σχήμα 2.4). Το νόημα του προβλήματος παραμένει ίδιο με πριν. Τα αντικείμενα του προβλήματος παραμένουν ίδια αλλά εδώ δεν δηλώνονται οι τύποι τους. Η δήλωση των τύπων των αντικειμένων γίνεται με έμμεσο τρόπο μέσον των κατηγορημάτων που έχουν προστεθεί στην αρχική κατάσταση. Παρατηρήστε ότι αντικείμενα που είχαν τύπο simple ή complex, αναφέρονται σε αυτά τα κατηγόρημα simple και complex αντίστοιχα αλλά και κατηγορήματα molecule.Έτσι όσες ουσίες υπάρχουν στο πρόβλημα αυτό και είναι απλές ή σύνθετες, είναι επίσης και μόρια.

Pathways Propositional Problem Ορισμός προβλήματος στο strips υποσύνολο της pddl

```
 (define (problem Pathways-Propositional-pr2)
(:domain Pathways-Propositional-2)
(:objects
      p107-E2F4-DP12p1-gE2
      p53p1
      p130
      Max
```

HDAC1-pRbp1-E2F4-DP12
HDAC1-pRbp1-E2F13-DP12
HDAC1-p130-E2F4p1-DP12
…..
cdk1p1p2
cdk1p1p2-Gadd45
c-Myc-Max
……
HDAC1-pRbp1-E2F13-DP12-gE2
HDAC1-pRbp1-E2F4-DP12-gE2
Mdm2-E2F13p1-DP12
…..
p19ARF
po1
p107p1
p107
10
11
12
13)

(:init
(molecule p53p1)
(molecule p130)
(molecule Max)
…
(simple p53p1)
(simple p130)
(simple Max)
….
(simple gE2)
….
(complex cdk1p1p2-Gadd45)

(complex c-Myc-Max)
(complex E2F13p1-DP12-gE2)
(complex E2F6-DP12p1-gE2)
(complex HBP1-p130)
….
(level 10)
(level 11)
(level 12)
(level 13)
(possible p53p1)
(possible p130)
(possible Max)
(possible HDAC1-pRbp1-E2F4-DP12)
….
(association-reaction cdk1p1p2 Gadd45 cdk1p1p2-Gadd45)
(association-reaction c-Myc Max c-Myc-Max)
(synthesis-reaction E2F13p1-DP12-gE2 c-Myc)
(synthesis-reaction E2F13p1-DP12-gE2 cycA)
(synthesis-reaction E2F13p1-DP12-gE2 cycD)
….
(association-reaction Mdm2 E2F13p1-DP12 Mdm2-E2F13p1-DP12)
(association-reaction p107-E2F4-DP12p1 gE2 p107-E2F4-DP12p1-gE2)
….
(synthesis-reaction p53p1 c-Fos)
(synthesis-reaction p53p1 Gadd45)
(synthesis-reaction p53p1 Mdm2)
(synthesis-reaction p53p1 p21)
(num-subs 10)
(next 11 10)
(next 12 11)
(next 13 12))

```
(:goal
    (and
    (goal1)
    (goal2)))
)
```

Σχήμα 2.4: Ορισμός προβλήματος που αναφέρεται στον Pathways Propositional κόσμο σε STRIPS.

Στο Σχήμα 2.5 που ακολουθεί, παρουσιάζεται η λύση που δίνει ο προγραμματιστής δράσης Satplan. Το βιοχημικό μονοπάτι που δημιουργείται αποτελείται από επτά βήματα.

Αρχικά επιλέγονται οι ουσίες E2F13P1-DP12 και GE2 οι οποίες είναι απλά μόρια και με την εκτέλεση της δράσης αυξάνεται η συγκέντρωση τους από 10 σε 11 και από 11 σε 12 αντίστοιχα. Στην συνέχεια, μέσον της INITIALIZE δράσης γίνεται η αρχικοποίηση τους και καθίστανται διαθέσιμες για συμμετοχή στις αντιδράσεις.

Έπειτα, επιλέγεται και μια τρίτη ουσία, η E2F4-DP12P1 με επίπεδο συγκέντρωσης 12, (μέσο της CHOOSE δράσης) το οποίο αυξάνεται σε 13. Μέσο της ASSOCIATE, συσχετίζονται οι E2F13P1-DP12 και GE2 ουσίες με αποτέλεσμα την παραγωγή της E2F13P1-DP12-GE2, η οποία πλέον είναι διαθέσιμη να συμμετέχει σε αντιδράσεις. Από την ουσία E2F13P1-DP12-GE2 που παράχθηκε, συνθέτονται δύο άλλες ουσίες (SYNTHESIZE δράση), οι CYCDP και P107 οι οποίες είναι πλέον διαθέσιμες. Αφού αρχικοποιηθεί (INITIALIZE),η E2F4-DP12P1 ουσία και γίνει διαθέσιμη συμμετέχει σε αντίδραση συσχετισμού(ASSOCIATE) με την P107 ουσία και ως αποτέλεσμα αυτού παράγεται η P107-E2F4-DP12P1 η οποία είναι διαθέσιμη. Έτσι, μπορούν να εκτελεστούν οι DUMMY-ACTION-1C και DUMMY-ACTION-2B δράσεις (αφού ικανοποιήθηκαν οι προαπαιτήσεις τους) και εκπληρωθούν και τα δύο μέρη του στόχου (goal1,goal2).

Λύση από Satplan Planner για παράδειγμα

; Time 0.29

; ParsingTime 0.00

; MakeSpan 7

0: (CHOOSE E2F13P1-DP12 11 10) [1]

1: (CHOOSE GE2 12 11) [1]

1: (INITIALIZE E2F13P1-DP12) [1]

2: (INITIALIZE GE2) [1]

3: (CHOOSE E2F4-DP12P1 13 12) [1]

3: (ASSOCIATE E2F13P1-DP12 GE2 E2F13P1-DP12-GE2) [1]

4: (SYNTHESIZE E2F13P1-DP12-GE2 CYCDP1) [1]

4: (SYNTHESIZE E2F13P1-DP12-GE2 P107) [1]

4: (INITIALIZE E2F4-DP12P1) [1]

5: (ASSOCIATE P107 E2F4-DP12P1 P107-E2F4-DP12P1) [1]

6: (DUMMY-ACTION-1C) [1]

6: (DUMMY-ACTION-2B) [1]

Σχήμα 2.5: Λύση για πρόβλημα Pathways-02 από satplan.

Κεφάλαιο 3

Αναδιοργάνωση γονιδιώματος και προγραμματισμός δράσης (Genome Rearrangement and Planning)

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο, θα μελετήσουμε το πρόβλημα της αναδιοργάνωσης γονιδιώματος από πλευράς προγραμματισμού δράσης. Αφού αναλυθούν κάποιοι σημαντικοί όροι για την κατανόηση του προβλήματος, θα παρουσιαστεί η κωδικοποίηση δράσεων του προβλήματος στην γλώσσα του TLPLAN προγραμματιστή δράσης(προτασιακή λογική γραμμένη με σύνταξη της γλώσσας lisp).Οι δράσεις αυτές είναι γραμμένες στο ύφος της adl (υποσύνολο της pddl) και παρουσιάζονται στο άρθρο Genome Rearrangement and Planning των Ersa Erdem kai Elisabeth Tiller[2].Στην συνέχεια, θα παρουσιαστούν οι δράσεις με ακριβέστερη κωδικοποίηση στην adl.

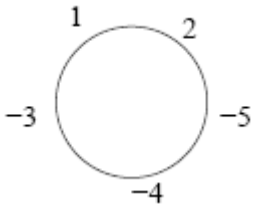
3.2 Αναδιοργάνωση γονιδιώματος

Το γονιδίωμα του κάθε οργανισμού, περιέχει το σύνολο των γενετικών πληροφοριών του. Αποτελείται από γονίδια. Με τον όρο αναδιοργάνωση γονιδιώματος, αναφερόμαστε σε κάποια «γεγονότα», μέσον των οποίων αλλάζει η σειρά των γονιδίων στο γονιδίωμα. Μέσον των γεγονότων αναδιοργάνωσης, μπορούμε να οδηγηθούμε από ένα γονιδίωμα ενός οργανισμού, στο γονιδίωμα κάποιου άλλου οργανισμού. Όσο πιο κοντά γενετικά βρίσκονται (φυλογενετικά δέντρα) δύο οργανισμοί, τόσο λιγότερα γεγονότα

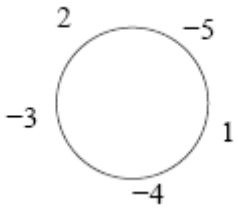
αναδιοργάνωσης γονιδιώματος, χρειάζονται για να μετατραπεί το γονιδίωμα του ενός οργανισμού στου άλλου.

Το πρόβλημα της γονιδιακής αναδιοργάνωσης αναφέρεται στην εύρεση του ελαχίστου αριθμού γεγονότων που απαιτούνται για την μετατροπή γονιδιώματος ενός οργανισμού σε κάποιου άλλου.
Από πλευράς προγραμματισμού δράσης, το πρόβλημα μεταφέρεται ως εξής: Δεδομένου δύο γονιδιώματα και ενός θετικού ακεραίου αριθμού k, πρέπει να βρεθεί μια ακολουθία από το πολύ k γεγονότα τα οποία μετατρέπουν το ένα γονιδίωμα στο άλλο. Εμείς θα ασχοληθούμε με απλά την επίλυση του προβλήματος χωρίς την εύρεση της πιο σύντομης λύσης.

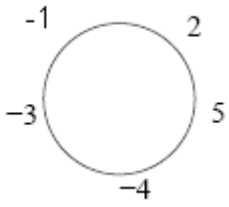
Στον προγραμματισμό δράσης, αναπαριστούμε το γονιδίωμα οργανισμών με μονά χρωματοσώματα με κυκλικούς σχηματισμούς, οι οποίοι έχουν αριθμούς 1, . . . ,n με πρόσημα – ή +. Οι αριθμοί ±1, . . . ,±n ονομάζονται ετικέτες (labels). Ο αριθμός αναπαριστά γονίδιο και το πρόσημο τον προσανατολισμό του γονιδίου.



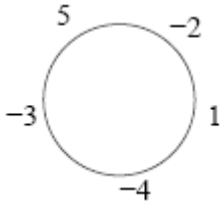
Σχήμα 3.1α



Σχήμα 3.1β



Σχήμα 3.1δ



Σχήμα 3.1γ

Σχήμα 3.1:Παραδείγματα γονιδιωμάτων

Οι κυκλικοί σχηματισμοί αναπαρίστανται ως (l1, . . . , ln), όπου l1, . . . , ln είναι οι ετικέτες, οι οποίες απαριθμούνται δεξιόστροφα. Π.χ τα (1, 2,−5,−4,−3), (2,−5,−4,−3, 1) αναπαριστούν το γονιδίωμα του Σχήματος 3.1α.
Η αναδιοργάνωση γονιδιώματος γίνεται μέσον γεγονότων αντιστροφής (inversions),μετατόπισης (transpositions), και μετατοπισμένης αντιστροφής (transversions).

Μετατόπιση (transposition):Δεδομένου δύο γονιδίων g και g′, το g′ είναι η μετατόπιση του g, αν για κάποιες ετικέτες l1, . . . , ln και αριθμούς k, m (0 < k, m ≤ n), g = (l1, . . . , ln) και g′ = (lk, . . . , lm, l1, . . . , lk−1, lm+1, . . . , ln).Δηλαδή παίρνουμε ένα «κομμάτι» (lk, . . . , lm) αρχικού γονιδιώματος g του και το μεταθέτουμε στην «αρχή» του γονιδιώματος, έτσι προκύπτει το g′ γονιδίωμα. Π.χ στο Σχήμα 3.1 το γονιδίωμα του Σχήματος 3.1β αποτελεί μετατόπιση του Σχήματος 3.1α.

Αντιστροφή (inversions): Δεδομένου δύο γονιδίων g και g′, το g′ είναι η αντιστροφή του g, αν για κάποιες ετικέτες l1, . . . , ln και αριθμό m (0 < m ≤ n), g = (l1, . . . , ln) και g′ = (−lm−1, . . . ,−l1, lm+1, . . . , ln).Δηλαδή, παίρνουμε ένα «κομμάτι» (l1….lm) του αρχικού γονιδιώματος g και αντιστρέφουμε την σειρά των γονιδίων του αντιστρέφοντας και τον προσανατολισμό τους, έτσι προκύπτει το g′ γονιδίωμα. Π.χ. στο Σχήμα 3.1 το γονιδίωμα του Σχήματος 3.1γ αποτελεί αντιστροφή του Σχήματος 3.1β.

Μετατοπισμένη αντιστροφή (transversion): Δεδομένου δύο γονιδίων g και g′, το g′ είναι η transversion (ή η αντιστραμμένη μετατόπιση) του g, αν για κάποιες ετικέτες l1, . . . , ln και αριθμούς k, m (0 < k, m ≤ n), g = (l1, . . . , ln) και g′ = (−lm, . . . ,−lk, l1, . . . , lk−1, lm+1, . . . , ln).Δηλαδή παίρνουμε ένα «κομμάτι» (lk, . . . , lm) του αρχικού γονιδιώματος g και το μεταθέτουμε στην «αρχή» του γονιδιώματος, αφού το αντιστρέψουμε πρώτα. Π.χ. στο Σχήμα 3.1 το γονιδίωμα του Σχήματος 3.1δ αποτελεί αντιστροφή του Σχήματος 3.1γ.

Το πρόβλημα που προσπαθούμε να λύσουμε, είναι να μας δοθεί κάποιο γονιδίωμα(αρχική κατάσταση) και μέσα από την εκτέλεση δράσεων

μετατόπισης, αντιστροφής και μετατοπισμένης αντιστροφής να δημιουργηθεί το γονιδίωμα g′ το οποίο είναι και ο στόχος μας.

3.3 Αναδιοργάνωση γονιδιώματος στην ADL

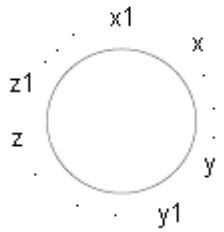
Πιο κάτω παρουσιάζονται αποσπασματικά, κάποιες από τις δράσεις του κόσμου στην γλώσσα του TLPLAN προγραμματιστή δράσης(προτασιακή λογική γραμμένη με σύνταξη της γλώσσας lisp),γραμμένες στο ύφος της adl,όπως παρουσιάζεται στο άρθρο Genome Rearrangement and Planning των Ersa Erdem και Elisabeth Tiller[2].Στην συνέχεια θα παρουσιαστούν οι δράσεις με κωδικοποίηση στην adl.

Στο Σχήμα3.2, παρουσιάζεται ο ορισμός της δράσης transpose η οποία μοντελοποιεί την λειτουργία της μετατόπισης. Η δράση αυτή παίρνει σαν παραμέτρους τρία γονίδια x, y, z. Η ακολουθία γονιδίων η οποία θα μετατοπιστεί, αρχίζει με τον γονίδιο x, τελειώνει με το y και θα τοποθετηθεί μετά το γονίδιο z. Προαπαιτήσεις της δράσης, είναι οι παραμέτροι να είναι ετικέτες (γονίδια) και να μπορεί να γίνει η μετατόπιση (κατηγόρημα cantranspose αληθές).Τα αποτελέσματα της δράσης, τα οποία είναι υπό συνθήκη, για να εκτελεστούν και να γίνει η μετατόπιση θα πρέπει πρώτα να βρεθεί η θέση που είχαν τα γονίδια x, y και z στο γονιδίωμα. Δηλαδή θα πρέπει να βεβαιωθούμε ότι υπάρχει κάποιο γονίδιο x1 το οποίο βρίσκεται αμέσως πριν το γονίδιο x,ότι υπάρχει κάποιο y1 το οποίο βρίσκεται αμέσως μετά το y και κάποιο z1 το οποίο βρίσκεται αμέσως μετά το z. Αυτό θα γίνει με την χρήση του υπαρξιακού τελεστή exists αλλά και με την χρήση του κατηγορήματος (clockwise ?d ?d1) το οποίο αναφέρεται στο ότι το γονίδιο d1 βρίσκεται αμέσως μετά το γονίδιο d (με την φορά των δεικτών στου ρολογιού).Το γονιδίωμα πριν από την εκτέλεση της δράσης(Σχήμα3.3α) είναι (?x1,?x..?y,?y1..?z,?z1,...) και μετά την μετατόπιση (Σχήμα3.3β) θα είναι (?x1,?y1..?z,?x..?y,?z1,...). Αφού γίνει αληθές η συνθήκη, τότε μπορούν να εκτελεστούν τα αποτελέσματα της δράσης τα οποία εκφράζονται με την χρήση λίστας προσθήκης – διαγραφής (add –delete list). Με την εκτέλεση της δράσης, προτίθενται τα εξής αληθή κατηγορήματα: (clockwise

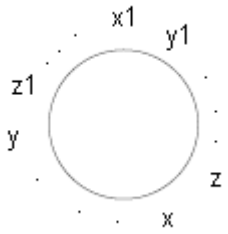
?x1 ?y1),(clockwise ?z ?x),(clockwise ?y ?z1),δηλαδή πλέον το γονίδιο y1 βρίσκεται αμέσως μετά το x1,το x αμέσως μετά από το z και το z αμέσως μετά το y.Επίσης πλέον είναι ψευδή τα εξής κατηγορήματα: (clockwise ?x1 ?x),(clockwise ?y ?y1),(clockwise ?z ?z1) τα οποία δηλώνουν τις αρχικές θέσεις των γονιδίων, αρχής και τέλους του μέρους του γονιδιώματος που μετατοπίστηκε (x,y) και του γονιδίου z, σημείου αναφοράς για την μετατόπιση.

```
(def-adl-operator (transpose ?x ?y ?z)
; preconditions
(pre (?x) (label ?x) (?y) (label ?y)(?z) (label ?z)(cantranspose ?x ?y ?z))

; insertion of ?x ?y after ?z  in (?x1,?x..?y,?y1..?z,?z1,...) is
; (?x1,?y1..?z,?x..?y,?z1,...)
(exists (?x1) (clockwise ?x1 ?x)(?y1) (clockwise ?y ?y1)
        (?z1) (clockwise ?z ?z1)
        (and (add (clockwise ?x1 ?y1)(clockwise ?z ?x)(clockwise ?y ?z1))
             (del (clockwise ?x1 ?x)(clockwise ?y ?y1)(clockwise ?z ?z1)))))
```

Σχήμα 3.2:Ορισμός δράσης μετατόπισης



Σχήμα 3.3α



Σχήμα 3.3β

Σχήμα 3.3: Στο Σχήμα 3.3α παρουσιάζεται το αρχικό γονιδίωμα και στο Σχήμα3.3β το γονιδίωμα μετά την δράση μετατόπισης.

Στο Σχήμα 3.4 παρουσιάζεται ο ορισμός του κατηγορήματος (cantranspose ?x ?y ?z).Το κατηγόρημα αυτό έχει αληθή τιμή αν δεν ισχύουν κάποιοι περιορισμοί

κάτω από τους οποίους δεν μπορεί να γίνει η μετατόπιση. Για να είναι εφικτή η μετατόπιση, πρέπει να ισχύουν όλοι οι περιορισμοί, γι' αυτό εκμεταλλευόμαστε την λειτουργία του τελεστή and(αν ένα από τα ορίσματα του είναι ψευδές, επιστρέφει ψευδή τιμή). Για να είναι εφικτή η μετατόπιση πρέπει: οι δράσεις που έχουν εκτελεστεί μέχρι τώρα να είναι λιγότερες από k,περιορισμός που τίθεται από τον ορισμό του προβλήματος, (< (plan-length) (k), η αρχή x και το τέλος y του μέρους του γονιδιώματος που θα μετατοπιστεί να μην ταυτίζεται με το σημείο αναφοράς z για την μετατόπιση(not (= ?x ?z), (not (= ?y ?z)).Επίσης, η μετατόπιση δεν θα είχε νόημα αν η αρχή x του μέρους του γονιδιώματος που θα μετατοπιστεί βρίσκεται ήδη μετά από το z (not (clockwise ?z ?x)) και αν το σημείο αναφοράς z για την μετατόπιση βρίσκεται μέσα στο μέρος του γονιδιώματος που θα μετατοπιστεί (notbetween ?z ?x ?y κατηγόρημα).

```
(def-defined-predicate  (cantranspose ?x ?y ?z)
(and (< (plan-length) (k)) (not (= ?x ?z)) (not (= ?y ?z))
       ; ?z is not followed by ?x
      (not (clockwise ?z ?x))
     ; ?z is not between ?x and ?y
      (notbetween ?z ?x ?y)))
```

Σχήμα 3.4:Ορισμός κατηγορήματος cantranspose

Ο ορισμός της δράσης αντιστροφής παρουσιάζεται στο Σχήμα 3.5.Σαν παραμέτρους, δέχεται την αρχή x και το τέλος y του κομματιού του γονιδιώματος το οποίο θα αναστραφεί. Προαπαιτήση της δράσης, είναι οι παράμετροι να είναι ετικέτες – γονίδια και η αντιστροφή να μπορεί να γίνει (κατηγόρημα caninvert αληθές).Σαν αποτέλεσμα της δράσης, η ετικέτα x αλλάζει πρόσημο, πολλαπλασιάζοντας την με -1(το γονίδιο x αλλάζει προσανατολισμό και έτσι το κατηγόρημα label ?x είναι ψευδής ενώ το label (* -1 ?x) αληθείς).Στην συνέχεια, για να αναστραφούν τα γονίδια z1,z2..μεταξύ του x και του y γίνεται χρήση του τελεστή συνεπαγωγής, implies.Εφόσον τα γονίδια x και y δεν ταυτίζονται(αν ταυτίζονταν δεν θα υπήρχαν ενδιάμεσα τους γονίδια για να αναστραφούν και η διαδικασία της αντιστροφής θα είχε ολοκληρωθεί σε

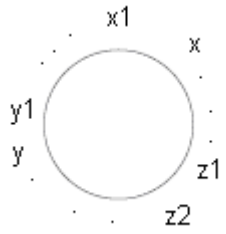
αυτό το σημείο),για κάθε γονίδιο z2 το οποίο βρίσκεται δεξιόστροφά από το z1,εφόσον τα z1,z2 βρίσκονται ενδιάμεσα των x,y(implies(and (in ?z1 ?x ?y) (in ?z2 ?x ?y)),αλλάζουν θέση και προσανατολισμό(add (label (* -1 ?z2))), (del (clockwise ?z1 ?z2)),(add (clockwise(* -1 ?z2).Δηλαδή γίνεται πολλαπλασιασμός με το -1 και αντιστρέφεται η σειρά τους. Έπειτα για να ολοκληρωθεί η διαδικασία αντιστροφής, θα πρέπει να αναστραφούν και τα x,y(η αρχή και το τέλος δηλαδή του μέρους του γονιδιώματος που δόθηκε σαν παράμετρος).Αυτό γίνεται, με την χρήση του υπαρξιακού τελεστή. Αφού υπάρχουν κάποια γονίδια x1 και y1((clockwise ?x1 ?x),(clockwise ?y ?y1) το x1 βρίσκεται αμέσως πριν από το x και το y1 αμέσως μετά το y με δεξιόστροφη φορά),αλλάζουμε τον προσανατολισμό των x,y αλλά και την θέση τους(add (clockwise ?x1 (* -1 ?y))(clockwise (* -1 ?x) ?y1)),(del (clockwise ?x1 ?x) (clockwise ?y ?y1)).Δηλαδή το -y πλέον ακολουθεί δεξιόστροφά το x1,ενώ το –x ακολουθείται από το y1. Το γονιδίωμα πριν από την εκτέλεση της δράσης (Σχήμα3.7α) είναι (?x1,?x..?z1,?z2..?y,?y1,...) και μετά την αντιστροφή (Σχήμα3.7β) θα είναι (?x1,-?y..?-z2,-?z1..-?x,?y1,...).

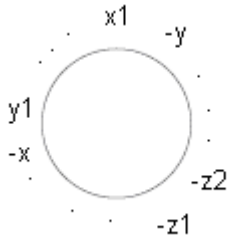
```
(def-adl-operator (invert ?x ?y)
; preconditions
(pre (?x) (label ?x) (?y) (label ?y)(caninvert ?x ?y))

; change the sign of ?x
(del (label ?x))
(add (label (* -1 ?x)))

; inversion of ?x..?y in  (?x1,?x..?z1,?z2..?y,?y1,...) is
;(?x1,-?y..?-z2,-?z1..-?x,?y1,...
; first invert every sequence ?z1 ?z2  in ?x..?y
 (implies
(not (= ?x ?y))
(forall (?z1 ?z2) (clockwise ?z1 ?z2)
        (implies (and (in ?z1 ?x ?y) (in ?z2 ?x ?y))
                 (and (del (label ?z2))(add (label (* -1 ?z2)))
```

(del (clockwise ?z1 ?z2))(add (clockwise(* -1 ?z2)(* -1 ?z1)))))))

; then change the neighbors of ?x1? and ?y1
(exists (?x1) (clockwise ?x1 ?x) (?y1) (clockwise ?y ?y1)
(and (add (clockwise ?x1 (* -1 ?y))(clockwise (* -1 ?x) ?y1))
(del (clockwise ?x1 ?x) (clockwise ?y ?y1)))))

Σχήμα 3.5:Ορισμός δράσης αντιστροφής





Σχήμα 3.6α *Σχήμα 3.6β*

Σχήμα 3.6: Στο Σχήμα 3.6 α παρουσιάζεται το αρχικό γονιδίωμα και στο Σχήμα3.6β το γονιδίωμα μετά την δράση αντιστροφής.

Ο ορισμός της μετατοπισμένης αντιστροφής προκύπτει από τον συνδυασμό των δράσεων μετατόπισης και αντιστροφής που μελετήσαμε πιο πάνω.

Σε αυτό το σημείο θα μελετήσουμε την κωδικοποίηση του κόσμου και του προβλήματος που έγινε στην adl.
Βασιζόμενη στην προηγούμενη υλοποίηση που μελετήθηκε πιο πάνω ορίσαμε τον κόσμο Genome_rearr,ο οποίος βασίζεται στο νόημα της πιο πάνω υλοποίησης αλλά δεν διατηρεί όλους τους τελεστές. Στο Παράρτημα Β υπάρχει ολόκληρος ο κώδικας ορισμού του κόσμου και ενός προβλήματος.

Όσον αφορά τα κατηγορήματα που υπάρχουν στον κόσμο, έχουν παραμείνει τα ίδια με πριν και έχουν προστεθεί κατηγορήματα για διαχωρισμό του πρόσημου κάποιου γονιδίου σε θετικό και αρνητικό,((positive_label ?x -gene) και (negative_label ?x -gene)),λόγο του ότι η γλώσσα μας περιορίζει να

χρησιμοποιούμε πρόσημα για τον προσανατολισμό του γονιδίου,εμείς χρησιμοποιήσαμε κατηγορήματα. Στο Σχήμα 3.9 περιγράφεται αναλυτικά το τί αντιπροσωπεύει το κάθε ένα κατηγόρημα.

```
(define (domain Genome_rearr)              ;Orismos domain Genome_rearr
      (:requirements :adl)                 ;Apaithsh domain h glwssa adl
      (:types gene - object)               ;xrhsimopihte o typos gene
      (:predicates                         ;Kathgorhmata domain:

  (clockwise ?x ?z  -gene)      ;clockwise: to gonidio ?z
                                ;brisketai amesws meta to gonidio
                                ;?x (me thn fora twn diktwn
                                ;toy rologioy)

  (in ?z ?x ?y -gene)           ;in: to gonidio ?z brisketai mesa
                                ;sthn akoloythia gonidiwn poy
                                ;arxizei apo ?x gonidio kai
                                ;teleiwnei sto ?y gonidio

  (positive_label ?x -gene)     ;positive_label:antiprwswpeyei
                                ;ton thetiko prosanatolismo
                                ;toy gonidioy x

  (negative_label ?x -gene)  ;positive_label:antiprwswpeyei
                                ;ton arnitiko prosanatolismo
                                ;toy gonidioy x
    )
```

Σχήμα 3.10: Ονομα και κατηγορήματα κόσμου Genome_rearr

Η δράση transpose που περιγράφηκε πιο πάνω, εξακολουθεί να έχει το ίδιο νόημα αλλά σε αυτή την υλοποίηση έχει σπάσει σε δύο δράσεις μια για την περίπτωση που το τμήμα του χρωμοσώματος που αποκόπτεται αποτελείται

από μόνο ένα γονίδιο και μια άλλη δράση στην περίπτωση που αποτελείται από πολλά γονίδια.

Ας εξετάσουμε αρχικά την transpose_single μέσον της οποίας αποκόπτεται ένα μόνο γονίδιο(το x) από το χρωμόσωμα και επανατοποθετήται μετά από κάποιο συγκεκριμένο γονίδιο (το z).

Αυτή η δράση δέχεται σαν παραμέτρους το αποκοπτώμενο γονίδιο x και το z γονίδιο μετά από το οποίο θα επανατοποθετηθεί x.Μοναδική προϋπόθεση της δράσης αυτής είναι τα δύο αυτά γονίδια παραμέτροι να μην είναι τα ίδια για να μπορεί να εκτελεστεί η δράση.

Σαν αποτέλεσμα της δράσης (Σχήμα 3.13) έχουμε την αποκοπή και επανατοποθέτηση του x μετά το z. Αφού εντοπιστούν τα ακριβώς γειτονικά γονίδια του x (και γίνουν κάποιοι απαραίτητοι ελέγχοι για το ότι το x και z δεν είναι ίδια με τα γειτονικά του x),αφαιρείται το x από ανάμεσα τους.Τώρα πλέον τα δυο γειτονικά γονίδια του x γειτνιάζουν επακριβώς μεταξύ τους. Επανακαθήσταται με αυτό τον τρόπο, το χρωμόσωμα σε εκείνο το σημείο και στην συνέχεια το x τοποθετείται μετά το z (αφού ενημερωθεί το αμέσως επόμενο γονίδιο μετά το z για την αλλαγή στον γείτονα του).Επειτά,αφού το ουσιαστικό μέρος των αποτελεσμάτων της δράσης έχει εκτελεστεί, μέσον της δράσης θα πρέπει να ενημερωθεί το κατηγόρημα in για τις νέες αλλαγές που έγιναν στο γονιδίομα.Έτσι λοιπόν, αν κάποιο γονίδιο βρισκόνταν μεταξύ του x και z (δεν ήταν ίδιο με αυτά),τώρα πλέον δεν μπορεί να βρίσκεται ανάμεσα τους αφού αυτά είναι ακολουθιακά. Επίσης, αν κάποια γονίδια m και m1,τα οποία δεν είναι ίδια αναμεταξύ τους αλλά ούτε είναι ίδια με το x και το z και το m εμπεριέχεται μεταξύ του z και m1,τότε με το πέρας της δράσης το m εμπεριέχεται στην ακολουθία από το x μέχρι το m1(λογικό επακόλουθο αφού πλέον το x είναι ακριβώς μετά το z).

```
:effect (and (forall  (?x1 ?x2 ?z1 -gene)
            (when (and (not (= ?x ?x1))
                  (not (= ?x ?x2))
```

```
        (not (= ?z ?x1))
        (not (= ?z ?x2))
        (not (= ?z ?z1))
        (clockwise ?x1 ?x)
        (clockwise ?x ?x2)
        (clockwise ?z ?z1))
   (and  (clockwise ?x1 ?x2)
        (clockwise ?z ?x)
        (clockwise ?x ?z1)
     (not (clockwise ?x1 ?x))
     (not (clockwise ?x ?x2))
     (not (clockwise ?z ?z1))
) ) )

(forall  (?m -gene)
(when (and (not (= ?x ?m))
          (not (= ?z ?m))
          (in ?m ?x ?z)
      )
      (and (not (in ?m ?x ?z)))
 ) )
(forall  (?m ?m1 -gene)
(when (and (not (= ?x ?m))
          (not (= ?z ?m))
          (not (= ?x ?m1))
          (not (= ?z ?m1))
          (not (= ?m ?m1))
          (in ?m ?z ?m1)
      )
      (and  (in ?m ?x ?m1))
 )        )    )
```

Σχήμα 3.13: Αποτελέσματα δράσης transpose_single

Στην συνέχεια θα επεξηγήσουμε την περίπτωση αποκοπής ακολουθίας πολλαπλών στοιχείων. Πρόκειται για την δράση transpose,η οποία έχει παίρνει σαν παραμέτρους την αρχή x, το τέλος της ακολουθίας αποκοπής y,το γονίδιο z μετά από το οποίο θα επανατοποθετηθεί η αποκοπτόμενη ακολουθία, το αμέσως προηγούμενο γονίδιο x1 από το x, το αμέσως επόμενο γονίδιο y1 από το y και το αμέσως επόμενο z1 του z.
Προαπαιτήσεις της δράσης (Σχήμα 3.14) αυτής είναι η αρχή και το τέλος της αποκοπτόμενης ακολουθίας να μην είναι τα ίδια. Ακόμα, το x και το y να μην είναι ίδια με το γονίδιο z μετά το οποίο θα τοποθετηθεί η ακολουθία με την αποκοπή. Το z δεν πρέπει να είναι ίδιο με το ακριβώς επόμενο του z1.
Το επόμενο y1 και το προηγούμενο x1 του αποκοπτόμενου τμήματος δεν πρέπει να είναι τα ίδια. Το x1 να μην είναι ίδιο με το z, το y1και το z1 να μην είναι ίδια με το x, το y να μην είναι ίδιο με το x1 αλλά ούτε το z1 να είναι ίδιο με το y1. Το τέλος y και η αρχή x της αποκοπτόμενης ακολουθίας δεν θα πρέπει να βρίσκονται ήδη μετά από το γονίδιο z μετά από το οποίο θα γίνει η επανατοποθέτηση αλλά και η αρχή της ακολουθίας αποκοπής να μην βρίσκεται ακριβώς μια θέση πριν το z .Τέλος,για να μπορεί να πραγματοποιηθεί η δράση θα πρέπει το γονίδιο που μετά από αυτό θα τοποθετηθεί η αποκομμένη ακολουθία να μην περιλαμβάνεται σε αυτή.

```
;proapaithseis ths drashs
    :precondition (and  (not (= ?x ?y))
                        (not (= ?x ?z))
                        (not (= ?y ?z))
                        (not (= ?z ?z1))
                        (not (= ?x1 ?y1))
                        (not(= ?x1 ?z))
                        (not(= ?z1 ?x))
                        (not (= ?y1 ?x))
                        (not (= ?y ?x1))
                        (not (= ?z1 ?y1))
                        (not (clockwise ?z ?x))
```

```
(not (clockwise ?z ?y))
(not (clockwise ?x ?z))
(not (in ?z ?x ?y))
(clockwise ?x1 ?x)
(clockwise ?y ?y1)
(clockwise ?z ?z1)   )
```

Σχήμα 3.14: Προαπαιτήσεις δράσης transpose

Τα αποτελέσματα της δράσης τα έχουμε χωρίσει σε πέντε μέρη για να μπορούν να επεξηγηθούν καλύτερα. Με το ίδιο σκεπτικό των αποτελεσμάτων της προηγούμενης δράσης αφού γίνει η επανατοποθέτηση της αποκοπτόμενης ακολουθίας στο γονιδίωμα,ενημερώνεται ανάλογα με την περίπτωση το κατηγόρημα in.

Στο πρώτο μέρος - Σχήμα 3.15,βλεπουμε την αποκατάσταση της ακολουθίας στο μέρος όπου έγινε η αποκοπή αμέσως προηγούμενο και το αμέσως επόμενο γονίδιο από την αποκοπτόμενη ακολουθία,τώρα πλέον το ένα ακολουθεί το άλλο αντίστοιχα.Η αποκομμένη ακολουθία επανατοποθετείται πίσω στο γονιδίωμα (το x είναι πλέον ακριβώς μετά το z και το y είναι ακριβώς πριν το z1).Ακόμα τα γονίδια αρχή x και τέλος y της αποκοπτόμενης ακολουθίας βρίσκονται πλέον με την επανατοποθέτηση μεταξύ του z και του z1 γονιδίου.Βέβαια οι προηγούμενοι επακριβώς γείτονες των γονιδίων x,y,z δεν ισχύουν πλέον.

```
:effect (                    ;apotelesmata drashs

        and  (clockwise ?x1 ?y1)
             (clockwise ?z ?x)
             (clockwise ?y ?z1)
             (in ?x ?z ?z1)
             (in ?y ?z ?z1)
             (not (clockwise ?x1 ?x))
```

```
(not (clockwise ?y ?y1))
(not (clockwise ?z ?z1))
```

Σχήμα 3.15: Αποτελέσματα δράσης transpose

Στο δεύτερο μέρος των αποτελεσμάτων Σχήμα 3.16 εξετάζουμε την περίπτωση όπου ένα m γονίδιο, εμπεριέχεται στην αποκοπτόμενη ακολουθία x y και δεν είναι ίδιο ούτε με το x ,y ούτε με το z .Τότε, το γονίδιο αυτό θα εμπεριέχεται μετά το πέρας της δράσης και στην z z1 ακολουθία(λόγο της επανατοποθέτησης),αλλά δεν θα υπάρχει στην x1 y1 λόγω της αποκοπής.

```
(forall (?m -gene)
 (when
 (and (in ?m ?x ?y)
 (not (= ?m ?x))
 (not (= ?m ?y))
 (not (= ?m ?z))
 )
 (and  (in ?m ?z ?z1)
       (not (in ?m ?x1 ?y1)))
 )
)
```

Σχήμα 3.16:Δέυτερο μέρος των αποτελεσμάτων δράσης transpose

Με το τρίτο μέρος των αποτελεσμάτων Σχήμα 3.17, εξετάζουμε την περίπτωση όπου ένα m γονίδιο, εμπεριέχεται στην ακολουθία z y και όχι στην x y και δεν είναι ίδιο ούτε με το x ,y ούτε με το z .Τότε, το γονίδιο αυτό με την επανατοποθέτηση το γονίδιο αυτό δεν θα βρίσκεται στην z y ακολουθία διότι εκεί οι μόνες θέσεις που μπορεί να είναι μεταξύ του x ?y κάτι που δεν ισχύει.Ασφαλώς δεν μπορεί να βρίσκεται το m ούτε μεταξύ των z z1(αφού αυτά τα γονίδια είναι ακριβώς γειτονικά με το x y αντίστοιχα).

```
(forall  (?m -gene)
   (when
   (and (not (= ?x ?m))
   (not (= ?z ?m))
   (not (= ?y ?m))
   (in ?m ?z ?y)
   (not (in ?m ?x ?y))
    )


  (and (not(in ?m ?z ?y))(not(in ?m ?z ?z1)) )
)
)
```

Σχήμα 3.17: Τρίτο μερός των αποτελεσμάτων δράσης transpose

Με το τέταρτο μέρος των αποτελεσμάτων (Σχήμα 3.18), εξετάζουμε την περίπτωση όπου ένα m γονίδιο, εμπεριέχεται στην ακολουθία y z και δεν είναι ίδιο ούτε με το x ,y ούτε με το z . Με το πέρας της δράσης,δεν μπορεί να περιέχεται z y αλλά ούτε μεταξύ των x y (διότι λόγο της συνθήκης αρχικά θα μπορούσε να υπάρχει μεταξύ των y1 z και y z μόνο, με την επανατοποθέτηση θα μείνει στο ίδιο διάστημα και δεν μπορεί να υπάρξει μεταξύ ?z ?y).Πριν x1 x …y y1…m..z z1…,
Μετά x1 y1 …m…z x…y z1….

```
(forall  (?m -gene)
 (when
 (and (not (= ?x ?m))
 (not (= ?z ?m))
 (not (= ?y ?m))
 (in ?m ?y ?z)
  )
```

```
        (and (not(in ?m ?z ?y))(not(in ?m ?x ?y)))
    )
    )
```

Σχήμα 3.18: Τέταρτο μέρος των αποτελεσμάτων δράσης transpose

Στο τελευταίο μέρος αποτελεσμάτων της δράσης (Σχήμα 3.19), έχουμε μια πιο γενική περίπτωση,όπου το z είναι μεταξύ κάποιων m1 m2 γονιδίων.Για το m1 ισχύει ότι δεν είναι ίδιο με το x, ούτε με το z, ούτε με κάποιο m, το οποίο βρίσκεται μεταξύ των x y(και το m δεν είναι ίδιο με x z).Επίσης το m1 δεν πρέπει να βρίσκεται μεταξύ των x y (για να μην υπάρχει περίπτωση να είναι μετά από το m και να μη είναι ορθό το αποτέλεσμα της δράσης. Π.χ x1 x …m…y y1 …m1…z z1…m2… είναι μια περίπτωση του γονιδιώματος που πληρεί τις συνθήκες. Αν εκτελεστεί η συνθήκη, το m θα είναι μεταξύ του m1 m2 ,εφόσον με την επανατοποθέτηση
το αποκοπτόμενο x y τμήμα του γονιδιώματος που περιέχει το m θα τοποθετηθεί μετά το z. Π.χ μετά την αποκοπή x1 y1 …m1…z x …m…y z1..m2…

```
(forall  (?m ?m1 ?m2 -gene)
        (when (and (not (= ?x ?m))
                (not (= ?z ?m))
                (not (= ?x ?m1))
                (not (= ?z ?m1))
                (not (= ?m ?m1))
                (in ?z ?m1 ?m2)
                (not in ?m1 ?x ? y)
                (in ?m ?x ?y)
            )
                (in ?m ?m1 ?m2)
    )
    )
```

Σχήμα 3.19: Τελευταίο μέρος αποτελεσμάτων της δράσης transpose

Για σκοπούς απλοποίησης στην δράση invert γίνεται η αναστροφή κάποιου τμήματος γονιδίων με το σκεπτικό της προηγούμενης υλοποίησης με την διαφορά ότι δεν γίνεται αντιστροφή των προσήμων του επιλεγμένου για αναστροφή τμήματος του γονιδωματος.Αν υπάρχει ανάγκη για αντιστροφή προσήμων για την επίτευξη του στόχου του προβλήματος,τότε αυτό θα γίνει μέσα από τους νέους τελεστές δράσης sign_negative και sign_positive.

Με τον sign_negative τελεστή δράσης(Σχήμα 3.20) αν κάποιο γονίδιο έχει αρνητικό πρόσημο τότε με την δράση γίνεται θετικό και παύει να ισχύει το αρνητικό.

Με τον sign_positive τελεστή δράσης(Σχήμα 3.21) αν κάποιο γονίδιο έχει θετικό πρόσημο τότε με την δράση γίνεται αρνητικό και παύει να ισχύει το θετικό.

```
(:action sign_negative

 :parameters (?x -gene)

 ;proapaithseis ths drashs
 :precondition (and  (negative_label ?x) )

 ;apotelesmata drashs
 :effect ( and
          (positive_label ?x)
          (not (negative_label ?x))
        )
)
```

Σχήμα3.20: Ορισμός τελεστή δράσης sign_negative.

```
(:action sign_positive
 :parameters (?x -gene)

 ;proapaithseis ths drashs
```

```
 :precondition (and (positive_label ?x)  )

;apotelesmata drashs
:effect (and (not (positive_label ?x))   ;h arxh kai to telos ths akoloythias
          (negative_label ?x)            ;allazoyn prosanatolismo apo thetiko se
        )
)
```

Σχήμα3.21: Ορισμός τελεστή _ positive δράσης sign

Με την δράση invert_multiple_genes όπως αναφέραμε πιο πάνω γίνεται η αναστροφή ακολουθίας πολλαπλών γονιδίων,χωρίς την αλλαγή προσήμων.Οι παραμέτροι αυτής της δράσης είναι η αρχή και το τέλος του τμήματος του γονιδιώματος που θα αντιστραφή.

Μόνες προαπαιτήσεις της δράσης είναι η αρχή και το τέλος της ακολουθίας αναστροφής να μην είναι ίδια και να μην είναι άμεσα γειτονικά (όχι μόνο ένα γονίδιο στην ακολουθία).

```
;proapaithseis ths drashs
:precondition (and  (not (= ?x ?y))
                    (not (clockwise ?x ?y))
                    (not (clockwise ?y ?x))
                    )
```

Σχήμα 3.18: Προαπαιτήσεις της δράσης invert_multiple_genes

Στο πρώτο μέρος των αποτελεσμάτων της δράσης invert_multiple_genes (Σχήμα 3.19),γίνεται η αντιστροφή των δύο άκρων της ακολουθίας με τα αμέσως γειτονικά τους.

```
;apotelesmata drashs
:effect (and
            (forall (?m -gene)
```

```
(when (clockwise ?m ?x)
(and (clockwise ?m ?y)
    (not (clockwise ?m ?x)))
)
)

(forall (?m -gene)
(when (clockwise ?x ?m)
(and (clockwise ?m ?x)
    (not (clockwise ?x ?m)))
)
)

(forall (?m -gene)
(when (clockwise ?m ?y)
(and (clockwise ?y ?m)
    (not (clockwise ?m ?y)))
)
)

(forall (?m -gene)
(when (clockwise ?y ?m)
(and (clockwise ?x ?m)
    (not (clockwise ?y ?m)))
)
)
```

Σχήμα3.19:Πρώτο μέρος αποτελεσμάτων δράσης invert_multiple_genes

Στο Σχήμα 3.20 παρουσιάζεται το μέρος των αποτελεσμάτων της δράσης,στο οποίο γίνεται η αντιστροφή ανά δύο αμέσως γειτονικών γονιδίων,τα οποία περιέχονται μεταξύ των δύο άκρων (x , y) της ακολουθίας στην οποία γίνεται η αντιστροφή.

```
(forall (?m ?m1 -gene)
```

```
(when
(and (in ?m ?x ?y)
      (in ?m1 ?x ?y)
      (clockwise ?m1 ?m)
      (not (= ?m ?x))
      (not (= ?m1 ?y))    )
(and  (clockwise ?m ?m1)
     (not (clockwise ?m1 ?m)))
) )
```

Σχήμα 3.20: Δεύτερο μέρος των αποτελέσμάτων της δράσης invert_multiple_genes

Μέσον των αποτελεσμάτων της δράσης που παρουσιάζονται στο Σχήμα 3.21,αφού γίνει η αντιστροφή,ενημερώνεται το κατηγόρημα in για τις νέες ανακατατάξεις που έγιναν στο γονιδίωμα.

Αναλυτικότερα,αν έχουμε αρχικά μια ακολουθία x1 x…m1…m…m2…y y1…z z1 … τότε με την εκτέλεση της δράσης αυτής η ακολουθία διαμορφώνεται ως εξής: x1 y…m2…m…m1…x y1…z z1…
Αν έχουμε αρχικά μια ακολουθία x1 x…m…m1…y y1…z z1… τότε με το πέρας της δράσης η ακολουθία γίνεται x1 y…m1…m…x y1…z z1…
Αν η ακολουθία μας είναι x1 x…m1…m…y y1…z z1… τότε η ακολουθία γίνεται x1 y…m…m1…x y1…z z1…
Τέλος,αν η ακολουθία είναι x1 x…m…y y1…z z1… τότε γίνεται x1 y…m…x y1…z z1…

```
(forall (?m ?m1 ?m2 -gene)
                (when
                (and (in ?m ?x ?y)
                    (or (in ?m1 ?x ?y)
                       (= ?m1 ?x)
                    )
                    (or (in ?m2 ?x ?y)
```

```
            (= ?m2 ?y)
          )
          (in ?m ?m1 ?m2)
          (not (= ?m ?x))
          (not (= ?m1 ?y))
          (not (= ?m ?m1))
          (not (= ?m ?m2))
          (not (= ?m1 ?m2))
       )
       (and  (in ?m ?m2 ?m1)
            (not (in ?m ?m1 ?m2)))
     )
    )

(forall (?m ?m1 -gene)
  (when
  (and (in ?m ?x ?m1)
       (in ?m1 ?x ?y)
       (not(=?m ?y))
       (not(= ?m1 ?y))
       (not(=?m ?m1))
   )
   (and (in ?m ?m1 ?x)
        (not (in ?m ?x ?m1)))
   )
   )

  (forall (?m ?m1 -gene)
  (when
  (and (in ?m ?m1 ?y)
       (in ?m1 ?x ?y)
       (not(=?m ?x))
       (not(= ?m1 ?y))
```

```
        (not(=?m ?m1))
    )
    (and (in ?m ?y ?m1)
          (not (in ?m ?m1 ?y)))
    )
    )

  (forall (?m  -gene)
      (when
        (and (in ?m ?y ?x)
             (not (= ?m ?x))
        (not (= ?m ?y))
        )
        (and  (in ?m ?y ?x)
              (not (in ?m ?x ?y)))
       ))
```

Σχήμα 3.21: Τρίτο μέρος των αποτελέσμάτων της δράσης invert_multiple_genes

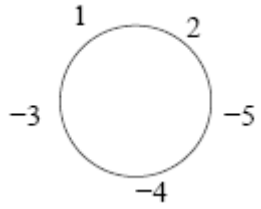
Στο Σχήματα που ακολουθούν παρουσιάζεται ενδεικτικά ένα πρόβλημα αναδιοργάνωσης γονιδιώματος και η λύση που πάραξε ο ff-linux planner.

Στο Σχήμα 3.22 παρουσιάζεται το πρώτο μέρος ορισμού του προβλήματος.Στο πρόβλημα αυτό έχουμε πέντε γονίδια το one,two,five,four,three.

```
(define (problem G_R_01)        ;Orismos problhmatos G_R_01
(:domain Genome_rearr)          ;gia to domain Genome_rearr
(:objects one - gene            ;yparxoyn pente antikeimena
        two - gene              ;se ayto to problhma ola typoy gene
        five - gene
        four - gene
        three - gene)
```

Σχήμα 3.22: Πρώτο μέρος ορισμού προβλήματος G_R_01

Στο Σχήμα 3.24 παρουσιάζονται τα κατηγορήματα που ισχύουν αρχικά.Το γονιδίωμα στην αρχική του κατάσταση έχει αυτή την μορφή του Σχήματος 3.23.



Σχήμα 3.23:Αρχική μορφή γονιδιώματος.

(:init ;Kathgorhmata poy einai alhthh sthn arxikh katastash

(clockwise one two)
(positive_label one)
(positive_label two)
(clockwise two five)
(negative_label five)
(clockwise five four)
(negative_label four)
(clockwise four three)
(negative_label three)
(clockwise three one)
(in two one five)
(in two one four)
(in two one three)
(in two three five)
(in two three four)
(in two four five)
(in five two four)
(in five two three)

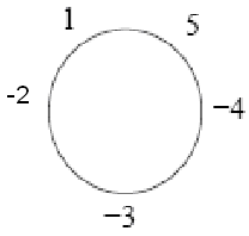
```
(in five two one)
(in five three four)
(in five one four)
(in five one three)
(in four five three)
(in four five one)
(in four five two)
(in four two three)
(in four two one)
(in four one three)
(in three four two)
(in three four five)
(in three four one)
(in three five one)
(in three five two)
(in three two one)
(in one three two)
(in one four two)
(in one three five)
(in one four five)
(in one three four)
(in one five two)
)
```

Σχήμα 3.24:Αρχική κατάσταση προβλήματος G_R_01

Στο Σχήμα 3.25 παρουσιάζεται το γονιδίωμα στόχος,γραφικά παρουσιάζεται στο Σχήμα 3.26.

```
(:goal (and                    ;stoxos poy prepei na ekplhrwthei
        (clockwise one five)
        (positive_label one)
        (positive_label five)
```

```
            (clockwise five four)
            (negative_label four)
            (clockwise four three)
            (negative_label three)
            (clockwise three two)
            (negative_label two)
            (clockwise two one)
        )
    )

)
```



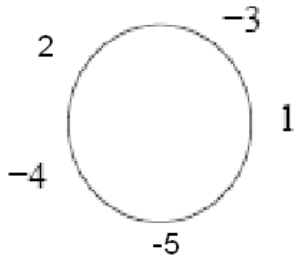
Σχήμα 3.26:Γραφική αναπαράσταση γονιδίου στόχου.

Στο Σχήμα 3.27 παρουσιάζεται η λύση που έδωσε για το πρόβλημα ο ff-linux planner.Γραφικά η λύση παρουσιάζεται στο Σχήμα 3.28.

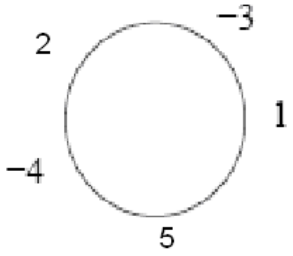
Λύση που πάραξε ο planner ff-linux:

0: TRANSPOSE THREE ONE TWO FOUR TWO FIVE
1: SIGN_NEGATIVE FIVE
2: TRANSPOSE_SINGLE THREE FOUR
3: SIGN_POSITIVE TWO

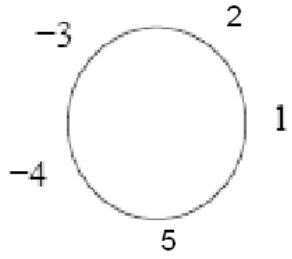
Σχήμα 3.27: Λύση που πάρηξε ο planner ff-linux για το πιο πάνω πρόβλημα.



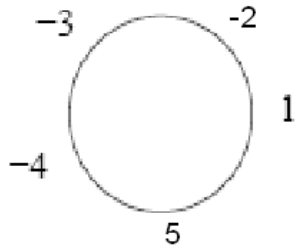
Σχήμα 3.28 α :Βήμα 0



Σχήμα 3.28 β :Βήμα 1



Σχήμα 3.28 γ :Βήμα 2



Σχήμα 3.28 δ :Βήμα 3

Σχήμα 3.28:Γραφική λύση που πάρηξε ο planner ff-linux για το πιο πάνω πρόβλημα.

Κεφάλαιο 4

Δυαδική ευθυγράμμιση ακολουθίας και προγραμματισμός δράσης

(Pairwise Sequence Alignment and planning)

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο, θα μελετήσουμε το πρόβλημα της δυαδικής ευθυγράμμισης ακολουθίας, από πλευράς προγραμματισμού δράσης. Αφού αναλυθούν κάποιοι σημαντικοί όροι για την κατανόηση του προβλήματος, θα παρουσιαστεί η μοντελοποίηση του κόσμου και του προβλήματος στο adl υποσύνολο της pddl γλώσσας.

4.2 Περιγραφή Προβλήματος

Γενικά μια ακολουθία αποτελείται από χαρακτήρες κάποιου αλφαβήτου A. A* είναι το σύνολο όλων των πεπερασμένων ακολουθιών χαρακτήρων από το A. Με μεταβλητές όπως s,u,v,t συμβολίζουμε τις ακολουθίες από το A*[8]. Αν παραδείγματος χάριν, το αλφάβητο είναι οι τέσσερις χαρακτήρες που αναπαριστούν τις βάσεις του DNA {A,G,C,T},τότε μιλάμε για μια ακολουθία μονόκλωνου DNA. Αν το αλφάβητο αποτελείται από τα σύμβολα των είκοσι αμινοξέων, τότε έχουμε ακολουθία πρωτεΐνης κ.τ.λ. Έστω ότι ονομάζουμε s μια ακολουθία, τότε |s| συμβολίζει το μήκος της. Για να μπορούμε να αναφερόμαστε με περισσότερη ευκολία σε ένα μέλος της ακολουθίας ή σε υποακολουθίες,

σημειώνουμε την θέση του κάθε χαρακτήρα στην ακολουθία. Παραδείγματος χάριν [8], στην ακολουθία s=ACCACGTA,οι χαρακτήρες που την αποτελούν είναι από το αλφάβητο του DNA, έχει μήκος |s| =8 και σημειώνουμε για τον κάθε χαρακτήρα την θέση του για να μπορούμε να αναφερόμαστε σε αυτόν $_0A_1C_2C_3A_4C_5G_6T_7A_8$.

Το πρόβλημα της δυαδικής ευθυγράμμισης ακολουθίας είναι, δεδομένου δύο ακολουθιών s και t οι οποίες έχουν το ίδιο μήκος, να διαπιστωθεί πόσο 'κοντά' είναι οι δυο ακολουθίες ως προς την ομοιότητα των χαρακτήρων ανά θέση και με ανακατατάξεις που θα γίνουν στην μια ακολουθία s να ταυτολογηθεί στο τέλος με την ακολουθία στόχος t.

4.3 Μοντελοποίηση Προβλήματος σε adl

Μετά από μελέτη της περιγραφής του προβλήματος, κατέληξα στο ότι για την μοντελοποίηση του ο κόσμος που θα δημιουργηθεί θα πρέπει να λαμβάνει ως είσοδο δυο ακολουθίες s και t και μετατρέποντας την s να οδηγηθούμε στον στόχο που είναι η t.Για τις διάφορες τροποποιήσεις που θα υποστεί η s ακολουθία θα χρειαστεί να υλοποιηθούν πέντε τελεστές δράσης, ένας για αντικατάσταση κάποιου χαρακτήρα της s από τον αντίστοιχο του χαρακτήρα στην t, ένας για εισαγωγή στην τελευταία θέση της ακολουθίας s (σε περίπτωση που έμεινε κενή μετά από κάποια διαγραφή),ένας για εισαγωγή του αντίστοιχου στοιχείου της t στην πρώτη θέση της s,ένας τελεστής για διαγραφή στοιχείου από την s ακολουθία και τέλος ένας άλλος τελεστής για μετακίνηση των κενών θέσεων της ακολουθίας μετά από μια διαγραφή.Με αυτό το σκεπτικό, όσον αφόρα τους τελεστές δράσης της ακολουθίας, διατρέχουμε την ακολουθία s και για κάθε στοιχείο της κοιτάζουμε το προηγούμενο και το επόμενο του για να προβούμε στις δράσεις αντικατάστασης και διαγραφής. Γι' αυτό το λόγο έχουν δημιουργηθεί οι τελεστές δράσης για εισαγωγή στην πρώτη και τελευταία θέση της ακολουθίας (λόγο έλλειψης για την πρώτη θέση προηγούμενης και την τελευταία επόμενης).Θεωρούμε επίσης ότι οι ακολουθίες που δίδονται ότι έχουν το ίδιο μήκος. Αναλυτικότερα, δημιούργησα τον κόσμο SEQ_ALIG ο οποίος αποτελείται από πέντε τελεστές δράσης: τον replacement για αντικατάσταση,

τον final_position_insertion για εισαγωγή στην τελευταία θέση αν είναι κενή, τον first_replacment για την αντικατάσταση του πρώτου στοιχείου,τον deletion για διαγραφή κάποιου στοιχείου και τον shift για μετατόπιση των υπολοίπων στοιχείων της ακολουθίας μετά από μια διαγραφή.Προτού εξηγηθούν αναλυτικά οι τελεστές, στο Σχήμα 4.1 παρουσιάζεται μέρος του πρώτο τμήματος ορισμού του κόσμου, στο οποίο δηλώνεται το όνομα του κόσμου, η απαίτηση χρήσης της adl γλώσσας στον ορισμό του κόσμου και οι δύο τύποι μεταβλητών που υπάρχουν στον κόσμο: ο seq_char τύπος για χαρακτήρα της ακολουθίας και ο seq_pos τύπος για θέση στην ακολουθία. Ολόκληρος ο κώδικας του κόσμου και του προβλήματος που πρέπει να επιλύσει υπάρχει στο Παράρτημα Β.

```
(define (domain SEQ_ALIG)                    ;Orismos toy domain SEQ_ALIG

        (:requirements :adl)                 ;Apaithsh toy domain  adl

      (:types seq_char,seq_pos -object)      ;yparxoyn dyo typoi:
                                             ;seq_char typos xarakt. ths akol.
                                             ;seq_pos typos thesh ths akol.
```

Σχήμα 4.1:Πρώτο μέρος του πρώτου τμήματος ορισμού του κόσμου

Στην συνέχεια του πρώτου τμήματος ορισμού του κόσμου δηλώνονται όλα τα κατηγορήματα που υπάρχουν στον κόσμο αυτό. Στο Σχήμα 4.2 υπάρχουν και τα ανάλογα επεξηγηματικά σχόλια για το κάθε κατηγόρημα.

```
(:predicates          ;kathgorhmata domain:

      ;kathgorhma poy ypodhlwnei oti o xarakthras ?character einai melos ths
      ;akolouthias s kai  brisketai  sthn thesh  ?position_number
      (seq_s_member ?character - seq_char,?position_number - seq_pos),

;kathgorhma poy ypodhlwnei oti o xarakthras ?character einai melos ths
;akolouthias t kai  brisketai  sthn thesh  ?position_number
```

```
(seq_t_member ?character -seq_char,?position_number -seq_pos)

;kathgorhma poy ypodhlwnei oti h thesh ?position_number1 einai
;megalyterh apo thn thesh  ?position_number2 sthn akoloythia
(greater_pos  ?position_number1 -seq_pos, ?position_number2 -seq_pos)

;kathgorhma poy ypodhlwnei ton arithmo thns teleytaias theshs sthn
;akoloythia s (h ?position_number einai h teleytai sthn s)
(final_position_s ?position_number -seq_pos)

;kathgorhma poy ypodhlwnei oti h thesh  ?next_position_number
;(arithmos theshs) akribws meta  thn ?position_number thesh arithmitika
(next_position ?position_number -seq_pos,?next_position_number -seq_pos)

;kathgorhma poy ypodhlwnei oti mia thesh einai kenh
(empty_position ?position_number -seq_pos)

;h ?position_number einai h teleytaia thesh ths akolouthias t
(final_position_t  ?position_number -seq_pos)

;h ?position_number einai h prwth thesh ths akolouthias s
(first_position_s ?position_number -seq_pos)

;h ?position_number einai h prwth thesh ths akolouthias s
(first_position_t ?position_number -seq_pos)

)    ;telos dhlwshs kathgorhmatwn domain
```

Σχήμα 4.2: Δήλωση κατηγορημάτων του κόσμου

Στο υπόλοιπο τμήμα ορισμού του κόσμου, βρίσκεται ο ορισμός των τελεστών δράσης. Στο Σχήμα 4.3 παρουσιάζεται το πρώτο μέρος του τελεστή

replacement (όνομα και παράμετροι της δράσης).Αναλυτικά σχόλια για τις παραμέτρους του τελεστή υπάρχουν στο Σχήμα 4.3.

```
(:action replacement
        :parameters(       ;dhlwsh parametrwn

;o xarakthras kai h thesi pou tha antikatastathoyn sthn s akoloythia xarakthrwn
?char_s -seq_char,?pos_in_s -seq_pos,

;o antistoixos xarakthras aytoy poy tha antiuaktastathei sthn akoloythia t
 ?char_t -seq_char, ?pos_in_t -seq_pos,

 ;o amesws prohgoymenos xarakthras kai  h prohgoumenh thesh
 ;(apo ayto poy tha antiaktastathei) sthn s akoloythia
 ?prev_char_s -seq_char,?prev_pos_s -seq_pos,

;o amesws prohgoymenos xarakthras kai h prohgoumenh thesh
; apo ayto poy tha antiaktastathei) sthn t akoloythia
?prev_char_t -seq_char,?prev_pos_t -seq_pos,

;o amesws epomenos xarakthras kai h epomenh thesh
;(toy antistoixoy apo ayto pou tha antikatastathei) sthn
;s akoloythia
?next_char_s -seq_char,?next_pos_s -seq_pos,
?next_char_t -seq_char,?next_pos_t -seq_pos
                    )
```

Σχήμα 4.3: Όνομα και παράμετροι της δράσης replacement

Οι προαπαιτήσεις του τελεστή δράσης replacement παρουσιάζονται στο πιο κάτω Σχήμα 4.4.Για να εκτελεστεί αυτή η δράση θα πρέπει ο χαρακτήρας που θα αντικατασταθεί να ανήκει στην s (αρχική ακολουθία),να υπάρχει ο αντίστοιχος του στην ακολουθία t(ακολουθία στόχος),ο χαρακτήρας που θα αντικατασταθεί να μην είναι ίδιος με αυτόν που θα τον αντικαταστήσει (οι δύο

αυτοί χαρακτήρες πρέπει να έχουν την ιδία θέση στις αντίστοιχες τους ακολουθίες).Επίσης, θα πρέπει να υπάρχει το αμέσως προηγούμενο και το αμέσως επόμενο στοιχείο από αυτό που θα αντικατασταθεί και να υπάρχουν και τα αντίστοιχα στην t ακολουθία (δηλαδή τα στοιχεία αυτά θα πρέπει να ανήκουν στην ακολουθία αλλά και να είναι προηγούμενα ή επόμενα από αυτά που θα αντικαταστήσουν ή θα αντικατασταθούν). Ακόμη, οι αμέσως προηγούμενοι χαρακτήρες(στην s και t ακολουθία) από αυτά που θα γίνει η αντικατάσταση, θα πρέπει να είναι ίδιοι και να έχουν τις ίδιες θέσεις στις δύο ακολουθίες. Το ίδιο ισχύει και για τα αμέσως επόμενα στοιχεία. Τέλος καθορίζουμε στις δυο ακολουθίες ποια θέση πρέπει να ακολουθεί ποια.

```
:precondition (and      ;proapaithseis drashs
;o xarakthras ?char_s sthn thesh ?pos_in_s na einai melos ths akoloythias s
(seq_s_member ?char_s,?pos_in_s)

;o xarakthras ?char_t sthn thesh ?pos_in_t na einai melos ths akoloythias t
(seq_t_member ?char_t,?pos_in_t)

;oi dyo antistoixei xarakthres gia antikatstash den prepei na einai oi idioi
;alla oi theseis toys prepei na einai idies
(not  (= ?char_s,?char_t))(= ?pos_in_s,?pos_in_t)

;na yparxei to prohgoymeno stoixeio sthn s akol. apo ayto poy tha antikastathei
 (seq_s_member ?prev_char_s,?prev_pos_s)

;na yparxei to prohgoymeno stoixeio sthn t akol. apo ayto poy tha
;antikastasteisei
(seq_t_member ?prev_char_t ,?prev_pos_t)

;na yparxei to epomeno stoixeio sthn s akol. apo ayto poy tha antikastathei
(seq_s_member ?next_char_s,?next_pos_s)

;na yparxei to epomeno stoixeio sthn t akol. apo ayto poy tha antikastasteisei
```

```
 (seq_t_member ?next_char_t,?next_pos_t)

;oi dyo prohgoymenoi xarakthres apo aytoys ths antikatstashs prepei
;na einai oi idioi,to idio kai oi prohgoymenes theseis
 (= ?prev_char_s,?prev_char_t)(= ?prev_pos_s,?prev_pos_t)

;oi epomenoi xarakthres prepei na einai idioi,to idio kai oi theseis toys
(= ?next_char_s,?next_char_t)(= ?next_pos_s,?next_pos_t)

;epomenh thesh apo thn ?pos_in_t einai h ?next_pos_t
 (next_position  ?pos_in_t,?next_pos_t)

 ;epomenh thesh apo thn ?pos_in_s einai h ?next_pos_s
  (next_position  ?pos_in_s,?next_pos_s)

;epomenh thesh apo thn ?prev_pos_s einai h ?pos_in_s
(next_position  ?prev_pos_s, ?pos_in_s)

;epomenh thesh apo thn ?pos_in_t einai h ?prev_pos_t
(next_position  ?prev_pos_t, ?pos_in_t)
              )
```

Σχήμα 4.4: Προαπαιτήσεις τελεστή replacement

Στο Σχήμα 4.5 που ακολουθεί παρουσιάζονται τα αποτελέσματα της δράσης replacement. Μετά την εκτέλεση της δράσης το μέλος της ακολουθίας s έχει πλέον αντικατασταθεί με το αντίστοιχο του της t και δεν ανήκει στην s ακολουθία.

```
:effect (and
                ;den isxyei pleon to kathgorhma ?char_s sthn thesh ?pos_in_s
                 (not(seq_s_member ?char_s,?pos_in_s))
```

```
            ;sthn thesh poy htan prwta to ?char_s twra einai to ?char_t
            (seq_s_member ?char_t,?pos_in_s)
    )
```

Σχήμα 4.5:Αποτελέσματα δράσης replacement

Ακολούθως θα εξετάσουμε την δράση final_position_insertion μέσον την οποίας γίνεται εισαγωγή στην τελευταία θέση της s ακολουθίας, της αντίστοιχης θέσης στην t ακολουθία. Βέβαια αυτό θα συμβεί υπό την προϋπόθεση ότι η τελευταία θέση στην s είναι κενή(αυτό προέκυψε από κάποια διαγραφή στοιχείου που έγινε στην ακολουθία).

Στο Σχήμα 4.6 παρουσιάζονται το πρώτο μέρος του τελεστή final_position_insertion (όνομα και παράμετροι της δράσης).Σαν παραμέτρους αυτή η δράση παίρνει τον τελευταίο χαρακτήρα και την θέση του για την ακολουθία t και την τελευταία θέση για την ακολουθία s.

```
(:action final_position_insertion
    :parameters(
            ;teleytaios xarakthras  kai h thesh toy sthn s
            ?pos_in_s -seq_pos,

            ;teleytaios xarakthras  kai h thesh toy sthn t
            ?char_t -seq_char, ?pos_in_t -seq_pos,

                )
```

Σχήμα 4.6:Δήλωση ονόματος και παραμέτρων της δράσης final_position_insertion

Προαπαιτήσεις της δράσης αυτής είναι η τελευταία θέση της ακολουθίας s να είναι κενή και να υπάρχει η αντίστοιχη τελευταία θέση στο γονιδίωμα. Επίσης να έχουν το ίδιο μήκος οι ακολουθίες.

```
 :precondition (and    ;proapaithseis drashs
                ;na einai melh ths akoloythias t to   ?char_t,?pos_in_t
                (seq_t_member ?char_t,?pos_in_t)
```

```
                ;h  pos_in_s na einai h teleytaia thesh ths s akoloythias
                 (final_position_s ?pos_in_s),

                 ;h pos_in_t na einai h teleytaia thesh ths akoloythias
                 (final_position_t ?pos_in_t),

                ;na exoyn to idio mhkos oi akoloythies
                 (=?pos_in_s,?pos_in_t),

                ;h teleytaia thesh toy s na einai adeia
                 (empty_position ?pos_in_s)
        )
```

Σχήμα 4.7:Προαπαιτήσεις της δράσης final_position_insertion

Σαν αποτέλεσμα της δράσης αυτής (Σχήμα 4.8) η τελευταία θέση της ακολουθίας s δεν είναι κενή, αλλά τοποθετήται σε αυτήν ο χαρακτήρας της τελευταίας θέσης της ακολουθίας t (μέλος πλέον της ακολουθίας s).

```
:effect (and

            ;sthn thesh pleon  ?pos_in_s yparxei to ?char_t
             (seq_s_member ?char_t,?pos_in_s)

             ;h teleytaia thesh den einai pleon adeia
             (not (empty_position  ?pos_in_s))

    )
```

Σχήμα 4.8: Αποτελέσματα της δράσης final_position_insertion

Μέσον της δράσης first_replacment αντικαταστείται ο πρώτος χαρακτήρας της ακολουθίας s με τον αντίστοιχο του της ακολουθίας t,υπό την προϋπόθεση το αμέσως επόμενα στοιχεία και στις δυο ακολουθίες να είναι τα ίδια ενώ τα πρώτα στοιχεία διαφορετικά. Αυτή η λειτουργία έπρεπε να υλοποιηθεί σαν μια ξεχωριστή δράση για τον λόγο ότι με την δράση αντικατάστασης και διαγραφής για κάθε στοιχείο που θα διαγραφεί ή θα αντικατασταθεί κοιτάζουμε το αμέσως προηγούμενο του και το αμέσως επόμενο του και στις δύο ακολουθίες.Σε αυτή την περίπτωση που τα πρώτα στοιχεία των δύο ακολουθιών δεν είναι τα ίδια

δεν θα μπορούσε να εφαρμοστεί ούτε η διαγραφή ούτε η αντικατάσταση οπότε χρειαζόμαστε την δημιουργία μιας νέας δράσης.

Στο πιο κάτω, στο Σχήμα 4.9παρουσίαζεται το πρώτο τμήμα ορισμού της δράσης(όνομα και παραμέτροι).Παραμέτροι αυτής της δράσης είναι το πρώτο στοιχείο και των δύο ακολουθιών και το αμέσως επόμενο του αντίστοιχα.

```
(:action first_replacment
        :parameters(

                ;xarakthras prwths thsehs sthn s
                ?char_s -seq_char,?pos_in_s -seq_pos,

                ;xarakthras  prwths theshs t
                 ?char_t -seq_char, ?pos_in_t -seq_pos,

               ;epomenos xarakthras kai thesh toy sthn s
                 ?next_char_s -seq_char,?next_pos_s -seq_pos,

               ;epomenos xarakthras kai thseh toy sthn t
                 ?next_char_t -seq_char,?next_pos_t -seq_pos,

                )
```

Σχήμα 4.9:Πρώτο τμήμα ορισμού της δράσης(όνομα και παραμέτροι).

Προϋποθέσεις της δράσης όπως φαίνονται και στο Σχήμα 4.10 είναι τα πρώτα στοιχεία να είναι μέλη των αντιστοίχων ακολουθιών ,να μην είναι τα ίδια αλλά βέβαια να έχουν την ίδια θέση και το αμέσως επόμενο τους στοιχείο να ανήκει στην αντίστοιχη ακολουθία και να είναι το ίδιο.

```
  :precondition (and

                ;na einai melon ths akoloythias
                (seq_s_member ?char_s,?pos_in_s)
                (seq_t_member ?char_t,?pos_in_t)
```

```
;prwth thseh ths akoloythias s
(first_position_s ?pos_in_s)

;prwth thesh akolythias t
(first_position_t ?pos_in_t)

;na exoyn thn idia prwth thesh  alla oi xarakthtres na mhn einai
;idioi
(= ?pos_in_s,?pos_in_t)(not (= ?char_s,?char_t))

;na yparxei h epomenh thesh
(seq_s_member ?next_char_s,?next_pos_s)
(seq_t_member ?next_char_t,?next_pos_t)

;epomenh thesh ths  ?pos_in_t einai h ?next_pos_t sthn t
(next_position ?pos_in_t,?next_pos_t)

;epomenh thesh ths  ?pos_in_s einai h ?next_pos_s sthn s
(next_position ?pos_in_s,?next_pos_s)

;epomenes theseis prepei na einai idies kai
;ws xarakthres kai ws  theseis
(= ?next_char_s,?next_char_t)(= ?next_pos_s,?next_pos_t)

   )
```

Σχήμα 4.10: Προαπαιτήσεις της δράσης first_replacment

Ως αποτέλεσμα της δράσης αυτής (Σχήμα 4.11), το πρώτο στοιχείο της s δεν είναι πλέον μέλος της ακολουθίας.Την θέση του παίρνει ο πρώτος χαρακτήρας της ακολουθίας t.

```
:effect (and
            ;pleon to ?char_t,?pos_in_s einai melos ths akolythias
            (seq_s_member ?char_t,?pos_in_s)

            ;pleon   ?char_s,?pos_in_s den einai melos ths akoloythias
            (not (seq_s_member ?char_s,?pos_in_s) )

             )
```

Σχήμα 4.11:Αποτελέσματα δράσης first_replacment

Μέσον της δράσης διαγραφής αφαιρείται από την ακολουθία κάποιος χαρακτήρας της ακολουθίας (βάση κάποιων κριτηρίων που θα αναφερθούν αργότερα) και η αμέσως επόμενη θέση παίρνει την θέση της διαγραμμένης και γίνεται αυτή κενή.

Η δράση αυτή (deletion) δέχεται σαν παραμέτρους το στοιχείο που θα διαγραφεί (χαρακτήρα και θέση),το αμέσως επόμενο και το αμέσως προηγούμενο του (από την s) και τα αντίστοιχα τους στοιχεία από την ακολουθία στόχος t. Ολόκληρος ο κώδικας της δράσης παρουσιάζεται αναλυτικά στο Παράρτημα Γ.

Προαπαιτήσεις τις δράσης είναι ο χαρακτήρας που θα διαγραφή(να μην ίδιος με τον αντίστοιχο του), ο επόμενος και ο προηγούμενος του να είναι μέλη της ακολουθίας s (και οι αντίστοιχοι τους να είναι μέλη της t ακολουθίας).Ακόμα ο επόμενος αυτού που θα διαγραφεί να μην είναι ίδιος με τον αντίστοιχο του,αλλά να είναι ίδιος με τον επόμενο αυτού που θα διαγραφή (ακολουθία s).

Τα αποτελέσματα της δράσης διαγραφής παρουσιάζονται στο Σχήμα 4.12. Γίνεται η διαγραφή ενός χαρακτήρα,δηλαδή στην θέση του χαρακτήρα που διαγράφηκε τοποθετήται πλέον ο αμέσως επόμενος του χαρακτήρας και η θέση του πλέον γίνεται κενή.

```
:effect      (and      ;to ?char_s,?pos_in_s den einai melos ths akoloythias
                       (not (seq_s_member ?char_s,?pos_in_s))
                       (not (seq_s_member ?next_char_s,?next_pos_s))

                        ;metakinhsh pros dexia ths apomenhs theshs apo ayth poy
                        ;diagrafhke
                        (seq_s_member ?next_char_s,?pos_in_s)

                        ;h epomenh thesh  pleon einai kenh
                        (empty_position ?next_pos_s)

    )  ;closing effects
```

Σχήμα 4.12: Αποτελέσματα δράσης διαγραφής

Μέσον της δράσης shift γίνεται η μετακίνηση των στοιχείων που ακολουθούν κάποιο ζεύγος(από αμέσως γειτονικά στοιχεία της ακολουθίας s) όπου, το πρώτο στοιχείο έχει διαγραφή και στην θέση του μπαίνει το αμέσως επόμενο στοιχείο μένοντας το ίδιο κενό.Αυτή η μετακίνηση της κενής θέσεις γίνεται ανά ζεύγη των δύο.

Σαν παραμέτρους (Σχήμα 4.13), αυτή η δράση δέχεται την θέση που διαγράφηκε και αντικαταστάθηκε από την επόμενη της και την επόμενη της διαγραμμένης που τώρα είναι κενή(προαπαιτήσεις).

```
(:action shift
      :parameters(    ;dhlwsh parametrwn
                   ?next_pos_s  -seq_pos,
                   ?pos_in_s  -seq_pos
                )   ;telos  dhlwshs parametrwn
```

```
:precondition   (and
                    (next_position ?pos_in_s,?next_pos_s)

                     ;h epomenh thesh  pleon einai kenh
                     (empty_position ?next_pos_s)
                    )
```

Σχήμα 4.13:Παραμέτροι και προαπαιτήσεις της δράσης shift

Σαν αποτέλεσμα της δράσης(Σχήμα 4.14),παίρνουμε ανά δύο(πρέπει να είναι αμέσως γειτονικά και να μην είναι ίδια), τα υπόλοιπα στοιχεία της ακολουθίας(να έχουν μεγαλύτερη θέση από τις δύο θέσεις που έχουν ήδη υποστεί διαγραφή) και μετακινούμε το δεύτερο στοιχείο του ζεύγους στο πρώτο και το δεύτερο γίνεται κενό και δεν ανήκει στην ακολουθία.

```
:effect
        (forall  (?epomeno1  ?epomeno2  -seq_pos  ?ch2  -seq_char)
                (when (and
                    ;h ?epomeno2 einai h epomenh thesh apo  ?epomeno1
                    (next_position ?epomeno1,?epomeno2)

                    ;to ?ch2,?epomeno2 einai stoixeio ths akoloythias
                    (seq_s_member ?ch2,?epomeno2)

                     (not(= ?epomeno1,?pos_in_s))
                     (not(= ?epomeno2,?pos_in_s))

                     ;h thesi ?epomeno2 einai megalyterh
                     ;apo thn ?pos_in_s
                     (greater_pos ?epomeno2,?pos_in_s)

                      ;h thesi ?epomeno2 einai megalyterh
                      ;apo thn ?epomeno1
                      (greater_pos  ?epomeno2,?epomeno1)
```

```
                ;h thesh ?ch1,?epomeno1 prepei na
                ;einai kenh
                (empty_position ?epomeno1)

                ;oi dyo theseis poy pairnoyme prepei
                ;na einai diaforetikes
                (not(=?epomeno1,?epomeno2))
                )

            ;an isxyei h synthikh tote
            (and
                ;metakinhsh toy ?ch2s xarakthra
                (seq_s_member ?ch2,?epomeno1)

                (not (empty_position ?epomeno1))

                ;to keno metaferetai dexiotera
                (empty_position  ?epomeno2)

                ;to ?ch2,?epomeno2 anhkei sthn akoloythia alla einai keno
                (not (seq_s_member ?ch2,?epomeno2))

                )
            )
        )
```

Σχήμα 4.14:Αποτελέσματα της δράσης shift

Εδώ τελειώνει ο ορισμός του κόσμου δυαδικής ευθυγράμμισης ακολουθίας.

Στο Παράρτημα Γ παρουσιάζεται το πρόβλημα P_S_A_1,το οποίο αφορά ευθυγράμμιση των ακολουθιών s=AGCACACA και t=ACACAC TA (χαρακτήρες ανήκουν στο αλφάβητο το DNA). Σαν αντικείμενα του προβλήματος δηλώνουμε τις θέσεις και τους χαρακτήρες των αντικειμένων.Στην αρχική κατάσταση δηλώνουμε τα μέλη και των δύο ακολουθιών,κτίζουμε την κάθε ακολουθία με τον να καθορίζουμε ποια θέση είναι μετά από ποιά,ποιές είναι μεγαλύτερες από ποιες και ποιος χαρακτήρας είναι μετά από ποιο και πριν από ποιό.Επίσης καθορίζεται ποια είναι η πρώτη και ποια είναι η τελευταία θέση της ακολουθίας.Σαν στόχος σε αυτό το πρόβλημα είναι να η ακολουθία s να έχει τα ίδια στοιχεία και να είναι με την ίδια σειρά με την ακολουθία t.

Στο Σχήμα 4.15 παρουσιάζεται η λύση που δίνει ο ff-linux planner για το συγκεκριμένο πρόβλημα.

0: DELETION G ONE C ONE A ZERO A ZERO C TWO A TWO
1: SHIFT TWO ONE
2: SHIFT THREE TWO
3: SHIFT FOUR THREE
4: SHIFT FIVE FOUR
5: SHIFT SIX FIVE
6: FINAL_POSITION_INSERTION SEVEN A SEVEN
7: REPLACEMENT A SIX T SIX C FIVE C FIVE A SEVEN A SEVEN

Σχήμα 4.15: Λύση που δίνει ο ff-linux planner για το πρόβλημα.

Κεφάλαιο 5

Συμπεράσματα

Γενικά και τα τρία προβλήματα που μελετήθηκαν και μοντελοποιήθηκαν,είχαν μεγάλο ενδιαφέρον καθώς οι λειτουργίες που εκτελούνται μέσον τους έχουν μεγάλο αντίκτυπο για ένα οργανισμό.Μέσα από απλές σχετικά,μοντελοποιήσεις μπορούμε να πάμε από το γονιδίωμα ενός οργανισμού σε κάποιο άλλο(πρόβλημα αναδιοργάνωσης γονιδιώματος),να βρούμε μια σειρά από βιοχημικές αντιδράσεις που διαδρματίζουν πάρα πολύ σημαντικό ρόλο στην κανονική λειτουργία κάποιου οργανισμού (Πρόβλημα Βιοχημικών Μονοπατιών) και τέλος μπορούμε να ταυποιήσουμε δύο μονόκλωνες αλυσίδες από DNA,πρωτείνες κ.τ.λ.(Πρόβλημα Δυαδικής Ευθυγράμμισης Ακολουθιών).

Τα εργαλεία που χρησιμοποιήθηκαν για την εύρεση λύσεων ήταν ικανοποιητικά(SatPlan και ff-linux Planners).Βέβαια υπήρξε κάποια δυσκολία,όσον αφορά την αποσφαλματοποίηση (debugging),όπου δεν παρέχεται αρκετή βοήθεια από τους Planners,αλλά αυτό οφείλεται στην φύση του προγραμματισμού δράσης.

Όσον αφορά το Πρόβλημα Βιοχημικών Μονοπατιών, στην μοντελοποίηση που έγινε προσπαθήσαμε να απλοποιήσουμε το υπάρχον μοντέλο (ήταν γραμμένο σε adl και γράφτηκε σε strips) και αυτό επιτεύχθηκε.Η strips όντας πιο περιοριστική γλώσσα από την adl,μας έδωσε την δυνατότητα να εκφράσουμε απλούστερα πιο πολύπλοκές διαδικασίες(έγινε χρήση μόνο δύο βασικών τελεστών σύζευξης και άρνησης).Αυτό είχε σαν αποτέλεσμα να βρεθεί το πλάνο δράσης πιο γρήγορα,όμως με το κόστος του ότι μπορεί να παραχθούν πλάνα με περισσότερα βήματα (λόγο του ότι αυξήθηκε ο αριθμός των δράσεων).

Σχετικά με το πρόβλημα της Αναδιοργάνωσης Γονιδιώματος,εμείς μοντελοποιήσαμε την πιο απλή έκδοση του (όχι την έυρεση της πιο σύντομης λύσης).Προσπαθήσαμε να κρατήσουμε το σκεπτικό που ακολουθήθηκε στο άθρο των Erdem Esra, Tillier Elisabeth [2]. Προσθέτοντας – τροποποιόντας

κάποια κατηγορήματα και δράσεις θελήσαμε να απλοποιήσουμε ως ένα σημείο τις δράσεις και τα καταφέραμε.Αυτό βέβαια μπορεί να μας κόστισε στο ότι παράχθηκαν μεγαλύτερα πλάνα αλλά δεν μας απασχολεί στο παρόν στάδιο αφού δεν ψάχναμε την βέλτιστη λύση.

Για το πρόβλημα της Δυαδικής Ευθυγράμμισης Ακολουθίας, δεν έγινε μελέτη κάποιου συγκεκριμένου μοντέλου αλλά μια γενική μελέτη του προβλήματος.Στο μοντέλο που υλοποιήθηκε μέσα από δική μας εμπειρεία δημιουργήθηκαν οι δράσεις του κόσμου.Προσπαθήσαμε όσον το δυνατόν να είναι πιο απλές(για να τρέχουν πιο γρήγορα στους planners).Για μεγάλο μέγεθος ακολουθιών,αν μετά από κάποια διαγραφή πρέπει να μετακινηθούν πολλά κενά,αυτό θα μας στοιχήσει στο ότι θα παραχθούν πολύ μεγάλα πλάνα(ανα δύο στοιχεία κινούνται τα κενά κάθε φορά που εκτελείτε η δράση).

Βιβλιογραφία

[1] Dimopoulos Yannis,Gerevini Alfonso,Haslum Patric, Saeti Alesandro:The Benchmark Domains of the Deterministic Part of IPC-5, in Abstract Booklet of the competing planners of the Fifth International Planning Competition - Satellite Event of the Sixteenth International Conference on Automated Planning & Scheduling (ICAPS-06), pp. 14-19, 2006.

[2] Erdem Esra, Tillier Elisabeth: Genome Rearrangement and Planning, AAAI 2005: 1139-1144.

[3] Geffner Hector: Functional STRIPS: A more flexible language for planning and problem solving. In Logic-Based Artificial Intelligence, Jack Minker (Ed.), Kluwer, 2000. Earlier version presented at Logic-based AI Workshop, Washington D.C., June 1999.

[4] Ghallab Malik, Howe Adele, Knoblock Craig, McDermott Drew, Ram Ashwin, Veloso Manuela, Weld Daniel, Wilkins David: PDDL - The Planning Domain Definition Language Version 1.2. Technical Report CVC TR-98-003, Yale Center for Computational Vision and Control, 1998.

[5] Gerevini Alfonso and Long Derek : BNF Description of PDDL3.0. Unpublished manuscript linked from the IPC-5 website, 2005.

[6] Gerevini Alfonso and Long Derek: Plan Constraints and Preferences in PDDL3 The Language of the Fifth International Planning Competition. Technical Report R. T. 2005-08-47, Dipartimento di Elettronica per l'Automazione, Università degli Studi di Brescia, 2005.

[7] Norvig Peter and Russell Stuart : Artificial Intelligence: A Modern Approach, 2nd Edition,Prentice Hall,2003.

[8] Wheeler David and Robert Giegerich: Pairwise Sequence Alignment, version 2.01, May 21, 1996. Available on internet: http://www.techfak.uni-bielefeld.de/bcd/Curric/PrwAli/prwali.html

[9] http://www.ida.liu.se/~TDDA13/labbar/planning/2003/writing.html

[10] http://en.wikipedia.org/wiki/STRIPS

Παράρτημα Α

Σε αυτό το παράρτημα παρουσιάζονται οι κώδικες του κόσμου και του προβλημάτος του Κεφαλαίου 2 (Βιοχημικά μονοπάτια).

Pathways Propositional - Ορισμός κόσμου στο adl υποσύνολο της pddl

```
(define (domain Pathways-Propositional)
(:requirements :typing :adl)

(:types level molecule - object
  simple complex -molecule)

(:constants c-Myc-Max cycDp1 p107-E2F4-DP12p1 p107-E2F4-DP12p1-gE2 -
complex)

(:predicates
        (association-reaction ?x1 ?x2 -molecule ?x3 -complex)
        (catalyzed-association-reaction ?x1 ?x2 -molecule ?x3 -complex)
        (synthesis-reaction ?x1 ?x2 - molecule)
        (possible ?x - molecule)
        (available ?x - molecule)
        (chosen ?s - simple)
        (next ?l1 ?l2 - level)
        (num-subs ?l1 -level)
        (goal1)
        (goal2))

  (:action choose
    :parameters (?x - simple ?l1 ?l2 - level)
    :precondition (and (possible ?x) (not (chosen ?x))
                       (num-subs ?l2) (next ?l1 ?l2))
    :effect (and (chosen ?x) (not (num-subs ?l2)) (num-subs ?l1)))
```

```
(:action initialize
   :parameters (?x - simple)
   :precondition (and (chosen ?x))
   :effect (and (available ?x)))


    (:action associate
      :parameters  (?x1 ?x2 - molecule ?x3 - complex)
     :precondition (and (association-reaction ?x1 ?x2 ?x3)
                        (available ?x1) (available ?x2))
     :effect (and (not (available ?x1)) (not (available ?x2)) (available ?x3)))


   (:action associate-with-catalyze
    :parameters  (?x1 ?x2 - molecule ?x3 - complex)
    :precondition (and (catalyzed-association-reaction ?x1 ?x2 ?x3)
                       (available ?x1) (available ?x2))
    :effect (and (not (available ?x1)) (available ?x3)))


 (:action synthesize
  :parameters (?x1 ?x2 - molecule)
  :precondition (and (synthesis-reaction ?x1 ?x2) (available ?x1))
  :effect (and (available ?x2)))


(:action DUMMY-ACTION-1
 :parameters ()
 :precondition (or (available p107-E2F4-DP12p1-gE2)
                   (available p107-E2F4-DP12p1))
 :effect (and (goal1)))
```

```
 (:action DUMMY-ACTION-2
 :parameters ()
 :precondition (or (available cycDp1)
                   (available c-Myc-Max))
 :effect(and (goal2)))
)
```

Pathways Propositional - Ορισμός κόσμου στο strips υποσύνολο της pddl

```
(define (domain Pathways-Propositional-2)
        (:requirements :strips)
        (:predicates
                (level ?l)
                (molecule ?m)
                 (simple ?s)
                    (complex ?c)
                    (association-reaction ?x1_m ?x2_m ?x3_c)
(catalyzed-association-reaction ?x1_m ?x2_m ?x3_c)
(synthesis-reaction ?x1_m ?x2_m)
                     (possible ?x_m)
                     (available ?x_m)
                     (chosen ?s_s)
                     (next ?l1_l ?l2_l)
                      (num-subs ?l1_l)
                     (goal1)
                     (goal2))

(:action choose
 :parameters (?x_s ?l1_l ?l2_l)
:precondition (and (possible ?x_s)(simple ?x_s)(level ?l1_l)
                   (level ?l2_l)(num-subs ?l2_l)
                   (num-subs ?l2_l) (next ?l1_l ?l2_l))
```

```
:effect (and (chosen ?x_s) (not (num-subs ?l2_l)) (num-subs ?l1_l)))


(:action initialize
:parameters (?x_s)
:precondition (and (chosen ?x_s)(simple ?x_s))
 :effect (and (available ?x_s)))
(:action associate
:parameters  (?x1_m ?x2_m ?x3_c)
:precondition (and (association-reaction ?x1_m ?x2_m ?x3_c)
                        (molecule ?x1_m)(molecule ?x2_m)(complex ?x3_c)
                        (available ?x1_m) (available ?x2_m))
:effect (and (not (available ?x1_m)) (not (available ?x2_m)) (available ?x3_c)))

(:action associate-with-catalyze
:parameters  (?x1_m ?x2_m ?x3_c)
:precondition (and (catalyzed-association-reaction ?x1_m ?x2_m ?x3_c)
                        (molecule ?x1_m)(molecule ?x2_m)(complex ?x3_c)
                        (available ?x1_m) (available ?x2_m))
 :effect (and (not (available ?x1_m)) (available ?x3_c)))


 (:action synthesize
 :parameters (?x1_m ?x2_m)
 :precondition (and (molecule ?x1_m)(molecule ?x2_m)
                         (synthesis-reaction ?x1_m ?x2_m)(available ?x1_m))
:effect (and (available ?x2_m)))


(:action DUMMY-ACTION-1a
 :parameters ()
 :precondition (and(available p107-E2F4-DP12p1-gE2)
                        (available p107-E2F4-DP12p1))
```

```
 :effect (and (goal1)))


(:action DUMMY-ACTION-1b
 :parameters ()
 :precondition (available p107-E2F4-DP12p1-gE2)
 :effect (and (goal1)))


 (:action DUMMY-ACTION-1c
 :parameters ()
 :precondition (available p107-E2F4-DP12p1)
 :effect (and (goal1)))

(:action DUMMY-ACTION-2a
 :parameters ()
 :precondition (and (available cycDp1)
                    (available c-Myc-Max))
 :effect(and (goal2)))


(:action DUMMY-ACTION-2b
 :parameters ()
 :precondition (available cycDp1)
 :effect(and (goal2)))


(:action DUMMY-ACTION-2c
 :parameters ()
 :precondition (available c-Myc-Max)
 :effect(and (goal2)))    )
```

Pathways Propositional Problem - Ορισμός προβλήματος σε pddl

```
(define (problem Pathways-02)
(:domain IPC5b)
(:objects
        p53p1 - simple
        p130 - simple
        Max - simple
        HDAC1-pRbp1-E2F4-DP12 - simple
          HDAC1-pRbp1-E2F13-DP12 - simple
          HDAC1-p130-E2F4p1-DP12 - simple
          HBP1 - simple
          gE2 - simple
          E2F6-DP12p1 - simple
          E2F4-DP12p1 - simple
          E2F13p1-DP12 - simple
          cdk1p1p2 - simple
          cdk1p1p2-Gadd45 - complex
          c-Myc-Max - complex
          E2F13p1-DP12-gE2 - complex
          E2F6-DP12p1-gE2 - complex
          HBP1-p130  - complex
          HDAC1-p130 - complex
         HDAC1-p130-E2F4p1-DP12-gE2 - complex
          HDAC1-pRbp1-E2F13-DP12-gE2 - complex
          HDAC1-pRbp1-E2F4-DP12-gE2  - complex
          Mdm2-E2F13p1-DP12 - complex
          p107-E2F4-DP12p1 - complex
          p130-E2F4-DP12p1-gE2 - complex
          p130-E2F4-DP12p1 - complex
          p21-Gadd45 - complex
          Mdm2 - complex
          Gadd45 - complex
```

p21 - complex
c-Fos - complex
c-Myc - complex
cycA - complex
cycD - complex
cycDp1 - complex
cycE - complex
cycEp1 - complex
p19ARF - complex
po1 - complex
p107p1 - complex
p107 - complex
10 -level
11 -level
12 -level
13 -level)

(:init
(possible p53p1)
(possible p130)
(possible Max)
(possible HDAC1-pRbp1-E2F4-DP12)
(possible HDAC1-pRbp1-E2F13-DP12)
(possible HBp1)
(possible gE2)
(possible E2F6-DP12p1)
(possible E2F4-DP12p1)
(possible E2F13p1-DP12)
(possible cdk1p1p2)
(association-reaction cdk1p1p2 Gadd45 cdk1p1p2-Gadd45)
(association-reaction c-Myc Max c-Myc-Max)
(synthesis-reaction E2F13p1-DP12-gE2 c-Myc)
(synthesis-reaction E2F13p1-DP12-gE2 cycA)

(synthesis-reaction E2F13p1-DP12-gE2 cycD)
(synthesis-reaction E2F13p1-DP12-gE2 cycDp1)
(synthesis-reaction E2F13p1-DP12-gE2 cycE)
(synthesis-reaction E2F13p1-DP12-gE2 cycEp1)
(association-reaction E2F13p1-DP12 gE2 E2F13p1-DP12-gE2)
(synthesis-reaction E2F13p1-DP12-gE2 p107)
(synthesis-reaction E2F13p1-DP12-gE2 p107p1)
(synthesis-reaction E2F13p1-DP12-gE2 p19ARF)
(synthesis-reaction E2F13p1-DP12-gE2 po1)
(association-reaction E2F6-DP12p1 gE2 E2F6-DP12p1-gE2)
(association-reaction HBP1 p130 HBP1-p130)
(association-reaction HDAC1-p130-E2F4p1-DP12 gE2 HDAC1-p130-E2F4p1-DP12-gE2)
(association-reaction HDAC1-pRbp1-E2F4-DP12 gE2 HDAC1-pRbp1-E2F4-DP12-gE2)
(association-reaction Mdm2 E2F13p1-DP12 Mdm2-E2F13p1-DP12)
(association-reaction p107-E2F4-DP12p1 gE2 p107-E2F4-DP12p1-gE2)
(association-reaction p107 E2F4-DP12p1 p107-E2F4-DP12p1)
(association-reaction p130-E2F4-DP12p1 gE2 p130-E2F4-DP12p1-gE2)
(association-reaction p130 E2F4-DP12p1 p130-E2F4-DP12p1)
(association-reaction p21 Gadd45 p21-Gadd45)
(synthesis-reaction p53p1 c-Fos)
(synthesis-reaction p53p1 Gadd45)
(synthesis-reaction p53p1 Mdm2)
(synthesis-reaction p53p1 p21)
(num-subs 10)
(next 11 10)
(next 12 11)
(next 13 12))

(:goal
(and

```
    (goal1)
    (goal2)))
)
```

Pathways Propositional Problem - Ορισμός προβλήματος στο strips υποσύνολο της pddl

```
 (define (problem Pathways-Propositional-pr2)
(:domain Pathways-Propositional-2)
(:objects
      p107-E2F4-DP12p1-gE2
      p53p1
      p130
      Max
        HDAC1-pRbp1-E2F4-DP12
        HDAC1-pRbp1-E2F13-DP12
        HDAC1-p130-E2F4p1-DP12
        HBP1
        gE2
        E2F6-DP12p1
        E2F4-DP12p1
        E2F13p1-DP12
        cdk1p1p2
        cdk1p1p2-Gadd45
        c-Myc-Max
        E2F13p1-DP12-gE2
        E2F6-DP12p1-gE2
        HBP1-p130
        HDAC1-p130
        HDAC1-p130-E2F4p1-DP12-gE2
        HDAC1-pRbp1-E2F13-DP12-gE2
        HDAC1-pRbp1-E2F4-DP12-gE2
        Mdm2-E2F13p1-DP12
```

p107-E2F4-DP12p1
p130-E2F4-DP12p1-gE2
p130-E2F4-DP12p1
p21-Gadd45
Mdm2
Gadd45
p21
c-Fos
c-Myc
cycA
cycD
cycDp1
cycE
cycEp1
p19ARF
po1
p107p1
p107
10
11
12
13)

(:init
(molecule p53p1)
(molecule p130)
(molecule Max)
(molecule HDAC1-pRbp1-E2F4-DP12)
(molecule HDAC1-pRbp1-E2F13-DP12)
(molecule HDAC1-p130-E2F4p1-DP12)
(molecule HBP1)
(molecule gE2)
(molecule E2F6-DP12p1)

(molecule E2F4-DP12p1)
(molecule E2F13p1-DP12)
(molecule cdk1p1p2)
(molecule c-Myc-Max)
(molecule cycDp1)
(molecule p107-E2F4-DP12p1)
(molecule p107-E2F4-DP12p1-gE2)
(molecule cdk1p1p2-Gadd45)
(molecule c-Myc-Max)
(molecule E2F13p1-DP12-gE2)
(molecule E2F6-DP12p1-gE2)
(molecule HBP1-p130)
(molecule HDAC1-p130)
(molecule HDAC1-p130-E2F4p1-DP12-gE2)
(molecule HDAC1-pRbp1-E2F13-DP12-gE2)
(molecule HDAC1-pRbp1-E2F4-DP12-gE2)
(molecule Mdm2-E2F13p1-DP12)
(molecule p107-E2F4-DP12p1)
(molecule p130-E2F4-DP12p1-gE2)
(molecule p130-E2F4-DP12p1)
(molecule p21-Gadd45)
(molecule Mdm2)
(molecule Gadd45)
(molecule p21)
(molecule c-Fos)
(molecule c-Myc)
(molecule cycA)
(molecule cycD)
(molecule cycDp1)
(molecule cycE)
(molecule cycEp1)
(molecule p19ARF)
(molecule po1)

(molecule p107p1)
(molecule p107)
(simple p53p1)
(simple p130)
(simple Max)
(simple HDAC1-pRbp1-E2F4-DP12)
(simple HDAC1-pRbp1-E2F13-DP12)
(simple HDAC1-p130-E2F4p1-DP12)
(simple HBP1)
(simple gE2)
(simple E2F6-DP12p1)
(simple E2F4-DP12p1)
(simple E2F13p1-DP12)
(simple cdk1p1p2)
(complex cdk1p1p2-Gadd45)
(complex c-Myc-Max)
(complex E2F13p1-DP12-gE2)
(complex E2F6-DP12p1-gE2)
(complex HBP1-p130)
(complex HDAC1-p130)
(complex HDAC1-p130-E2F4p1-DP12-gE2)
(complex HDAC1-pRbp1-E2F13-DP12-gE2)
(complex HDAC1-pRbp1-E2F4-DP12-gE2)
(complex Mdm2-E2F13p1-DP12)
(complex p107-E2F4-DP12p1)
(complex p130-E2F4-DP12p1-gE2)
(complex p130-E2F4-DP12p1)
(complex p21-Gadd45)
(complex Mdm2)
(complex Gadd45)
(complex p21)
(complex c-Fos)
(complex c-Myc)

(complex cycA)
(complex cycD)
(complex cycDp1)
(complex cycE)
(complex cycEp1)
(complex p19ARF)
(complex po1)
(complex p107p1)
(complex p107)
(complex c-Myc-Max)
(complex cycDp1)
(complex p107-E2F4-DP12p1)
(complex p107-E2F4-DP12p1-gE2)
(level 10)
(level 11)
(level 12)
(level 13)
(possible p53p1)
(possible p130)
(possible Max)
(possible HDAC1-pRbp1-E2F4-DP12)
(possible HDAC1-pRbp1-E2F13-DP12)
(possible HBp1)
(possible gE2)
(possible E2F6-DP12p1)
(possible E2F4-DP12p1)
(possible E2F13p1-DP12)
(possible cdk1p1p2)
(association-reaction cdk1p1p2 Gadd45 cdk1p1p2-Gadd45)
(association-reaction c-Myc Max c-Myc-Max)
(synthesis-reaction E2F13p1-DP12-gE2 c-Myc)
(synthesis-reaction E2F13p1-DP12-gE2 cycA)
(synthesis-reaction E2F13p1-DP12-gE2 cycD)

(synthesis-reaction E2F13p1-DP12-gE2 cycDp1)
(synthesis-reaction E2F13p1-DP12-gE2 cycE)
(synthesis-reaction E2F13p1-DP12-gE2 cycEp1)
(association-reaction E2F13p1-DP12 gE2 E2F13p1-DP12-gE2)
(synthesis-reaction E2F13p1-DP12-gE2 p107)
(synthesis-reaction E2F13p1-DP12-gE2 p107p1)
(synthesis-reaction E2F13p1-DP12-gE2 p19ARF)
(synthesis-reaction E2F13p1-DP12-gE2 po1)
(association-reaction E2F6-DP12p1 gE2 E2F6-DP12p1-gE2)
(association-reaction HBP1 p130 HBP1-p130)
(association-reaction HDAC1-p130-E2F4p1-DP12 gE2 HDAC1-p130-E2F4p1-DP12-gE2)
(association-reaction HDAC1-pRbp1-E2F4-DP12 gE2 HDAC1-pRbp1-E2F4-DP12-gE2)
(association-reaction Mdm2 E2F13p1-DP12 Mdm2-E2F13p1-DP12)
(association-reaction p107-E2F4-DP12p1 gE2 p107-E2F4-DP12p1-gE2)
(association-reaction p107 E2F4-DP12p1 p107-E2F4-DP12p1)
(association-reaction p130-E2F4-DP12p1 gE2 p130-E2F4-DP12p1-gE2)
(association-reaction p130 E2F4-DP12p1 p130-E2F4-DP12p1)
(association-reaction p21 Gadd45 p21-Gadd45)
(synthesis-reaction p53p1 c-Fos)
(synthesis-reaction p53p1 Gadd45)
(synthesis-reaction p53p1 Mdm2)
(synthesis-reaction p53p1 p21)
(num-subs 10)
(next 11 10)
(next 12 11)
(next 13 12))

(:goal
(and
(goal1)
(goal2))))

Παράρτημα Β

Σε αυτό το παράρτημα παρουσιάζονται οι κώδικες των κόσμων και των προβλημάτων που επιλύθηκαν στο Κεφάλαιο 3 (Genome Rearrangement).

Ορισμός Κόσμου Genome_rearr:

```
(define (domain Genome_rearr)              ;Orismos domain Genome_rearr
      (:requirements :adl)                 ;Apaithsh domain h glwssa adl
      (:types gene - object)               ;xrhsimopihte o typos gene
      (:predicates                         ;Kathgorhmata domain:


          (clockwise ?x ?z  -gene)         ;clockwise: to gonidio ?z
                                           ;brisketai amesws meta to gonidio
                                           ;?x (me thn fora twn diktwn
                                           ;toy rologioy)

            (in ?z ?x ?y -gene)            ;in: to gonidio ?z brisketai mesa
                                           ;sthn akoloythia gonidiwn poy
                                           ;arxizei apo ?x gonidio kai
                                           ;teleiwnei sto ?y gonidio


          (positive_label ?x -gene)        ;positive_label:antiprwswpeyei
                                           ;ton thetiko prosanatolismo
                                           ;toy gonidioy x

          (negative_label ?x -gene)        ;positive_label:antiprwswpeyei
                                           ;ton arnitiko prosanatolismo
                                           ;toy gonidioy x
      )
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                    ;
; action - transpose_single                          ;
;                                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(:action transpose_single
        :parameters(?x ?z -gene) ;parametroi: x,y arxh kai
                                 ;telos ths akoloythias
                                 ;z tha topothetithei meta apo
                                 ;ayto h akoloythia
                                 ;x1 gonidio poy brisketai mia
                                 ;thesi prin to x
                                 ;yi gonidio poy brisketai mia
                                 ;thesi meta to y
                                 ;gonidio poy brisketai mia
                                 ;thesi meta to z
        ;proapaithseis ths drashs
        :precondition (not (= ?x ?z))


       ;apotelesmata drashs
       :effect (and (forall  (?x1 ?x2 ?z1 -gene)
               (when (and (not (= ?x ?x1))
                          (not (= ?x ?x2))
                          (not (= ?z ?x1))
                          (not (= ?z ?x2))
                          (not (= ?z ?z1))
                          (clockwise ?x1 ?x)
                          (clockwise ?x ?x2)
                          (clockwise ?z ?z1))
                    (and  (clockwise ?x1 ?x2)
```

```
            (clockwise ?z ?x)
            (clockwise ?x ?z1)
          (not (clockwise ?x1 ?x))
          (not (clockwise ?x ?x2))
          (not (clockwise ?z ?z1))
     ) )
     )

     (forall  (?m -gene)
     (when (and (not (= ?x ?m))
             (not (= ?z ?m))
             (in ?m ?x ?z)
          )
          (and (not (in ?m ?x ?z)))
      )
      )

     (forall  (?m ?m1 -gene)
     (when (and (not (= ?x ?m))
             (not (= ?z ?m))
             (not (= ?x ?m1))
             (not (= ?z ?m1))
             (not (= ?m ?m1))
             (in ?m ?z ?m1)
          )
          (and  (in ?m ?x ?m1))
      )
      )

)
      )
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                      ;
```

```
; action - transpose                          ;
;                                             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(:action transpose
        :parameters(?x ?y ?z ?x1 ?y1 ?z1 -gene) ;parametroi: x,y arxh kai
                                        ;telos ths akoloythias
                                        ;z tha topothetithei meta apo
                                        ;ayto h akoloythia
                                        ;x1 gonidio poy brisketai mia
                                        ;thesi prin to x
                                        ;yi gonidio poy brisketai mia
                                        ;thesi meta to y
                                        ;gonidio poy brisketai mia
                                        ;thesi meta to z
        ;proapaithseis ths drashs
        :precondition (and  (not (= ?x ?y))
                        (not (= ?x ?z))
                        (not (= ?y ?z))
                        (not (= ?z ?z1))
                        (not (= ?x1 ?y1))
                        (not(= ?x1 ?z))
                        (not(= ?z1 ?x))
                        (not (= ?y1 ?x))
                        (not (= ?y ?x1))
                        (not (= ?z1 ?y1))
                        (not (clockwise ?z ?x))
                        (not (clockwise ?z ?y))
                        (not (clockwise ?x ?z))
                        (not (in ?z ?x ?y))
                        (clockwise ?x1 ?x)
                        (clockwise ?y ?y1)
                        (clockwise ?z ?z1)
```

```
        )

;apotelesmata drashs
:effect (
        and  (clockwise ?x1 ?y1)
            (clockwise ?z ?x)
            (clockwise ?y ?z1)
            (in ?x ?z ?z1)
            (in ?y ?z ?z1)
            (not (clockwise ?x1 ?x))
            (not (clockwise ?y ?y1))
            (not (clockwise ?z ?z1))

            (forall (?m -gene)
               (when
               (and (in ?m ?x ?y)
               (not (= ?m ?x))
               (not (= ?m ?y))
               (not (= ?m ?z))
               )
               (and  (in ?m ?z ?z1)
                    (not (in ?m ?x1 ?y1)))
               )
            )

            (forall  (?m -gene)
                 (when
                 (and (not (= ?x ?m))
                 (not (= ?z ?m))
                 (not (= ?y ?m))
                 (in ?m ?z ?y)
                 (not (in ?m ?x ?y))
                  )
```

```
                (and (not(in ?m ?z ?y))(not(in ?m ?z ?z1)) )


            )
            )

               (forall  (?m -gene)
                    (when
                    (and (not (= ?x ?m))
                    (not (= ?z ?m))
                    (not (= ?y ?m))
                    (in ?m ?y ?z)
                     )
                 (and (not(in ?m ?z ?y))(not(in ?m ?x ?y)))
            )
            )

            (forall  (?m ?m1 ?m2 -gene)
            (when (and (not (= ?x ?m))
                    (not (= ?z ?m))
                    (not (= ?x ?m1))
                    (not (= ?z ?m1))
                    (not (= ?m ?m1))
                    (not (in ?m1 ?x ?y))
                    (in ?z ?m1 ?m2)
                   (in ?m ?x ?y))

                 (in ?m ?m1 ?m2)
            )
            ) ) )
.....................................................................
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                   ;
; action - sign_negative                            ;
```

```
;                                                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(:action sign_negative

 :parameters (?x -gene)   ;parametroi: x,y arxh kai telos ths
                                  ;akoloythias
                                  ;x1 gonidio poy brisketai mia
                                  ;thesi prin to x
                                  ;yi gonidio poy brisketai mia
                                  ;thesi meta to y

 ;proapaithseis ths drashs
 :precondition (and
                 (negative_label ?x)
               )

 ;apotelesmata drashs
 :effect ( and
           (positive_label ?x)        ;h arxh kai to telos ths akoloythias
           (not (negative_label ?x))  ;allazoyn prosanatolismo apo arnhtiko se
        )
)



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                   ;
; action - sign_positive                            ;
;                                                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(:action sign_positive
 :parameters (?x -gene)

 ;proapaithseis ths drashs
 :precondition (and
                 (positive_label ?x)
               )



;apotelesmata drashs
:effect (and (not (positive_label ?x))   ;h arxh kai to telos ths akoloythias
            (negative_label ?x)          ;allazoyn prosanatolismo apo thetiko se
        )  )
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                              ;
; action - invert_multiple                     ;
;                                              ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(:action invert_mutliple
        :parameters(?x ?y -gene) ;parametroi: x,y arxh kai
                                 ;telos ths akoloythias
                                 ;z tha topothetithei meta apo
                                 ;ayto h akoloythia
                                 ;x1 gonidio poy brisketai mia
                                 ;thesi prin to x
                                 ;yi gonidio poy brisketai mia
                                 ;thesi meta to y
                                 ;gonidio poy brisketai mia
                                 ;thesi meta to z
        ;proapaithseis ths drashs
        :precondition (and  (not (= ?x ?y))
                      (not (clockwise ?x ?y))
```

```
            (not (clockwise ?y ?x))
        )

;apotelesmata drashs
:effect (and
            (forall (?m -gene)
            (when (clockwise ?m ?x)
            (and (clockwise ?m ?y)
                (not (clockwise ?m ?x)))
            )
            )

            (forall (?m -gene)
            (when (clockwise ?x ?m)
            (and (clockwise ?m ?x)
                (not (clockwise ?x ?m)))
            )
            )

            (forall (?m -gene)
            (when (clockwise ?m ?y)
            (and (clockwise ?y ?m)
                (not (clockwise ?m ?y)))
            )
            )


            (forall (?m -gene)
            (when (clockwise ?y ?m)
            (and (clockwise ?x ?m)
                (not (clockwise ?y ?m)))
            )
            )
```

```
(forall (?m ?m1 -gene)
   (when
   (and (in ?m ?x ?y)
        (in ?m1 ?x ?y)
        (clockwise ?m1 ?m)
   (not (= ?m ?x))
   (not (= ?m1 ?y))
   )
   (and  (clockwise ?m ?m1)
         (not (clockwise ?m1 ?m)))
 )
 )

(forall (?m ?m1 ?m2 -gene)
   (when
   (and (in ?m ?x ?y)
        (or (in ?m1 ?x ?y)
            (= ?m1 ?x)
        )
        (or (in ?m2 ?x ?y)
            (= ?m2 ?y)
        )
        (in ?m ?m1 ?m2)
        (not (= ?m ?x))
        (not (= ?m1 ?y))
        (not (= ?m ?m1))
        (not (= ?m ?m2))
        (not (= ?m1 ?m2))
   )
   (and  (in ?m ?m2 ?m1)
         (not (in ?m ?m1 ?m2)))
  )
```

```
    )


(forall (?m ?m1 -gene)
  (when
  (and (in ?m ?x ?m1)
        (in ?m1 ?x ?y)
        (not(=?m ?y))
        (not(= ?m1 ?y))
        (not(=?m ?m1))
    )
    (and (in ?m ?m1 ?x)
          (not (in ?m ?x ?m1)))
    )
    )



  (forall (?m ?m1 -gene)
  (when
  (and (in ?m ?m1 ?y)
        (in ?m1 ?x ?y)
        (not(=?m ?x))
        (not(= ?m1 ?y))
        (not(=?m ?m1))
    )
    (and (in ?m ?y ?m1)
          (not (in ?m ?m1 ?y)))
    )
    )

 (forall (?m  -gene)
      (when
```

```
                    (and (in ?m ?y ?x)
                         (not (= ?m ?x))
                         (not (= ?m ?y))
                    )
                    (and  (in ?m ?y ?x)
                          (not (in ?m ?x ?y)))
                   ))

)
)

)
```

Κώδικας προβλήματος G_R_01:

```
(define (problem G_R_01)        ;Orismos problhmatos G_R_01
(:domain Genome_rearr)          ;gia to domain Genome_rearr
(:objects one - gene            ;yparxoyn pente antikeimena
        two - gene              ;se ayto to problhma ola typoy gene
        five - gene
        four - gene
        three - gene)


(:init                  ;Kathgorhmata poy einai alhthh sthn arxikh katastash

       (clockwise one two)
       (positive_label one)
       (positive_label two)
         (clockwise two five)
         (negative_label five)
```

(clockwise five four)
(negative_label four)
(clockwise four three)
(negative_label three)
(clockwise three one)
(in two one five)
(in two one four)
(in two one three)
(in two three five)
(in two three four)
(in two four five)
(in five two four)
(in five two three)
(in five two one)
(in five three four)
(in five one four)
(in five one three)
(in four five three)
(in four five one)
(in four five two)
(in four two three)
(in four two one)
(in four one three)
(in three four two)
(in three four five)
(in three four one)
(in three five one)
(in three five two)
(in three two one)
(in one three two)
(in one four two)
(in one three five)
(in one four five)

```
        (in one three four)
        (in one five two)
)


 (:goal (and                    ;stoxos poy prepei na ekplhrwthei
            (clockwise one five)
            (positive_label one)
            (positive_label five)
            (clockwise five four)
            (negative_label four)
              (clockwise four three)
            (negative_label three)
              (clockwise three two)
            (negative_label two)
              (clockwise two one)
        )
 )


)
```

Παράρτημα Γ

Σε αυτό το παράρτημα παρουσιάζονται οι κώδικες των κόσμων και των προβλημάτων που επιλύθηκαν στο Κεφάλαιο 4 (Pairwise Sequence Alignment).

Ορισμός κόσμου:

```
(define (domain SEQ_ALIG)                    ;Orismos toy domain SEQ_ALIG

      (:requirements :adl)                    ;Apaithsh toy domain  adl

      (:types seq_char,seq_pos -object)       ;yparxoyn dyo typoi:
                                              ;seq_char typos xarakthras ths
akoloythias
                                              ;seq_pos typos thesh ths
akoloythias

      (:predicates          ;kathgorhmata domain:


      ;kathgorhma poy ypodhlwnei oti o xarakthras ?character einai melos ths
      ;akolouthias s kai  brisketai  sthn thesh  ?position_number

      (seq_s_member ?character - seq_char,?position_number - seq_pos),



      ;kathgorhma poy ypodhlwnei oti o xarakthras ?character einai melos ths
      ;akolouthias t kai  brisketai  sthn thesh  ?position_number

      (seq_t_member ?character -seq_char,?position_number -seq_pos)



       ;kathgorhma poy ypodhlwnei oti h thesh ?position_number1 einai
       ;megalyterh apo thn thesh  ?position_number2 sthn akoloythia

       (greater_pos  ?position_number1 -seq_pos, ?position_number2 -
seq_pos)



       ;kathgorhma poy ypodhlwnei ton arithmo thns teleytaias theshs sthn
       ;akoloythia s (h ?position_number einai h teleytai sthn s)
```

```
        (final_position_s ?position_number -seq_pos)


        ;kathgorhma poy ypodhlwnei oti h thesh  ?next_position_number
        ;(arithmos theshs) akribws meta  thn ?position_number thesh arithmitika

        (next_position ?position_number -seq_pos,?next_position_number -
seq_pos)


        ;kathgorhma poy ypodhlwnei oti mia thesh einai kenh

        (empty_position ?position_number -seq_pos)


       ;h ?position_number einai h teleytaia thesh ths akolouthias t

       (final_position_t  ?position_number -seq_pos)


       ;h ?position_number einai h prwth thesh ths akolouthias s

       (first_position_s ?position_number -seq_pos)


        ;h ?position_number einai h prwth thesh ths akolouthias s

       (first_position_t ?position_number -seq_pos)


        )    ;telos dhlwshs kathgorhmatwn domain




(:action replacement
       :parameters(       ;dhlwsh parametrwn

                     ;o xarakthras kai h thesi pou tha antikatastathoyn
                     ;sthn s akoloythia xarakthrwn
                     ?char_s -seq_char,?pos_in_s -seq_pos,


                     ;o antistoixos xarakthras aytoy poy tha
                     ;antiuaktastathei sthn akoloythia t
```

```
                ?char_t -seq_char, ?pos_in_t -seq_pos,


                 ;o amesws prohgoymenos xarakthras kai
                ;h prohgoumenh thesh(apo ayto poy tha antiaktastathei)
                ; sthn s akoloythia

                ?prev_char_s -seq_char,?prev_pos_s -seq_pos,


              ;o amesws prohgoymenos xarakthras kai
                ;h prohgoumenh thesh(apo ayto poy tha antiaktastathei)
                ; sthn t akoloythia
                ?prev_char_t -seq_char,?prev_pos_t -seq_pos,



                 ;o amesws epomenos xarakthras kai
                 ;h epomenh thesh(toy antistoixoy apo
                 ; ayto pou tha antikatastathei) sthn s akoloythia
                 ?next_char_s -seq_char,?next_pos_s -seq_pos

                   ?next_char_t -seq_char,?next_pos_t -seq_pos


            )




   :precondition (and      ;proapaithseis drashs

                    ;o xarakthras ?char_s sthn thesh ?pos_in_s
                    ;na einai melos ths akoloythias s
                   (seq_s_member ?char_s,?pos_in_s)


              ;o xarakthras ?char_t sthn thesh ?pos_in_t
              ;na einai melos ths akoloythias t
              (seq_t_member ?char_t,?pos_in_t)


             ;oi dyo antistoixei xarakthres gia antikatstash den prepei na einai
oi idioi
             ;alla oi theseis toys prepei na einai idies
              (not  (= ?char_s,?char_t))(= ?pos_in_s,?pos_in_t)

             ;na yparxei to prohgoymeno stoixeio sthn s akol. apo
```

```
                ;ayto poy tha antikastathei
                (seq_s_member ?prev_char_s,?prev_pos_s)


                 ;na yparxei to prohgoymeno stoixeio sthn t akol. apo
                ;ayto poy tha antikastasteisei
                (seq_t_member ?prev_char_t ,?prev_pos_t)


                ;na yparxei to epomeno stoixeio sthn s akol. apo
                ;ayto poy tha antikastathei
                (seq_s_member ?next_char_s,?next_pos_s)


                    ;na yparxei to epomeno stoixeio sthn t akol. apo
                ;ayto poy tha antikastasteisei
                (seq_t_member ?next_char_t,?next_pos_t)



                ;oi dyo prohgoymenoi xarakthres apo aytoys ths
                ;antikatstashs prepei na einai oi idioi,to idio kai oi
                ;prohgoymenes theseis
                (= ?prev_char_s,?prev_char_t)(= ?prev_pos_s,?prev_pos_t)


                ;oi epomenoi xarakthres prepei na einai idioi,to idio kai oi theseis
toys
               (= ?next_char_s,?next_char_t)(= ?next_pos_s,?next_pos_t)



                  ;epomenh thesh apo thn ?pos_in_t einai h ?next_pos_t
                  (next_position  ?pos_in_t,?next_pos_t)

                  ;epomenh thesh apo thn ?pos_in_s einai h ?next_pos_s
               (next_position  ?pos_in_s,?next_pos_s)

                ;epomenh thesh apo thn ?prev_pos_s einai h ?pos_in_s
               (next_position  ?prev_pos_s, ?pos_in_s)

                ;epomenh thesh apo thn ?pos_in_t einai h ?prev_pos_t
              (next_position  ?prev_pos_t, ?pos_in_t)


            )

 :effect (and
              ;den isxyei pleon to kathgorhma ?char_s sthn thesh ?pos_in_s
```

```
                (not(seq_s_member ?char_s,?pos_in_s))

                ;sthn thesh poy htan prwta to ?char_s twra einai to ?char_t
                (seq_s_member ?char_t,?pos_in_s)


        )

)


(:action final_position_insertion

        :parameters(

                ;teleytaios xarakthras  kai h thesh toy sthn s

                ?pos_in_s -seq_pos,


                ;teleytaios xarakthras  kai h thesh toy sthn t

                 ?char_t -seq_char, ?pos_in_t -seq_pos,




                )

    :precondition (and    ;proapaithseis drashs


                        ;na einai melh ths akoloythias t to   ?char_t,?pos_in_t
                        (seq_t_member ?char_t,?pos_in_t)

                       ;h  pos_in_s na einai h teleytaia thesh ths s akoloythias
                        (final_position_s ?pos_in_s),

                        ;h pos_in_t na einai h teleytaia thesh ths akoloythias
                        (final_position_t ?pos_in_t),

                       ;na exoyn to idio mhkos oi akoloythies
                        (=?pos_in_s,?pos_in_t),
```

```
                    ;h teleytaia thesh toy s na einai adeia
                     (empty_position ?pos_in_s)

            )

 :effect (and

              ;sthn thesh pleon  ?pos_in_s yparxei to ?char_t
               (seq_s_member ?char_t,?pos_in_s)

               ;h teleytaia thesh den einai pleon adeia
               (not (empty_position  ?pos_in_s))

      )

)



(:action first_replacment

       :parameters(

                  ;xarakthras prwths thsehs sthn s
                  ?char_s -seq_char,?pos_in_s -seq_pos,

                   ;xarakthras  prwths theshs t
                    ?char_t -seq_char, ?pos_in_t -seq_pos,

                 ;epomenos xarakthras kai thesh toy sthn s
                   ?next_char_s -seq_char,?next_pos_s -seq_pos,

                ;epomenos xarakthras kai thseh toy sthn t
                   ?next_char_t -seq_char,?next_pos_t -seq_pos,

                   )


   :precondition (and

                    ;na einai meloi ths akoloythias
                    (seq_s_member ?char_s,?pos_in_s)
```

```
                    (seq_t_member ?char_t,?pos_in_t)


                    ;prwth thseh ths akoloythias s
                    (first_position_s ?pos_in_s)

                    ;prwth thesh thesh akolythias t
                    (first_position_t ?pos_in_t)

                    ;na exoyn thn idia prwth thesh  alla oi xarakthtres na mhn
einai idioi
                    (= ?pos_in_s,?pos_in_t)(not (= ?char_s,?char_t))

                   ;na yparxei h epomenh thesh
                  (seq_s_member ?next_char_s,?next_pos_s)

                  (seq_t_member ?next_char_t,?next_pos_t)

                 ;epomenh thesh ths  ?pos_in_t einai h ?next_pos_t sthn t
                  (next_position ?pos_in_t,?next_pos_t)

                    ;epomenh thesh ths  ?pos_in_s einai h ?next_pos_s sthn s
                     (next_position ?pos_in_s,?next_pos_s)


                    ;epomenes theseis prepei na einai idies kai ws xarakthres kai
ws
                    ;theseis
                     (= ?next_char_s,?next_char_t)(= ?next_pos_s,?next_pos_t)




                )


:effect (and
                        ;pleon to ?char_t,?pos_in_s einai melos ths akolythias
                         (seq_s_member ?char_t,?pos_in_s)

                        ;pleon   ?char_s,?pos_in_s den einai melos ths
akoloythias
                        (not (seq_s_member ?char_s,?pos_in_s) )



                )

 )
```

```
(:action deletion

        :parameters(    ;dhlwsh parametrwn

                        ;o xarakthras kai h thesi pou tha diagrfh
                        ;sthn s akoloythia xarakthrwn

                        ?char_s  -seq_char,?pos_in_s  -seq_pos,



                        ;o antistoixos xarakthras aytoy poy tha
                        ;diagrafh sthn akoloythia t

                        ?char_t  -seq_char, ?pos_in_t  -seq_pos,



                        ;o amesws prohgoymenos xarakthras kai
                        ;h prohgoumenh thesh(apo ayto poy tha diagrafh)
                        ; sthn s akoloythia

                        ?prev_char_s  -seq_char,?prev_pos_s  -seq_pos,



                         ;o amesws prohgoymenos xarakthras kai
                         ;h prohgoumenh thesh(toy antistoixoy apo
                         ; ayto pou tha diagrafei) sthn t akoloythia

                         ?prev_char_t  -seq_char,?prev_pos_t  -seq_pos,



                        ;o amesws epomenos xarakthras kai
                        ;h epomenh thesh(apo ayto poy tha diagrafei)
                        ; sthn s akoloythia

                         ?next_char_s  -seq_char,?next_pos_s  -seq_pos,



                         ;o amesws epomenos xarakthras kai
                         ;h epomenh thesh(toy antistoixoy apo
                         ; ayto pou tha diagrafei) sthn t akoloythia
```

```
                ?next_char_t  -seq_char,?next_pos_t  -seq_pos,

            )   ;telos  dhlwshs parametrwn

:precondition (and       ;proapaithseis drashs

                ;o xarakthras ?char_s sthn thesh ?pos_in_s
                ;na einai melos ths akoloythias s

                (seq_s_member ?char_s,?pos_in_s)


                ;o xarakthras ?char_t sthn thesh ?pos_in_t
                 ;na einai melos ths akoloythias t

                 (seq_t_member ?char_t,?pos_in_t)


                ;o xarakthras ?char_s den tha prepei na
                ;einai idios me ton antistoixo toy ?char_t
                ;sthn akoloythia t

                (not  (= ?char_s,?char_t))(= ?pos_in_s,?pos_in_t)


               ;o xarakthras ?prev_char_s sthn thesh ?prev_pos_s
               ;einai melos ths akoloythias s

               (seq_s_member ?prev_char_s,?prev_pos_s)


              ;o xarakthras ?prev_char_t sthn thesh ?prev_pos_t
              ;einai melos ths akoloythias t

              (seq_t_member ?prev_char_t ,?prev_pos_t)


              ;o xarakthras ?next_char_s sthn thesh ?next_pos_s
              ;einai melos ths akoloythias s

              (seq_s_member ?next_char_s,?next_pos_s)
```

```
                ;o xarakthras ?next_char_t sthn thesh ?next_pos_t
                ;einai melos ths akoloythias t

                (seq_t_member ?next_char_t,?next_pos_t)



              ;oi prohgorymenoi xarakthres apo ta ?char_t kai ?char_s
antistoixa
              ;einai idioi kai briskontai se antistoixes theseis stis dyo
akoloythies
              ;t kai s

              (= ?prev_char_s,?prev_char_t)(= ?prev_pos_s,?prev_pos_t)



              ;oi epomenoi  xarakthres apo ta ?char_t kai ?char_s antistoixa
              ;den prepei na einai idioi briskontai se antistoixes theseis stis dyo
akoloythies
              ;t kai s


              ;oi epomenes theseis twn  ?char_t kai ?char_s den prepei na
              ;exoyn idioyd xarakthres alla arithmitikes na einai idies

              (not(= ?next_char_s,?next_char_t))(= ?next_pos_s,?next_pos_t)



             ;o epomenos xarakthras meta to ?char_s (dhladh o ?next_char_s)
             ;prepei na einai o idios me ton antistoixo aytoy poy tha diagrafei
             ;shn akoloythia t

              (= ?next_char_s,?char_t)

             ;epomenh thesh apo thn ?pos_in_t einai h ?next_pos_t
             (next_position  ?pos_in_t,?next_pos_t)

            ;epomenh thesh apo thn ?pos_in_s einai h ?next_pos_s
             (next_position  ?pos_in_s,?next_pos_s)

             ;epomenh thesh apo thn ?prev_pos_s einai h ?pos_in_s
             (next_position  ?prev_pos_s, ?pos_in_s)
```

```
                ;epomenh thesh apo thn ?pos_in_t einai h ?prev_pos_t
              (next_position  ?prev_pos_t, ?pos_in_t)


              ;pernoyme ana dyo an yparxoyn ta ypoloipa  stoixeia ths
akoloythias
              ;an yparxoyn kai metakinoyntai oi xarakthres ths akoloythias gia
              ;na kalyfthei h kenh thesh

        ) ;closing and preconditions


 :effect      (and      ;to ?char_s,?pos_in_s den einai melos ths akoloythias
                        (not (seq_s_member ?char_s,?pos_in_s))

                        (not (seq_s_member ?next_char_s,?next_pos_s))

                         ;metakinhsh pros dexia ths apomenhs theshs apo ayth poy
                         ;diagrafhke
                         (seq_s_member ?next_char_s,?pos_in_s)



                         ;h epomenh thesh  pleon einai kenh
                         (empty_position ?next_pos_s)


     )  ;closing effects


)    ;telos drashs diagrafhs



 (:action shift

       :parameters(    ;dhlwsh parametrwn


                  ?next_pos_s  -seq_pos,


                   ?pos_in_s  -seq_pos
```

```
                    )   ;telos  dhlwshs parametrwn


  :precondition   (and

                        (next_position ?pos_in_s,?next_pos_s)

                         ;h epomenh thesh  pleon einai kenh
                          (empty_position ?next_pos_s)
                       )
:effect
                        (forall  (?epomeno1  ?epomeno2  -seq_pos  ?ch2  -
seq_char)

                      (when (and

                                    ;h ?epomeno2 einai h epomenh thesh apo
                                    ;?epomeno1
                                    (next_position ?epomeno1,?epomeno2)


                                   ;to ?ch2,?epomeno2 einai stoixeio ths
                                    ;akoloythias
                                    (seq_s_member ?ch2,?epomeno2)


                                    ;h thesi ?epomeno1 einai megalyterh
                                    ;apo thn ?pos_in_s
                                    (greater_pos ?epomeno1,?pos_in_s)

                                    (not(= ?epomeno1,?pos_in_s))

                                   (not(= ?epomeno2,?pos_in_s))

                                    ;h thesi ?epomeno2 einai megalyterh
                                    ;apo thn ?pos_in_s
                                    (greater_pos ?epomeno2,?pos_in_s)

                                    ;h thesi ?epomeno2 einai megalyterh
                                    ;apo thn ?epomeno1
                                    (greater_pos  ?epomeno2,?epomeno1)


                                    ;h thesh ?ch1,?epomeno1 prepei na
                                    ;einai kenh
                                    (empty_position ?epomeno1)
```

```
                                    ;oi dyo theseis poy pairnoyme prepei
                                    ;na einai diaforetikes
                                    (not(=?epomeno1,?epomeno2))
                                )

                                ;an isxyei h synthikh tote
                            (and

                                ;metakinhsh toy ?ch2s xarakthra
                                (seq_s_member ?ch2,?epomeno1)


                                (not (empty_position ?epomeno1))

                                ;to keno metaferetai dexiotera
                                (empty_position  ?epomeno2)

                                ;to ?ch2,?epomeno2 anhkei sthn akoloythia alla
einai keno
                                (not (seq_s_member ?ch2,?epomeno2))



                            )
                        )

                )
        )

)           ;telos domain
```

Ορισμός προβλήματος P_S_A_1:

```
(define (problem P_S_A_1)
(:domain SEQ_ALIG)
(:objects A -seq_char,
        G -seq_char,
        C -seq_char,
        T -seq_char,
        zero -seq_pos,
        one -seq_pos,
        two -seq_pos,
        three -seq_pos,
```

```
        four -seq_pos,
        five -seq_pos,
        six -seq_pos,
        seven -seq_pos
)


(:init (next_position zero,one)
       (next_position one,two)
       (next_position two,three)
      (next_position three,four)
       (next_position four,five)
       (next_position five,six)
      (next_position six,seven)
      (seq_s_member A, zero)
      (seq_s_member G,one)
      (seq_s_member C,two)
      (seq_s_member A,three)
      (seq_s_member C,four)
      (seq_s_member A,five)
      (seq_s_member C ,six)
      (seq_s_member A ,seven )

      (seq_t_member A , zero )
      (seq_t_member C ,one )
      (seq_t_member A ,two )
      (seq_t_member C ,three )
      (seq_t_member A ,four )
      (seq_t_member C ,five )
      (seq_t_member T ,six )
      (seq_t_member A ,seven )
```

(final_position_s seven)
(greater_pos one,zero)
(greater_pos two,one)
(greater_pos two,zero)
(greater_pos three,zero)
(greater_pos three,one)
(greater_pos three,two)
(greater_pos four,zero)
(greater_pos four,one)
(greater_pos four,two)
(greater_pos four,three)
(greater_pos five,zero)
(greater_pos five,one)
(greater_pos five,two)
(greater_pos five,three)
(greater_pos five,four)
(greater_pos six,zero)
(greater_pos six,one)
(greater_pos six,two)
(greater_pos six,three)
(greater_pos six,four)
(greater_pos six,five)
(greater_pos seven,zero)
(greater_pos seven,one)
(greater_pos seven,two)
(greater_pos seven,three)
(greater_pos seven,four)
(greater_pos seven,five)
(greater_pos seven,six)
(final_position_t seven)
(first_position_t zero)
(first_position_s zero)

```
)

(:goal (and

          (seq_s_member A ,zero )
          (seq_s_member C,one )
          (seq_s_member A,two )
          (seq_s_member C,three)
          (seq_s_member A,four)
          (seq_s_member C,five)

         (seq_s_member T ,six)
          (seq_s_member A ,seven )


      )
)

)
```