

Δεύτερο μέρος εργαστηριακής εργασίας στο μάθημα Τεχνητής Νοημοσύνης

Ιάκωβος Μαστρογιαννόπουλος

2021-01-15

Contents

1	Πρόλογος	1
2	Κατανόηση προβλήματος - Πρόβλημα του Parking	2
3	Βήμα 1	3
3.1	Ζητούμενα πρώτου βήματος	3
3.2	Templates πρώτου βήματος	3
3.3	Γεγονόντα πρώτου βήματος	3
3.4	Κανόνες πρώτου βήματος	5
3.4.1	Κανόνας εκκίνησης	5
3.4.2	Κανόνας goal_found	5
3.4.3	Τυπικός κανόνας κίνησης	6
3.5	Βήμα 1 source code	7
3.6	Ενδεικτικά τρεξίματα του πρώτου βήματος	11
3.6.1	Πριν την εκκίνηση	11
3.6.2	Κανόνας εκκίνησης	12
3.6.3	Κανόνας κίνησης	12
3.6.4	Μετά την εκκίνηση	12
3.6.5	Σχόλια	12
4	Παρατηρήσεις ενδιάμεσα των βήματων	13
5	Βήμα 2	13
5.1	Ζητούμενα δεύτερου βήματος	13
5.2	Τι μπορούμε να χρησιμοποιήσουμε από το πρώτο βήμα	13
5.2.1	Τροποποιημένα templates πρώτου βήματος	13
5.2.2	Τροποποιημένα γεγονόντα πρώτου βήματος	14
5.2.3	Τροποποιημένοι κανόνες πρώτου βήματος	15
5.3	Templates δεύτερου βήματος	15
5.4	Γεγονόντα δεύτερου βήματος	16
5.5	Κανόνες δεύτερου βήματος	16
5.5.1	Κανόνας prompt	16
5.5.2	Τυπικός κανόνας επιστροφής στο s2	17
5.5.3	Τυπικός κανόνας εύρεσης αυτοκίνητου	18
5.6	Βήμα 2 source code	19
5.7	Ενδεικτικά τρεξίματα του δεύτερου βήματος	29
5.7.1	Κανόνας κίνησης	29
5.7.2	Κανόνας prompt	29
6	Σχολιασμός - Βελτιώσεις	29

Listings

1	Κανόνας εκκίνησης	5
2	Κανόνας goal_found	5
3	Τυπικός κανόνας κίνησης	6
4	Βήμα 1	7
5	Τροποποιημένος γενικός κανόνας κίνησης	15
6	Κανόνας prompt	16
7	Τυπικός κανόνας επιστροφής στο s2	17
8	Τυπικός κανόνας εύρεσης αυτοκίνητου	18
9	Βήμα 2	19

Abstract

Αυτό είναι το δεύτερο μέρος της εργαστηριακής εργασίας του μαθήματος
Τεχνητής Νοημοσύνης του Ιάκωβου Μαστρογιαννόπουλου, ΑΜ: cse 242017102
εξάμηνο 7.

1 Πρόλογος

Στη συγκεκριμένη εργασία του μαθήματος «Τεχνητής Νοημοσύνης» είχαμε να μελετήσουμε το πρόβλημα του parking. Βέβαια, για να μπορέσουμε να φτάσουμε στο σημείο της υλοποίησης του κωδικά, πρώτα πρέπει να εξηγηθούν μερικά πράγματα με το ποιο είναι το πρόβλημα, πώς μπορούμε να το υλοποιήσουμε και ποια θα είναι τα πιθανά αποτελέσματα που θα μπορούσαμε να πάρουμε πίσω. Σε αυτό το μέρος χρησιμοποιήθηκε το εργαλείο της Clips.

2 Κατανόηση προβλήματος - Πρόβλημα του Parking

Όπως και στο πρώτο μέρος της εργασίας, το πρόβλημα που έχουμε να λύσουμε είναι το πρόβλημα του Parking. Θεωρητικά, υπάρχει ένας αυτόματος οδηγός ο οποίος προσπαθεί να βρει ελεύθερο χώρο για να γεμίσει τα αμάξια που θέλουν να μπουν μέσα στο Parking. Υπάρχουν N spaces από τα οποία θα μπορούσαν μερικά από αυτά να ήταν τελείως άδεια, ενώ κάποια αλλά να είχαν μια πλατφόρμα. Στο δεύτερο μέρος της εργασίας, όμως υπάρχουν και επιπλέον όροφοι M , όπου κάθε M όροφος έχει N spaces.

Παρόλο που στο προηγούμενο μέρος χρησιμοποιήθηκαν αλγόριθμοι αναζήτησης για την επιλύσή του, αυτή την φορά χρησιμοποιούμε εμπείρα συστήματα. Ο στόχος όμως δεν αλλάζει, ο οποίος είναι να βρίσκει τις πλατφόρμες και από εκεί να καταλαμβάνει ποιες από αυτές έχουν ελεύθερο χώρο και να τις γεμίζει. Ένα εμπείρο σύστημα αποτελείται από κανόνες και γεγονότα. Στο σύστημα που θα φτιάξουμε θα πρέπει να έχει τα εξής γεγονότα:

- Τα spaces και οι γειτονικές τους καταστάσεις
- Οι πλατφόρμες (platforms)
- Το info της τρέχουσας κατάστασης του συστήματος
- Και τα αυτοκίνητα που είναι παρκαρισμένα

Ενώ οι κανόνες είναι οι εξής:

- Ο αρχικός κανόνας που αρχικοποιεί το info και ζητάει να μάθει πόσα αυτοκίνητα είναι στην ουρά και περιμένουν
- Οι κανόνες που περιγράφουν πως κουνιέται βορεία, νότια, ανατολικά και δυτικά η πλατφόρμα αναλόγως με τις γειτονικές καταστάσεις που έχουν τα spaces
- Και ο τελικός κανόνας που λειτουργεί μόνο όταν δεν υπάρχει άλλο αμάξι στην ουρά

Η εργασία είναι χωρισμένη σε δύο βήματα, όπου θα τα αναλύσουμε ξεχωριστά στο PDF.

3 Βήμα 1

3.1 Ζητούμενα πρώτου βήματος

Το πρώτο βήμα του δευτέρου μέρους της εργασίας μας ζητάει να φτιάξουμε ένα parking με 6 spaces και 5 πλατφόρμες σε έναν όροφο. Έχουμε 6 spaces (s1-s6) όπου η s2 είναι η είσοδος των αυτοκινήτων και στα υπόλοιπα έχουμε από μία πλατφόρμα (p1-p5). Ο αυτόματος παρκαδόρος μπορεί να κινηθεί βόρεια, νοτεία, δυτικά και ανατολικά. Το αυτοκίνητο τοποθετείται στην πλατφόρμα την οποία ήταν άδεια.

3.2 Templates πρώτου βήματος

Τα templates του πρώτου βήματος είναι η εξής:

```
1      (deftemplate space
2        (slot name)
3        (slot level)
4        (slot state)
5      )
6
7      (deftemplate platform
8        (slot name)
9        (slot space)
10       (slot plat_state)
11     )
12
13     (deftemplate car
14       (slot cars_left)
15       (slot state)
16     )
17
```

3.3 Γεγονόντα πρώτου βήματος

Τα γεγονόντα του πρώτου βήματος είναι η εξής:

```
1      (def facts space-init
2        (space (name s1) (level 1) (state Y))
3        (space (name s2) (level 1) (state N))
4        (space (name s3) (level 1) (state Y))
5        (space (name s4) (level 1) (state Y))
6        (space (name s5) (level 1) (state Y))
7        (space (name s6) (level 1) (state Y))
8      )
9
10     (def facts plat-init
11       (platform (name p1) (space s1) (plat_state E))
12       (platform (name p2) (space s3) (plat_state E))
13       (platform (name p3) (space s4) (plat_state E))
14       (platform (name p4) (space s5) (plat_state E))
15       (platform (name p5) (space s6) (plat_state E))
16     )
17
```

```
18      (deffacts east
19          (east s1 s2)
20          (east s2 s3)
21          (east s4 s5)
22          (east s5 s6)
23      )
24
25      (deffacts north
26          (north s1 s4)
27          (north s2 s5)
28          (north s3 s6)
29      )
30
31      (deffacts west
32          (west s2 s1)
33          (west s3 s2)
34          (west s5 s4)
35          (west s6 s5)
36      )
37
38      (deffacts south
39          (south s4 s1)
40          (south s5 s2)
41          (south s6 s3)
42      )
43
```


3.4 Κανόνες πρώτου βήματος

Παμέ να δούμε αναλητικά το κάθε κανόνα ξεχωριστά.

3.4.1 Κανόνας εκκίνησης

```
1 ; Kanonas pou ksekinaei to programma
2 (defrule startup
3   (declare (salience 100)) ; Dinei protereotita ston kanona
4   (initial-fact)
5   =>
6   (set-strategy random)
7   (printout t "Starting the program!" crlf)
8
9   (printout t "Enter the ammount of cars waiting in the line."
10    crlf)
11   (bind ?c(read)) ; Eisagwgh arithmwn autokitwn pou perimenei
12   sthn oura
13   (if (<= ?c 0)
14     then
15       (printout t "The ammount of cars cannot be 0 or less than
16        0." crlf)
17       (assert (car (cars_left 0) (state GOAL_NOT_FOUND)))
18     else
19       (assert (car (cars_left ?c) (state GOAL_NOT_FOUND)))
20     )
21  )
22 )
```

Listing 1: Κανόνας εκκίνησης

Εδώ ο κανόνας παίρνει από τον χρηστη των αριθμό των αυτοκινήτων που περιμένει στην ουρά για να μπουν μέσα. Εάν του δωθεί αριθμός μικρότερος ή ισός του μηδενός, τότε έγινε κάποιο λάθος και το πρόγραμμα λήγει εδώ. Επιπρόσθετα, όπως είναι και σχολιασμένο, έχει προτερεότητα σε σχέση με άλλους κανόνες και τρέχει πάντα πρώτο.

3.4.2 Κανόνας goal_found

```
1 (defrule goal_found
2   (declare (salience 100))
3   ?c<-(car (cars_left 0) (state GOAL_NOT_FOUND))
4   =>
5   (modify ?c (state GOAL_FOUND))
6   (printout t "Goal found. End of program." crlf)
7 )
8
```

Listing 2: Κανόνας goal_found

Ο συγκεκριμένος κανόνας έχει ως όρισμα το car να έχει cars_left 0 και state GOAL_NOT_FOUND. Τότε, και μόνο τότε, αλλάζει το state σε GOAL_FOUND και λήγει εδώ το πρόγραμμα με το goal να έχει βρεθεί.

3.4.3 Τυπικός κανόνας κίνησης

```
1 ; Kanonas pou kounaei anatolika ton automato parkadoro
2 (defrule moveEast
3   ?z<-(space (name ?y) (state N)) ; Trexon space
4   (east ?x ?y) ; Edw allazei mono
5   ?p<-(space (name ?x) (state Y)) ; Space pou tha paei o
   parkadoros
6   ?q<-(platform (space ?x) (plat_state ?ps)) ; Platforma pou
   uparxei sto space
7   ?c<-(car (cars_left ?cars) (state GOAL_NOT_FOUND)) ; Katastasi
   autokinhtwn
8 =>
9   (modify ?p (state N)) ; Allagi space state se N gia to space
   pou tha pame
10  (modify ?z (state Y)) ; Allagi space state se Y gia to space
   pou eimastan
11  (if (eq ?ps F)
12    then
13      (modify ?q (space ?y))
14
15    else
16      (modify ?q (space ?y) (plat_state F)) ; Ean den einai
   gemato, vazei to autokinito
17      (modify ?c (cars_left (- ?cars 1))) ; Afairei 1 apo ta
   autokinhta pou apomenoun
18  )
19 )
20
```

Listing 3: Τυπικός κανόνας κίνησης

Αυτός είναι ένας τυπικός κανόνας μετακίνησης του αυτόματου παρκαδόρου (συγκεκριμένα είναι του ανατολικού αλλά δεν αλλάζει πολύ ο κάθε κωδικός). Σαν ορίσματα παίρνει το τρέχον space, το γειτονικό του και την πλατφόρμα, και πρέπει το car να έχει state GOAL_NOT_FOUND. Στην συνέχεια, θα αλλάξει το state του κάθε space, αφού το ένα θα πάρει την πλατφόρμα του άλλου. Μετά, ελέγχει εάν η πλατφόρμα είναι γεμμάτη, τότε απλώς αλλάζει την πλατφόρμα θέση, αλλιώς αλλάζει την πλατφόρμα θέση και αλλάζει το state της πλατφόρμας έτσι ώστε να είναι γεμμάτη και να αφαιρεί 1 αυτοκίνητο από το car.

3.5 Βήμα 1 source code

```
1 ; Deutero meros ergasias
2 ; Foithths: Iakovos Mastrogiannopoulos
3 ; AM: cse242017102
4 ; Programma Spoudwn: PADA (5etes)
5
6 (deftemplate space
7   (slot name)
8   (slot level)
9   (slot state)
10 )
11
12 (def facts space-init
13   (space (name s1) (level 1) (state Y))
14   (space (name s2) (level 1) (state N))
15   (space (name s3) (level 1) (state Y))
16   (space (name s4) (level 1) (state Y))
17   (space (name s5) (level 1) (state Y))
18   (space (name s6) (level 1) (state Y))
19 )
20
21 (deftemplate platform
22   (slot name)
23   (slot space)
24   (slot plat_state)
25 )
26
27 (def facts plat-init
28   (platform (name p1) (space s1) (plat_state E))
29   (platform (name p2) (space s3) (plat_state E))
30   (platform (name p3) (space s4) (plat_state E))
31   (platform (name p4) (space s5) (plat_state E))
32   (platform (name p5) (space s6) (plat_state E))
33 )
34
35 (deftemplate car
36   (slot cars_left)
37   (slot state)
38 )
39
40 ; Kanonas pou ksekinaei to programma
41 (defrule startup
42   (declare (salience 100)) ; Dinei protereotita ston kanona
43   (initial-fact)
44 =>
45   (set-strategy random)
46   (printout t "Starting the program!" crlf)
47
48   (printout t "Enter the ammount of cars waiting in the line."
49     crlf)
50   (bind ?c(read)) ; Eisagwgh arithmwn autokitwn pou perimenei
51   sthn oura
52   (if (<= ?c 0)
53     then
54     (printout t "The ammount of cars cannot be 0 or less than
55       0." crlf)
```

```

53     (assert (car (cars_left 0) (state GOAL_NOT_FOUND)))
54     else
55         (assert (car (cars_left ?c) (state GOAL_NOT_FOUND)))
56     )
57 )
58
59 ; Kanonas pou leitourgh mono ean vrethike to goal
60 (defrule goal_found
61     (declare (salience 100))
62     ?c<-(car (cars_left 0) (state GOAL_NOT_FOUND))
63 =>
64     (modify ?c (state GOAL_FOUND))
65     (printout t "Goal found. End of program." crlf)
66 )
67
68 (deffacts east
69     (east s1 s2)
70     (east s2 s3)
71     (east s4 s5)
72     (east s5 s6)
73 )
74
75 ; Kanonas pou kounaei anatolika ton automato parkadoro
76 (defrule moveEast
77     ?z<-(space (name ?y) (state N)) ; Trexon space
78     (east ?x ?y)
79     ?p<-(space (name ?x) (state Y)) ; Space pou tha paei o
    parkadoros
80     ?q<-(platform (space ?x) (plat_state ?ps)) ; Platforma pou
    uparxei sto space
81     ?c<-(car (cars_left ?cars) (state GOAL_NOT_FOUND)) ; Katastasi
    autokinhtwn
82 =>
83     (modify ?p (state N)) ; Allagi space state se N gia to space
    pou tha pame
84     (modify ?z (state Y)) ; Allagi space state se Y gia to space
    pou eimastan
85     (if (eq ?ps F)
86     then
87         (modify ?q (space ?y))
88     else
89         (modify ?q (space ?y) (plat_state F)) ; Ean den einai
    gemato, vazei to autokinito
90         (modify ?c (cars_left (- ?cars 1))) ; Afairei 1 apo ta
    autokinhtha pou apomenoun
91     )
92 )
93 )
94
95 (deffacts north
96     (north s1 s4)
97     (north s2 s5)
98     (north s3 s6)
99 )
100
101 ; Kanonas pou kounaei voreia ton automato parkadoro
102 (defrule moveNorth

```

```

103 ?z<-(space (name ?y) (state N))
104 (north ?x ?y)
105 ?p<-(space (name ?x) (state Y))
106 ?q<-(platform (space ?x) (plat_state ?ps))
107 ?c<-(car (cars_left ?cars) (state GOAL_NOT_FOUND))
108 =>
109 (modify ?p (state N))
110 (modify ?z (state Y))
111 (if (eq ?ps F)
112 then
113 (modify ?q (space ?y))
114
115 else
116 (modify ?q (space ?y) (plat_state F))
117 (modify ?c (cars_left (- ?cars 1)))
118 )
119 )
120
121 (def facts west
122 (west s2 s1)
123 (west s3 s2)
124 (west s5 s4)
125 (west s6 s5)
126 )
127
128 ; Kanonas pou kounaei dutika ton automato parkadoro
129 (defrule moveWest
130 ?z<-(space (name ?y) (state N))
131 (west ?x ?y)
132 ?p<-(space (name ?x) (state Y))
133 ?q<-(platform (space ?x) (plat_state ?ps))
134 ?c<-(car (cars_left ?cars) (state GOAL_NOT_FOUND))
135 =>
136 (modify ?p (state N))
137 (modify ?z (state Y))
138 (if (eq ?ps F)
139 then
140 (modify ?q (space ?y))
141
142 else
143 (modify ?q (space ?y) (plat_state F))
144 (modify ?c (cars_left (- ?cars 1)))
145 )
146 )
147
148 (def facts south
149 (south s4 s1)
150 (south s5 s2)
151 (south s6 s3)
152 )
153
154 ; Kanonas pou kounaei notia ton automato parkadoro
155 (defrule moveSouth
156 ?z<-(space (name ?y) (state N))
157 (south ?x ?y)
158 ?p<-(space (name ?x) (state Y))
159 ?q<-(platform (space ?x) (plat_state ?ps))

```

```

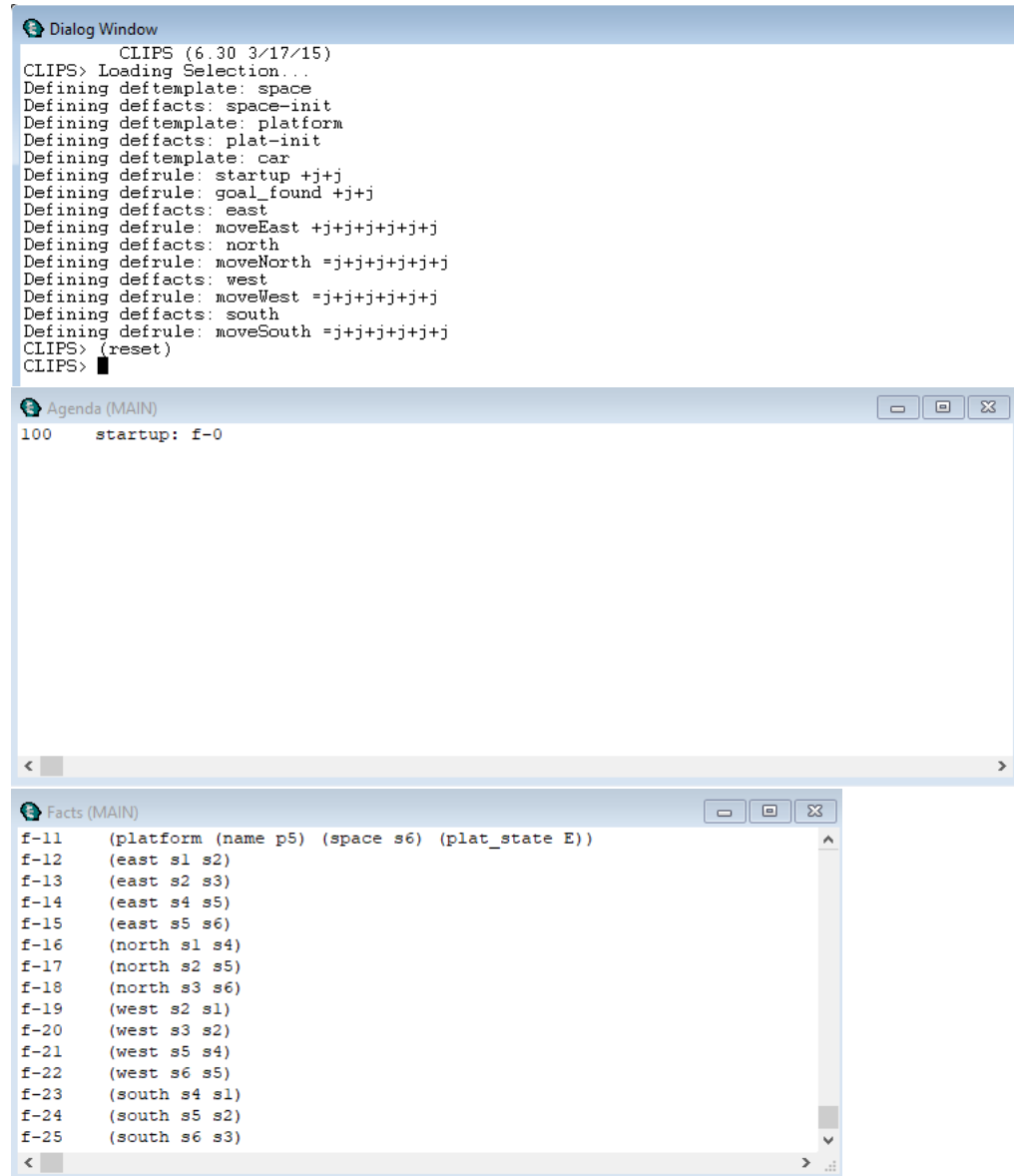
160      ?c<-(car (cars_left ?cars) (state GOAL_NOT_FOUND))
161 =>
162      (modify ?p (state N))
163      (modify ?z (state Y))
164      (if (eq ?ps F)
165          then
166              (modify ?q (space ?y))
167
168          else
169              (modify ?q (space ?y) (plat_state F))
170              (modify ?c (cars_left (- ?cars 1)))
171      )
172 )

```

Listing 4: Βήμα 1

3.6 Ενδεικτικά τρεξίματα του πρώτου βήματος

3.6.1 Πριν την εκκίνηση



The screenshot displays three windows from the CLIPS (Common LISP Integrated Programming System) environment:

- Dialog Window:** Shows the CLIPS version (6.30 3/17/15) and a series of initialization commands being executed. These include loading a selection, defining default templates (space, platform, car), default facts (space-init, plat-init, east, north, west, south), and default rules (startup, goal_found, moveEast, moveNorth, moveWest, moveSouth). The commands are prefixed with 'CLIPS>' and end with a cursor.
- Agenda (MAIN):** Displays the current state of the agenda, showing a single entry: '100 startup: f-0'.
- Facts (MAIN):** Lists the current facts in the knowledge base, numbered f-11 through f-25. Each fact is a list of arguments for a template, such as '(platform (name p5) (space s6) (plat_state E))' for f-11 and '(south s6 s3)' for f-25.

3.6.2 Κανόνας εκκίνησης

```
CLIPS> (run 1)
Starting the program!
Enter the amount of cars waiting in the line.
3
CLIPS>
f-25 (south s6 s3)
f-26 (car (cars_left 3) (state GOAL_NOT_FOUND))
```

Agenda (MAIN)

0	moveSouth: f-2,f-24,f-5,f-10,f-26
0	moveEast: f-2,f-12,f-1,f-7,f-26
0	moveWest: f-2,f-20,f-3,f-8,f-26

3.6.3 Κανόνας κίνησης

```
f-30 (car (cars_left 2) (state GOAL_NOT_FOUND))
```

3.6.4 Μετά την εκκίνηση

```
CLIPS> (run 1)
CLIPS> (run 1)
Goal found. End of program.
```

f-37	(platform (name p1) (space s4) (plat_state F))
f-39	(car (cars_left 0) (state GOAL_FOUND))

3.6.5 Σχόλια

Εδώ βλέπουμε ότι φορτώνει τα γεγονότα και τους κανόνες χωρίς πρόβλημα, διαβάζει από το πληκτρολόγιο πόσα αυτοκίνητα είναι στην λίστα, αφαιρεί ένα αμάξι από την λίστα όταν το παρκάρει στην σωστή πλατφόρμα και τελειώνει όταν όλα τα αυτοκίνητα στην λίστα έχουν παρκάρει. Σε ένα άλλο ενδεικτικό τρέξιμο, με αυτοκίνητα πάνω από 5, το πρόγραμμα μπαίνει σε loop αφού δεν υπάρχει τρόπος να παρκάρει τα έξτρα αυτοκίνητα.

4 Παρατηρήσεις ενδιάμεσα των βήματων

Από ότι φαίνεται, αυτός ο τρόπος λύσης συγκριτικά με τον άλλον τρόπο λύσης του πρώτου μερούς της εργασίας είναι τυχαίος και ίσως πιο αργός. Στην υλοποίηση, όμως, είναι πολύ πιο εύκολος, όποτε είναι πιο απλός για τον προγραμματιστή. Μπορεί να κολλήσει και να μην βρει ποτέ λύση, που δεν φαντάζομαι να είναι εφικτό αυτό στον προγραμματικό κόσμο.

5 Βήμα 2

5.1 Ζητούμενα δευτέρου βήματος

Το βήμα 2 μας ζητάει να κάνουμε μια προέκταση του πρώτου βήματος, δηλαδή θα προσθέσουμε δύο παραπάνω όρους, με ξεχωριστά spaces και platforms. Ο δεύτερος όρος έχει τα spaces s7 έως s12, όπου ο s7 και ο s12 είναι άδεια ενώ οι υπόλοιποι έχουν τις πλατφόρμες p6 έως p9. Ο τρίτος όρος έχει τα spaces s13 έως s18, όπου ο s13 είναι άδειος και οι υπόλοιποι έχουν τις πλατφόρμες p10 έως p14. Θα μπορεί να αλλάζει όροφο από τα spaces s5, s11 και το s17 με την χρήση ενός ανεγκυστήρα. Επιπρόσθετα, μας ζητάει όποτε μπαίνει ένα νέο αμάξι να δίνουμε τον αριθμό κυκλοφορίας του με στόχο να μπορούμε μετά να το εντοπίζουμε και να το ξεπαρκάρουμε με ακριβώς την αντίθετη διαδικασία.

5.2 Τι μπορούμε να χρησιμοποιήσουμε από το πρώτο βήμα

- Τα templates του πρώτου βήματος
- Τα γεγονότα του πρώτου βήματος
- Οι κανόνες του πρώτου βήματος

Παρόλου που είναι σχεδόν έτοιμα, θα πρέπει να τα τροποποιήσουμε λίγο.

5.2.1 Τροποποιημένα templates πρώτου βήματος

```
1 (deftemplate info
2   (slot cars_left)
3   (slot current_space)
4   (slot state)
5 )
6
```

Το template car μετονομάστηκε σε info.

5.2.2 Τροποποιημένα γεγονότα πρώτου βήματος

```
1 (def facts space-init
2   (space (name s1) (level 1) (state Y))
3   (space (name s2) (level 1) (state N))
4   (space (name s3) (level 1) (state Y))
5   (space (name s4) (level 1) (state Y))
6   (space (name s5) (level 1) (state Y))
7   (space (name s6) (level 1) (state Y))
8   (space (name s7) (level 2) (state N))
9   (space (name s8) (level 2) (state Y))
10  (space (name s9) (level 2) (state Y))
11  (space (name s10) (level 2) (state Y))
12  (space (name s11) (level 2) (state Y))
13  (space (name s12) (level 2) (state N))
14  (space (name s13) (level 3) (state N))
15  (space (name s14) (level 3) (state Y))
16  (space (name s15) (level 3) (state Y))
17  (space (name s16) (level 3) (state Y))
18  (space (name s17) (level 3) (state Y))
19  (space (name s18) (level 3) (state Y))
20 )
21
22 (def facts plat-init
23   (platform (name p1) (space s1) (plat_state E))
24   (platform (name p2) (space s3) (plat_state E))
25   (platform (name p3) (space s4) (plat_state E))
26   (platform (name p4) (space s5) (plat_state E))
27   (platform (name p5) (space s6) (plat_state E))
28   (platform (name p6) (space s8) (plat_state E))
29   (platform (name p7) (space s9) (plat_state E))
30   (platform (name p8) (space s10) (plat_state E))
31   (platform (name p9) (space s11) (plat_state E))
32   (platform (name p10) (space s14) (plat_state E))
33   (platform (name p11) (space s15) (plat_state E))
34   (platform (name p12) (space s16) (plat_state E))
35   (platform (name p13) (space s17) (plat_state E))
36   (platform (name p14) (space s18) (plat_state E))
37 )
38
```

5.2.3 Τροποποιημένοι κανόνες πρώτου βήματος

```
1 (defrule moveEast
2   ?z<-(space (name ?y) (state N))
3   (east ?y ?x)
4   ?p<-(space (name ?x) (state Y))
5   ?q<-(platform (name ?pn) (space ?x) (plat_state ?ps))
6   ?c<-(info (cars_left ?cars) (current_space ?y) (state
7     GOAL_NOT_FOUND)) ; Koitame kai to space pou eimaste twra
8   =>
9   (modify ?p (state N))
10  (modify ?z (state Y))
11
12  (if (eq ?ps F)
13    then
14      (modify ?q (space ?y))
15      (modify ?c (current_space ?x))
16    else
17      (printout t "Enter the license plate." crlf)
18      (bind ?lp(read))
19      (assert (car (license_plate ?lp) (platform ?pn)))
20      (modify ?q (space ?y) (plat_state F))
21      (modify ?c (cars_left (- ?cars 1)) (current_space ?x) (
22        state PROMPT)) ; State pou mas paei sto prompt
23      )
24  )
```

Listing 5: Τροποποιημένος γενικός κανόνας κίνησης

Μπορούμε να δούμε ότι έχουμε προσθέσει ένα παραπάνω slot το οποίο κοιτάει το τρέχον space στο οποίο βρίσκεται αυτή την στιγμή ο αυτόματος παρκαδόρος. Αυτό γίνεται, επειδή πλέον έχουμε παραπάνω από ένα empty space, και κάπως θα πρέπει να το ξεχωρίζουμε ποιο είναι αυτό στο οποίο βρίσκεται ο παρκαδόρος. Επιπλέον, παρατηρούμε ότι όταν βάζουμε το αυτοκίνητο, ζητάμε από τον χρήστη να μας δώσει τον αριθμό κυκλοφορίας του και επίσης αλλάζει το state σε prompt.

5.3 Templates δεύτερου βήματος

Αυτά είναι τα templates του δεύτερου βήματος:

```
1 (deftemplate car_to_find
2   (slot license_plate)
3 )
4
5 (deftemplate car
6   (slot license_plate)
7   (slot platform)
8 )
9
```

5.4 Γεγονόντα δεύτερου βήματος

Αυτά είναι τα γεγονόντα του δεύτερου βήματος:

```
1 (def facts up
2   (up s5 s11)
3   (up s11 s17)
4 )
5
6 (def facts down
7   (down s11 s5)
8   (down s17 s11)
9 )
10
```

5.5 Κανόνες δεύτερου βήματος

Τώρα θα δούμε αναλυτικά τους κανόνες του δεύτερου βήματος. Πλέον για τους κανόνες κινήσεις, αφού έχουμε πολυεπίπεδο parking με διαφόρους όροφους, δεν μας φτάνει να μπορεί να πηγαίνει ανατολικά, βορεια, νότια και δυτικά αλλά και πάνω και κάτω. Οπότε, προσθέσαμε δύο παραπάνω κανόνες κινήσεις για κάθε τυπικό κανόνα που υπάρχει.

5.5.1 Κανόνας prompt

```
1 ; Kanonas pou rwtaei ton xristi ean thelei na kseparkarei
   autokinhto
2 (defrule prompt
3   (declare (salience 100))
4   ?c<-(info (state PROMPT))
5 =>
6   (printout t "Do you want to remove a car? y/n" crlf)
7   (bind ?qu(read))
8   (if (eq ?qu y)
9     then
10      (printout t "Give license plate." crlf)
11      (bind ?lp(read)) ; Ean nai, tote pairnei to license plate
      kai arxizei na to psaxnei
12      (modify ?c (state SEARCH_CAR))
13      (assert (car_to_find (license_plate ?lp)))
14
15     else
16      (modify ?c (state BACK_TO_S2)) ; Ean oxi, tote gurnaei pish
      sto s2
17   )
18 )
19
```

Listing 6: Κανόνας prompt

Κανόνας που ρώταει από τον χρήστη εάν θέλει να ξεπαρκάρει κάποιο αυτοκίνητο. Εάν απαντήσει ναι, τότε του ζητάει τον αριθμό κυκλοφορίας του αυτοκινήτου και αλλάζει το state σε SEARCH_CAR. Η προϋπόθεση για να δουλέψει αυτός ο

κανόνας είναι να έχουμε state PROMPT, που το παίρνουμε όταν τοποθετείται τουλάχιστον ένα αυτοκίνητο.

5.5.2 Τυπικός κανόνας επιστροφής στο s2

```
1 (defrule backToS2East
2   ?z<-(space (name ?y) (state N))
3   (east ?y ?x)
4   ?p<-(space (name ?x) (state Y))
5   ?q<-(platform (space ?x))
6   ?c<-(info (state BACK_TO_S2) (current_space ?y))
7 =>
8   (modify ?p (state N))
9   (modify ?z (state Y))
10  (modify ?q (space ?y))
11  (if (eq ?x s2)
12    then
13      (modify ?c (current_space ?x) (state GOAL_NOT_FOUND))
14    else
15      (modify ?c (current_space ?x))
16  )
17 )
18 )
19
```

Listing 7: Τυπικός κανόνας επιστροφής στο s2

Αυτός ο κανόνας είναι αρκετά παρόμοιος με τον τυπικό κανόνα κίνησης, με την διαφορά ότι αντί να ψάχνει αδεία πλατφόρμα, ψάχνει το space που έχει όνομα s2. Εάν το βρει, δίνει το state GOAL_NOT_FOUND.

5.5.3 Τυπικός κανόνας εύρεσης αυτοκίνητου

```
1 (defrule findCarEast
2   ?z<-(space (name ?y) (state N))
3   (east ?y ?x)
4   ?p<-(space (name ?x) (state Y))
5   ?q<-(platform (name ?pn) (space ?x))
6   ?car<-(car (license_plate ?lp) (platform ?pn)) ; Autokinhto pou
   vriskete sthn pn platforma kai exei lp license plate
7   ?c<-(info (state SEARCH_CAR))
8   ?cf<-(car_to_find (license_plate ?lpf)) ; Euresi lpf license
   plate
9 =>
10  (modify ?p (state N))
11  (modify ?z (state Y))
12  (modify ?q (space ?y))
13  (if (eq ?lp ?lpf)
14    then
15      (retract ?car ?cf) ; Afairesh autokinhtou apo thn lista
16      (modify ?c (current_space ?x) (state BACK_TO_S2))
17    else
18      (modify ?c (current_space ?x))
19    )
20  )
21 )
22 )
```

Listing 8: Τυπικός κανόνας εύρεσης αυτοκίνητου

Εδώ ψάχνουμε το αυτοκίνητο, όπου βρίσκεται σε μία πλατφόρμα pn. Για να ξεπαρκάρι το αυτοκίνητο θα πρέπει ο αριθμός κυκλοφορίας του να είναι ίσος με τον αριθμό κυκλοφορίας από το αυτοκίνητο που ψάχνουμε. Αφού το αφαιρέσει με την χρήση της εντολής retract, θα αλλάξει το state σε BACK_TO_S2 για να γυρίσουμε πίσω στο s2.

5.6 Βήμα 2 source code

```
1 ; Deutero meros ergasias
2 ; Foithths: Iakovos Mastrogiannopoulos
3 ; AM: cse242017102
4 ; Programma Spoudwn: PADA (5etes)
5
6 (deftemplate space
7   (slot name)
8   (slot level)
9   (slot state)
10 )
11
12 (def facts space-init
13   (space (name s1) (level 1) (state Y))
14   (space (name s2) (level 1) (state N))
15   (space (name s3) (level 1) (state Y))
16   (space (name s4) (level 1) (state Y))
17   (space (name s5) (level 1) (state Y))
18   (space (name s6) (level 1) (state Y))
19   (space (name s7) (level 2) (state N))
20   (space (name s8) (level 2) (state Y))
21   (space (name s9) (level 2) (state Y))
22   (space (name s10) (level 2) (state Y))
23   (space (name s11) (level 2) (state Y))
24   (space (name s12) (level 2) (state N))
25   (space (name s13) (level 3) (state N))
26   (space (name s14) (level 3) (state Y))
27   (space (name s15) (level 3) (state Y))
28   (space (name s16) (level 3) (state Y))
29   (space (name s17) (level 3) (state Y))
30   (space (name s18) (level 3) (state Y))
31 )
32
33 (deftemplate platform
34   (slot name)
35   (slot space)
36   (slot plat_state)
37 )
38
39 (def facts plat-init
40   (platform (name p1) (space s1) (plat_state E))
41   (platform (name p2) (space s3) (plat_state E))
42   (platform (name p3) (space s4) (plat_state E))
43   (platform (name p4) (space s5) (plat_state E))
44   (platform (name p5) (space s6) (plat_state E))
45   (platform (name p6) (space s8) (plat_state E))
46   (platform (name p7) (space s9) (plat_state E))
47   (platform (name p8) (space s10) (plat_state E))
48   (platform (name p9) (space s11) (plat_state E))
49   (platform (name p10) (space s14) (plat_state E))
50   (platform (name p11) (space s15) (plat_state E))
51   (platform (name p12) (space s16) (plat_state E))
52   (platform (name p13) (space s17) (plat_state E))
53   (platform (name p14) (space s18) (plat_state E))
54 )
55
```

```

56 (deftemplate info
57   (slot cars_left)
58   (slot current_space)
59   (slot state)
60 )
61
62 (deftemplate car_to_find
63   (slot license_plate)
64 )
65
66 (deftemplate car
67   (slot license_plate)
68   (slot platform)
69 )
70
71 (defrule startup
72   (declare (salience 100))
73   (initial-fact)
74 =>
75   (set-strategy random)
76   (printout t "Starting the program!" crlf)
77
78   (printout t "Enter the ammount of cars waiting in the line."
79     crlf)
80   (bind ?c(read))
81   (if (<= ?c 0)
82     then
83     (printout t "The ammount of cars cannot be 0 or less than
84       0." crlf)
85     (assert (info (cars_left 0) (current_space s2) (state
86       GOAL_NOT_FOUND)))
87   else
88   (assert (info (cars_left ?c) (current_space s2) (state
89     GOAL_NOT_FOUND)))
90 )
91 )
92
93 (defrule goal_found
94   (declare (salience 100))
95   ?c<-(info (cars_left 0) (state GOAL_NOT_FOUND))
96 =>
97   (modify ?c (state GOAL_FOUND))
98   (printout t "Goal found. End of program." crlf)
99 )
100
101 ; Kanonas pou rwtaei ton xristi ean thelei na kseparkarei
102   autokinhto
103 (defrule prompt
104   (declare (salience 100))
105   ?c<-(info (state PROMPT))
106 =>
107   (printout t "Do you want to remove a car? y/n" crlf)
108   (bind ?qu(read))
109   (if (eq ?qu y)
110     then
111     (printout t "Give license plate." crlf)

```



```

108      (bind ?lp(read)) ; Ean nai, tote pairnei to license plate
      kai arxizei na to psaxnei
109      (modify ?c (state SEARCH_CAR))
110      (assert (car_to_find (license_plate ?lp)))
111
112      else
113      (modify ?c (state BACK_TO_S2)) ; Ean oxi, tote gurnaei pisw
      sto s2
114      )
115  )
116
117  (def facts east
118      (east s1 s2)
119      (east s2 s3)
120      (east s4 s5)
121      (east s5 s6)
122      (east s7 s8)
123      (east s8 s9)
124      (east s10 s11)
125      (east s11 s12)
126      (east s13 s14)
127      (east s14 s15)
128      (east s16 s17)
129      (east s17 s18)
130  )
131
132  (defrule moveEast
133      ?z<-(space (name ?y) (state N))
134      (east ?y ?x)
135      ?p<-(space (name ?x) (state Y))
136      ?q<-(platform (name ?pn) (space ?x) (plat_state ?ps))
137      ?c<-(info (cars_left ?cars) (current_space ?y) (state
      GOAL_NOT_FOUND)) ; Koitame kai to space pou eimaste twra
138  =>
139      (modify ?p (state N))
140      (modify ?z (state Y))
141
142      (if (eq ?ps F)
143      then
144          (modify ?q (space ?y))
145          (modify ?c (current_space ?x))
146
147      else
148          (printout t "Enter the license plate." crlf)
149          (bind ?lp(read))
150          (assert (car (license_plate ?lp) (platform ?pn)))
151          (modify ?q (space ?y) (plat_state F))
152          (modify ?c (cars_left (- ?cars 1)) (current_space ?x) (
      state PROMPT)) ; State pou mas paei sto prompt
153      )
154  )
155
156  (defrule backToS2East
157      ?z<-(space (name ?y) (state N))
158      (east ?y ?x)
159      ?p<-(space (name ?x) (state Y))
160      ?q<-(platform (space ?x))

```

```

161 ?c<-(info (state BACK_TO_S2) (current_space ?y))
162 =>
163 (modify ?p (state N))
164 (modify ?z (state Y))
165 (modify ?q (space ?y))
166 (if (eq ?x s2)
167 then
168 (modify ?c (current_space ?x) (state GOAL_NOT_FOUND))
169
170 else
171 (modify ?c (current_space ?x))
172 )
173 )
174
175 (defrule findCarEast
176 ?z<-(space (name ?y) (state N))
177 (east ?y ?x)
178 ?p<-(space (name ?x) (state Y))
179 ?q<-(platform (name ?pn) (space ?x))
180 ?car<-(car (license_plate ?lp) (platform ?pn)) ; Autokinhto pou
181 vriskete sthn pn platforma kai exei lp license plate
182 ?c<-(info (state SEARCH_CAR))
183 ?cf<-(car_to_find (license_plate ?lpf)) ; Euresi lpf license
184 plate
185 =>
186 (modify ?p (state N))
187 (modify ?z (state Y))
188 (modify ?q (space ?y))
189 (if (eq ?lp ?lpf)
190 then
191 (retract ?car ?cf) ; Afairesh autokinhtou apo thn lista
192 (modify ?c (current_space ?x) (state BACK_TO_S2))
193
194 else
195 (modify ?c (current_space ?x))
196 )
197 )
198
199 (deffacts north
200 (north s1 s4)
201 (north s2 s5)
202 (north s3 s6)
203 (north s7 s10)
204 (north s8 s11)
205 (north s9 s12)
206 (north s13 s16)
207 (north s14 s17)
208 (north s15 s18)
209 )
210
211 (defrule moveNorth
212 ?z<-(space (name ?y) (state N))
213 (north ?y ?x)
214 ?p<-(space (name ?x) (state Y))
215 ?q<-(platform (name ?pn) (space ?x) (plat_state ?ps))
216 ?c<-(info (cars_left ?cars) (current_space ?y) (state
217 GOAL_NOT_FOUND))

```

```

215 =>
216     (modify ?p (state N))
217     (modify ?z (state Y))
218
219     (if (eq ?ps F)
220     then
221         (modify ?q (space ?y))
222         (modify ?c (current_space ?x))
223
224     else
225         (printout t "Enter the license plate." crlf)
226         (bind ?lp(read))
227         (assert (car (license_plate ?lp) (platform ?pn)))
228         (modify ?q (space ?y) (plat_state F))
229         (modify ?c (cars_left (- ?cars 1)) (current_space ?x) (
state PROMPT))
230     )
231 )
232
233 (defrule backToS2North
234     ?z<-(space (name ?y) (state N))
235     (north ?y ?x)
236     ?p<-(space (name ?x) (state Y))
237     ?q<-(platform (space ?x))
238     ?c<-(info (state BACK_TO_S2) (current_space ?y))
239 =>
240     (modify ?p (state N))
241     (modify ?z (state Y))
242     (modify ?q (space ?y))
243     (if (eq ?x s2)
244     then
245         (modify ?c (current_space ?x) (state GOAL_NOT_FOUND))
246
247     else
248         (modify ?c (current_space ?x))
249     )
250 )
251
252 (defrule findCarNorth
253     ?z<-(space (name ?y) (state N))
254     (north ?y ?x)
255     ?p<-(space (name ?x) (state Y))
256     ?q<-(platform (name ?pn) (space ?x))
257     ?car<-(car (license_plate ?lp) (platform ?pn))
258     ?c<-(info (state SEARCH_CAR))
259     ?cf<-(car_to_find (license_plate ?lpf))
260 =>
261     (modify ?p (state N))
262     (modify ?z (state Y))
263     (modify ?q (space ?y))
264     (if (eq ?lp ?lpf)
265     then
266         (retract ?car ?cf)
267         (modify ?c (current_space ?x) (state BACK_TO_S2))
268
269     else
270         (modify ?c (current_space ?x))

```

```

271 )
272 )
273
274 (deffacts west
275   (west s2 s1)
276   (west s3 s2)
277   (west s5 s4)
278   (west s6 s5)
279   (west s8 s7)
280   (west s9 s8)
281   (west s11 s10)
282   (west s12 s11)
283   (west s14 s13)
284   (west s15 s14)
285   (west s17 s16)
286   (west s18 s17)
287 )
288
289 (defrule moveWest
290   ?z<-(space (name ?y) (state N))
291   (west ?y ?x)
292   ?p<-(space (name ?x) (state Y))
293   ?q<-(platform (name ?pn) (space ?x) (plat_state ?ps))
294   ?c<-(info (cars_left ?cars) (current_space ?y) (state
GOAL_NOT_FOUND))
295 =>
296   (modify ?p (state N))
297   (modify ?z (state Y))
298
299   (if (eq ?ps F)
300   then
301     (modify ?q (space ?y))
302     (modify ?c (current_space ?x))
303
304   else
305     (printout t "Enter the license plate." crlf)
306     (bind ?lp(read))
307     (assert (car (license_plate ?lp) (platform ?pn)))
308     (modify ?q (space ?y) (plat_state F))
309     (modify ?c (cars_left (- ?cars 1)) (current_space ?x) (
state PROMPT))
310   )
311 )
312
313 (defrule backToS2West
314   ?z<-(space (name ?y) (state N))
315   (west ?y ?x)
316   ?p<-(space (name ?x) (state Y))
317   ?q<-(platform (space ?x))
318   ?c<-(info (state BACK_TO_S2) (current_space ?y))
319 =>
320   (modify ?p (state N))
321   (modify ?z (state Y))
322   (modify ?q (space ?y))
323   (if (eq ?x s2)
324   then
325     (modify ?c (current_space ?x) (state GOAL_NOT_FOUND))

```

```

326
327     else
328         (modify ?c (current_space ?x))
329     )
330 )
331
332 (defrule findCarWest
333     ?z<-(space (name ?y) (state N))
334     (west ?y ?x)
335     ?p<-(space (name ?x) (state Y))
336     ?q<-(platform (name ?pn) (space ?x))
337     ?car<-(car (license_plate ?lp) (platform ?pn))
338     ?c<-(info (state SEARCH_CAR))
339     ?cf<-(car_to_find (license_plate ?lpf))
340 =>
341     (modify ?p (state N))
342     (modify ?z (state Y))
343     (modify ?q (space ?y))
344     (if (eq ?lp ?lpf)
345     then
346         (retract ?car ?cf)
347         (modify ?c (current_space ?x) (state BACK_T0_S2))
348     else
349         (modify ?c (current_space ?x))
350     )
351 )
352 )
353
354 (deffacts south
355     (south s4 s1)
356     (south s5 s2)
357     (south s6 s3)
358     (south s10 s7)
359     (south s11 s8)
360     (south s12 s9)
361     (south s16 s13)
362     (south s17 s14)
363     (south s18 s15)
364 )
365
366 (defrule moveSouth
367     ?z<-(space (name ?y) (state N))
368     (south ?y ?x)
369     ?p<-(space (name ?x) (level 1) (state Y))
370     ?q<-(platform (name ?pn) (space ?x) (plat_state ?ps))
371     ?c<-(info (cars_left ?cars) (current_space ?y) (state
GOAL_NOT_FOUND))
372 =>
373     (modify ?p (state N))
374     (modify ?z (state Y))
375
376     (if (eq ?ps F)
377     then
378         (modify ?q (space ?y))
379         (modify ?c (current_space ?x))
380     else
381

```

```

382         (printout t "Enter the license plate." crlf)
383         (bind ?lp(read))
384         (assert (car (license_plate ?lp) (platform ?pn)))
385         (modify ?q (space ?y) (plat_state F))
386         (modify ?c (cars_left (- ?cars 1)) (current_space ?x) (
state PROMPT))
387     )
388 )
389
390 (defrule findCarSouth
391     ?z<-(space (name ?y) (state N))
392     (south ?y ?x)
393     ?p<-(space (name ?x) (state Y))
394     ?q<-(platform (name ?pn) (space ?x))
395     ?car<-(car (license_plate ?lp) (platform ?pn))
396     ?c<-(info (state SEARCH_CAR))
397     ?cf<-(car_to_find (license_plate ?lpf))
398 =>
399     (modify ?p (state N))
400     (modify ?z (state Y))
401     (modify ?q (space ?y))
402     (if (eq ?lp ?lpf)
403     then
404         (retract ?car ?cf)
405         (modify ?c (current_space ?x) (state BACK_TO_S2))
406     else
407         (modify ?c (current_space ?x))
408     )
409 )
410 )
411
412 (defrule backToS2South
413     ?z<-(space (name ?y) (state N))
414     (south ?y ?x)
415     ?p<-(space (name ?x) (state Y))
416     ?q<-(platform (space ?x))
417     ?c<-(info (state BACK_TO_S2) (current_space ?y))
418 =>
419     (modify ?p (state N))
420     (modify ?z (state Y))
421     (modify ?q (space ?y))
422     (if (eq ?x s2)
423     then
424         (modify ?c (current_space ?x) (state GOAL_NOT_FOUND))
425     else
426         (modify ?c (current_space ?x))
427     )
428 )
429 )
430
431 (def facts up
432     (up s5 s11)
433     (up s11 s17)
434 )
435
436 (defrule moveUp
437     ?z<-(space (name ?y) (state N))

```

```

438     (up ?y ?x)
439     ?p<-(space (name ?x) (state Y))
440     ?q<-(platform (name ?pn) (space ?x) (plat_state ?ps))
441     ?c<-(info (cars_left ?cars) (current_space ?y) (state
GOAL_NOT_FOUND))
442 =>
443     (modify ?p (state N))
444     (modify ?z (state Y))
445
446     (if (eq ?ps F)
447     then
448         (modify ?q (space ?y))
449         (modify ?c (current_space ?x))
450
451     else
452         (printout t "Enter the license plate." crlf)
453         (bind ?lp(read))
454         (assert (car (license_plate ?lp) (platform ?pn)))
455         (modify ?q (space ?x) (plat_state F))
456         (modify ?c (cars_left (- ?cars 1)) (current_space ?x) (
state PROMPT))
457     )
458 )
459
460 (defrule findCarUp
461     ?z<-(space (name ?y) (state N))
462     (up ?y ?x)
463     ?p<-(space (name ?x) (state Y))
464     ?q<-(platform (name ?pn) (space ?x))
465     ?car<-(car (license_plate ?lp) (platform ?pn))
466     ?c<-(info (state SEARCH_CAR))
467     ?cf<-(car_to_find (license_plate ?lpf))
468 =>
469     (modify ?p (state N))
470     (modify ?z (state Y))
471     (modify ?q (space ?y))
472     (if (eq ?lp ?lpf)
473     then
474         (retract ?car ?cf)
475         (modify ?c (current_space ?x) (state BACK_TO_S2))
476
477     else
478         (modify ?c (current_space ?x))
479     )
480 )
481
482 (def facts down
483     (down s11 s5)
484     (down s17 s11)
485 )
486
487 (defrule moveDown
488     ?z<-(space (name ?y) (state N))
489     (down ?y ?x)
490     ?p<-(space (name ?x) (state Y))
491     ?q<-(platform (name ?pn) (space ?x) (plat_state ?ps))

```

```

492 ?c<-(info (cars_left ?cars) (current_space ?y) (state
GOAL_NOT_FOUND))
493 =>
494 (modify ?p (state N))
495 (modify ?z (state Y))
496
497 (if (eq ?ps F)
498 then
499 (modify ?q (space ?y))
500 (modify ?c (current_space ?x))
501
502 else
503 (printout t "Enter the license plate." crlf)
504 (bind ?lp(read))
505 (assert (car (license_plate ?lp) (platform ?pn)))
506 (modify ?q (space ?y) (plat_state F))
507 (modify ?c (cars_left (- ?cars 1)) (current_space ?x) (
state PROMPT))
508 )
509 )
510
511 (defrule findCarDown
512 ?z<-(space (name ?y) (state N))
513 (down ?x ?y)
514 ?p<-(space (name ?x) (state Y))
515 ?q<-(platform (name ?pn) (space ?x))
516 ?car<-(car (license_plate ?lp) (platform ?pn))
517 ?c<-(info (state SEARCH_CAR))
518 ?cf<-(car_to_find (license_plate ?lpf))
519 =>
520 (modify ?p (state N))
521 (modify ?z (state Y))
522 (modify ?q (space ?y))
523 (if (eq ?lp ?lpf)
524 then
525 (retract ?car ?cf)
526 (modify ?c (current_space ?x) (state BACK_TO_S2))
527
528 else
529 (modify ?c (current_space ?x))
530 )
531 )

```

Listing 9: Βήμα 2

5.7 Ενδεικτικά τρεξίματα του δεύτερου βήματος

5.7.1 Κανόνας κίνησης

```
CLIPS> (run 1)
Enter the license plate.
de
```

5.7.2 Κανόνας prompt

```
CLIPS> (run 1)
Do you want to remove a car? y/n
y
Give license plate.
de
CLIPS> █
```

6 Σχολιασμός - Βελτιώσεις

Επειδή και στα δύο βήματα έχουμε τυχαία στρατηγική, πολλές φορές δεν λείγει το πρόγραμμα, ενώ άλλες φορές λείγει.

Από πλευράς βελτιώσεων, θα μπορούσα να χρησιμοποιούσα καλύτερη ονοματολογία των μεταβλητών.