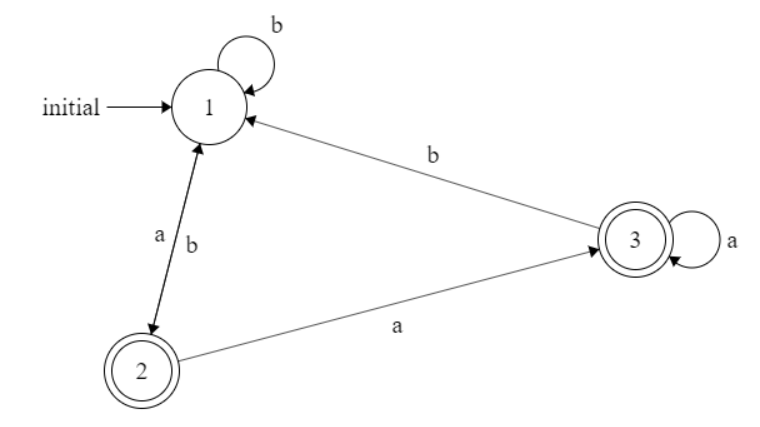
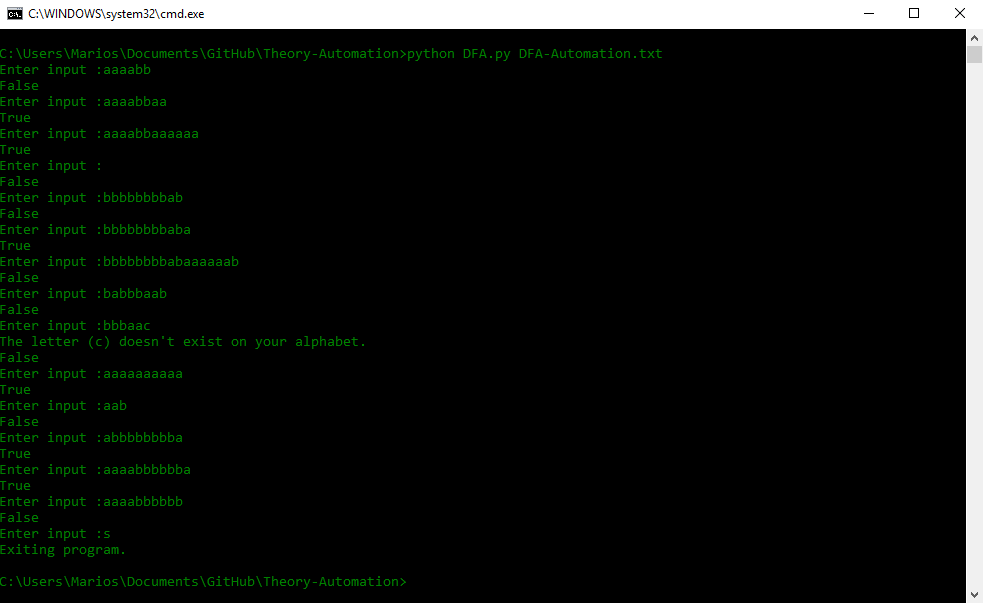
1η εργασία αυτόματα δημιουργία αυτομάτου μη ντετερμινιστικόυ με ε-μεταβάσεις Μάριος Μαρίνος(dai17147)

# Ενδεικτικά screenshots από ντετερμενιστικό, μη και με ε-μεταβάσεις.

1. Ντενερμινιστικο αυτόματο (DFA) . Με το a πάμε από την κατάσταση 1 στην 2 και με το b από την 2 στην 1. (DFA-Automation.txt)





1. Μη ντετερμινιστικό αυτόματο χωρίς e-μεταβάσεις(NFA-Solved3.txt).

**q**

**p**

**r**

**s**

**0**

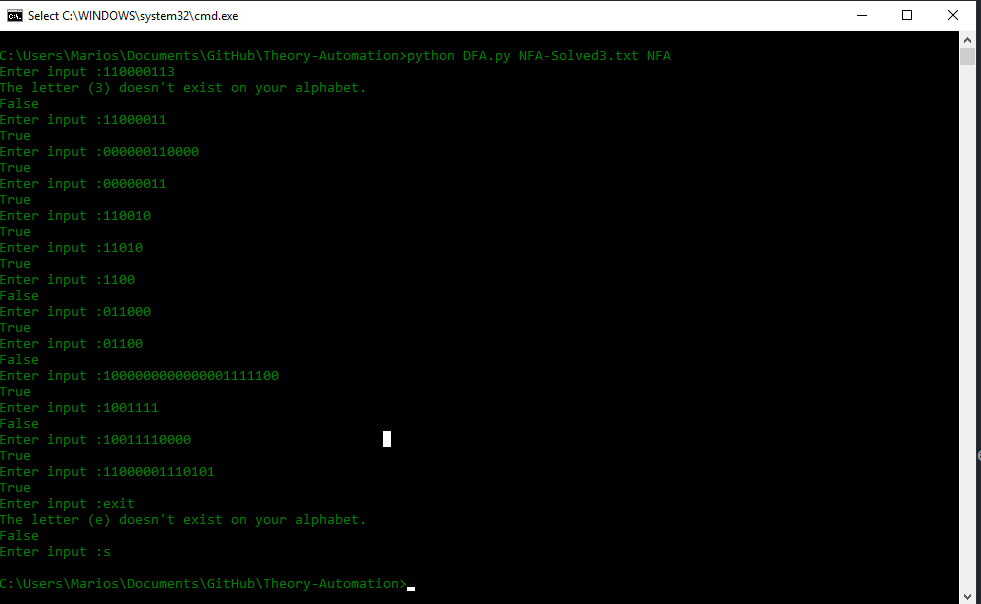
**0,1**

**0,1**

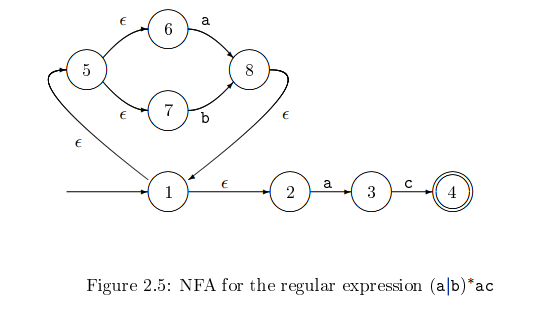
**0**

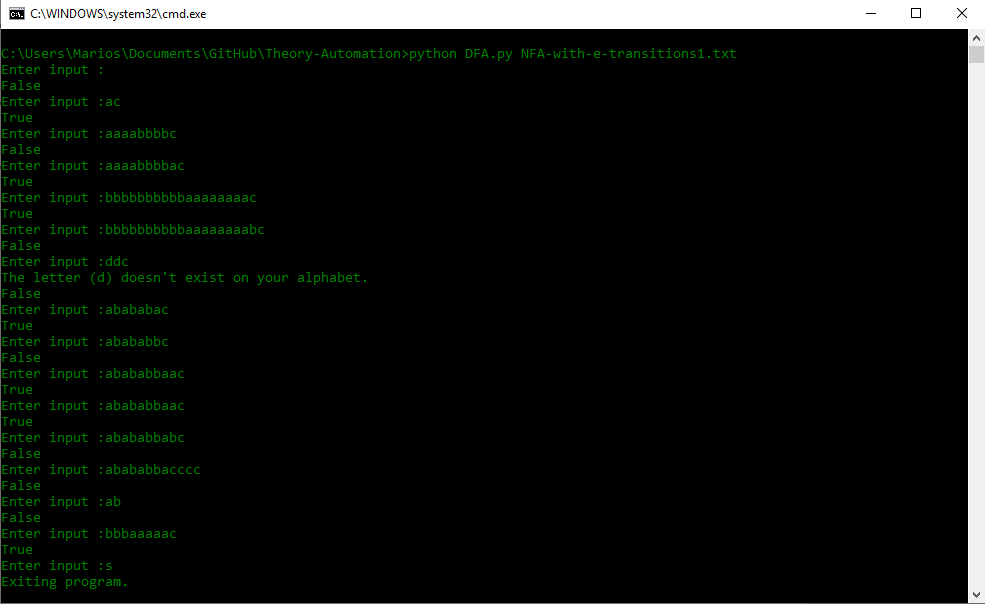
**0,1**

Παρατηρούμε οτι η γλώσσα που είναι η εξής όσα 0 ή 1 θέλουμε έπειτα 0 έπειτα 0 ή 1 έπειτα πάλί 0 και τέλος πάλι όσα 0 ή 1 θέλουμε. Άρα (0+1)\*0(0+1)0(0+1)\*

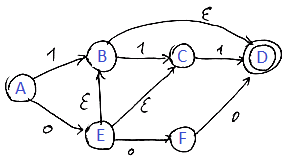


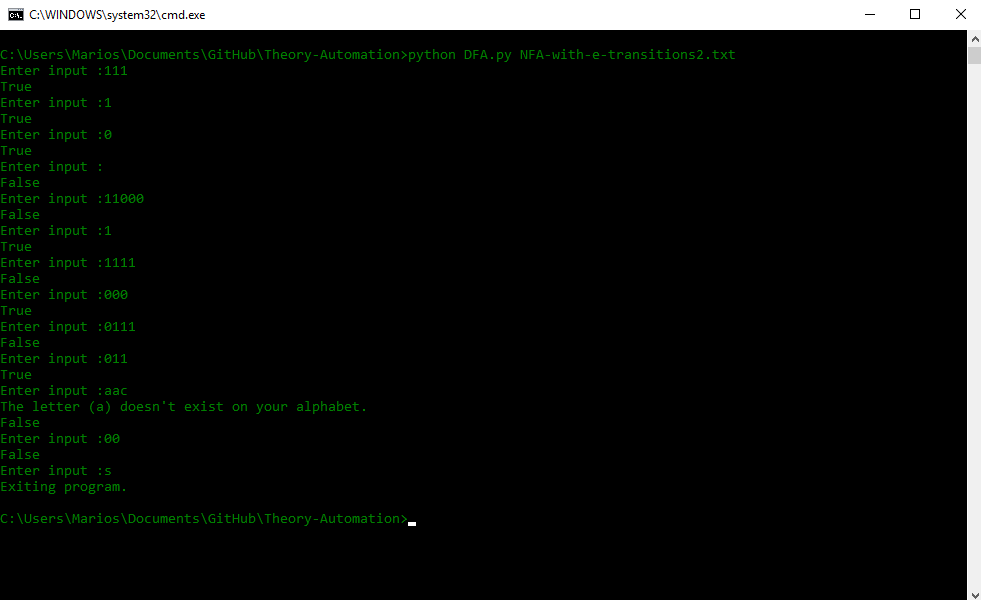
1. Μη ντετερμινιστικά αυτόματα με e-μεταβάσεις.
2. Το παρακάτω(NFA-with-e-transitions1.txt) όπως δείχνει και στην εικόνα αναγνωρίζει λέξεις της γλώσσας (a+b)\*ac.



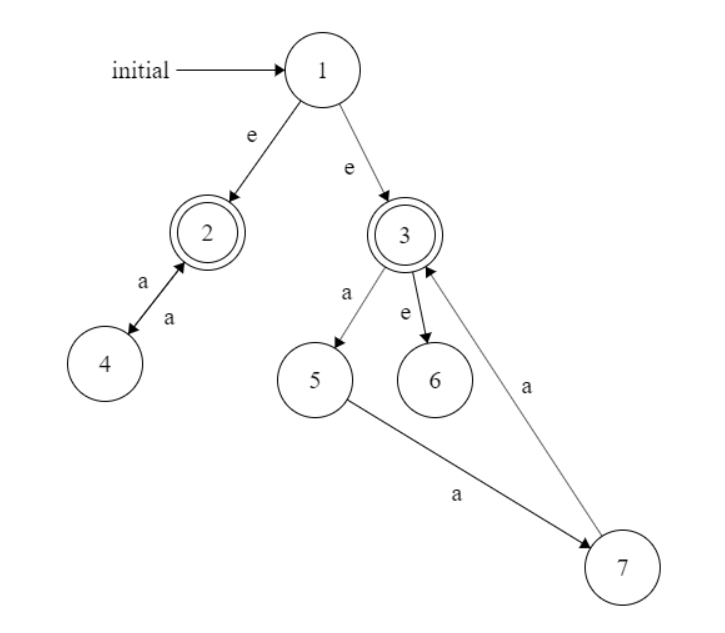


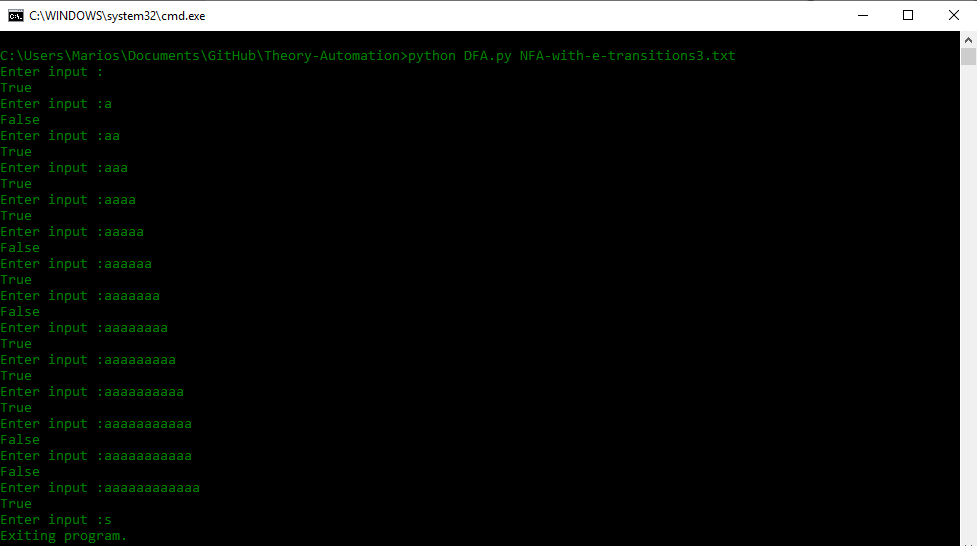
1. Ένα δεύτερο(NFA-with-e-transitions2.txt) παράδειγμα με e-transitions αυτόματο που αναγνωρίζει λέξεις.





1. Και ένα τρίτο(NFA-with-e-transitions3.txt) και τελευταίο το οποιό έχει και ως αρχική κατάσταση τελική άρα δέχεται και την κενή λέξη. Το αυτόματο αναγωνρίζει γλώσσες που αριθμός των γραμμάτων a είναι ζυγός ή διαιρείται με το 3.





## Τεκμηρίωση κώδικα και δομές δεδομένων που χρησιμοποιήθηκαν.

Ο χρήστης πρέπει να τρέξει το πρόγραμμα από την γραμμή εντολών ως εξής : python DFA.py <name\_file.txt>. Αρχικά, το πρόγραμμα περιμένει από τον χρηστή να εισάγει μία λέξη για να ελέξει το αυτόματο αν ανήκει ή όχι στην γλώσσα που περιγράφτηκε στο αρχείο. Στο αρχείο επίσης έχω εισάγει πριν τους αριθμούς των καταστάσεων, αρχικής κλπ το που αντιστοιχεί η κάθε γραμμή. Για παράδειγμα, στην 1η γραμμή αντί για το 3 σκέτο έχω states 3, στην 2η αντί για 1 έχω initial 1 κοκ. Το πρόγραμμα σταματάει μετά την εισαγωγή του γράμματος ‘s’ από το stop, υποθέτωντας οτι δεν μπορεί να το έχει μια γλώσσα αλλά θα μπορούσε εύκολα να ήταν κάποιο άλλο σύμβολο πχ το #. Για την είσοδο της κενής λέξης εισάγεται ένα space και πατήστε enter. Ο αλγόριθμος αρχικά έχει υλοποιηθεί ως κλάση και δέχεται ως εισόδους 5 παραμέτρους που είναι όλες οι καταστάσεις, οι μεταβάσεις, η αρχική, οι τελικές και των αριθμό των μεταβάσεων. Για τις καταστάσεις χρησιμοποιήθηκε structure dictionary της python το οποιό έχει ως κλειδί την αρχική κατάσταση και το στοιχείο με το οποιό πάει στις επόμενες καταστάσεις και δεξιά σαν τιμή έχω ένα set με τις επόμενες καταστάσεις. Για παράδειγμα σε ένα μη ντεντερμινιστικό αυτόματο με ε-μεταβάσεις όπως το παράδειγμα 3 από την 1 με ε πάει στην 2 και στην 3, άρα στο dictionary θα έχουμε (1, ‘@’) : {2,3}. Αφού έχει δημιουργηθεί το αντικείμενο (δηλ το συγκεκριμένο αυτόματο που περιγράψαμε) με τις παραπάνω παραμέτρους, και αφού διαβάσει την λέξη από την χρηστή καλεί μία μέθοδο του αυτομάτου run\_with\_input\_list(self, input\_list) που δέχεται σαν είσοδο την λέξη. Αρχικά θέτει ως τρέχουσα κατάσταση την αρχική κατάσταση που έχει από την είσοδο και μετέπειτα υπολογίζει το αλφάβητο της γλώσσας βάζοντας μέσα και την κενή λέξη είτε είναι είτε όχι(ο έλεγχος για το αν δέχεται την κενή λέξη το αυτόματο γίνεται παρακάτω). Αμέσως μετά μπαίνει σε μία for loop για και για κάθε ένα γράμμα της λέξης αρχικά ελέγχει αν το γράμμα είναι στην αλφάβητο και αν δεν είναι επιστρέφει False και περιμένει την επόμενη λέξη, αλλιώς καλεί την μέθοδο transition\_to\_state\_with\_input(inp) όπου inp το τρέχον γράμμα της λέξης. Τώρα αρχικά δημιουργείται ένα set που ονομάζεται next\_states και θα περιέχει όλες τις επόμενες καταστάσεις από την τρέχον κατάσταση. Έπειτα υπάρχει μία επανάληψη for που επαναλμβάνεται **για κάθε** κατάσταση που είμαστε στην τωρινή κατάσταση μιας και μιλάμε για μη ντετερμινιστικό αυτόματο και μπορουμε να βρισκόμαστε σε πολλές καταστάσεις την ίδια χρονική στιγμή. Αρχικά δημιουργώ 2 ακόμη sets τα οποιά ονομάζω e\_closure\_states και last\_used\_set αντίστοιχα και προσθέτω την κατάσταση στο set e\_closure\_states. Επίσης υπολογίζω και ένα set\_difference = e\_closure\_states - last\_used\_set για εξοικονόμηση χρόνου ώστε να μην ελέγχω τις ίδιες ε-μεταβάσεις σε κάθε επανάληψη. Αμέσως μετά υπάρχει μια while bool(set\_difference) που ουσιαστικά επαναλαμβάνεται όσο υπάρχουν στοιχεία στο set\_difference και αυτό που κάνει πρακτικά είναι να υπολογίζει το σύνολο **ΟΛΩΝ** των επόμενων καταστάσεων που μπορούμε να βρεθούμε με ε-μεταβάσεις μέχρις ότου δεν υπάρχουν άλλες και τις προσθέτει στο σύνολο e\_closure\_states. Αφού τελειώσει και δεν υπάρχει άλλη ε-μετάβαση υπάρχει μία for loop η οποία για κάθε κατάσταση που είναι στο e\_closure\_states ελέγχει αν υπάρχει κάποια μετάβαση σε άλλη κατάσταση με το τρέχον γράμμα (input\_value) και την/τις προσθέτει στο σύνολο next\_states. Τέλος τα προσθέτει στις τρέχον καταστάσεις εκτός αν έχουμε είσοδο την κενή λέξη ‘ ‘ τότε θα προσθέσει τις καταστάσεις που είναι στο σύνολο e\_closure\_states αφού δεν έχουμε κάποια μετάβαση κάπου άρα αν υπάρχει τελική κατάσταση θα βρίσκεται στο σύνολο των ε-μεταβάσεων. Η παραπάνω διαδικασία επαναλαμβάνεται για όλες τις καταστάσεις των τρέχων καταστάσεων και έπειτα για κάθε γράμμα της λέξης. Αφού έχει διαβαστεί όλη η λέξη και έχει πραγματοιποιηθεί η παραπάνω διαδικασία για όλα τα γράμματα τότε καλούμε την μέθοδο in\_accept\_state() που απλά ελέγχει αν κάποια από τις τρέχον καταστάσεις ανήκει στις τελικές ώστε να επιστρέψει True αλλιώς επιστρέφει False.

**Μάριος Μαρίνος (Dai17147).**