Τεχνητή Νοημοσύνη ΙΙ
Εαρινό Εξάμηνο 2011-2012

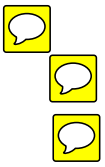
Εργασία 2: Σχεδιασμός
Ημερομηνία Παράδοσης: 19 Απριλίου κατά την έναρξη του μαθήματος ή ηλεκτρονικά ΠΡΙΝ την έναρξη του

Από τη σελίδα του μαθήματος στην ενότητα "Εργασίες" κατεβάστε το συνοδευτικό αρχείο "homework2-ai2.zip" της Εργασίας 2, το οποίο περιέχει τα αρχεία που θα χρειαστείτε στις παρακάτω ασκήσεις.

Για κάποιες από τις ασκήσεις θα χρειαστεί να γράψετε ή να χρησιμοποιήσετε αρχεία στην γλώσσα Planning Domain Definition Language (PDDL) στην πιο απλή της εκδοχή. Δεν θα χρειαστείτε παραπάνω πληροφορίες για τη γλώσσα PDDL από αυτές που είπαμε στο μάθημα. Μπορείτε να βρείτε μια συνοπτική περιγραφή του STRIPS υποσυνόλου της PDDL στον παρακάτω σύνδεσμο: http://www.ida.liu.se/~TDDC17/info/labs/planning/2004/writing.html. Στην περιγραφή αυτή αγνοείστε αυτά που αναφέρονται σε "types" και ":equality".

1. Στην PDDL τα προβλήματα σχεδιασμού περιγράφονται χωριστά σε δύο αρχεία, ένα αρχείο το οποίο περιγράφει τα κατηγορήματα και τα σχήματα ενεργειών του πεδίου που μας ενδιαφέρει, και ένα αρχείο το οποίο περιγράφει τις σταθερές που αντιστοιχούν στα αντικείμενα ενός συγκεκριμένου προβλήματος, την αρχική κατάσταση, και τον στόχο του προβλήματος. Θα αναφερόμαστε στο πρώτο ως το αρχείο domain και στο δεύτερο ως το αρχείο problem.

   Για την άσκηση αυτή θα χρησιμοποιήσετε τα παρακάτω αρχεία:

   - problems/gripper-domain.txt
   - problems/gripper-problem1.txt
   - problems/gripper-problem2.txt
   - blackbox.exe
   - cygwin1.dll

   Το αρχείο "gripper-domain.txt" είναι μια περιγραφή του domain του κόσμου των κύβων σε PDDL με βάση τις ενέργειες και τα λεκτικά που αναφέρονται στην πρώτη άσκηση της Εργασίας 1. Αντίστοιχα, το αρχείο "gripper-problem1.txt" είναι μια περιγραφή των αντικειμένων (objects), της αρχικής κατάστασης, και του στόχου που αναφέρονται στη δεύτερη άσκηση της Εργασίας 1. Το αρχείο "gripper-problem2.txt" περιγράφει τον ίδιο στόχο αλλά για μια διαφορετική αρχική κατάσταση.

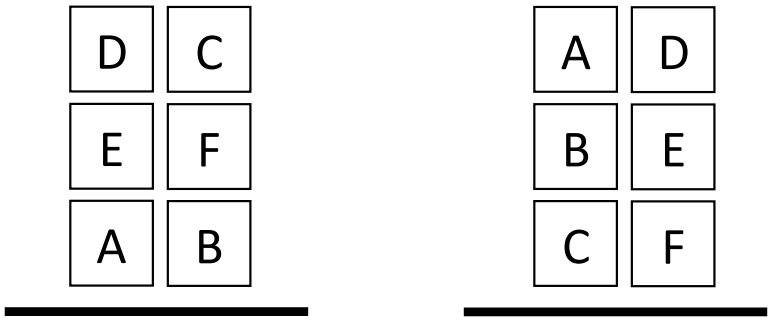
   Το εκτελέσιμο αρχείο "blackbox.exe" είναι η υλοποίηση ενός planner που συνδυάζει τα γραφήματα σχεδιασμού με τεχνικές επίλυσης προβλημάτων ικανοποιησιμότητας (SAT). Περισσότερες πληροφορίες μπορείτε να βρείτε στη σελίδα του ερευνητή που τον δημιούργησε, Henry Kautz: http://www.cs.rochester.edu/~kautz/satplan/blackbox/.

   Το αρχείο "cygwin1.dll" χρειάζεται να βρίσκεται στον ίδιο φάκελο με το "blackbox.exe" ώστε να μπορεί να εκτελεστεί το τελευταίο. Η τυπική χρήση του planner είναι ως εξής:

   blackbox –o gripper-domain.txt –f gripper-problem1.txt

   Για την άσκηση αυτή πρέπει να κάνετε τα παρακάτω:

   (i) Χρησιμοποιήστε τον blackbox για να βρείτε μια λύση στα προβλήματα σχεδιασμού (gripper-domain.txt, gripper-problem1.txt) και (gripper-domain.txt, gripper-problem2.txt). Σε κάθε μια από τις δυο περιπτώσεις εκτέλεσης του blackbox, παρατηρείστε το output του planner και εντοπίστε το πλάνο που προκύπτει ως λύση του προβλήματος. Ως απάντηση σε αυτή την ερώτηση πρέπει απλώς να αντιγράψετε τα δύο αυτά πλάνα που προκύπτουν στο αρχείο κειμένου που θα παραδώσετε.



Εικόνα 1: Η αρχική κατάσταση (αριστερά) και ο στόχος (δεξιά) για το πρόβλημα "gripper-problem3.txt"

(ii) Χρησιμοποιώντας το αρχείο "gripper-domain.txt" ως περιγραφή των operators του κόσμου των κύβων, δημιουργείστε το αρχείο "gripper-problem3.txt" το οποίο περιγράφει ένα πρόβλημα σχεδιασμού που εμπλέκει 6 κύβους αυτή τη φορά, και στο οποίο η αρχική κατάσταση και ο στόχος είναι όπως στην Εικόνα 1. Χρησιμοποιείστε τον blackbox για να βρείτε μια λύση στο πρόβλημα σχεδιασμού (gripper-domain.txt, gripper-problem3.txt). Ως απάντηση σε αυτή την ερώτηση πρέπει να παραδώσετε το αρχείο "gripper-problem3.txt", και να αντιγράψετε το αρχείο αυτό και το πλάνο που προκύπτει από την εκτέλεση του blackbox στο αρχείο κειμένου που θα παραδώσετε.

2. Για την άσκηση αυτή θα υλοποιήσετε σε Prolog έναν planner που βρίσκει λύσεις σε προβλήματα σχεδιασμού γραμμένα σε PDDL. Ειδικότερα, θα υλοποιήσετε μια αναδρομική εκδοχή της DepthFirstPlanner που χρησιμοποιήσατε για την άσκηση 2 της Εργασίας 1. Για την άσκηση αυτή θα χρησιμοποιήσετε αποκλειστικά την SWI-Prolog την οποία μπορείτε να κατεβάσετε από τον παρακάτω σύνδεσμο: http://www.swi-prolog.org/download/stable, και κάποια βοηθητικά αρχεία τα οποία παρέχουν έτοιμες τις βασικές συναρτήσεις που θα χρειαστούν για την υλοποίηση.

   Για την άσκηση αυτή θα χρησιμοποιήσετε τα παρακάτω αρχεία:

   - problems/gripper-domain.txt
   - problems/gripper-problem0.txt
   - problems/gripper-problem1.txt
   - problems/gripper-problem2.txt
   - planner.pl
   - τα αρχεία του φακέλου utils

   Στο αρχείο "planner.pl" υπάρχει μια ημιτελής υλοποίηση ενός planner που βασίζεται στα παρακάτω τρία κατηγορήματα:

   - **planner(+DomainFile, +ProblemFile)**

     Διαβάζει τα PDDL αρχεία και αρχικοποιεί την αναζήτηση. Όταν υλοποιηθούν τα κομμάτια που λείπουν από την αναζήτηση, το παρακάτω ερώτημα στην Prolog θα υπολογίζει και θα εκτυπώνει στην οθόνη μια λύση για το πρόβλημα (gripper-domain.txt, gripper-problem1.txt):

     ?- planner('gripper-domain.txt', 'gripper-problem1.txt').

     Αυτό το κατηγόρημα δεν χρειάζεται να τροποποιηθεί.

- **search(-Solution)**

  Αρχικοποιεί τις λίστες OpenStates και VisitedStates όπως στα δύο πρώτα βήματα της DepthFirstPlanner στην Εργασία 1, και καλεί την συνάρτηση dfs(OpenStates, VisitedStates, Solution).

  Αυτό το κατηγόρημα δεν χρειάζεται να τροποποιηθεί.

- **dfs(+OpenStates, +VisitedStates, -Solution)**

  Το κατηγόρημα αυτό πρέπει να τροποποιηθεί ώστε να υλοποιεί μια αναδρομική εκδοχή της αναζήτησης που πραγματοποιείται με το loop της DepthFirstPlanner στην Εργασία 1. Προς το παρόν δεν κάνει κάτι τέτοιο, απλώς παίρνει το πρώτο ζεύγος της λίστας OpenStates και ελέγχει αν η κατάσταση που περιγράφεται από το δεύτερο στοιχείο του ζεύγους ικανοποιεί το στόχο.

Για την άσκηση αυτή πρέπει να κάνετε τα παρακάτω:

(i) Κάντε consult το αρχείο "planner.pl" και κάντε τα παρακάτω ερωτήματα στην Prolog:

```
?- planner('problems/gripper-domain.txt', 'problems/gripper-problem0.txt').
?- planner('problems/gripper-domain.txt', 'problems/gripper-problem1.txt').
```

Ως απάντηση σε αυτή την ερώτηση εξηγήστε γιατί η ημιτελής υλοποίηση του planner βρίσκει λύση στην πρώτη περίπτωση, ενώ αδυνατεί να βρει λύση στη δεύτερη.

(ii) Τροποποιήστε το κατηγόρημα dfs/3 κατάλληλα ώστε να υλοποιεί μια αναδρομική εκδοχή της αναζήτησης που πραγματοποιείται με το loop της DepthFirstPlanner στην Εργασία 1. Χρησιμοποιήστε τα προβλήματα (gripper-domain.txt, gripper-problem1.txt) και (gripper-domain.txt, gripper-problem2.txt) για να βεβαιωθείτε ότι δουλεύει σωστά.

Ως απάντηση σε αυτή την ερώτηση πρέπει να παραδώσετε το αρχείο "planner.pl", και να αντιγράψετε την υλοποίηση του dfs/3 στο αρχείο κειμένου που θα παραδώσετε. Επίσης, αντιγράψτε τα αποτελέσματα του planner για τα προβλήματα (gripper-domain.txt, gripper-problem1.txt) και (gripper-domain.txt, gripper-problem2.txt) στο αρχείο κειμένου που θα παραδώσετε.

Δεν χρειάζεται να τροποποιήσετε κανένα από τα αρχεία που βρίσκονται στο φάκελο "utils", ο φάκελος χρειάζεται όμως γιατί παρέχει κάποια κατηγορήματα που θα χρησιμοποιήσετε στην υλοποίηση του "planner.pl". Για μια περιγραφή των κατηγορημάτων που είναι διαθέσιμα δείτε τα σχόλια στην αρχή του αρχείου "planner.pl" και το παράρτημα στο τέλος του κειμένου.

3. Εφόσον έχετε τελειώσει με την άσκηση 2 και ο planner λειτουργεί σωστά για τα προβλήματα (gripper-domain.txt, gripper-problem1.txt) και (gripper-domain.txt, gripper-problem2.txt), δοκιμάστε κάποια πιο δύσκολα προβλήματα.

   Για την άσκηση αυτή θα χρησιμοποιήσετε επιπλέον τα παρακάτω αρχεία:

   - problems/sokoban-domain.txt
   - problems/sokoban-problem1.txt
   - problems/sokoban-problem2.txt
   - problems/sokoban-problem3.txt

Για την άσκηση αυτή πρέπει να κάνετε τα παρακάτω:

(i) Χρησιμοποιείστε τον planner που φτιάξατε στην προηγούμενη άσκηση για να λύσετε τα προβλήματα (sokoban-domain.txt, sokoban-problem1.txt), (sokoban-domain.txt, sokoban-problem2.txt) και (sokoban-domain.txt, sokoban-problem3.txt). Θα δείτε ότι για τα προβλήματα αυτά ο planner σας καθυστερεί σημαντικά σε σχέση με τα απλά προβλήματα της προηγούμενης άσκησης και το "Search length" που αναφέρει ο planner είναι πολύ μεγάλο.

Προσοχή: Για να βλέπετε το σωστό "Search length" πρέπει στην υλοποίηση του κατηγορήματος dfs/3 να καλείτε το κατηγόρημα "stat_node" κάθε φορά που η dfs/3 αφαιρεί ένα στοιχείο από τη λίστα OpenStates και προσθέτει διάδοχες καταστάσεις.

Ως απάντηση σε αυτή την ερώτηση πρέπει να αντιγράψετε τα αποτελέσματα του planner για τα προβλήματα (sokoban-domain.txt, sokoban-problem1.txt), (sokoban-domain.txt, sokoban-problem2.txt) και (sokoban-domain.txt, sokoban-problem3.txt) στο αρχείο κειμένου που θα παραδώσετε.

(ii) Χρησιμοποιήστε την υλοποίηση του dfs/3 ως βάση για να γράψετε το κατηγόρημα astar/3 το οποίο υλοποιεί μια αναδρομική εκδοχή της συνάρτησης AStarPlanner που χρησιμοποιήσατε για την άσκηση 3 της Εργασίας 1. Τροποποιήστε το κατηγόρημα search/1 ώστε να αρχικοποιεί και να χρησιμοποιεί το κατηγόρημα astar/3 για την αναζήτηση αντί για το dfs/3.

Ως απάντηση σε αυτή την ερώτηση πρέπει να παραδώσετε το αρχείο "planner.pl", και να αντιγράψετε την υλοποίηση του bfs/3 στο αρχείο κειμένου που θα παραδώσετε. Επίσης, αντιγράψτε τα αποτελέσματα του planner για τα προβλήματα (sokoban-domain.txt, sokoban-problem1.txt), (sokoban-domain.txt, sokoban-problem2.txt) και (sokoban-domain.txt, sokoban-problem3.txt) στο αρχείο κειμένου που θα παραδώσετε.

(iii) Φτιάξτε ένα πινακάκι που δείχνει το χρόνο και το "Search length" για την επίλυση των προβλημάτων (sokoban-domain.txt, sokoban-problem1.txt), (sokoban-domain.txt, sokoban-problem2.txt) και (sokoban-domain.txt, sokoban-problem3.txt) από τον planner όταν χρησιμοποιήθηκε το κατηγόρημα dfs/3 στο ερωτημα (i) και όταν χρησιμοποιήθηκε το κατηγόρημα bfs/3 στο ερωτημα (ii). Τι παρατηρείτε;

4. **Bonus ερώτηση.** Για την άσκηση αυτή θα χρησιμοποιήσετε τα παρακάτω αρχεία:

- problems/simplegame-domain.txt
- problems/simplegame-problem1.txt

Για την άσκηση αυτή πρέπει να τροποποιήσετε το αρχείο "simplegame-domain.txt" ώστε να περιγράφει έναν (πολύ) απλοποιημένο κόσμο ενός video game όπως εξηγείται παρακάτω.

Σε αυτόν τον κόσμο υπάρχει ένας χαρακτήρας (npc) και ένας άνθρωπος-παίκτης (player), καθώς και διάφορα αντικείμενα ανάμεσα στα οποία κάποια είναι όπλα και κάποια μαχαίρια. Η περιγραφή του domain και του problem αρχείου γίνεται από την οπτική του χαρακτήρα.

Η θέση του χαρακτήρα αναπαριστάται με το λεκτικό (npc-at ?l). Ο χαρακτήρας μπορεί να κινηθεί από το ένα σημείο του χώρου στο άλλο μόνο προς την κατεύθυνση που κοιτάει η οποία αναπαριστάται με το λεκτικό (npc-facing ?d). Ο χαρακτήρας μπορεί να κρατάει το πολύ ένα αντικείμενο και η αναπαράσταση αυτής της πληροφορίας γίνεται με τα λεκτικά (npc-holding ?o) και (npc-emptyhands).

Η θέση του ανθρώπου-παίκτη αναπαριστάται με το λεκτικό (player-at ?l). Οι θέσεις των διαφόρων αντικειμένων στο χώρο αναπαριστώνται με το λεκτικό (object-at ?o ?l). Το λεκτικό

(adjacent ?l1 ?l2 ?d) αναπαριστά ότι από τη θέση ?l1 κοιτώντας προς την κατεύθυνση ?d μπορούμε να κινηθούμε στη θέση ?l2.

Τα λεκτικά (knife ?k), (gun ?g), και (dir ?d) αναπαριστούν ότι ένα αντικείμενο είναι μαχαίρι, όπλο, και μια από τις διαθέσιμες κατευθύνσεις που μπορεί να κοιτάει ο χαρακτήρας, αντίστοιχα. Τέλος, το λεκτικό (player-down) αναπαριστά ότι ο άνθρωπος-παίκτης έχει χτυπηθεί θανάσιμα από τον χαρακτήρα.

Ο χαρακτήρας μπορεί να πραγματοποιήσει τις παρακάτω ενέργειες:

- move(?froml ?tol ?dir) με την οποία μετακινείται κατά μια θέση ως προς την κατεύθυνση πού κοιτά
- turn(?fromd ?tod) με την οποία αλλάζει την κατεύθυνση στην οποία κοιτά
- pick-up(?o ?l) με την οποία παίρνει ένα αντικείμενο το οποίο βρίσκεται στην ίδια θέση με αυτόν
- stab(?loc ?k) με την οποία επιτίθεται στον άνθρωπο-παίκτη υπό την προϋπόθεση ότι βρίσκεται στην ίδια θέση με αυτόν και κρατάει ένα μαχαίρι
- shoot(?locn ?locp ?dir ?g) με την οποία επιτίθεται στον άνθρωπο-παίκτη υπό την προϋπόθεση ότι ο άνθρωπος-παίκτης βρίσκεται στη θέση που είναι δίπλα από τη θέση του χαρακτήρα ως προς την κατεύθυνση που κοιτάει ο χαρακτήρας, και ο τελευταίος κρατάει ένα όπλο.

Για την άσκηση αυτή πρέπει να συμπληρώστε την περιγραφή των σχημάτων ενεργειών στο ημιτελές αρχείο “simplegame-domain.txt” και χρησιμοποιήστε τον blackbox ή τον planner που φτιάξατε στην προηγούμενη άσκηση για να βρείτε μια λύση στο πρόβλημα (“simplegame-domain.txt”, “simplegame-problem1.txt”).

Παράρτημα: Βοηθητικές συναρτήσεις που παρέχονται από το φάκελο “utils” μέσω του αρχείου “common.pl”

- **parsePDDL(+DomainFile, +ProblemFile)**

  Διαβάζει τα PDDL αρχεία domain και problem και αρχικοποιεί κάποιες global δομές από τις οποίες μπορείτε να παίρνετε τις πληροφορίες που χρειάζεστε χρησιμοποιώντας τις παρακάτω συναρτήσεις.

- **get_goal(-Goal)**

  Επιστρέφει την περιγραφή του στόχου ως λίστα θετικών λεκτικών.

- **get_init(-InitialState)**

  Επιστρέφει την περιγραφή της αρχικής κατάστασης ως λίστα θετικών λεκτικών.

- **satisfies_goal(+State)**

  Ελέγχει αν η μεταβλητή State (η οποία αποτελεί μια λίστα θετικών λεκτικών) ικανοποιεί το στόχο.

- **progress(+State, -Action, -NextState)**

  Παράγει όλες τις δυνατές διάδοχες καταστάσεις από την κατάσταση State. Η μεταβλητή Action είναι μια εφαρμόσιμη ενέργεια στην κατάσταση State και η κατάσταση NextState είναι η διάδοχη κατάσταση που προκύπτει από την εφαρμογή των επιδράσεων της Action στην State.

- **h(+State, -Value)**

  Ευρετική συνάρτηση η οποία δίνει μια προσέγγιση για το πόσο απέχει η State από το στόχο, υπολογίζοντας τον αριθμό των υποστόχων που δεν ικανοποιούνται στην State.