ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΜΕΡΟΣ Α2

ΚΩΔΙΚΟΠΟΙΗΣΗ ΑΥΤΟΜΑΤΩΝ ΠΕΠΕΡΑΣΜΕΝΩΝ ΚΑΤΑΣΤΑΣΕΩΝ ΜΕΣΩ FSM

ΣΤΟΙΧΕΙΑ ΟΜΑΔΑΣ / ΕΡΓΑΣΙΑΣ

ΑΡΙΘΜΟΣ ΟΜΑΔΑΣ: 2
ΣΤΟΙΧΕΙΑ ΦΟΙΤΗΤΗ 1: Αθανασίου Βασίλειος Ευάγγελος (ΠΑΔΑ-19390005)
ΣΤΟΙΧΕΙΑ ΦΟΙΤΗΤΗ 2: Θεοχάρης Γεώργιος (ΠΑΔΑ-19390283)
ΣΤΟΙΧΕΙΑ ΦΟΙΤΗΤΗ 3: Τάτσης Παντελής (ΠΑΔΑ-20390226)
ΣΤΟΙΧΕΙΑ ΦΟΙΤΗΤΗ 4: Ηλιού Ιωάννης (ΠΑΔΑ-19390066)
ΣΤΟΙΧΕΙΑ ΦΟΙΤΗΤΗ 5: Δομινάρης Βασίλειος (ΠΑΔΑ-21390055)

ΤΜΗΜΑ ΕΡΓΑΣΤΗΡΙΟΥ: Β1 Τετάρτη 12:00-14:00
ΥΠΕΥΘΥΝΟΣ ΕΡΓΑΣΤΗΡΙΟΥ: Ιορδανάκης Μιχάλης
ΠΡΟΘΕΣΜΙΑ ΥΠΟΒΟΛΗΣ: 07/05/2024
ΗΜΕΡΟΜΗΝΙΑ ΟΛΟΚΛΗΡΩΣΗΣ: 07/05/2024

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

ΠΕΡΙΕΧΟΜΕΝΑ

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

Εισαγωγή

Την συγγραφή την ανέλαβε ο φοιτητής

Αθανασίου Βασίλειος Ευάγγελος (ΠΑΔΑ-19390005)

Αναφορά

Το μέρος “A2 Κωδικοποίηση αυτομάτων πεπερασμένων καταστάσεων μέσω FSM” αποτελεί το 1ο μέρος της εργασίας “Δημιουργία Ανεξάρτητου Λεκτικού Αναλυτή με τη γεννήτρια FLEX” και παρουσιάζει μέσα από βήματα που αναλύονται διεξοδικά στην παρούσα αναφορά, την κωδικοποίηση αυτόματων πεπερασμένων καταστάσεων για τις λεκτικές μονάδες της γλώσσας Uni-C (η γλώσσα Uni-C αποτελεί ένα υποσύνολο της γλώσσας προγραμματισμού C). Στο τέλος, προσομοιώνεται και το ενιαίο αυτόματο.

Οργάνωση ομάδας

Μέσα από το κλίμα συνεργασίας, τα μέλη της ομάδας οργανώθηκαν και οι εργασίες μοιράστηκαν ισομερώς ώστε ο φόρτος εργασίας να είναι δίκαιος και ωφέλιμος για όλους με απώτερο σκοπό να αποκτήσουν όλοι σε ικανοποιητικό βαθμό τις γνώσεις και τις δεξιότητες που παρέχει η υλοποίηση του μέρους αυτού. Η ομάδα χωρίστηκε σε υπο-ομάδες, οι οποίες ανέλαβαν κάθε μία χωριστά, την προσομοίωση των αντίστοιχων επιμέρους λεκτικών μονάδων της Uni-C. Τέλος, όλη η ομάδα ανασυγκροτείται για την προσομοίωση του ενιαίου αυτόματου, με απώτερο σκοπό να καλυφθούν τυχόν δυσκολίες που αντιμετωπίστηκαν κατά την διάρκεια της προσομοίωσης των επιμέρους.

Περιεχόμενο αναφοράς

Η παρουσίαση περιλαμβάνει την εκτέλεση μιας σειράς από πεπερασμένα βήματα που περιγράφουν τις λεκτικές μονάδες της γλώσσας Uni-C και είναι τα εξής:

1. Αναφορά του πρότυπου αναγνώρισης
2. Σχεδίαση κανονικών εκφράσεων
3. Σχεδίαση αυτόματων πεπερασμένων καταστάσεων
4. Σχεδίαση πινάκων μεταβάσεων
5. Κωδικοποίηση των αυτόματων πεπερασμένων καταστάσεων μέσω του μετα-εργαλείου FSM
6. Σημείωση περιπτώσεων ελέγχου μέσα από εξαντλητικές δοκιμαστικές εκτελέσεις

Επιπρόσθετα, για κάθε λεκτική μονάδα αναγράφονται:

1. Ποια υπο-ομάδα υλοποίησε την προσομοίωση
2. Σχολιασμός των περιπτώσεων ελέγχου
3. Τυχόν δυσκολίες που προέκυψαν και τρόποι αντιμετώπισης
4. Αναφορά τυχόν ελλείψεων και ορθότητας ως προς την εκτέλεση του κώδικα

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

Ανάλυση αρμοδιοτήτων/ρόλων όλων των μελών της ομάδας

Αθανασίου Βασίλειος Ευάγγελος (ΠΑΔΑ-19390005)

Στον ρόλο του γενικού συντονιστή, ο φοιτητής συμμετείχε στις εργασίες:

- Εισαγωγή
- 3. Ακέραιοι
- 4. Αριθμοί κινούμενης υποδιαστολής
- Ενιαίο

Ήταν συντονιστής στην οργάνωση της ομάδας για τον σχεδιασμό των πεπερασμένων αυτομάτων και για τη συγγραφή της αναφοράς.

Θεοχάρης Γεώργιος (ΠΑΔΑ-19390283)

Στον ρόλο του προγραμματιστή, ο φοιτητής συμμετείχε στις εργασίες:

- 1. Αναγνωρίσματα
- 3. Ακέραιοι
- 5. Σχόλια
- Ενιαίο

Ήταν συντονιστής στην οργάνωση της ομάδας για την κωδικοποίηση των αυτόματων πεπερασμένων καταστάσεων μέσω του μετα-εργαλείου FSM και για την σχεδίαση των πινάκων μεταβάσεων.

Τάτσης Παντελής (ΠΑΔΑ-20390226)

Στον ρόλο του δοκιμαστή (tester), ο φοιτητής συμμετείχε στις εργασίες:

- 2. Λεκτικά κυριολεκτικά
- 3. Ακέραιοι
- 6. White_spaces χαρακτήρες
- Ενιαίο

Ήταν συντονιστής στην οργάνωση της ομάδας για την επιλογή κατάλληλων περιπτώσεων ελέγχου για την αξιοπιστία της κωδικοποίησης των αυτόματων πεπερασμένων, καθώς και στον σχολιασμό των αποτελεσμάτων.

Ηλιού Ιωάννης (ΠΑΔΑ-19390066)

Στον ρόλο του σχολιαστή, ο φοιτητής συμμετείχε στις εργασίες:

- 1. Αναγνωρίσματα
- 2. Λεκτικά κυριολεκτικά

- 4. Αριθμοί κινούμενης υποδιαστολής
- Ενιαίο

Ήταν συντονιστής στην οργάνωση της ομάδας για τον σχολιασμό των αποτελεσμάτων των περιπτώσεων ελέγχου, καθώς και στην κατάλληλη επιλογή τους για την αξιοπιστία της κωδικοποίησης των αυτόματων πεπερασμένων.

Δομινάρης Βασίλειος (ΠΑΔΑ-21390055)

Στον ρόλο του αναλυτή, ο φοιτητής συμμετείχε στις εργασίες:

- 2. Λεκτικά κυριολεκτικά
- 5. Σχόλια
- 6. White_spaces χαρακτήρες
- Ενιαίο

Ήταν συντονιστής στην οργάνωση για την σχεδίαση των κανονικών εκφράσεων

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

1. Αναγνωριστικά (ονόματα)

Την προσομοίωση την ανέλαβαν οι φοιτητές

Θεοχάρης Γεώργιος (ΠΑΔΑ-19390283) – Ηλιού Ιωάννης (ΠΑΔΑ-19390066)

1.1 Πρότυπο αναγνώρισης

Η λειτουργία του αυτομάτου είναι να αναγνωρίζει αναγνωρίσματα που υποστηρίζει η γλώσσα Uni-C με βάση ένα πρότυπο αναγνώρισης.

Το πρότυπο αναγνώρισης αναγνωρίζει τα λεξήματα που αποτελούνται από ένα ή περισσότερα πεζά/κεφαλαία λατινικά γράμματα ή και από την κάτω παύλα (_) και, εκτός από τον αρχικό χαρακτήρα, από αριθμούς.

Επιπρόσθετα, επιστρέφονται ως tokens με το αναγνωριστικό όνομα ***identifier***.

1.2 Κανονική Έκφραση

Η κανονική έκφραση για τις λεκτικές μονάδες που αντιστοιχούν σε αλφαριθμητικές μεταβλητές είναι η εξής:

```
[a-zA-Z_][a-zA-Z0-9_]{0,31}
```

Αυτή η κανονική έκφραση χρησιμοποιείται για να αντιστοιχίσει ένα αλφαριθμητικό που αποτελείται από ένα γράμμα (κεφαλαίο ή μικρό), κάτω παύλα ή αριθμούς, με τουλάχιστον έναν χαρακτήρα και έως 32 χαρακτήρες.

Συγκεκριμένα, η έκφραση αυτή σημαίνει:

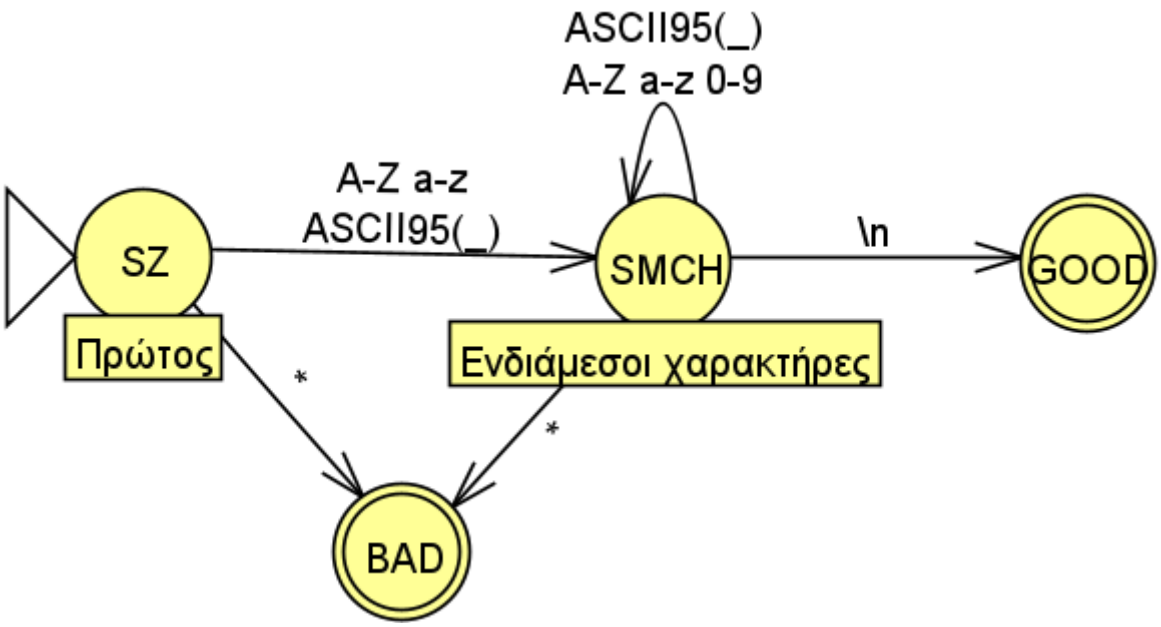
```
[a-zA-Z_]
```

Ένα γράμμα (κεφαλαίο ή μικρό) ή τον χαρακτήρα _ (κάτω παύλα).

```
[a-zA-Z0-9_]{0,31}
```

Ακολουθούμενος από μηδέν έως 31 χαρακτήρες, οι οποίοι μπορεί να είναι γράμματα (κεφαλαία ή μικρά), αριθμοί ή ο χαρακτήρας _ (κάτω παύλα).

1.3 Αυτόματο πεπερασμένης κατάστασης ή ΔΜ



Εικόνα 1.3.1 Πεπερασμένο αυτόματο αναγνωριστικών της γλώσσας Uni-C

Αρχική κατάσταση (SZ) για τον πρώτο χαρακτήρα:

Αν ο επόμενος χαρακτήρας είναι γράμμα (κεφαλαίο ή μικρό) ή κάτω παύλα (_), μετάβαση στην κατάσταση SMCH.

Αν ο επόμενος χαρακτήρας είναι οποιοσδήποτε άλλος χαρακτήρας, μετάβαση στην κατάσταση εξόδου BAD.

Κατάσταση SMCH για τους ενδιάμεσους χαρακτήρες:

Αν ο επόμενος χαρακτήρας είναι γράμμα (κεφαλαίο ή μικρό), κάτω παύλα (_) ή ψηφίο, παραμένουμε στην ίδια κατάσταση (SMCH).

Αν ο επόμενος χαρακτήρας είναι αλλαγή γραμμής (\n), το αναγνωριστικό είναι έγκυρο, οπότε μετάβαση στην κατάσταση εξόδου GOOD.

Αν ο επόμενος χαρακτήρας είναι οποιοσδήποτε άλλος χαρακτήρας, το αναγνωριστικό δεν είναι έγκυρο, οπότε μετάβαση στην κατάσταση εξόδου BAD.

Κατάσταση εξόδου GOOD:

Επιτυχής αναγνώριση αναγνωριστικού.

1.4 Πίνακας μεταβάσεων

	a-z	A-Z	0-9	-	/n	ΚΑΤΑΣΤΑΣΗ ΕΞΟΔΟΥ
SZ	SMCH	SMCH	BAD	SMCH	BAD	OXI
SMCH	SMCH	SMCH	SMCH	SMCH	GOOD	OXI
GOOD						NAI
BAD						NAI

Πίνακας 1.4.1 Πίνακας μεταβάσεων των αναγνωριστικών της γλώσσας Uni-C

Ακρωνύμιο Κατάστασης	Όνομα Κατάστασης	Πληροφορίες Κατάστασης
SZ	State Z	Η κατάσταση για τον 1^{ο} χαρακτήρα
SMCH	State Mid Characters	Η κατάσταση για τους ενδιάμεσους χαρακτήρες

Πίνακας 1.4.2 Πίνακας καταστάσεων των αναγνωριστικών της γλώσσας Uni-C

1.5 Κωδικοποίηση μέσω FSM

Η κωδικοποίηση μέσω FSM του αυτομάτου των αναγνωριστικών της γλώσσας Uni-C βρίσκεται στο πηγαίο αρχείο **1_identifiers.fsm**

```
// ================== 1_Αναγνωρίσματα ==================


START = SZ              // ==== Αρχική κατάσταση είναι η SZ ====


SZ:                     // --- [State Z] Κατάσταση για τον 1ο χαρακτήρα ---
    A-Z a-z -> SMCH         // Πεζό ή κεφαλαίο λατινικό γράμμα
    _       -> SMCH         // Κάτω παύλα
    *       -> BAD          // Το αναγνώρισμα είναι μη έγκυρο
```

```
SMCH:                   // --- [State Mid Characters] Κατάσταση για τους ενδιάμεσους
                                                       χαρακτήρες ---
    A-Z a-z -> SMCH         // Πεζά ή κεφαλαία λατινικά γράμματα
    _       -> SMCH         // Κάτω παύλα
    0-9     -> SMCH         // Ψηφία
    \n      -> GOOD         // Το αναγνώρισμα είναι έγκυρο
    *       -> BAD          // Τα αναγνώρισμα είναι μη έγκυρο

BAD:                    // ==== Κατάσταση εξόδου: ΟΧΙ ====
    * -> BAD

GOOD(OK):               // ==== Κατάσταση εξόδου: ΝΑΙ ====
```

1.6 Περιπτώσεις ελέγχου

#1 Το αυτόματο δέχεται τη συγκεκριμένη συμβολοσειρά, διότι ξεκινάει με λατινικό γράμμα (a-z) και στη συνέχεια περιέχει έναν αριθμό. Συμβαδίζει δηλαδή, με τους κανόνες των αναγνωριστικών στην γλώσσα Uni-C.

```
./fsm 1_identifiers.fsm
```

a1

```
YES
```

#2 Η συμβολοσειρά είναι αποδεκτή από το αυτόματο επειδή αρχίζει με αποδεκτό χαρακτήρα (κάτω παύλα) και στη συνέχεια περιέχει έναν αριθμό.

```
./fsm 1_identifiers.fsm
```

_1

```
YES
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

#3 Το αυτόματο αποδέχεται την παρακάτω συμβολοσειρά, καθώς όλοι οι χαρακτήρες (a-z, A-Z) είναι αποδεκτοί για την ονοματοδοσία ενός αναγνωριστικού στην γλώσσα Uni-C.

```
./fsm 1_identifiers.fsm

Spectacular

YES
```

#4 Ο χαρακτήρας "a" είναι αποδεκτός από την γλώσσα Uni-C ως αναγνωριστικό καθώς ξεκινάει με λατινικό γράμμα (a-z) και δεν περιέχει κάποιον μη αποδεκτό ειδικό χαρακτήρα. Δεν υπάρχει όριο χαρακτήρων που να το εμποδίζει στο να μην είναι έγκυρο.

```
./fsm 1_identifiers.fsm

a

YES
```

#5 Η παρακάτω είσοδος είναι αποδεκτή από το αυτόματο, διότι η συμβολοσειρά ξεκινάει με λατινικό γράμμα (A-Z). Η συμβολοσειρά δεν περιέχει κάποιον ειδικό χαρακτήρα που θα ήταν μη αποδεκτός στην ονοματοδοσία αναγνωριστικών από την γλώσσα Uni-C.

```
./fsm 1_identifiers.fsm

A1

YES
```

#6 Η συμβολοσειρά είναι αποδεκτή από το αυτόματο, διότι ξεκινάει με ένα λατινικό γράμμα (A-Z) και περιέχει μόνο αποδεκτούς χαρακτήρες (αριθμούς, λατινικά γράμματα, κάτω παύλα).

```
./fsm 1_identifiers.fsm

SPEctacular_15_

YES
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

#7 Η συμβολοσειρά είναι αποδεκτή από το αυτόματο, επειδή επιτρέπεται στη γλώσσα Uni-C ένα αναγνωριστικό να ξεκινάει με τον ειδικό χαρακτήρα κάτω παύλα (_). Δεν υπάρχει κάποιος μη αποδεκτός χαρακτήρας στη συγκεκριμένη συμβολοσειρά.

```
./fsm 1_identifiers.fsm

_a

YES
```

#8 Η συμβολοσειρά είναι αποδεκτή από το αυτόματο, καθώς επιτρέπεται στη γλώσσα Uni-C ένα αναγνωριστικό να ξεκινάει με τον ειδικό χαρακτήρα (_). Όπως και στο προηγούμενο παράδειγμα δεν υπάρχει κάποια διαφοροποίηση στο αποτέλεσμα επειδή δώσαμε κεφαλαίο αντί για πεζό χαρακτήρα.

```
./fsm 1_identifiers.fsm

_W

YES
```

#9 Η παρακάτω συμβολοσειρά δεν είναι αποδεκτή από το αυτόματο για τον ορισμό ενός αναγνωριστικού, διότι ο κενός χαρακτήρας δεν μπορεί να δοθεί σε αναγνωριστικό της Uni-C.

```
./fsm 1_identifiers.fsm

_ _A

NO
```

#10 Η συμβολοσειρά είναι αποδεκτή από το αυτόματο, καθώς ξεκινάει με λατινικό γράμμα (a-z) όπως ορίζει η γλώσσα Uni-C στα ονόματα και παράλληλα όλοι οι χαρακτήρες που περιέχει είναι αποδεκτοί.

```
./fsm 1_identifiers.fsm

a_b_c_1_2_3

YES
```

#11 Η σειρά χαρακτήρων είναι αποδεκτή για την ονοματοδοσία αναγνωριστικών, διότι ξεκινάει με λατινικό γράμμα (A-Z) και δεν περιέχει κάποιον χαρακτήρα μη αποδεκτό για τον ορισμό αναγνωριστικών.

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

```
./fsm 1_identifiers.fsm

AA4_F

YES
```

```
#12  Η παρακάτω συμβολοσειρά βλέπουμε ότι ξεκινάει με αριθμό, κάτι το οποίο δεν είναι συμβατό με τους κανόνες της Uni-C όσον αφορά τον ορισμό αναγνωριστικού. Για τον λόγο αυτό το αυτόματο απορρίπτει την συγκεκριμένη σειρά χαρακτήρων.

./fsm 1_identifiers.fsm

1_id

NO
```

```
#13  Η ακόλουθη σειρά χαρακτήρων δεν μπορεί να γίνει αποδεκτή από το αυτόματο για την ονομασία αναγνωριστικού, καθώς περιέχει τον ειδικό χαρακτήρα '/'. Οποιοσδήποτε ειδικός χαρακτήρας δοθεί σε κάποιο αναγνωριστικό πέραν της '_' καθιστά τη συμβολοσειρά μη αποδεκτή.

./fsm 1_identifiers.fsm

/id

NO
```

```
#14  Η παρακάτω σειρά χαρακτήρων δεν είναι αποδεκτή από το αυτόματο, διότι περιέχει τον ειδικό χαρακτήρα της τελείας '.'. Ο χαρακτήρας αυτός δεν είναι αποδεκτός στην ονοματοδοσία αναγνωριστικών στη γλώσσα Uni-C.

./fsm 1_identifiers.fsm

id.student

NO
```

```
#15  Η παρακάτω συμβολοσειρά δεν είναι αποδεκτή από το αυτόματο για τον ορισμό ενός αναγνωριστικού, διότι περιέχει κενούς χαρακτήρες ' ' και ο κενός χαρακτήρας δεν μπορεί να δοθεί σε αναγνωριστικό της Uni-C.

./fsm 1_identifiers.fsm

i d student
```

```
NO
```

#16 Η συμβολοσειρά δεν είναι αποδεκτή από το αυτόματο, καθότι ο χαρακτήρας ‘Ε’ είναι ελληνικός χαρακτήρας. Οι ελληνικοί χαρακτήρες δεν μπορούν να χρησιμοποιηθούν για τον ορισμό κάποιου ονόματος στην γλώσσα Uni-C.

```
./fsm -trace 1_identifiers.fsm

E2

sz \316 -> smch

fsm: in 1_identifiers.fsm, state 'smch' input \225 not accepted
```

#17 Η ακόλουθη συμβολοσειρά είναι αποδεκτή, διότι μπορεί ένα αναγνωριστικό της γλώσσας Uni-C να ξεκινάει με τον ειδικό χαρακτήρα ‘_’ και να ακολουθούν λατινικά γράμματα, αριθμοί ή και κάτω παύλα. Οποιοσδήποτε άλλος ειδικός χαρακτήρας δεν είναι αποδεκτός.

```
./fsm 1_identifiers.fsm

_user

YES
```

#18 Η συμβολοσειρά δεν είναι αποδεκτή από το αυτόματο διότι ξεκινάει με τον κενό χαρακτήρα ‘ ’, αυτό απαγορεύεται βάση των κανόνων ονοματοδοσίας αναγνωριστικών στη γλώσσα Uni-C.

```
./fsm 1_identifiers.fsm

 user

NO
```

#19 Η συμβολοσειρά δεν είναι αποδεκτή καθώς για την εισαγωγή της στο αυτόματο δεν χρησιμοποιήθηκε ο χαρακτήρας newline ‘\n’, αντίθετα χρησιμοποιήθηκε το EOF και γι’ αυτό το λόγο δεν έχουμε και θετικό αποτέλεσμα από το αυτόματο.

```
./fsm -trace 1_identifiers.fsm

hisz h -> smch

smch i -> smch
```

```
smch EOF -> bad

NO
```

1.7 Προβλήματα και τρόποι αντιμετώπισης

Πρόβλημα αντιμετωπίσαμε όταν για είσοδο η ομάδα δοκίμασε να χρησιμοποιήσει ελληνικούς χαρακτήρες.

```
./fsm -trace 1_identifiers.fsm

Γειά

sz \316 -> smch

Segmentation fault
```

Η κωδικοποίηση ASCII των ελληνικών χαρακτήρων δημιούργησε επιπλοκές στην αναγνώριση χαρακτήρων από το μετα-εργαλείο FSM, καθώς, τα γράμματα Γ, Κ, λ είχαν τον ίδιο ASCII κωδικό (\316). Ακόμα και σε μη αποδεκτές εισόδους με λατινικά γράμματα η κατάσταση εξόδου ήταν NO και όχι όπως φαίνεται στα δύο αυτά παραδείγματα.

```
./fsm -trace 1_identifiers.fsm

Καλημέρα

sz \316 -> smch

smch \232 -> bad

fsm: in 1_identifiers.fsm, state 'bad' input \316 not accepted
```

Τρόπος αντιμετώπισης στο πρόβλημα δεν βρέθηκε, καθώς έτσι και αλλιώς το πρότυπο της Uni-C δεν αναγνωρίζει αναγνωρίσματα με ελληνικούς χαρακτήρες αλλά ούτε το αλφάβητο της γλώσσας περιλαμβάνει ελληνικούς χαρακτήρες στις λεκτικές μονάδες της.

Άλλο πρόβλημα που αντιμετωπίσαμε ήταν στο όριο χαρακτήρων. Στο αυτόματο και συνεπώς, στην κωδικοποίηση FSM δεν βρήκαμε τρόπο να αναπαραστήσουμε/κωδικοποιήσουμε αντίστοιχα το όριο χαρακτήρων που προτείνει το πρότυπο της Uni-C (32 χαρακτήρες). Συνεπώς, το αυτόματο δέχεται αναγνωρίσματα από 1 εώς άπειρους χαρακτήρες σε αντίθεση με την κανονική έκφραση που αναγνωρίζει συμβολοσειρές εώς 32 χαρακτήρες.

1.8 Ρητή αναφορά ελλείψεων

Η αναφορά περιλαμβάνει όλα τα ζητούμενα που απαιτούνται:

- Πρότυπο αναγνώρισης

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

- Κανονική έκφραση με περιγραφή
- Αυτόματο πεπερασμένης κατάστασης με περιγραφή
- Πίνακας μεταβάσεων με πίνακα καταστάσεων
- Κωδικοποίηση μέσω FSM με σχολιασμό κώδικα
- Περιπτώσεις ελέγχου με εξαντλητικές δοκιμαστικές εκτελέσεις και σχολιασμό αποτελεσμάτων
- Προβλήματα που αντιμετωπίσαμε και τρόποι αντιμετώπισης

Ο κώδικας εκτελέστηκε σε περιβάλλον Windows με εγκατάσταση Windows Subsystem for Linux (WSL). Το τερματικό που έτρεξε η ομάδα τον κώδικα ήταν σε Ubuntu 22.04 LTS, συνεπώς οι εντολές που χρησιμοποιήσαμε για την εκτέλεση του κώδικα είναι οι εξής:

```
gcc -o fsm fsm.c

./fsm 1_identifiers.fsm

./fsm -list 1_identifiers.fsm

./fsm -trace 1_identifiers.fsm
```

2. Λεκτικά κυριολεκτικά

Την προσομοίωση την ανέλαβαν οι φοιτητές

Δομινάρης Βασίλειος (ΠΑΔΑ-21390055) – Ηλιού Ιωάννης (ΠΑΔΑ-19390066) – Τάτσης Παντελής (ΠΑΔΑ-20390226)

2.1 Πρότυπο αναγνώρισης

Η λειτουργία του αυτομάτου είναι να αναγνωρίζει λεκτικά κυριολεκτικά που υποστηρίζει η γλώσσα Uni-C με βάση ένα πρότυπο αναγνώρισης.

Το πρότυπο αναγνώρισης αναγνωρίζει τις συμβολοσειρές που περικλείονται μέσα σε διπλά εισαγωγικά (“) και περιλαμβάνουν οποιονδήποτε χαρακτήρα πέρα του backslash (\), της νέας γραμμής (\n) ή του διπλού εισαγωγικού που για τη χρήση τους απαιτούν χρήση σειράς διαφυγής.

Επιπρόσθετα, επιστρέφονται ως tokens με το αναγνωριστικό όνομα ***string***.

2.2 Κανονική Έκφραση

Η κανονική έκφραση αυτή περιγράφει ένα πρότυπο που αντιστοιχεί σε έναν συνδυασμό χαρακτήρων που μπορεί να περιέχει συμβολοσειρές μέσα σε διπλά εισαγωγικά, με δυνατότητα διαφυγής με χρήση του χαρακτήρα backslash (\). Με άλλα λόγια, η έκφραση αναγνωρίζει έγκυρα λεκτικά που περικλείονται μέσα σε διπλά εισαγωγικά.

```
"([^"\\]*(\\[\\n"][^"\\]*)*)"
```

Εδώ είναι η ανάλυση των τμημάτων της κανονικής έκφρασης:

```
"
```

Αντιστοιχεί σε ένα διπλό εισαγωγικό που σηματοδοτεί την έναρξη του λεκτικού.

```
([^"\\]*(\\[\\n"][^"\\]*)*)
```

Αυτό το μέρος αντιστοιχεί σε μια συμβολοσειρά που περιέχει δυνατότητα διαφυγής.

```
[^"\\]*
```

Αντιστοιχεί σε μηδέν ή περισσότερους χαρακτήρες που δεν είναι διπλά εισαγωγικά (" ή \\).

```
(\\[\\n"][^"\\]*)*
```

Αυτό το τμήμα αντιστοιχεί σε μια ακολουθία με σειρά διαφυγής.

```
\\
```

Αντιστοιχεί σε έναν χαρακτήρα backslash (\).

```
[\\n"]
```

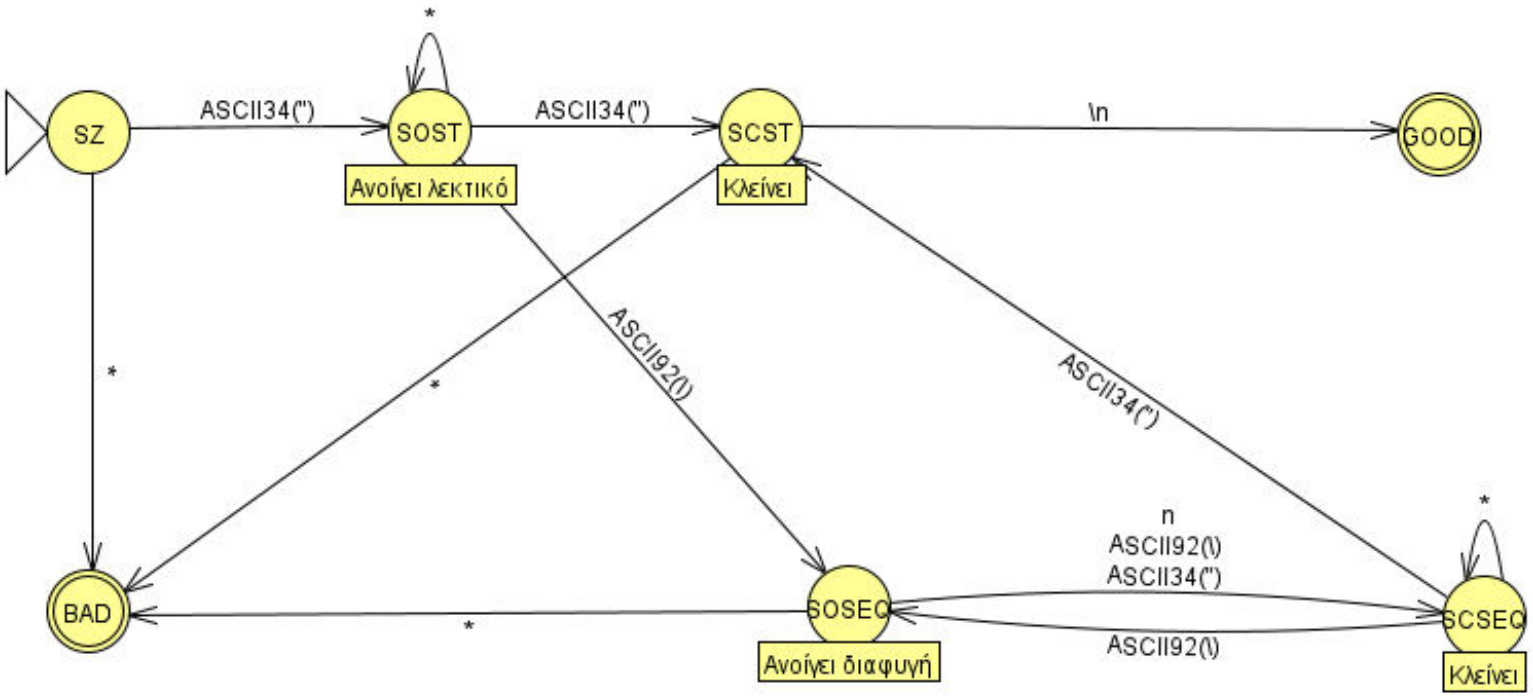
Αντιστοιχεί σε έναν χαρακτήρα που μπορεί να ακολουθείται με σειρά διαφυγής (\, n ή ").

```
[^"\\]*
```

Αντιστοιχεί σε μηδέν ή περισσότερους χαρακτήρες που δεν είναι διπλά εισαγωγικά (" ή \\).

Συνοψίζοντας, η κανονική έκφραση αυτή αντιστοιχεί σε μια συμβολοσειρά που περιέχει δυνατότητα διαφυγής για διπλά εισαγωγικά και τους χαρακτήρες \, n, " μέσα σε αυτά.

2.3 Αυτόματο πεπερασμένης κατάστασης ή ΔΜ



Εικόνα 2.3.1 Πεπερασμένο αυτόματο λεκτικών της γλώσσας Uni-C

Το αυτόματο ξεκινάει από την αρχική κατάσταση SZ, όπου το λεκτικό πρέπει να ξεκινάει με διπλά εισαγωγικά ".

Αν διαβαστεί διπλό εισαγωγικό, το αυτόματο μεταβαίνει στην κατάσταση SOST. Σε αυτή την κατάσταση, αν διαβαστεί διπλό εισαγωγικό, περνάει στην κατάσταση SCST, αν διαβαστεί \, περνάει στην κατάσταση SOSEQ, αλλιώς παραμένει στην ίδια κατάσταση SOST.

Στην SCST, αν διαβαστεί \n, το λεκτικό θεωρείται έγκυρο και το αυτόματο μεταβαίνει στην κατάσταση εξόδου GOOD, ενώ σε οποιοδήποτε άλλο χαρακτήρα μεταβαίνει στην κατάσταση εξόδου BAD.

Στις καταστάσεις SOSEQ, SCSEQ, το αυτόματο διαβάζει τους χαρακτήρες που μπορούν να εμφανιστούν με σειρά διαφυγής. Αν διαβαστεί ", \, ή n, το αυτόματο πηγαίνει στην κατάσταση SCSEQ, ενώ σε οποιοδήποτε άλλο χαρακτήρα μεταβαίνει στην κατάσταση εξόδου BAD.

Στην SCSEQ, αν διαβαστεί ", το αυτόματο πηγαίνει στην κατάσταση SCST, αν διαβαστεί \ πηγαίνει πίσω στην SOSEQ, ενώ σε οποιοδήποτε άλλο χαρακτήρα παραμένει στην ίδια κατάσταση SCSEQ.

Η κατάσταση εξόδου είναι GOOD όταν το λεκτικό κλείσει σωστά με διπλά εισαγωγικά και δεν περιέχει μη έγκυρους χαρακτήρες.

2.4 Πίνακας μεταβάσεων

	“	n	\	\n	ΚΑΤΑΣΤΑΣΗ ΕΞΟΔΟΥ
SZ	SOST	BAD	BAD	BAD	OXI
SOST	SCST	SOST	SOSEQ	SOST	OXI
SOSEQ	SCSEQ	SCSEQ	SCSEQ	BAD	OXI
SCSEQ	SCST	SCSEQ	SOSEQ	SCST	OXI
SCST	BAD	BAD	BAD	GOOD	OXI
GOOD					NAI
BAD					NAI

Πίνακας 2.4.1 Πίνακας μεταβάσεων των λεκτικών κυριολεκτικών της γλώσσας Uni-C

Ακρωνύμιο Κατάστασης	Όνομα Κατάστασης	Πληροφορίες Κατάστασης
SZ	State Z	Η κατάσταση για το άνοιγμα του λεκτικού
SOST	State Open String	Η κατάσταση για το περιεχόμενο του λεκτικού

SOSEQ	State Open Sequence	Η κατάσταση για το άνοιγμα σειράς διαφυγής μέσα στο λεκτικό
SCEQ	State Close Sequence	Η κατάσταση για τον χαρακτήρα που ακολουθείται με σειρά διαφυγής μέσα στο λεκτικό
SCST	State Close String	Η κατάσταση για το κλείσιμο του λεκτικού

Πίνακας 2.4.2 Πίνακας καταστάσεων των λεκτικών της γλώσσας Uni-C

2.5 Κωδικοποίηση μέσω FSM

Η κωδικοποίηση μέσω FSM του αυτομάτου των λεκτικών κυριολεκτικών της γλώσσας Uni-C βρίσκεται στο πηγαίο αρχείο **2_strings.fsm**

```
// ================== 2_Λεκτικά ==================


START = SZ              // ==== Αρχική κατάσταση είναι η SZ ====


SZ:                     // --- [State Z] Κατάσταση για το άνοιγμα του λεκτικού ---
    "   -> SOST             // Το λεκτικό ανοίγει με διπλό εισαγωγικό
    *    -> BAD             // Το λεκτικό είναι μη έγκυρο


SOST:                   // --- [State Open String] Κατάσταση για το περιεχόμενο του
                                    λεκτικού ---
    "   -> SCST             // Το λεκτικό κλείνει με διπλό εισαγωγικό
    \\   -> SOSEQ           // Ακολουθεί σειρά διαφυγής
    *    -> SOST            // Ακολουθεί οποιοσδήποτε άλλος χαρακτήρας


SOSEQ:                  // --- [State Open Sequence] Κατάσταση για χαρακτήρες με σειρά
                                    διαφυγής ---
    \\   -> SCSEQ           // Ακολουθεί ο χαρακτήρας backslash (\)
    "   -> SCSEQ            // Ακολουθεί διπλό εισαγωγικό
```

```
    n   -> SCSEQ            // Ακολουθεί το n για την αλλαγή γραμμής

    *   -> BAD              // Το λεκτικό είναι μη έγκυρο


SCSEQ:                  // --- [State Close Sequence] Κατάσταση για κλείσιμο λεκτικού ή
                                                        σειρά διαφυγής ---

    "  -> SCST              // Το λεκτικό κλείνει με διπλό εισαγωγικό

    \\  -> SOSEQ            // Επιστροφή στην κατάσταση για χαρακτήρες με σειρά διαφυγής

    *   -> SCSEQ            // Ακολουθεί οποιοσδήποτε άλλος χαρακτήρας


SCST:                   // --- [State Close String] Κατάσταση για το το κλείσιμο του
                                                    λεκτικού ---

    \n -> GOOD              // Το λεκτικό είναι έγκυρο

    *  -> BAD               // Το λεκτικό είναι άκυρο


BAD:                    // ==== Κατάσταση εξόδου: OXI ====

    * -> BAD


GOOD(OK):               // ==== Κατάσταση εξόδου: NAI ====
```

2.6 Περιπτώσεις ελέγχου

#1 Είναι αποδεκτό διότι ανοίγει και κλείνει με διπλά εισαγωγικά (“) και ακολουθείται από έναν χαρακτήρα αλλαγής γραμμής (\n).

```
./fsm 2_strings.fsm

"KALHMERA"

YES
```

#2 Είναι αποδεκτό, διότι στην κατάσταση “SCS”, όταν ολοκληρώνεται ακολουθεί χαρακτήρας αλλαγής γραμμής (\n), τότε η κατάσταση προχωρά σε κατάσταση εξόδου που είναι ορισμένη ως “YES”.

```
./fsm 2_strings.fsm
```

```
""

YES
```

```
#3   Είναι αποδεκτό, διότι ανοίγει με διπλό εισαγωγικό (“), έπειτα προχωράμε στην κατάσταση SOS όπου συναντάμε ένα διπλό εισαγωγικό (“) και στην συνέχεια συναντάει (\n). Ο χαρακτήρας διπλό εισαγωγικό χρησιμοποιείται ορθά μέσα στο λεκτικό με σειρά διαφυγής (\”), όπως και η αλλαγή γραμμής (\n).

./fsm 2_strings.fsm

"hello, \"world!\"\n"

YES
```

```
#4 Είναι αποδεκτό διότι ανοίγει και κλείνει με ένα διπλό εισαγωγικό (“). Οι χαρακτήρες που χρησιμοποιούνται με σειρά διαφυγής (\, “, newline) στην γλώσσα Uni-C, χρησιμοποιούνται ορθά μέσα στο λεκτικό.

./fsm -trace 2_strings.fsm

"\"\\n\"\"\\n\""

sz " -> sost

sost \ -> soseq

soseq " -> scseq

scseq \ -> soseq

soseq \ -> scseq

scseq n -> scseq

scseq \ -> soseq

soseq " -> scseq

scseq \ -> soseq

soseq " -> scseq

scseq \ -> soseq

soseq \ -> scseq

scseq n -> scseq

scseq \ -> soseq
```

```
soseq " -> scseq

scseq " -> scst

scst \n -> good

YES
```

```
#5 Είναι λάθος γιατί το λεκτικό δεν ξεκινάει με διπλό εισαγωγικό (“).

./fsm 2_strings.fsm

f"number of loops is {counter}"

NO
```

```
#6    Είναι αποδεκτό γιατί ανοίγει με διπλό εισαγωγικό (“), και προχωράει στην κατάσταση όπου ακολουθούν οι χαρακτήρες "6", "/", "3", "=", "2". Το λεκτικό ορθά κλείνει με διπλό εισαγωγικό (“).

./fsm 2_strings.fsm

"6/3=2"

YES
```

```
#7    Δεν είναι αποδεκτό γιατί το λεκτικό ανοίγει και κλείνει με διπλό εισαγωγικό (“) και η γλώσσα Uni-C αποδέχεται την χρήση του διπλού εισαγωγικού (“) ως  χαρακτήρα μέσα στο λεκτικό, μόνο με χρήση σειρά διαφυγής (\”).

./fsm 2_strings.fsm

"He said "Why Brutus?""

NO
```

```
#8    Δεν είναι αποδεκτό γιατί η γλώσσα Uni-C αποδέχεται την χρήση του χαρακτήρα backslash (\) ως χαρακτήρα μέσα στο λεκτικό, μόνο με χρήση σειράς διαφυγής (\\).

./fsm -trace 2_strings.fsm

"6\3=2"

sz " -> sost

sost 6 -> sost
```

```
sost \ -> soseq

soseq 3 -> bad

bad = -> bad

bad 2 -> bad

bad " -> bad

bad \n -> bad

bad EOF -> bad

NO
```

#9 Είναι έγκυρο, διότι το λεκτικό ανοίγει και κλείνει με διπλό εισαγωγικό και ο χαρακτήρας backslash (\) χρησιμοποιείται μέσα στο λεκτικό με σειρά διαφυγής.

```
./fsm -trace 2_strings.fsm

"6\\3=2"

sz " -> sost

sost 6 -> sost

sost \ -> soseq

soseq \ -> scseq

scseq 3 -> scseq

scseq = -> scseq

scseq 2 -> scseq

scseq " -> scst

scst \n -> good

YES
```

#10 Δεν είναι έγκυρο διότι η Uni-C δεν αναγνωρίζει τον χαρακτήρα d με σειρά διαφυγής μέσα σε λεκτικό.

```
./fsm -trace 2_strings.fsm

"\n\"\\\d"

sz " -> sost
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

```
sost \ -> soseq

soseq n -> scseq

scseq \ -> soseq

soseq " -> scseq

scseq \ -> soseq

soseq \ -> scseq

scseq \ -> soseq

soseq d -> bad

bad " -> bad

bad \n -> bad

bad EOF -> bad

NO
```

#11 Δεν είναι έγκυρο γιατί δεν έχουμε ορίσει κανόνες μετάβασης για ελληνικά γράμματα.

```
./fsm 2_strings.fsm

"Καλημέρα κόσμε"

Segmentation fault (core dumped)
```

#12 Δεν είναι έγκυρο γιατί το λεκτικό ανοίγει και κλείνει με μονό εισαγωγικό (‘) αντί για διπλό (“).

```
./fsm 2_strings.fsm

'La vida loca'

NO
```

#13 Είναι έγκυρο γιατί το λεκτικό ξεκινάει και κλείνει με διπλά εισαγωγικά(“)

```
./fsm 2_strings.fsm

"Hi Mark"

YES
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

#14 Είναι έγκυρο διότι ξεκινά με διπλό εισαγωγικό(“) και το λεκτικό κλείνει με διπλό εισαγωγικό(“)

```
./fsm 2_strings.fsm

"Tests"

YES
```

#15 Είναι έγκυρο διότι ξεκινάει με διπλά εισαγωγικά (“), περιέχει αλλαγή γραμμής (\n) και το λεκτικό κλείνει με διπλά εισαγωγικά(“)

```
./fsm 2_strings.fsm

"Mark said, \"Boo!\"\n"

YES
```

#16 Είναι έγκυρο διότι ανοίγει και κλείνει με διπλά εισαγωγικά(“), και η αλλαγή γραμμής χρησιμοποιείται με σειρά διαφυγής μέσα στο λεκτικό (\n)

```
./fsm 2_strings.fsm

"Hi/n\n"

YES
```

#17 Δεν είναι έγκυρο γιατί διακόπτεται η ανάγνωση χαρακτήρων με EOF αντί για αλλαγή γραμμής

```
./fsm -trace 2_strings.fsm

"hi"sz " -> sost

sost h -> sost

sost i -> sost

sost " -> scst

scst EOF -> bad

NO
```

2.7 Προβλήματα και τρόποι αντιμετώπισης

Πρόβλημα αντιμετωπίσαμε πάλι στην αναγνώριση των ελληνικών χαρακτήρων, καθώς, υφίστανται να υπάρχουν λεκτικά κυριολεκτικά με ελληνικά γράμματα. Τρόπος αντιμετώπισης δεν βρέθηκε και το πόρισμα είναι παρόμοιο που αναγράφεται αναλυτικά εδώ. Συνεπώς, η Uni-C υποστηρίζει λεκτικά κυριολεκτικά μόνο με λατινικά γράμματα, ψηφία, ειδικούς χαρακτήρες και whitespace χαρακτήρες.

Πρόβλημα επίσης, αντιμετωπίσαμε κατά την κωδικοποίηση του FSM, καθώς όταν πήγαμε να τρέξουμε τον κώδικα μας έβγαζε μήνυμα λάθους ότι το αυτόματο δεν είναι ντετερμινιστικό.

```
./fsm 2_strings.fsm

fsm: in 2_strings.fsm, Non-Deterministic, state='sost', no input and input='"'
```

Το μετα-εργαλείο FSM αναγνωρίζει μόνο ντετερμινιστικά αυτόματα, ωστόσο, η αιτία του προβλήματος δεν ήταν ότι είχαμε παραπάνω από μία μετάβαση από μία κατάσταση σε μία άλλη με τον ίδιο χαρακτήρα. Η αιτία ήταν στην κατάσταση SOST (State Open String), η οποία είχε αυτές τις μεταβάσεις όταν εμφάνιζε το πρόβλημα.

```
SOST:                   // --- Κατάσταση για το περιεχόμενο του λεκτικού ---

    "   -> SCST             // Το λεκτικό κλείνει με διπλό εισαγωγικό

    \   -> SOSEQ            // Ακολουθεί σειρά διαφυγής

    *    -> SOST            // Ακολουθεί οποιοσδήποτε άλλος χαρακτήρας
```

Ο χαρακτήρας backslash (\) δηλώνει σειρά διαφυγής και προφανώς εμείς χρειαζόμασταν τον ίδιο τον χαρακτήρα για την μετάβαση στην SOSEQ (State Open Sequence). Με το μονό backslash δηλώσαμε απλά μία κενή διαφυγή (no input), πράγμα που δημιουργούσε την αιτία το αυτόματο να είναι μη ντετερμινιστικό.

Το πρόβλημα αντιμετωπίστηκε με το διπλό backslash, γιατί η χρήση του όπως και στους χαρακτήρες διαφυγής (newline, tab, space), γίνεται με σειρά διαφυγής.

```
SOST:                   // --- Κατάσταση για το περιεχόμενο του λεκτικού ---

    "   -> SCST             // Το λεκτικό κλείνει με διπλό εισαγωγικό

    \\  -> SOSEQ            // Ακολουθεί σειρά διαφυγής

    *    -> SOST            // Ακολουθεί οποιοσδήποτε άλλος χαρακτήρας
```

2.8 Ρητή αναφορά ελλείψεων

Η αναφορά περιλαμβάνει όλα τα ζητούμενα που απαιτούνται:

- Πρότυπο αναγνώρισης
- Κανονική έκφραση με περιγραφή
- Αυτόματο πεπερασμένης κατάστασης με περιγραφή
- Πίνακας μεταβάσεων με πίνακα καταστάσεων
- Κωδικοποίηση μέσω FSM με σχολιασμό κώδικα
- Περιπτώσεις ελέγχου με εξαντλητικές δοκιμαστικές εκτελέσεις και σχολιασμό αποτελεσμάτων
- Προβλήματα που αντιμετωπίσαμε και τρόποι αντιμετώπισης

Ο κώδικας εκτελέστηκε σε περιβάλλον Windows με εγκατάσταση Windows Subsystem Linux (WSL). Το τερματικό που έτρεξε η ομάδα τον κώδικα ήταν σε Ubuntu 22.04 LTS, συνεπώς οι εντολές που χρησιμοποιήσαμε για την εκτέλεση του κώδικα είναι οι εξής:

```
gcc -o fsm fsm.c

./fsm 2_strings.fsm

./fsm -list 2_strings.fsm

./fsm -trace 2_strings.fsm
```

3. Ακέραιοι

Την προσομοίωση την ανέλαβαν οι φοιτητές

Αθανασίου Βασίλειος Ευάγγελος (ΠΑΔΑ-19390005) – Θεοχάρης Γεώργιος (ΠΑΔΑ-19390283) – Τάτσης Παντελής (ΠΑΔΑ-20390226)

3.1 Πρότυπο αναγνώρισης

Η λειτουργία του αυτομάτου είναι να αναγνωρίζει μη προσημασμένους ακεραίους.

Συγκεκριμένα, αναγνωρίζει ακέραιους που ανήκουν στο δεκαδικό, δεκαεξαδικό ή οκταδικό σύστημα αρίθμησης. Ο δεκαδικός ακέραιος ξεκινάει με μη μηδενικό ψηφίο (1-9) και ακολουθεί προαιρετικά οποιοσδήποτε αριθμός ψηφίων (0-9). Το 0 αναγνωρίζεται ως η μόνη εξαίρεση δεκαδικού ακεραίου που ξεκινάει με 0. Ο δεκαεξαδικός ακέραιος αναγνωρίζεται όταν ξεκινάει με 0x ή 0X και ακολουθούν ένα ή περισσότερα δεκαεξαδικά ψηφία (0-9 ή A-F). Τέλος, ο οκταδικός ακέραιος αναγνωρίζεται όταν ξεκινάει με 0 και ακολουθεί ένα ή περισσότερα οκταδικά (0-7) ψηφία.

Το πρότυπο τους αναγνωρίζει με το αναγνωριστικό όνομα ***integer*** και επιστρέφονται ως token.

3.2 Κανονική Έκφραση

Η κανονική έκφραση για τις λεκτικές μονάδες των ακεραίων της γλώσσας Uni-C είναι η εξής:

```
([1-9][0-9]*|0[x|X][0-9A-F]+|0[0-7]+|0)
```

Η έκφραση αναγνωρίζει τις συμβολοσειρές εισόδου που αντιστοιχούν σε δεκαδικούς, δεκαεξαδικούς και οκταδικούς ακέραιους αριθμούς.

- **Δεκαδικοί: `[1-9][0-9]*|0`**

  Η έκφραση αναγνωρίζει τις συμβολοσειρές που ξεκινάνε με χαρακτήρα από το εύρος “1-9” και ενδέχεται να ακολουθεί χαρακτήρας που ανήκει στο εύρος “0-9” μηδέν ή περισσότερες φορές. Επιπρόσθετα, αναγνωρίζει και το 0 ως τη μόνη εξαίρεση δεκαδικού ακεραίου που ξεκινάει με 0.

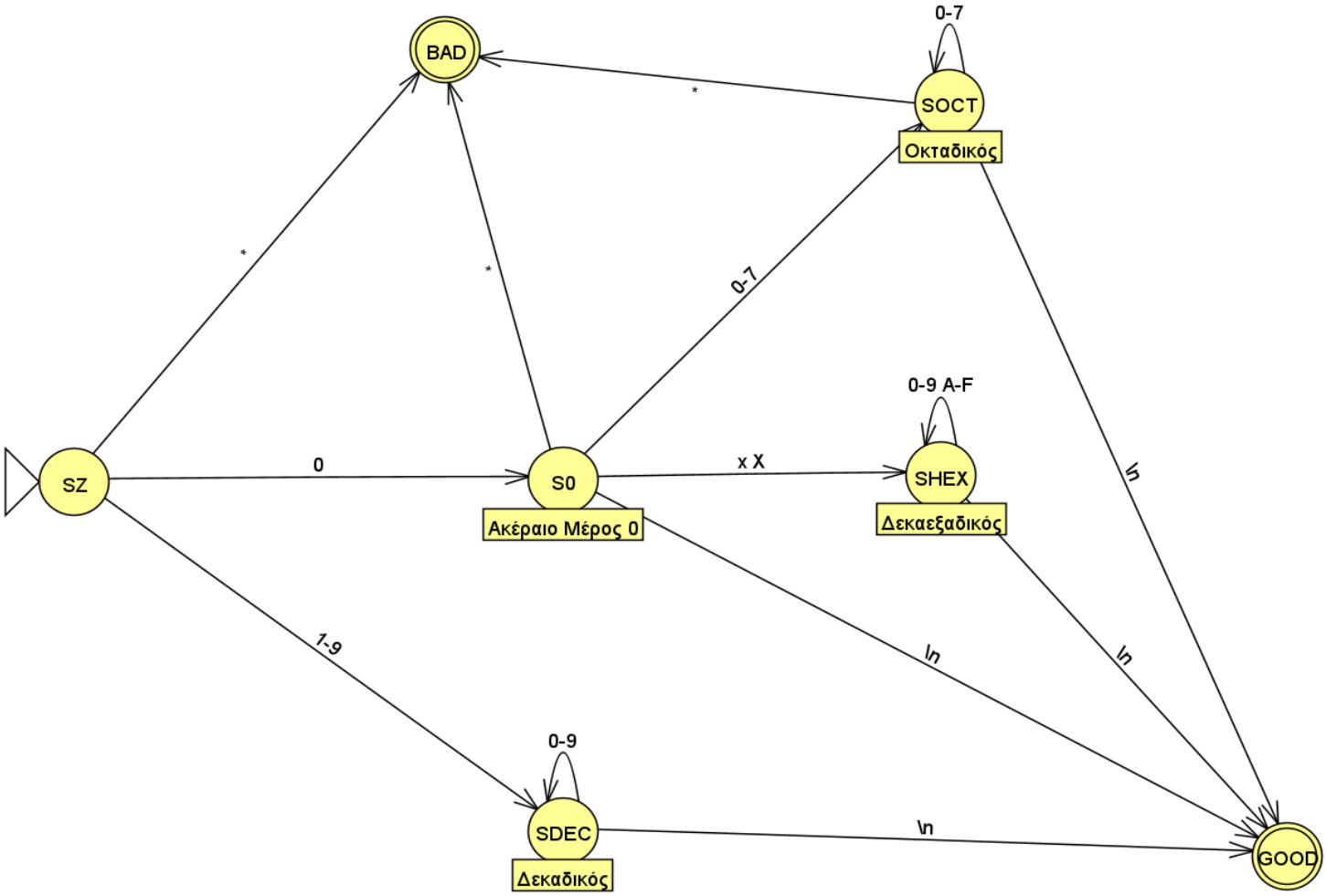
- **Δεκαεξαδικοί: `0[x|X][0-9A-F]+`**

Η έκφραση αναγνωρίζει τις συμβολοσειρές που ξεκινάνε με τον χαρακτήρα “0”, ακολουθούμενοι είτε από τον χαρακτήρα “x” είτε από τον “X” και τέλος από χαρακτήρα που ανήκει στο εύρος “0-9” ή στο εύρος “A-F” μία ή περισσότερες φορές.

- **Οκταδικοί: `0[0-7]+`**

Η έκφραση αναγνωρίζει τις συμβολοσειρές που ξεκινάνε με τον χαρακτήρα “0”, ακολουθούμενοι από τον χαρακτήρα που ανήκει στο εύρος “0-7” μία ή περισσότερες φορές.

3.3 Αυτόματο πεπερασμένης κατάστασης ή ΔΜ



Εικόνα 3.3.1 Πεπερασμένο αυτόματο ακεραίων της γλώσσας Uni-C

Το αυτόματο αυτό αναγνωρίζει έγκυρους ακεραίους που μπορούν να είναι δεκαδικοί, οκταδικοί ή δεκαεξαδικοί.

Αρχικά βρίσκεται στην κατάσταση SZ, περιμένοντας το πρώτο σύμβολο του ακεραίου.

Αν το πρώτο σύμβολο είναι 0, το αυτόματο μεταβαίνει στην κατάσταση S0, ελέγχοντας τα επόμενα σύμβολα για να καθορίσει αν ο ακέραιος είναι οκταδικός ή δεκαεξαδικός.

Αν το πρώτο σύμβολο είναι από 1 έως 9, το αυτόματο μεταβαίνει στην κατάσταση SDEC, καθώς πρόκειται για δεκαδικός ακέραιος. Στη συνέχεια, περιμένει περισσότερα δεκαδικά ψηφία ή το τέλος γραμμής για να αποδεχτεί τον ακέραιο.

Στις καταστάσεις S0, SDEC ελέγχει τα σύμβολα που ακολουθούν:

Στην κατάσταση S0, αν το επόμενο σύμβολο είναι από 0 έως 7, ο αριθμός είναι οκταδικός. Αν το επόμενο σύμβολο είναι το x ή το X, τότε ο αριθμός είναι δεκαεξαδικός. Αν είναι το τέλος γραμμής, ο αριθμός 0 είναι έγκυρος ως η μόνη εξαίρεση δεκαδικού ακεραίου που ξεκινάει με το 0 και το αυτόματο μεταβαίνει στην κατάσταση GOOD. Σε οποιαδήποτε άλλη περίπτωση, ο αριθμός δεν είναι έγκυρος (BAD).

Στην κατάσταση SDEC, αν το επόμενο σύμβολο είναι από 0 έως 9, ο αριθμός συνεχίζεται να είναι δεκαδικός. Αν είναι το τέλος γραμμής, ο αριθμός είναι έγκυρος και το αυτόματο μεταβαίνει στην κατάσταση GOOD.

Στις καταστάσεις SHEX και SOCT, το αυτόματο ελέγχει τα σύμβολα που ακολουθούν:

Στην κατάσταση SHEX των δεκαεξαδικών ακεραίων, αν το επόμενο σύμβολο είναι από 0 έως 9 ή τα γράμματα A έως F, το αυτόματο παραμένει στην ίδια κατάσταση. Αν είναι το τέλος γραμμής, ο αριθμός είναι έγκυρος και το αυτόματο μεταβαίνει στην κατάσταση GOOD.

Στην κατάσταση SOCT των οκταδικών ακεραίων, αν το επόμενο σύμβολο είναι από 0 έως 7, το αυτόματο παραμένει στην ίδια κατάσταση. Αν είναι το τέλος γραμμής, ο αριθμός είναι έγκυρος και το αυτόματο μεταβαίνει στην κατάσταση GOOD.

3.4 Πίνακας μεταβάσεων

	0	1-7	8-9	x X	A-F	\n	ΚΑΤΑΣΤΑΣΗ ΕΞΟΔΟΥ
SZ	S0	SDEC	SDEC	BAD	BAD	BAD	OXI
S0	SOCT	SOCT	BAD	SHEX	BAD	GOOD	OXI
SDEC	SDEC	SDEC	SDEC	BAD	BAD	GOOD	OXI
SHEX	SHEX	SHEX	SHEX	BAD	SHEX	GOOD	OXI
SOCT	SOCT	SOCT	BAD	BAD	BAD	GOOD	OXI
GOOD							NAI
BAD							NAI

Πίνακας 3.4.1 Πίνακας μεταβάσεων των ακεραίων της γλώσσας Uni-C

Ακρωνύμιο Κατάστασης	Όνομα Κατάστασης	Πληροφορίες Κατάστασης
SZ	State Z	Η κατάσταση για το 1ο ψηφίο
S0	State 0	Η κατάσταση που ο ακέραιος ξεκινάει με 0
SDEC	State Decimal	Η κατάσταση που αναγνωρίζει ακεραίους στο δεκαδικό σύστημα
SHEX	State Hexadecimal	Η κατάσταση που αναγνωρίζει ακεραίους στο δεκαεξαδικό σύστημα
SOCT	State Octal	Η κατάσταση που αναγνωρίζει ακεραίους στο οκταδικό σύστημα

Πίνακας 3.4.2 Πίνακας καταστάσεων των ακεραίων της γλώσσας Uni-C

3.5 Κωδικοποίηση μέσω FSM

Η κωδικοποίηση μέσω FSM του αυτομάτου των ακεραίων της γλώσσας Uni-C βρίσκεται στο πηγαίο αρχείο **3_integers.fsm**

```
// ================== 3_Ακέραιοι ==================


START = SZ              // ==== Αρχική κατάσταση είναι η SZ ====


SZ:                     // --- [State Z] Κατάσταση για τους ακεραίους ---
    0   -> S0               // Ακέραιος που ξεκινάει με 0
    1-9 -> SDEC             // Θετικός δεκαδικός ακέραιος
    *   -> BAD              // Ο ακέραιος είναι μη έγκυρος


S0:                     // --- [State 0] Κατάσταση για τους ακεραίους που ξεκινάνε με 0
---
    0-7 -> SOCT             // Οκταδικός ακέραιος
    x X -> SHEX             // Δεκαεξαδικός ακέραιος
```

```
    \n  -> GOOD             // Έγκυρος ακέραιος 0
    *   -> BAD              // Ο ακέραιος είναι μη έγκυρος

SDEC:                   // --- [State Decimal] Κατάσταση θετικού δεκαδικού ακεραίου ---
    0-9 -> SDEC             // Ακολουθούν κι άλλα ψηφία από 0-9
    \n  -> GOOD             // Ο ακέραιος είναι έγκυρος
    *   -> BAD              // Ο ακέραιος είναι μη έγκυρος

SHEX:                   // --- [State Hexadecimal] Κατάσταση δεκαεξαδικού ακεραίου ---
    0-9 A-F -> SHEX         // Ακολουθούν κι άλλα ψηφία από 0-9 και γράμματα από A-F
    \n      -> GOOD         // Ο ακέραιος είναι έγκυρος
    *       -> BAD          // Ο ακέραιος είναι μη έγκυρος

SOCT:                   // --- [State Octal] Κατάσταση οκταδικού ακεραίου ---
    0-7 -> SOCT             // Ακολουθούν κι άλλα ψηφία από 0-7
    \n  -> GOOD             // Ο ακέραιος είναι έγκυρος
    *   -> BAD              // Ο ακέραιος είναι μη έγκυρος

BAD:                    // ==== Κατάσταση εξόδου: OXI ====
    * -> BAD

GOOD(OK):               // ==== Κατάσταση εξόδου: NAI ====
```

3.6 Περιπτώσεις ελέγχου

```
#1  Το 0 είναι αποδεκτή κατάσταση ακεραίου στη γλώσσα Uni-C, η καταχώρηση του γίνεται με το \n συνεπώς σωστά το αυτόματο δέχεται την ακόλουθη είσοδο.

./fsm 3_integers.fsm

0

YES
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

#2 Το 3 είναι αποδεκτή κατάσταση ακεραίου στη γλώσσα Uni-C, η καταχώρηση του γίνεται με το \n συνεπώς σωστά το αυτόματο δέχεται την ακόλουθη είσοδο.

```
./fsm 3_integers.fsm

3

YES
```

#3 Η ακόλουθη είσοδος βλέπουμε ότι έχει μόνο ακέραιο μέρος και όχι δεκαδικό, συνεπώς σωστά γίνεται αποδεκτή από το αυτόματο μιάς και πρόκειται για την αναπαράσταση ενός ακεραίου αριθμού στη γλώσσα Uni-C.

```
./fsm 3_integers.fsm

214748

YES
```

#4 Στη γλώσσα Uni-C τα αριθμητικά δεν έχουν πρόσημο, επομένως σωστά δεν δέχεται το αυτόματο την είσοδο -50.

```
./fsm 3_integers.fsm

-50

NO
```

#5 Επειδή το αυτόματο αναγνωρίζει συμβολοσειρές που ξεκινάνε με 0 ακολουθούνται από τον χαρακτήρα x και βρίσκονται στο εύρος [0-F], ορθώς δέχεται την συγκεκριμένη είσοδο που αναπαριστά έναν 16-δικο αριθμό.

```
./fsm 3_integers.fsm

0x4F

YES
```

#6 Επειδή το αυτόματο αναγνωρίζει συμβολοσειρές που ξεκινάνε με 0 ακολουθούνται από τον χαρακτήρα X και βρίσκονται στο εύρος [0-F], σωστά δέχεται την συγκεκριμένη είσοδο που αναπαριστά έναν 16-δικο αριθμό.

```
./fsm 3_integers.fsm
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

0X88AA

YES

#7 Το αυτόματο δέχεται την ακόλουθη συμβολοσειρά καθώς πρόκειται για ακέραιο του 8-δικου συστήματος αρίθμησης. Παρατηρούμε ότι όλα τα ψηφία που εισάγονται είναι εντός του [0-7], ακολουθούν δηλαδή τους κανόνες του 8-δικου συστήματος.

./fsm 3_integers.fsm

063

YES

#8 Το αυτόματο όπως είδαμε και στο προηγούμενο παράδειγμα δέχεται αριθμούς του 8-δικου συστήματος αρίθμησης. Για τον λόγο αυτό η είσοδος '00' είναι αποδεκτή.

./fsm -trace 3_integers.fsm

00

sz 0 -> s0

s0 0 -> soct

soct \n -> good

YES

#9 Η είσοδος '01' πάλι πρόκειται για ακέραιο αριθμό του 8-δικου συστήματος αρίθμησης, εφόσον δεν υπάρχει κάποιο πρόσημο ή κάποιο ψηφίο εκτός του πεδίου [0-7], η είσοδος είναι έγκυρη.

./fsm -trace 3_integers.fsm

01

YES

#10 Η είσοδος '09' δεν είναι αποδεκτή από το αυτόματο, καθώς ο αριθμός 9 δεν περιέχεται στους αριθμούς [0-7].

./fsm -trace 3_integers.fsm

09

```
sz 0 -> s0

s0 9 -> bad

bad \n -> bad

bad EOF -> bad

NO
```

#11 Το αυτόματο απορρίπτει την ακόλουθη είσοδο, διότι το 'G' που εισάγεται είναι εκτός των ορίων του 16-δικου συστήματος αρίθμησης, το οποίο περιέχει αριθμούς από [0-9] ή γράμματα από [A-F]. Δεν αναπαριστά 16-δικό αριθμό στην γλώσσα Uni-C.

```
./fsm -trace 3_integers.fsm

0XFGA9

sz 0 -> s0

s0 X -> shex

shex F -> shex

shex G -> bad

bad A -> bad

bad 9 -> bad

bad \n -> bad

bad EOF -> bad

NO
```

#12 Η παρακάτω συμβολοσειρά απορρίπτεται από το αυτόματο εξαιτίας της εισόδου του αριθμού '8', διότι δεν είναι αριθμός εντός του [0-7] για να μπορεί να είναι έγκυρος οκταδικός ακέραιος.

```
./fsm 3_integers.fsm

01578

NO
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

#13 Η συμβολοσειρά ξεκινάει με 2 μηδενικά, επομένως δεν αναπαριστά σωστά έναν 16-δικο αριθμό στην γλώσσα Uni-C. Για τον λόγο αυτό η συμβολοσειρά απορρίπτεται από το αυτόματο.

```
./fsm 3_integers.fsm

00xAFB1

NO
```

#14 Η γλώσσα Uni-C στην αναπαράσταση ακεραίων δεν δέχεται προσημασμένους αριθμούς, λόγω του προσήμου λοιπόν η παρακάτω συμβολοσειρά δεν θα γίνει αποδεκτή από το αυτόματο.

```
./fsm 3_integers.fsm

-001

NO
```

#15 Η γλώσσα Uni-C στην αναπαράσταση ακεραίων δεν δέχεται προσημασμένους αριθμούς, λόγω του προσήμου λοιπόν η παρακάτω συμβολοσειρα δεν θα γίνει αποδεκτή από το αυτόματο.

```
./fsm 3_integers.fsm

-0xAF01

NO
```

#16 Οι αριθμοί f στην παρακάτω συμβολοσειρά είναι γραμμένοι με πεζά γράμματα. Για την σωστή αναπαράσταση ενός 16-δικου αριθμού θα έπρεπε να είναι γραμμένοι με κεφαλαία. Γι' αυτό το λόγο η συμβολοσειρά δεν είναι αποδεκτή από το αυτόματο.

```
./fsm 3_integers.fsm

0xff23

NO
```

#17 Στη γλώσσα Uni-C δεν υπάρχει αναπαράσταση προσημασμένων ακεραίων, συνεπώς λόγω του προσήμου η παρακάτω είσοδος θα απορριφθεί από το αυτόματο.

```
./fsm 3_integers.fsm
```

```
+01

NO
```

```
#18  Στη γλώσσα Uni-C δεν υπάρχει αναπαράσταση προσημασμένων ακεραίων, συνεπώς λόγω του προσήμου η παρακάτω είσοδος θα απορριφθεί από το αυτόματο.

./fsm 3_integers.fsm

+56

NO
```

```
#19  Δεν μπορεί να υπάρξη αναπαράσταση ενός ακεραίου αριθμού όταν εντός της συμβολοσειράς βρίσκεται κάποιο πρόσημο. Γι' αυτό το λόγο, το αυτόματο απορρίπτει την ακόλουθη συμβολοσειρά.

./fsm 3_integers.fsm

0x-FF

NO
```

```
#20  Η συμβολοσειρά δεν είναι αποδεκτή καθώς για την εισαγωγή της στο αυτόματο δεν χρησιμοποιήθηκε ο χαρακτήρας newline '\n', αντίθετα χρησιμοποιήθηκε το EOF και γι' αυτό το λόγο δεν έχουμε κάποιο θετικό αποτέλεσμα από το αυτόματο.

./fsm -trace 3_integers.fsm

619sz 6 -> sdec

sdec 1 -> sdec

sdec 9 -> sdec

sdec EOF -> bad

NO
```

3.7 Προβλήματα και τρόποι αντιμετώπισης

Πρόβλημα αντιμετωπίσαμε στον σχεδιασμό της κανονικής έκφρασης, καθώς η αρχική εκτίμηση ήταν αυτή.

```
(0|[1-9][0-9]*|0[x|X][0-9A-F]+|0[0-7]+)
```

Τα αποτελέσματα αναγνώρισης από το εργαλείο Regexpal ήταν τα εξής:

```
0

0 0 7

10

0x10
```

Η κανονική έκφραση είναι ουσιαστικά 4 επιμέρους κανονικές εκφράσεις σε μία λογική παράσταση με τελεστή το OR. Η σειρά ελέγχου της συμβολοσειράς εισόδου από τους κανόνες γίνεται από τα αριστερά προς τα δεξιά. Συνεπώς, η κανονική έκφραση δεν αναγνώριζε τους δεκαεξαδικούς και τους οκταδικούς ακεραίους, διότι ο πρώτος κανόνας που τηρούσαν ήταν ο κανόνας του 0 ως η μόνη εξαίρεση δεκαδικού ακεραίου που ξεκινάει με 0.

Το πρόβλημα αντιμετωπίστηκε αλλάζοντας την κανονική έκφραση.

```
([1-9][0-9]*|0[x|X][0-9A-F]+|0[0-7]+|0)
```

Ο κανόνας του 0 ως η μόνη εξαίρεση δεκαδικού ακεραίου που ξεκινάει με 0 ελέγχεται τελευταίος, ώστε να ελέγχονται πρώτα οι πρόσθετοι κανόνες που ισχύουν αντίστοιχα στους δεκαεξαδικούς και στους οκταδικούς ακεραίους.

3.8 Ρητή αναφορά ελλείψεων

Η αναφορά περιλαμβάνει όλα τα ζητούμενα που απαιτούνται:

- Πρότυπο αναγνώρισης
- Κανονική έκφραση με περιγραφή
- Αυτόματο πεπερασμένης κατάστασης με περιγραφή
- Πίνακας μεταβάσεων με πίνακα καταστάσεων
- Κωδικοποίηση μέσω FSM με σχολιασμό κώδικα
- Περιπτώσεις ελέγχου με εξαντλητικές δοκιμαστικές εκτελέσεις και σχολιασμό αποτελεσμάτων
- Προβλήματα που αντιμετωπίσαμε και τρόποι αντιμετώπισης

Στο διάγραμμα μετάβασης δεν σχεδιάστηκαν μη επιτρεπτές εξόδους (BAD) σε όλες τις καταστάσεις (συγκεκριμένα δεν σχεδιάστηκαν στις καταστάσεις SDEC, SHEX) για λόγους αναγνωσιμότητας του διαγράμματος. Ωστόσο, μη επιτρεπτές έξοδοι κωδικοποιήθηκαν στο FSM και σημειώθηκαν στον πίνακα μεταβάσεων σε όλες τις καταστάσεις.

Ο κώδικας εκτελέστηκε σε περιβάλλον Windows με εγκατάσταση Windows Subsystem for Linux (WSL). Το τερματικό που έτρεξε η ομάδα τον κώδικα ήταν σε Ubuntu 22.04 LTS, συνεπώς οι εντολές που χρησιμοποιήσαμε για την εκτέλεση του κώδικα είναι οι εξής:

```
gcc -o fsm fsm.c
```

```
./fsm 3_integers.fsm

./fsm -list 3_integers.fsm

./fsm -trace 3_integers.fsm
```

4. Αριθμοί κινούμενης υποδιαστολής

Την προσομοίωση την ανέλαβαν οι φοιτητές

Αθανασίου Βασίλειος Ευάγγελος (ΠΑΔΑ-19390005) – Ηλιού Ιωάννης (ΠΑΔΑ-19390066)

4.1 Πρότυπο Αναγνώρισης

Το αυτόματο αναγνωρίζει κυριολεκτικά κινητής υποδιαστολής της γλώσσας Uni-C με βάση ένα πρότυπο αναγνώρισης.

Ένας δεκαδικός αριθμός αποτελείται από το ακέραιο μέρος και το δεκαδικό μέρος που διαχωρίζονται με μία τελεία '.'. Τόσο το ακέραιο μέρος όσο και το δεκαδικό ορίζονται σύμφωνα με τους κανόνες που περιγράφηκαν εδώ για τους απλούς ακεραίους του δεκαδικού αριθμητικού συστήματος. Υπάρχει επίσης η δυνατότητα ορισμού δυνάμεων με χρήση του χαρακτήρα 'e' ή 'E'. Στην περίπτωση αυτή το ακέραιο ή/και δεκαδικό μέρος υψώνεται στην ακέραια δύναμη που ακολουθεί μετά τον χαρακτήρα 'e' ή 'E'.

Το πρότυπο τους αναγνωρίζει με το αναγνωριστικό όνομα ***float*** και επιστρέφονται ως token.

4.2 Κανονική Έκφραση

Η κανονική έκφραση για τις λεκτικές μονάδες των αριθμών κινητής υποδιαστολής της γλώσσας Uni-C είναι η εξής:

```
(?:[1-9][0-9]*|0)(?:\.(?:[1-9][0-9]*|0*[1-9]+))?(?:[eE](?:-?[1-9][0-9]*|0))?
```

Η έκφραση αναγνωρίζει τις εξής μορφές αριθμών:

- Ένας ακέραιος αριθμός με ένα ή περισσότερα αριθμητικά ψηφία (ισχύει ο κανόνας που ορίστηκε και στο πρότυπο των ακεραίων).
- Ένας ακέραιος αριθμός που μπορεί να περιλαμβάνει δεκαδικό μέρος, ακολουθούμενο από έναν ή περισσότερα αριθμητικά ψηφία.

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

- Ένας ακέραιος αριθμός στο ρόλο της βάσης που μπορεί να περιλαμβάνει εκθέτη (ακέραιος εκθέτης με τον ίδιο κανόνα που ορίστηκε και στο πρότυπο των ακεραίων) για την αναπαράσταση δύναμης.
- Ένας ακέραιος αριθμός που μπορεί να περιλαμβάνει δεκαδικό μέρος, ακολουθούμενο από έναν ή περισσότερα αριθμητικά ψηφία και το δεκαδικό μέρος να περιλαμβάνει εκθέτη για αναπαράσταση δύναμης.

Η έκφραση αναλύεται ως εξής:

```
(?:[1-9][0-9]*|0)
```

Αυτό το τμήμα αντιστοιχεί στο ακέραιο μέρος του αριθμού. Αναλύεται ως εξής:

```
?:
```

Αυτό το τμήμα υποδεικνύει ένα μη αναδρομικό group, δηλαδή ένα group χωρίς αποθήκευση του αποτελέσματός του.

```
[1-9][0-9]*
```

Αυτό αντιστοιχεί σε έναν μη μηδενικό ψηφίο ακολουθούμενο από μηδέν ή περισσότερα αριθμητικά ψηφία.

```
|
```

Αυτό δηλώνει εναλλαγή ανάμεσα στα δύο τμήματα της έκφρασης. Σε αυτή την περίπτωση, είτε ο αριθμός είναι θετικός είτε είναι το μηδέν.

```
0
```

Αντιστοιχεί στο μηδέν.

```
(?:\.(?:[1-9][0-9]*|0*[1-9]+))?
```

Αυτό το τμήμα αντιστοιχεί στο δεκαδικό μέρος του αριθμού. Αναλύεται ως εξής:

```
?:
```

Πάλι, αυτό υποδεικνύει ένα μη αναδρομικό group.

```
\.
```

Αντιστοιχεί στην υποδιαστολή.

```
?:
```

Μη αναδρομικό group.

```
[1-9][0-9]*
```

Αυτό αντιστοιχεί σε έναν μη μηδενικό ψηφίο ακολουθούμενο από μηδέν ή περισσότερα αριθμητικά ψηφία.

```
|
```

Αυτό δηλώνει εναλλαγή ανάμεσα στα δύο τμήματα της έκφρασης. Σε αυτή την περίπτωση, είτε ο αριθμός είναι θετικός είτε είναι το μηδέν.

```
0*[1-9]+
```

Αυτό αντιστοιχεί στο ψηφίο 0 μηδέν ή περισσότερες φορές ακολουθούμενο με ψηφίο από 1-9 μία ή περισσότερες φορές.

```
?
```

Αντιστοιχεί σε μηδέν ή μία φορά δηλώνοντας ότι η ύπαρξη δεκαδικού μέρους είναι προαιρετική.

```
(?:[eE](?:-?[1-9][0-9]*|0))?
```

Αυτό το τμήμα αντιστοιχεί στον εκθέτη της επιστημονικής μορφής (εάν υπάρχει). Αναλύεται ως εξής:

```
?:
```

Πάλι, μη αναδρομικό group.

```
[eE]
```

Αντιστοιχεί στο γράμμα "e" ή "E", που χρησιμοποιείται για την επιστημονική μορφή της αναπαράστασης του εκθέτη της δύναμης.

```
(?:-?[1-9][0-9]*|0)
```

Αντιστοιχεί στον αριθμό μετά το "e" ή "E". Αναλύεται ως εξής:

```
-?
```

Αντιστοιχεί σε μηδέν ή ένα αρνητικό πρόσημο, εάν υπάρχει.

```
[1-9][0-9]*
```

Αντιστοιχεί σε έναν μη μηδενικό ψηφίο ακολουθούμενο από μηδέν ή περισσότερα αριθμητικά ψηφία.

```
|
```

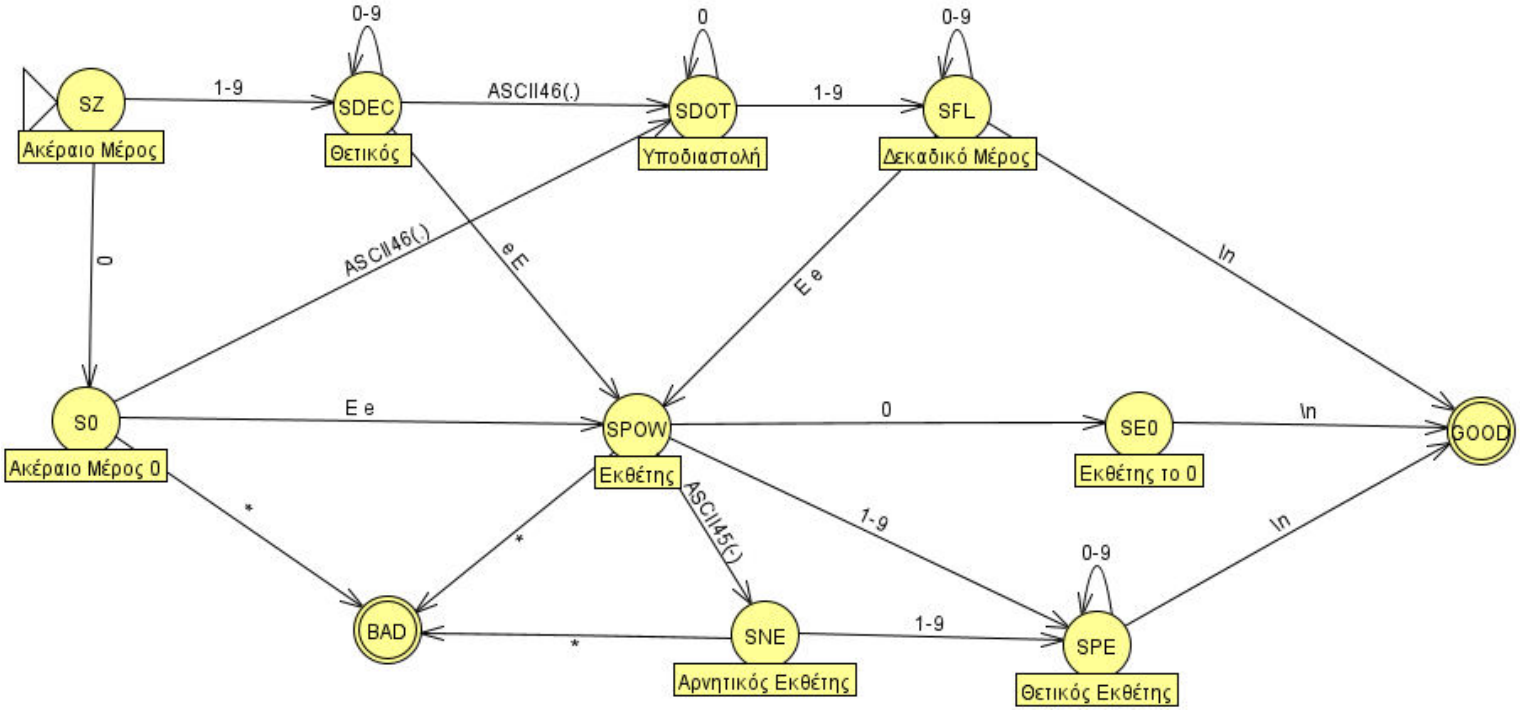
Εναλλαγή μεταξύ των δύο επιλογών.

```
0
```

Αντιστοιχεί στο μηδέν.

Συνοψίζοντας, αυτή η κανονική έκφραση αντιστοιχεί σε αριθμούς που μπορούν να είναι ακέραιοι, δεκαδικοί ή δυνάμεις σε επιστημονική μορφή.

4.3 Αυτόματο πεπερασμένης κατάστασης ή ΔΜ



Εικόνα 4.3.1 Πεπερασμένο αυτόματο αριθμών κινούμενης υποδιαστολής της γλώσσας Uni-C

Το αυτόματο αναγνωρίζει τη δομή ενός αριθμού, ο οποίος μπορεί να αποτελείται από τρία μέρη: το ακέραιο μέρος, το δεκαδικό μέρος και τη δύναμη. Η ακολουθία των καταστάσεων απεικονίζεται ως εξής:

- Η αρχική κατάσταση είναι η SZ, που αντιστοιχεί στο ακέραιο μέρος του αριθμού.
- Αν ξεκινάει με "0", πηγαίνουμε στην κατάσταση S0.
- Αν ξεκινάει με ψηφίο από 1 έως 9, πηγαίνουμε στην κατάσταση SDEC για θετικό ακέραιο.
- Ο ακέραιος μπορεί να ακολουθείται από το δεκαδικό μέρος, το οποίο αναγνωρίζεται από τις καταστάσεις SDOT και SFL.
- Αν ακολουθεί τελεία, πηγαίνουμε στην κατάσταση SDOT.
- Αν ακολουθεί "E" ή "e", πηγαίνουμε στην κατάσταση SPOW για να αναγνωρίσουμε τη δύναμη του αριθμού.
- Οι καταστάσεις SDOT και SFL αναγνωρίζουν το δεκαδικό μέρος.

- Η δύναμη αναγνωρίζεται από τις καταστάσεις SPOW, SE0, SNE και SPE.

Ο αριθμός είναι έγκυρος όταν φτάσει στην κατάσταση εξόδου GOOD, που σημαίνει ότι έχουν αναγνωριστεί το ακέραιο μέρος, το δεκαδικό μέρος (αν υπάρχει) και η δύναμη του αριθμού, και έπειτα ακολουθεί το τέλος γραμμής (\n). Αν συναντηθεί οποιοδήποτε μη έγκυρο σύμβολο, ο αριθμός θεωρείται μη έγκυρος.

4.4 Πίνακας μεταβάσεων

	-	0	1-9	.	E e	\n	ΚΑΤΑΣΤΑΣΗ ΕΞΟΔΟΥ
SZ	BAD	S0	SDEC	BAD	BAD	BAD	OXI
S0	BAD	BAD	BAD	SDOT	SPOW	BAD	OXI
SDEC	BAD	SDEC	SDEC	SDOT	SPOW	BAD	OXI
SDOT	BAD	SDOT	SFL	BAD	BAD	BAD	OXI
SFL	BAD	SFL	SFL	BAD	SPOW	GOOD	OXI
SPOW	SNE	SE0	SPE	BAD	BAD	BAD	OXI
SNE	BAD	BAD	SPE	BAD	BAD	BAD	OXI
SPE	BAD	SPE	SPE	BAD	BAD	GOOD	OXI
SE0	BAD	BAD	BAD	BAD	BAD	GOOD	OXI
GOOD							NAI
BAD							NAI

Πίνακας 4.4.1 Πίνακας μεταβάσεων των αριθμών κινητής υποδιαστολής της γλώσσας Uni-C

Ακρωνύμιο Κατάστασης	Όνομα Κατάστασης	Πληροφορίες Κατάστασης
SZ	State Z	Η κατάσταση για το 1ο ψηφίο
S0	State 0	Η κατάσταση που ο αριθμός ξεκινάει με 0 στο ακέραιος μέρος

SDEC	State Decimal	Η κατάσταση που αναγνωρίζει ακεραίους στο δεκαδικό σύστημα για το ακέραιο μέρος
SDOT	State Dot	Η κατάσταση που αναγνωρίζει την υποδιαστολή
SFL	State Float	Η κατάσταση που αναγνωρίζει ακεραίους στο δεκαδικό σύστημα για το δεκαδικό μέρος
SPOW	State Power	Η κατάσταση που αναγνωρίζει τον εκθέτη σε επιστημονική μορφή (E ή e) για την ανάγνωση δύναμης
SNE	State Negative Exponential	Η κατάσταση που αναγνωρίζει αρνητικό πρόσημο στον εκθέτη
SPE	State Positive Exponential	Η κατάσταση που αναγνωρίζει θετικό ακέραιο στον εκθέτη
SE0	State Exponential 0	Η κατάσταση που αναγνωρίζει το 0 στον εκθέτη

Πίνακας 4.4.2 Πίνακας καταστάσεων των αριθμών κινούμενης υποδιαστολής της γλώσσας Uni-C

4.5 Κωδικοποίηση μέσω FSM

Η κωδικοποίηση μέσω FSM του αυτομάτου των αριθμών κινητής υποδιαστολής της γλώσσας Uni-C βρίσκεται στο πηγαίο αρχείο **4_floats.fsm**

```
// ================== 4_Αριθμοί κινούμενης υποδιαστολής ==================


START = SZ                // ==== Η αρχική κατάσταση είναι η SZ ====


SZ:                       // --- [State Z] Κατάσταση για το ακέραιο μέρος ---

    0   -> S0                // Το ακέραιο μέρος είναι το 0
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

```
    1-9 -> SDEC             // Το ακέραιο μέρος ξεκινάει από 1-9
    *   -> BAD              // Ο αριθμός είναι μη έγκυρος


S0:                     // --- [State 0] Κατάσταση για το 0 ως ακέραιο μέρος ---
    .   -> SDOT             // Το ακέραιο μέρος είναι το 0 και ακολουθεί υποδιαστολή για
                               το δεκαδικό μέρος
    E e -> SPOW             // Η βάση της δύναμης είναι το 0 και ακολουθεί η
                               επιστημονική μορφή του εκθέτη
    *   -> BAD              // Ο αριθμός είναι μη έγκυρος


SDEC:                   // --- [State Decimal] Κατάσταση για το ακέραιο μέρος ---
    .   -> SDOT             // Ακολουθεί υποδιαστολή για το δεκαδικό μέρος
    0-9 -> SDEC             // Ακολουθούν κι άλλα ψηφία στο ακέραιο μέρος από 0-9
    E e -> SPOW             // Ακολουθεί η επιστημονική μορφή του εκθέτη
    *   -> BAD              // Ο αριθμός είναι μη έγκυρος


SDOT:                   // --- [State Dot] Κατάσταση για την υποδιαστολή ---
    0   -> SDOT             // Ακολουθεί το ψηφίο 0
    1-9 -> SFL              // Ακολουθούν ψηφία για το δεκαδικό μέρος από 1-9
    *   -> BAD              // Ο αριθμός είναι μη έγκυρος


SFL:                    // --- [State Float] Κατάσταση για το δεκαδικό μέρος ---
    0-9 -> SFL              // Ακολουθούν κι άλλα ψηφία για το δεκαδικό μέρος από 0-9
    E e -> SPOW             // Ακολουθεί η επιστημονική μορφή του εκθέτη
    \n  -> GOOD             // Ο αριθμός είναι έγκυρος
    *   -> BAD              // Ο αριθμός είναι μη έγκυρος


SPOW:                   // --- [State Power] Κατάσταση για τις δυνάμεις ---
    -   -> SNE              // Ακολουθεί αρνητικό πρόσημο για τον εκθέτη
    0   -> SE0              // Ο εκθέτης είναι 0
    1-9 -> SPE              // Ακολουθεί θετικός εκθέτης
    *   -> BAD              // Ο αριθμός είναι μη έγκυρος
```

```
SNE:                    // --- [State Negative Exponential] Κατάσταση για αρνητικό
                                                            εκθέτη ---

    1-9 -> SPE             // Ακολουθούν ψηφία από 1-9 για τον αρνητικό εκθέτη

    *   -> BAD             // Ο αριθμός είναι μη έγκυρος


SPE:                    // --- [State Positive Exponential] Κατάσταση για θετικό εκθέτη
                                                            ---

    0-9 -> SPE             // Ακολουθούν ψηφία από 1-9 για τον θετικό εκθέτη

    \n  -> GOOD            // Ο αριθμός είναι έγκυρος

    *   -> BAD             // Ο αριθμός είναι μη έγκυρος


SE0:                    // --- [State Exponential 0] Κατάσταση για το ψηφίο 0 για
                                                      εκθέτης στην δύναμη ---

    \n  -> GOOD            // Ο αριθμός είναι έγκυρος

    *   -> BAD             // Ο αριθμός είναι μη έγκυρος


BAD:                    // ==== Κατάσταση εξόδου: ΟΧΙ ====

    * -> BAD


GOOD(OK):               // ==== Κατάσταση εξόδου: ΝΑΙ ====
```

4.6 Περιπτώσεις ελέγχου

```
#1   Είναι έγκυρο διότι το ακέραιο μέρος ξεκινάει από 1-9 (SDEC), ακολουθεί η
μορφή του εκθέτη (SPOW) και ο εκθέτης είναι 8 (SPE). Τηρεί τους κανόνες των
ακεραίων και στην βάση και στον εκθέτη.

./fsm 4_floats.fsm

9e8

YES
```

```
#2   Είναι έγκυρο διότι το ακέραιο μέρος ξεκινάει από 1-9 (SDEC), ακολουθεί η
μορφή του εκθέτη (SPOW), περιέχει αρνητικό πρόσημο για τον εκθέτη (SNE) καθώς
```

και ο αρνητικός εκθέτης είναι 8. Τηρεί τους κανόνες των ακεραίων και στην βάση και στον εκθέτη.

```
./fsm 4_floats.fsm

9E-8

YES
```

#3 Είναι έγκυρο διότι το ακέραιο μέρος ξεκινάει από 1-9 (SDEC), ακολουθεί υποδιαστολή για το δεκαδικό μέρος (SDOT) και ακολουθούν ψηφία για το δεκαδικό μέρος από 0-9 (SFL). Τηρεί τους κανόνες των ακεραίων και στο ακέραιο και στο δεκαδικό μέρος.

```
./fsm 4_floats.fsm

3.14

YES
```

#4 Δεν είναι έγκυρο διότι το 10 μπορεί να αναπαρασταθεί ως ακέραιος, οπότε η αναπαράσταση του σε δεκαδική μορφή είναι περιττή.

```
./fsm -trace 4_floats.fsm

10.0

sz 1 -> sdec

sdec 0 -> sdec

sdec . -> sdot

sdot 0 -> sdot

sdot \n -> bad

bad EOF -> bad

NO
```

#5 Είναι έγκυρο διότι το ακέραιο μέρος είναι το 10 (SDEC), ακολουθεί υποδιαστολή για το δεκαδικό μέρος(SDOT) και το δεκαδικό μέρος περιέχει τα ψηφία 001(SFL). Τηρεί τους κανόνες των ακεραίων και στο ακέραιο και στο δεκαδικό μέρος.

```
./fsm -trace 4_floats.fsm
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

```
10.0001

sz 1 -> sdec

sdec 0 -> sdec

sdec . -> sdot

sdot 0 -> sdot

sdot 0 -> sdot

sdot 0 -> sdot

sdot 1 -> sfl

sfl \n -> good

YES
```

```
#6    Είναι έγκυρο διότι το ακέραιο μέρος ξεκινάει από 1-9 (SDEC), ακολουθεί η μορφή του εκθέτη (SPOW), περιέχει αρνητικό πρόσημο για τον εκθέτη (SNE) και ο αρνητικός εκθέτης είναι 15 (SPE). Τηρεί τους κανόνες των ακεραίων και στη βάση και στον εκθέτη.

./fsm -trace 4_floats.fsm

5e-15

sz 5 -> sdec

sdec e -> spow

spow - -> sne

sne 1 -> spe

spe 5 -> spe

spe \n -> good

YES
```

```
#7    Είναι έγκυρο διότι το ακέραιο μέρος ξεκινά από 1-9 (SDEC), ακολουθεί η μορφή του εκθέτη (SPOW), ακολουθεί ο θετικός εκθέτης (SPE) και ο εκθέτης είναι 100 (SPE). Τηρεί τους κανόνες των ακεραίων και στη βάση και στον εκθέτη.

./fsm 4_floats.fsm

1e100
```

```
YES
```

#8 Είναι έγκυρο διότι το ακέραιο μέρος ξεκινά από το 1-9 (SDEC), ακολουθεί υποδιαστολή για το δεκαδικό μέρος (SDOT), το δεκαδικό μέρος περιέχει τον αριθμό 1 (SFL), ακολουθεί η μορφή του εκθέτη (SPOW) και ο εκθέτης είναι 0 (SE0). Τηρεί τους κανόνες των ακεραίων και στο ακέραιο μέρος και στο δεκαδικό μέρος.

```
./fsm -trace 4_floats.fsm

3.1E0

sz 3 -> sdec

sdec . -> sdot

sdot 1 -> sfl

sfl E -> spow

spow 0 -> se0

se0 \n -> good

YES
```

#9 Είναι έγκυρο διότι το ακέραιο μέρος είναι 0 (S0), ακολουθεί η μορφή του εκθέτη (SPOW) και ο εκθέτης είναι 0 (SE0). Τηρεί τους κανόνες των ακεραίων και στο ακέραιο και στο δεκαδικό μέρος.

```
./fsm 4_floats.fsm

0e0

YES
```

#10 Δεν είναι έγκυρο διότι το ακέραιο μέρος δεν υποστηρίζει προσημασμένους ακέραιους σύμφωνα με το πρότυπο αναγνώρισης των ακεραίων της Uni-C. Πάραυτα η Uni-C υποστηρίζει προσημασμένους ακεραίους μόνο στον εκθέτη δύναμης (e-100).

```
./fsm 4_floats.fsm

-5.1e-100

NO
```

#11 Δεν είναι έγκυρο διότι το 0 δεν είναι αριθμός κινούμενης υποδιαστολής.

```
./fsm 4_floats.fsm

0

NO
```

#12 Δεν είναι έγκυρο διότι το ακέραιο μέρος δεν υποστηρίζει προσημασμένους ακέραιους σύμφωνα με το πρότυπο αναγνώρισης των ακεραίων της Uni-C.

```
./fsm 4_floats.fsm

-0.5

NO
```

#13 Δεν είναι έγκυρο διότι δεν τηρεί τον κανόνα για το δεκαδικό μέρος, καθώς δεν ακολουθεί δεκαδικό ψηφίο(SFL) μετά την υποδιαστολή(SDOT). Επίσης, δεν τηρεί τον κανόνα για την μορφή του εκθέτη, καθώς η βάση της δύναμης δεν έχει ψηφίο από 0-9(SPOW).

```
./fsm 4_floats.fsm

5.e1

NO
```

#14 Δεν είναι έγκυρο παρόλο που η μορφή της δύναμης φαίνεται συμβατή με ψηφίο στην βάση και ψηφίο στον εκθέτη. Ο λόγος είναι γιατί η επιστημονική μορφή του εκθέτη, δηλαδή το E, είναι ελληνικός χαρακτήρας και όχι λατινικός. Το πρόβλημα με τους ελληνικούς χαρακτήρες αναλύεται εδώ.

```
./fsm -trace 4_floats.fsm

1E1

sz 1 -> sdec

sdec \316 -> bad

Segmentation fault (core dumped)
```

#15 Δεν είναι έγκυρο διότι το ακέραιο μέρος δεν τηρεί τον κανόνα των ακεραίων να ξεκινάει με ψηφίο από 1-9. Το 0 αποτελεί τη μόνη εξαίρεση δεκαδικού ακεραίου

που ξεκινάει με 0, συνεπώς, η είσοδος 00 δεν είναι συμβατή με το πρότυπο της Uni-C.

```
./fsm 4_floats.fsm

00.01

NO
```

#16 Δεν είναι έγκυρο διότι ο εκθέτης δεν τηρεί τον κανόνα των ακεραίων να ξεκινάει με ψηφίο από 1-9. Το 0 αποτελεί τη μόνη εξαίρεση δεκαδικού ακεραίου που ξεκινάει με 0, συνεπώς, η είσοδος 01 δεν είναι συμβατή με το πρότυπο της Uni-C.

```
./fsm -trace 4_floats.fsm

0e01

sz 0 -> s0

s0 e -> spow

spow 0 -> se0

se0 1 -> bad

bad \n -> bad

bad EOF -> bad

NO
```

#17 Δεν είναι έγκυρο διότι η Uni-C δεν υποστηρίζει τέτοιες μορφές δυνάμεων. Μετά από χαρακτήρα εκθέτη σε επιστημονική μορφή (SPOW) μόνο δεκαδικός ακέραιος μπορεί να ακολουθεί (SPE), ή αρνητικός (SNE) ή το 0 (SE0).

```
./fsm 4_floats.fsm

5e1.5e1

NO
```

#18 Δεν είναι έγκυρο διότι δεν τηρεί τον κανόνα για τον εκθέτη, καθώς η Uni-C υποστηρίζει μόνο ακέραιους δεκαδικούς και όχι αριθμούς κινητής υποδιαστολής.

```
./fsm 4_floats.fsm

1E1.2
```

```
NO
```

#19 Δεν είναι έγκυρο διότι ο χαρακτήρας που οδηγεί στην επιτρεπτή κατάσταση εξόδου είναι το EOF (End-of-File) και όχι το newline (\n).

```
./fsm -trace 4_floats.fsm

0.1sz 0 -> s0

s0 . -> sdot

sdot 1 -> sfl

sfl EOF -> bad

NO
```

#20 Δεν είναι έγκυρο διότι δεν τηρεί τον κανόνα για το δεκαδικό μέρος, καθώς αντί για το χαρακτήρα “.” χρησιμοποιείται η “,” (SDOT) και δεν υπάρχει συνέχεια μετά την υποδιαστολή “,” για να δοθεί το δεκαδικό μέρος (SFL).

```
./fsm 4_floats.fsm

6,20

NO
```

#21 Δεν είναι έγκυρο διότι το 10 μπορεί να αναπαρασταθεί ως ακέραιος, οπότε η αναπαράσταση του σε δεκαδική μορφή είναι περιττή.

```
./fsm -trace 4_floats.fsm

0.0

sz 0 -> s0

s0 . -> sdot

sdot 0 -> sdot

sdot \n -> bad

bad EOF -> bad

NO
```

4.7 Προβλήματα και τρόποι αντιμετώπισης

Πρόβλημα αντιμετωπίσαμε στον σχεδιασμό της κανονικής έκφρασης, καθώς η αρχική εκτίμηση ήταν αυτή.

```
(?:[1-9][0-9]*|0)(?:\.\d+)?(?:[eE](?:-?[1-9][0-9]*|0))?
```

Τα αποτελέσματα αναγνώρισης από το εργαλείο Regexpal ήταν τα εξής:

```
0

0.0

0.00

5

5.0

5.00
```

Η κανονική έκφραση αναγνώριζε σωστά και τους ακέραιους όπως ορίζει το πρότυπο των ακεραίων, ωστόσο, το 0 δεν υφίστανται να αναγνωριστεί σε μορφή δεκαδικού (0.0). Το πρόβλημα αντιμετωπίστηκε στην έκφραση **\d** που αναγνωρίζει οποιοδήποτε ψηφίο από 0-9. Την αντικαταστήσαμε με αυτή την κανονική έκφραση.

```
(?:[1-9][0-9]*|0*[1-9]+)
```

Τα αποτελέσματα αναγνώρισης από το εργαλείο Regexpal ήταν τα εξής:

```
0

0.0

5

5.0

0.01

5.01
```

Πλέον, η κανονική έκφραση δεν αναγνωρίζει το 0.0 αλλά και οποιοδήποτε ακέραιο αριθμό σε δεκαδική μορφή (5.0). Επιλέξαμε να μην επεκτείνουμε την κανονική έκφραση να αναγνωρίζει οποιοδήποτε ακέραιο αριθμό σε δεκαδική μορφή εκτός από το 0, ώστε να μην βγει πολύ μεγάλη η κανονική έκφραση. Επιπρόσθετα θα δημιουργήσε πρόβλημα στην αναγνωσιμότητα του διαγράμματος μετάβασης. Με λίγα λόγια, οι ακέραιοι αναγνωρίζονται, απλά όχι στην δεκαδική τους μορφή.

4.8 Ρητή αναφορά ελλείψεων

Η αναφορά περιλαμβάνει όλα τα ζητούμενα που απαιτούνται:

- Πρότυπο αναγνώρισης
- Κανονική έκφραση με περιγραφή
- Αυτόματο πεπερασμένης κατάστασης με περιγραφή
- Πίνακας μεταβάσεων με πίνακα καταστάσεων
- Κωδικοποίηση μέσω FSM με σχολιασμό κώδικα
- Περιπτώσεις ελέγχου με εξαντλητικές δοκιμαστικές εκτελέσεις και σχολιασμό αποτελεσμάτων
- Προβλήματα που αντιμετωπίσαμε και τρόποι αντιμετώπισης

Στο διάγραμμα μετάβασης δεν σχεδιάστηκαν μη επιτρεπτές εξόδους (BAD) σε όλες τις καταστάσεις (συγκεκριμένα δεν σχεδιάστηκαν στις καταστάσεις SZ, SDEC, SDOT, SFL, SE0, SPE) για λόγους αναγνωσιμότητας του διαγράμματος. Ωστόσο, μη επιτρεπτές έξοδοι κωδικοποιήθηκαν στο FSM και σημειώθηκαν στον πίνακα μεταβάσεων σε όλες τις καταστάσεις.

Έλλειψη υπάρχει επίσης στον κώδικα FSM που δεν αναγνωρίζει τους ακεραίους (5, 5.0 κλπ), καθώς, καλύπτεται στο ενιαίο.

Ο κώδικας εκτελέστηκε σε περιβάλλον Windows με εγκατάσταση Windows Subsystem for Linux (WSL). Το τερματικό που έτρεξε η ομάδα τον κώδικα ήταν σε Ubuntu 22.04 LTS, συνεπώς οι εντολές που χρησιμοποιήσαμε για την εκτέλεση του κώδικα είναι οι εξής:

```
gcc -o fsm fsm.c

./fsm 4_floats.fsm

./fsm -list 4_floats.fsm

./fsm -trace 4_floats.fsm
```

5. Σχόλια

Την προσομοίωση την ανέλαβαν οι φοιτητές

Θεοχάρης Γεώργιος (ΠΑΔΑ-19390283) – Δομινάρης Βασίλειος (ΠΑΔΑ-21390055)

5.1 Πρότυπο Αναγνώρισης

Η λειτουργία του αυτομάτου είναι να αναγνωρίζει σχόλια της Uni-C με βάση ένα πρότυπο αναγνώρισης.

Το πρότυπο αναγνωρίζει δύο ειδών σχόλια: τα σχόλια μιας γραμμής και τα σχόλια πολλών γραμμών. Ένα σχόλιο μιας γραμμής αρχίζει με δύο χαρακτήρες '/' στη σειρά (δηλαδή //), που δεν αποτελούν μέρος της συμβολοσειράς του σχολίου, και ολοκληρώνεται με το τέλος της γραμμής (\n). Ένα σχόλιο πολλών γραμμών ξεκινάει με τους χαρακτήρες /* και ολοκληρώνεται μονάχα με τους χαρακτήρες */ ανεξάρτητα με τον αριθμό των γραμμών που έχουν μεσολαβήσει.

Το πρότυπο τους αναγνωρίζει με το αναγνωριστικό όνομα ***comment*** αλλά δεν επιστρέφονται ως token (αγνοούνται από τον συντακτικό αναλυτή).

5.2 Κανονική Έκφραση

Η κανονική έκφραση για τις λεκτικές μονάδες των σχολίων της γλώσσας Uni-C είναι η εξής:

```
(?:\/\/.*\n|\/\*[\s\S]*?\*\/)
```

Η κανονική έκφραση αναλύεται ως εξής:

- `(?: ... )`

  Ομαδοποίηση μη αναδρομικής αναφοράς. Αυτό σημαίνει ότι δεν δημιουργεί καταγραφή ομάδας.

- `\/\/.*\n`

  Αυτό το μέρος ταιριάζει με σχόλια που ξεκινούν με // και εκτείνονται έως το τέλος της γραμμής (\n).

- `|`

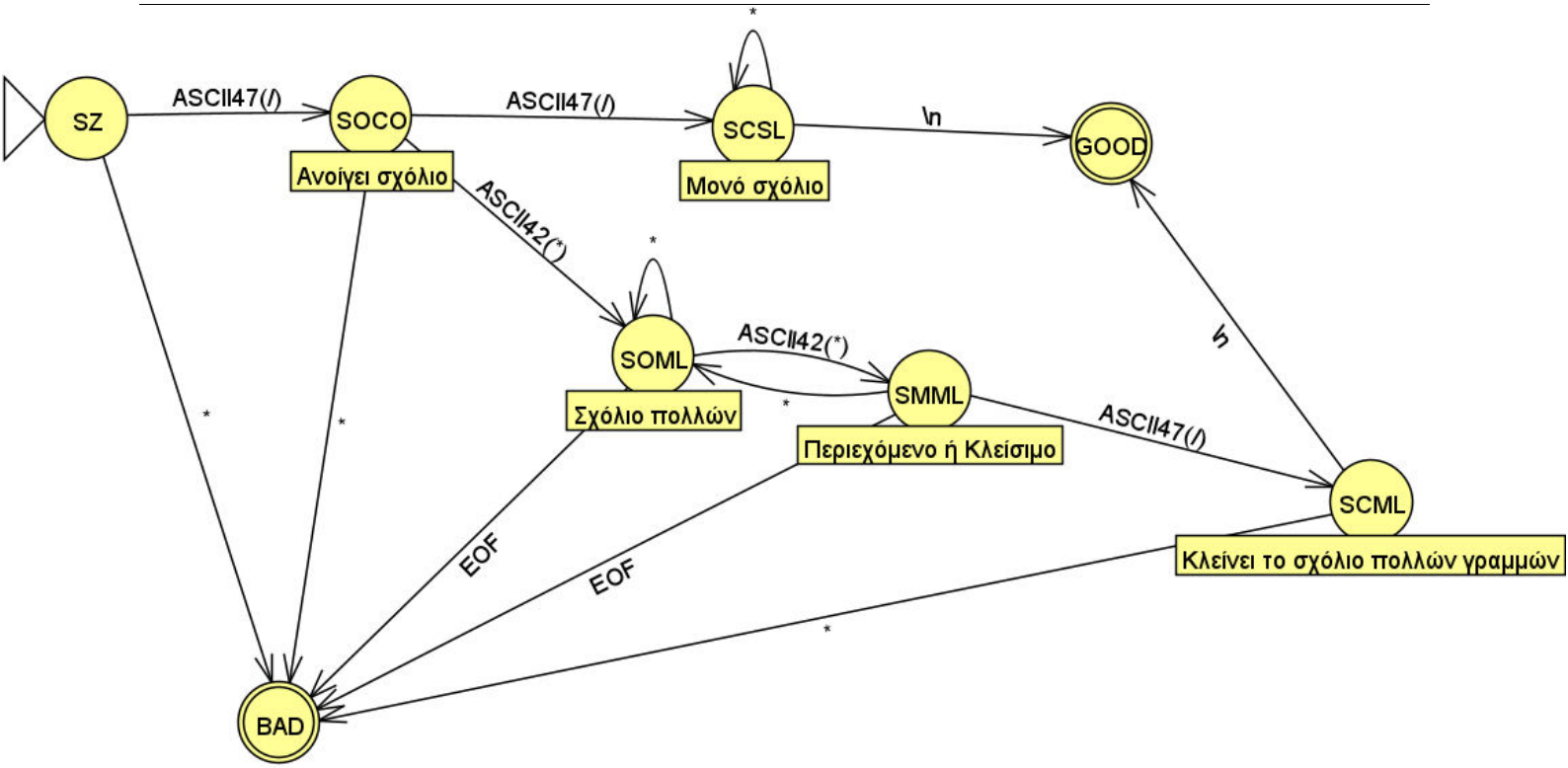
  Συμβολίζει την εναλλαγή, δηλαδή μπορεί να ταιριάξει είτε το ένα είτε το άλλο μέρος.

- `\/\*[\s\S]*?\*\/`

Αυτό το μέρος ταιριάζει με σχόλια που ξεκινούν με /* και τελειώνουν με */. Το [\s\S] ταιριάζει με οποιοδήποτε χαρακτήρα, συμπεριλαμβανομένων και των κενών και των νέων γραμμών, έτσι ώστε να μπορεί να ταιριάξει πολλαπλές γραμμές σχολίων.

Συνοπτικά, αυτή η κανονική έκφραση ταιριάζει με σχόλια σε κώδικα, είτε είναι μονή γραμμή που ξεκινά με //, είτε πολλαπλών γραμμών που ξεκινάνε με /* και τελειώνουν με */.

5.3 Αυτόματο πεπερασμένης κατάστασης ή ΔΜ



Εικόνα 5.3.1 Πεπερασμένο αυτόματο σχολίων της γλώσσας Uni-C

Το παραπάνω αυτόματο περιγράφει τον τρόπο λειτουργίας ενός προγράμματος που ανιχνεύει σχόλια σε κώδικα Uni-C. Αρχικά, η αρχική κατάσταση είναι η SZ, η οποία αναπαριστά την αναμονή για το άνοιγμα ενός σχολίου. Όταν εντοπίζεται το σύμβολο /, το αυτόματο μεταβαίνει στην κατάσταση SOCO για να ελέγξει αν πρόκειται για ένα σχόλιο μίας γραμμής ή ένα πολλαπλών γραμμών σχόλιο. Αν εντοπιστεί οποιοδήποτε άλλο σύμβολο στην αρχική κατάσταση, το αυτόματο μεταβαίνει στην κατάσταση BAD, αντικαθιστώντας το σχόλιο ως μη έγκυρο.

Οι καταστάσεις SOML και SMML ελέγχουν τα πολλαπλών γραμμών σχόλια, μεταβαίνοντας από την SOML στην SMML όταν εντοπίζεται ένα ακόμα *, και από την SMML στην SCML όταν βρεθεί ένα / μετά από ένα *, υποδηλώνοντας το τέλος του σχολίου πολλαπλών γραμμών.

Οι καταστάσεις SCML και SCSL αντιπροσωπεύουν το τέλος ενός έγκυρου σχολίου πολλαπλών γραμμών ή μονής γραμμής αντίστοιχα. Το αυτόματο επιστρέφει στην αρχική κατάσταση μετά από ένα έγκυρο σχόλιο.

Τέλος, η κατάσταση GOOD δηλώνει ότι ένα σχόλιο έχει ανιχνευθεί επιτυχώς.

5.4 Πίνακας μεταβάσεων

	/	*	EOF	\n	Άλλος Χαρακτήρας	ΚΑΤΑΣΤΑΣΗ ΕΞΟΔΟΥ
SZ	SOCO	BAD	BAD	BAD	BAD	OXI
SOCO	SCSL	SOML	BAD	BAD	BAD	OXI
SOML	SOML	SMML	BAD	SOML	SOML	OXI
SMML	SCML	SOML	BAD	SOML	SOML	OXI
SCML	BAD	BAD	BAD	GOOD	BAD	OXI
SCSL	BAD	BAD	BAD	GOOD	BAD	OXI
GOOD						NAI
BAD						NAI

Πίνακας 5.4.1 Πίνακας μεταβάσεων των σχολίων της γλώσσας Uni-C

Ακρωνύμιο Κατάστασης	Όνομα Κατάστασης	Πληροφορίες Κατάστασης
SZ	State Z	Η κατάσταση για το άνοιγμα σχολίου
SOCO	State Open Comment	Η κατάσταση για το άνοιγμα σχολίου μονής γραμμής ή πολλών
SOML	State Open Multiple Line (Comment)	Η κατάσταση για το άνοιγμα σχολίου πολλών γραμμών και ανάγνωσης περιεχομένου
SMML	State Mid Multiple Line (Comment)	Η κατάσταση για το κλείσιμο σχολίου πολλών γραμμών ή ανάγνωση

		περιεχομένου
SCML	State Close Multiple Line (Comment)	Η κατάσταση για το κλείσιμο του σχολίου πολλών γραμμών
SCSL	State Close Single Line (Comment)	Η κατάσταση για το κλείσιμο σχολίου μονής γραμμής ή ανάγνωση περιεχομένου

Πίνακας 5.4.2 Πίνακας καταστάσεων των σχολίων της γλώσσας Uni-C

5.5 Κωδικοποίηση μέσω FSM

Η κωδικοποίηση μέσω FSM του αυτομάτου των σχολίων της γλώσσας Uni-C βρίσκεται στο πηγαίο αρχείο **5_comments.fsm**

```
// ================== 5_Σχόλια ==================


START = SZ          // ==== Η αρχική κατάσταση είναι η SZ ====


SZ:                 // --- [State Z] Κατάσταση για το άνοιγμα σχολίου ---
    / ->  SOCO          // Ανοίγει σχόλιο
    * ->  BAD           // Το σχόλιο είναι μη έγκυρο


SOCO:               // --- [State Open Comment] Κατάσταση ανοιχτού σχολίου ---
    /  -> SCSL          // Ανοίγει σχόλιο μίας γραμμής
    \* -> SOML          // Ανοίγει σχόλιο πολλών γραμμών
    *  -> BAD           // Το σχόλιο είναι μη έγκυρο


SOML:               // --- [State Open Multiple Line (Comment)] Κατάσταση σχολίου πολλών
                                                            γραμμών ---
    \*  -> SMML       // Πρόθεση για κλείσιμο του σχολίου
    *   -> SOML       // Περιεχόμενο σχολίου
```

```
    EOF -> BAD          // Το σχόλιο είναι μη έγκυρο


SMML:               // --- [State Mid Multiple Line (Comment)] Κατάσταση σχολίου πολλών
                                                              γραμμών ---
    /   -> SCML         // Κλείνει το σχόλιο πολλών γραμμών
    *   -> SOML         // Επιστροφή στην κατάσταση για περιεχόμενο σχολίου
    EOF -> BAD          // Το σχόλιο είναι μη έγκυρο


SCML:               // --- [State Close Multiple Line (Comment)] Κατάσταση έγκυρου
                                                                  σχολίου πολλών γραμμών
                                                                  ---
    \n  -> GOOD         // Το σχόλιο είναι έγκυρο
    *   -> BAD          // Το σχόλιο είναι μη έγκυρο


SCSL:               // --- [State Close Single Line (Comment)] Κατάσταση έγκυρου σχολίου
                                                              μίας γραμμής ---
    \n  -> GOOD         // Το σχόλιο είναι έγκυρο
    *   -> SCSL         // Το σχόλιο είναι μη έγκυρο


BAD:                // ==== Κατάσταση εξόδου: OXI ====
    * -> BAD


GOOD(OK):           // ==== Κατάσταση εξόδου: NAI ====
```

5.6 Περιπτώσεις ελέγχου

#1 Η είσοδος γίνεται αποδεκτή από το αυτόματο μιας και αποτελεί ένα μονό σχόλιο, ξεκινάει με '//' και καταλήγει σε '\n'.

```
./fsm 5_comments.fsm

// hello world!!

YES
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

#2 Εφόσον δεν ξεκινάει η συμβολοσειρά με '//' ή '/*' δεν μπορεί να αποτελεί σχόλιο στη γλώσσα Uni-C.

```
./fsm 5_comments.fsm

/hello world!!/

NO
```

#3 Τα σχόλια στη γλώσσα Uni-C πρέπει να ξεκινάνε είτε με '//' είτε με '/*', συνεπώς η ακόλουθη συμβολοσειρά δεν είναι αποδεκτή από το αυτόματο. Το ακόλουθο σχόλιο υποστηρίζεται στη γλώσσα Python.

```
./fsm 5_comments.fsm

# hello world!

NO
```

#4 Η παρακάτω συμβολοσειρά δεν είναι αποδεκτή από το αυτόματο, διότι ο χαρακτήρας 'ο' που εισάγεται πριν το '\n' είναι ελληνικός χαρακτήρας. Η Uni-C δεν υποστηρίζει ελληνικούς χαρακτήρες.

```
./fsm -trace 5_comments.fsm

// hello

sz / -> soco

soco / -> scsl

scsl \s -> scsl

scsl h -> scsl

scsl e -> scsl

scsl l -> scsl

scsl l -> scsl

scsl \316 -> soml

soml \277 -> soml

soml \n -> soml

soml EOF -> bad

NO
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

#5 Η συγκεκριμένη είσοδος δεν είναι αποδεκτή από το αυτόματο, καθώς δεν υπάρχει κάποιος ειδικός χαρακτήρας που να παραπέμπει σε συγγραφή σχολίου. Η συμβολοσειρά δεν υφίσταται ως σχόλιο.

```
./fsm 5_comments.fsm

hello world

NO
```

#6 Η ακόλουθη συμβολοσειρά περικλείεται από τους ειδικούς χαρακτήρες '/*' και '*/' πριν δοθεί το '\n', συνεπώς πρόκειται για σχόλιο και γίνεται αποδεκτό από το αυτόματο.

```
./fsm 5_comments.fsm

/* hello world!! */

YES
```

#7 Η συμβολοσειρά είναι αποδεκτή από το αυτόματο ως σχόλιο καθώς ξεκινάει με τους ειδικούς χαρακτήρες '//' και στη συνέχεια δεν υπάρχει κάποιο ελληνικό γράμμα ή έλλειψη του '\n'.

```
./fsm 5_comments.fsm

//hello//world//!!

YES
```

#8 Η συμβολοσειρά δεν είναι αποδεκτή από το αυτόματο, διότι αντί για τους ειδικούς χαρακτήρες '//' που ορίζουν ένα σχόλιο υπάρχουν λανθασμένα οι ειδικοί χαρακτήρες '\\'.

```
./fsm 5_comments.fsm

\\ hello world!!

NO
```

#9 Η συμβολοσειρά περικλείεται από τους ειδικούς χαρακτήρες '/*' και '*/' συνεπώς αποτελεί σχόλιο και άρα γίνεται αποδεκτή από το αυτόματο.

```
./fsm 5_comments.fsm
/*
 * hello
 * world
 * !!
 */
YES
```

```
#10  Η συμβολοσειρά ξεκινάει με τους ειδικούς χαρακτήρες ‘/*’ και τελειώνει με τους ειδικούς χαρακτήρες ‘*/’ και τον ειδικό χαρακτήρα ‘/n’. Για τον λόγο αυτό αποτελεί σχόλιο και γίνεται αποδεκτή από το αυτόματο.

./fsm -trace 5_comments.fsm
/*
sz / -> soco
soco * -> soml
soml \s -> soml
soml \n -> soml
hi
soml h -> soml
soml i -> soml
soml \n -> soml
*/
soml * -> smml
smml / -> scml
scml \n -> good
YES
```

#11 Αντί για τον ειδικό χαρακτήρα '/' στη συγκεκριμένη συμβολοσειρά χρησιμοποιείται ο ειδικός χαρακτήρας '\'. Για τον λόγο αυτό δεν αποτελεί σωστή παράθεση σχολίου και απορρίπτεται από το αυτόματο.

```
./fsm 5_comments.fsm

\* hello world! *\

NO
```

#12 Οι ειδικοί χαρακτήρες '/*' στην ακόλουθη συμβολοσειρά δεν ακολουθούνται στη συνέχεια από τους ειδικούς χαρακτήρες '*/' επομένως δεν υπάρχει σωστή παράθεση σχολίου. Η συμβολοσειρά απορρίπτεται από το αυτόματο.

```
./fsm 5_comments.fsm

/*//

//hi

NO
```

#13 Εφόσον η συμβολοσειρά ξεκινάει με τους ειδικούς χαρακτήρες '//', δεν περιέχει στο ενδιάμεσο κάποιον χαρακτήρα που απαγορεύεται και καταλήγει σε '\n' πρόκειται για ένα σωστό σχόλιο μονής γραμμής στη γλώσσα Uni-C. Το αυτόματο αποδέχεται την συμβολοσειρά.

```
./fsm 5_comments.fsm

//* hi *//

YES
```

#14 Η συμβολοσειρά δεν έχει κάποιον ειδικό χαρακτήρα που βρίσκεται σε σχόλια της Uni-C συνεπώς δεν αποτελεί σχόλιο και απορρίπτεται από το αυτόματο. Το σχόλιο υποστηρίζεται στη γλώσσα Matlab.

```
./fsm 5_comments.fsm

%% hi

NO
```

#15 Η συμβολοσειρά παρόλο που αποτελεί σχόλιο μονής γραμμής, απορρίπτεται από το αυτόματο. Αυτό συμβαίνει διότι ενώ ξεκινάει με τους ειδικούς χαρακτήρες '//',

δεν τελειώνει με το '\n' ώστε να ήταν σχόλιο. Αντίθετα, τελειώνει με τελευταίο χαρακτήρα το EOF.

```
./fsm -trace 5_comments.fsm

// hisz / -> soco

soco / -> scsl

scsl \s -> scsl

scsl h -> scsl

scsl i -> scsl

scsl EOF -> scsl

NO
```

#16 Η ακόλουθη συμβολοσειρά γίνεται αποδεκτή από το αυτόματο, διότι ξεκινάει με τους ειδικούς χαρακτήρες '//' και τελειώνει με το '\n' χωρίς στο ενδιάμεσο να προκύπτει κάποιος χαρακτήρας που απαγορεύεται.

```
./fsm 5_comments.fsm

// hi */

YES
```

#17 Η συμβολοσειρά ξεκινάει με τους ειδικούς χαρακτήρες '/*' και για να αποτελεί σχόλιο πρέπει να τελειώνει και με τους ειδικούς χαρακτήρες '*/'. Αυτό δεν συμβαίνει και η συμβολοσειρά απορρίπτεται από το αυτόματο.

```
./fsm 5_comments.fsm

/* hi //

NO
```

#18 Η συμβολοσειρά δεν είναι αποδεκτή από το αυτόματο για να είναι σχόλιο της Uni-C, καθώς, το περιεχόμενο του σχολίου είναι με ελληνικά γράμματα. Το πρόβλημα με τους ελληνικούς χαρακτήρες περιγράφεται εδώ.

```
./fsm -trace 5_comments.fsm

// Αυτό είναι ένα σχόλιο

sz / -> soco
```

```
soco / -> scsl

scsl \s -> scsl

scsl \316 -> soml

fsm: in 5_comments.fsm, state 'soml' input \221 not accepted
```

5.7 Προβλήματα και τρόποι αντιμετώπισης

Πρόβλημα αντιμετωπίσαμε πάλι στην αναγνώριση των ελληνικών χαρακτήρων, καθώς, υφίστανται να υπάρχουν σχόλια με ελληνικά γράμματα. Τρόπος αντιμετώπισης δεν βρέθηκε και το πόρισμα είναι παρόμοιο που αναγράφεται αναλυτικά εδώ. Συνεπώς, η Uni-C υποστηρίζει σχόλια μόνο με λατινικά γράμματα, ψηφία, ειδικούς χαρακτήρες και whitespace χαρακτήρες.

Επίσης, πρόβλημα αντιμετωπίσαμε στην κωδικοποίηση μέσω FSM και κυρίως στην κατάσταση εξόδου BAD, καθώς, εξαρχής δεν είχαμε εξερχόμενη μεταβάση από την BAD και στα αυτόματα των προαναφερόμενων λεκτικών μονάδων της Uni-C. Συνεπώς, δεν εμφανιζόταν το αποτέλεσμα NO σ' όλες τις μη επιτρεπτές συμβολοσειρές, αλλά το παρακάτω αποτέλεσμα από ένα ενδεικτικό παράδειγμα.

```
./fsm 5_comments.fsm

this is a comment

fsm: in 5_comments.fsm, state 'bad' input h not accepted
```

Το πρόβλημα λύθηκε προσθέτοντας στον κώδικα FSM την παρακάτω μετάβαση.

```
BAD:                // ==== Κατάσταση εξόδου: ΟΧΙ ====

    * -> BAD
```

Συνεπώς τώρα το αποτέλεσμα της μη επιτρεπτής συμβολοσειράς είναι το παρακάτω.

```
./fsm 5_comments.fsm

this is a comment

NO
```

5.8 Ρητή αναφορά ελλείψεων

Η αναφορά περιλαμβάνει όλα τα ζητούμενα που απαιτούνται:

- Πρότυπο αναγνώρισης
- Κανονική έκφραση με περιγραφή

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

- Αυτόματο πεπερασμένης κατάστασης με περιγραφή
- Πίνακας μεταβάσεων με πίνακα καταστάσεων
- Κωδικοποίηση μέσω FSM με σχολιασμό κώδικα
- Περιπτώσεις ελέγχου με εξαντλητικές δοκιμαστικές εκτελέσεις και σχολιασμό αποτελεσμάτων
- Προβλήματα που αντιμετωπίσαμε και τρόποι αντιμετώπισης

Ο κώδικας εκτελέστηκε σε περιβάλλον Windows με εγκατάσταση Windows Subsystem for Linux (WSL). Το τερματικό που έτρεξε η ομάδα τον κώδικα ήταν σε Ubuntu 22.04 LTS, συνεπώς οι εντολές που χρησιμοποιήσαμε για την εκτέλεση του κώδικα είναι οι εξής:

```
gcc -o fsm fsm.c

./fsm 5_comments.fsm

./fsm -list 5_comments.fsm

./fsm -trace 5_comments.fsm
```

6. White_spaces χαρακτήρες

Την προσομοίωση την ανέλαβαν οι φοιτητές

Δομινάρης Βασίλειος (ΠΑΔΑ-21390055) – Τάτσης Παντελής (ΠΑΔΑ-20390226)

6.1 Πρότυπο Αναγνώρισης

Η λειτουργία του αυτομάτου είναι να αναγνωρίζει τους white_spaces χαρακτήρες της Uni-C με βάση ένα πρότυπο αναγνώρισης.

Το πρότυπο αναγνωρίζει τις ακολουθίες διαχωριστικών χαρακτήρων (κενά, tabs ή συνδυασμοί τους), που χρησιμοποιούνται για το διαχωρισμό των λεξημάτων.

Το πρότυπο επίσης τους αναγνωρίζει με το αναγνωριστικό όνομα ***white_spaces*** αλλά δεν επιστρέφονται ως tokens (αγνοούνται από τον συντακτικό αναλυτή).

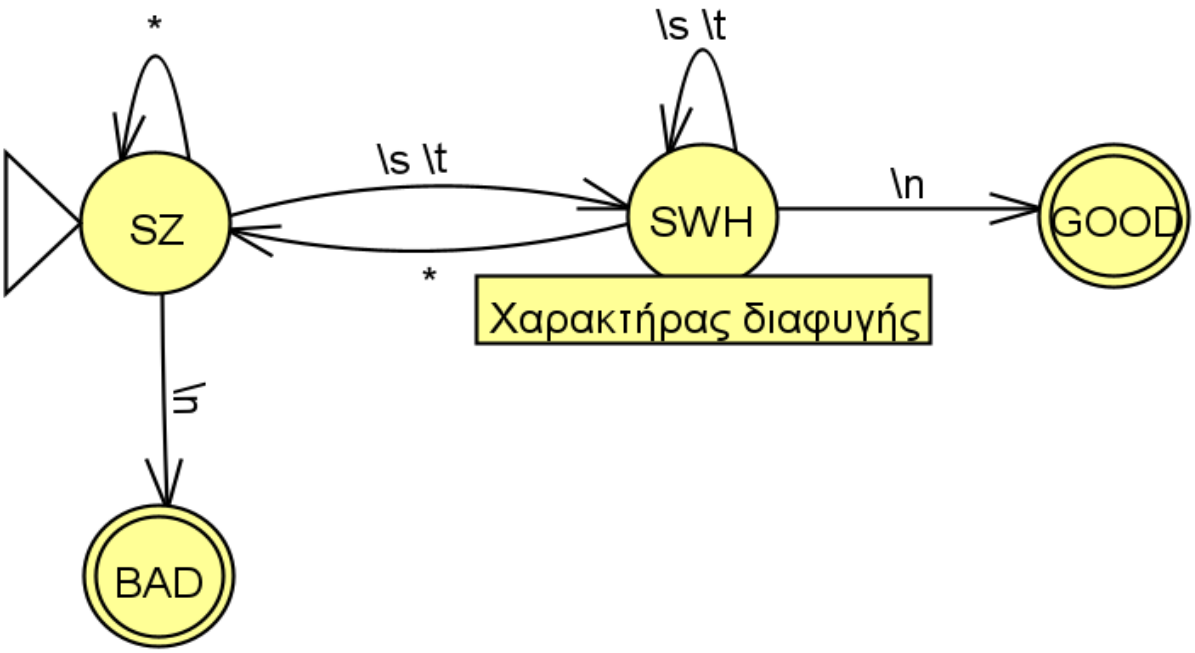
6.2 Κανονική Έκφραση

Η κανονική έκφραση για τις λεκτικές μονάδες των whites_spaces χαρακτήρων της γλώσσας Uni-C είναι η εξής:

```
\s+
```

Οι whitespace χαρακτήρες της γλώσσας Uni-C είναι το space (\s), το tab (\t), ο χαρακτήρας newline(\n), καθώς και το τέλος αρχείου (EOF). Μια κανονική έκφραση που αναγνωρίζει αυτούς τους whitespace χαρακτήρες είναι η \s+. Η κανονική έκφραση αναπαριστά ένα χαρακτήρα κενού (space), σύμβολο tab ή οποιοδήποτε άλλο χαρακτήρα κενού, και επαναλαμβάνεται μία ή περισσότερες φορές. Με άλλα λόγια, ταιριάζει με οποιοδήποτε αριθμό από έναν ή περισσότερους χαρακτήρες κενού, ανεξάρτητα από το πόσους χαρακτήρες κενού βρίσκονται μαζί.

6.3 Αυτόματο πεπερασμένης κατάστασης ή ΔΜ



Εικόνα 6.3.1 Πεπερασμένο αυτόματο χαρακτήρων διαφυγής της γλώσσας Uni-C

Το διάγραμμα μεταβάσεων ξεκινάει από το SZ. Αν διαβαστεί οποιοσδήποτε white space χαρακτήρας από τους space, tab (\s \t) πηγαίνει στην κατάσταση SWH. Αν διαβαστεί οποιοσδήποτε χαρακτήρας πέρα από τους χαρακτήρες white space που αναφέρθηκαν προηγουμένως, παραμένει στην κατάσταση SZ. Αν από την SZ διαβαστεί αλλαγή γραμμής πηγαίνει στην κατάσταση εξόδου BAD. Για την SWH εφόσον διαβάζονται white space χαρακτήρες, η κατάσταση παραμένει όπως έχει στην SWH, αλλιώς αν διαβαστεί οποιοσδήποτε άλλος χαρακτήρας πέρα από τους white spaces ξαναπηγαίνει στο SZ. Αν από την SWH διαβαστεί αλλαγή γραμμής πηγαίνει στην κατάσταση εξόδου GOOD.

6.4 Πίνακας μεταβάσεων

	\s	\t	Άλλος χαρακτήρας	EOF	\n	ΚΑΤΑΣΤΑΣΗ ΕΞΟΔΟΥ
SZ	SWH	SWH	SZ	BAD	BAD	OXI
SWH	SWH	SWH	SZ	BAD	GOOD	OXI

GOOD						NAI
BAD						NAI

Πίνακας 6.4.1 Πίνακας μεταβάσεων των χαρακτήρων διαφυγής της γλώσσας Uni-C

Ακρωνύμιο Κατάστασης	Όνομα Κατάστασης	Πληροφορίες Κατάστασης
SZ	State Z	Η κατάσταση για την ανάγνωση του 1ου χαρακτήρα διαφυγής
SWH	State Whitespaces	Η κατάσταση για την ανάγνωση οποιουδήποτε άλλου χαρακτήρα μαζί με χαρακτήρα διαφυγής

Πίνακας 6.4.2 Πίνακας καταστάσεων των χαρακτήρων διαφυγής της γλώσσας Uni-C

6.5 Κωδικοποίηση μέσω FSM

Η κωδικοποίηση μέσω FSM του αυτομάτου των χαρακτήρων διαφυγής της γλώσσας Uni-C βρίσκεται στο πηγαίο αρχείο **6_white_spaces.fsm**

```
// ================== 6_White_spaces ==================


START = SZ              // ==== Η αρχική κατάσταση είναι η SZ ====


SZ:                     // --- [State Z] Κατάσταση για ανάγνωση χαρακτήρα διαφυγής ---

    \s \t -> SWH           // Αναγνωρίζει χαρακτήρα διαφυγής

    *     -> SZ            // Αναγνωρίζει οποιοδήποτε άλλον χαρακτήρα διαβάζει

    \n    -> BAD           // Δεν διαβάστηκε χαρακτήρας διαφυγής στο τέλος του αρχείου
                              πέρα από το newline


SWH:                    // --- [State Whitespace] Κατάσταση για ανάγνωση χαρακτήρα
                                                    διαφυγής ---

    *     -> SZ            // Αναγνωρίζει οποιοδήποτε άλλο χαρακτήρα διαβάζεται,
                              επιστροφή στην αρχική κατάσταση

    \n    -> GOOD          // Τελειώνει με χαρακτήρα διαφυγής και ακολουθεί newline
```

```
    \s \t -> SWH          // Ακολουθεί χαρακτήρας διαφυγής, παραμονή στην ίδια κατάσταση


BAD:                  // ==== Κατάσταση εξόδου: ΟΧΙ ====
    * -> BAD


GOOD(OK):             // ==== Κατάσταση εξόδου: ΝΑΙ ====
```

6.6 Περιπτώσεις ελέγχου

#1 Είναι έγκυρο διότι πριν διαβαστεί το newline ακολουθούμενο από EOF, ακολουθείται το tab (SWH) που αποτελεί whitespace χαρακτήρας στην γλώσσα Uni-C.

```
./fsm -trace 6_white_spaces.fsm


sz \s -> swh

swh \t -> swh

swh \n -> good

YES
```

#2 Δεν είναι έγκυρο διότι δεν περιλαμβάνει tabs ή κενά ή συνδυασμοί τους πριν διαβαστεί το newline ακολουθούμενο από EOF. Το newline και το EOF παρόλο που αποτελούν whitespace χαρακτήρες, στην γλώσσα Uni-C αναγνωρίζονται και επιστρέφονται ως tokens με διαφορετικά αναγνωριστικά ονόματα.

```
./fsm -trace 6_white_spaces.fsm

hi

sz h -> sz

sz i -> sz

sz \n -> bad

bad EOF -> bad

NO
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

#3 Δεν είναι έγκυρο διότι δεν υπάρχουν κενά, tabs ή συνδυασμοί τους και ούτε καταλήγει με newline αλλά με EOF. Το EOF παρόλο που αποτελεί whitespace χαρακτήρας, στην γλώσσα Uni-C αναγνωρίζεται και επιστρέφεται ως token με διαφορετικό αναγνωριστικό όνομα.

```
./fsm -trace 6_white_spaces.fsm

sz EOF -> sz

NO
```

#4 Δεν είναι έγκυρο διότι δεν υπάρχουν κενά, tab ή συνδυασμοί τους πριν διαβαστεί το newline ακολουθούμενο από EOF.

```
./fsm -trace 6_white_spaces.fsm

        hi

sz \t -> swh

swh h -> sz

sz i -> sz

sz \n -> bad

bad EOF -> bad

NO
```

#5 Είναι έγκυρο διότι διαβάζει κενό πριν διαβαστεί το newline και το τέλος του αρχείου EOF.

```
./fsm -trace 6_white_spaces.fsm

hi Mark

sz h -> sz

sz i -> sz

sz \s -> swh

swh M -> sz

sz a -> sz

sz r -> sz
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

```
sz k -> sz

sz \s -> swh

swh \n -> good

YES
```

#6 Είναι έγκυρο διότι ακολουθεί τον κανόνα και μεταβαίνει από την SZ στην SWH όταν διαβάζεται ο χαρακτήρας (\t), επίσης παραμένει στην κατάσταση SWH όταν διαβάζεται ο χαρακτήρας διαφυγής (\t) και έπειτα όταν διαβάζεται ο χαρακτήρας νέας γραμμής μεταβαίνει στην GOOD.

```
./fsm -trace 6_white_spaces.fsm


sz \t -> swh

swh \t -> swh

swh \n -> good

YES
```

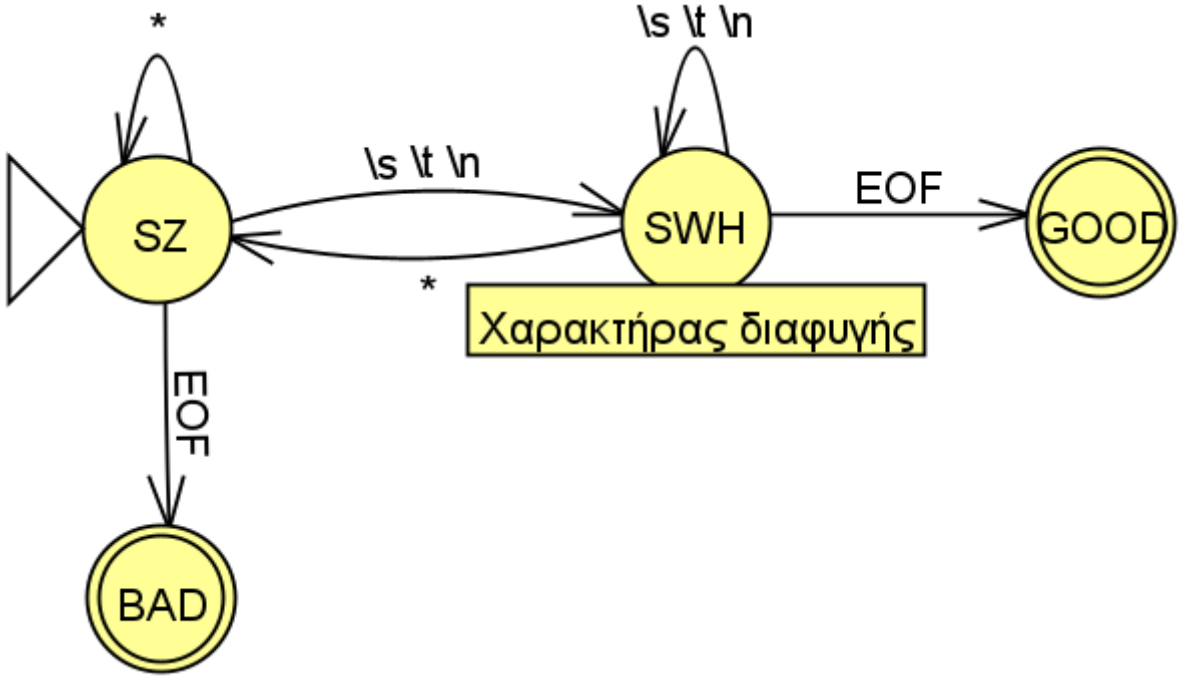
#7 Δεν είναι έγκυρο διότι δεν ξεκινάει με τον χαρακτήρα διαφυγής(\t) που αναμένεται για να μεταβεί στην κατάσταση SWH, δεν ακολουθεί μετά το "h" χαρακτήρας διαφυγής(\t) για να παραμείνει στην κατάσταση SWH και δεν διαβάζει (\n) για να δώσει GOOD.

```
./fsm -trace 6_white_spaces.fsm

hi mark sz h -> sz

sz i -> sz

sz \s -> swh

swh m -> sz

sz a -> sz

sz r -> sz

sz k -> sz

sz \s -> swh

swh EOF -> sz
```

NO

6.7 Προβλήματα και τρόποι αντιμετώπισης

Πρόβλημα αντιμετωπίσαμε στον σχεδιασμό του διαγράμματος μετάβασης, καθώς, σ' όλα τα διαγράμματα μετάβασης των προαναφερόμενων λεκτικών μονάδων της γλώσσας Uni-C, χρησιμοποιούσαμε τον χαρακτήρα αλλαγής γραμμής (\n) για την μετάβαση στην κατάσταση εξόδου GOOD. Το αρχικό διάγραμμα μετάβασης συμπεριλάμβανε και τον χαρακτήρα newline πέρα από τα tabs και τα κενά για τις ενδιάμεσες καταστάσεις (καθώς πρόκειται και αυτό για whitespace χαρακτήρα), ενώ, ο τελικός χαρακτήρας για την μετάβαση στην GOOD ήταν το EOF.



Εικόνα 6.7.1 Αρχικό πεπερασμένο αυτόματο χαρακτήρων διαφυγής της γλώσσας Uni-C

Μετά από νέα σύσκεψη της ομάδας συνειδητοποιήσαμε ότι και το EOF αποτελεί ένας έγκυρος whitespace χαρακτήρας οπότε ο προβληματισμός παρέμενε έντονος για το τελικό διάγραμμα.

Διαβάζοντας καλύτερα το αλφάβητο της Uni-C και τις βασικές κατηγορίες αναγνωριστικών που χρησιμοποιούνται στην αναγνώριση προτύπων στην Uni-C, εμπεδώσαμε ότι παρόλο που ο χαρακτήρας αλλαγής γραμμής (\n) και το τέλος του αρχείου (EOF) αναγνωρίζονται ως whitespaces χαρακτήρες, επιστρέφονται ως tokens με δικά τους αναγνωριστικά ονόματα (***newline*** για την αλλαγή γραμμής και ***eof*** για το τέλος του αρχείου). Επομένως, το πρόβλημα αντιμετωπίστηκε αφήνοντας τον χαρακτήρα αλλαγή γραμμής για τις μεταβάσεις στις καταστάσεις εξόδου GOOD, BAD όπως κάναμε και στα διαγράμματα των προαναφερόμενων λεκτικών μονάδων της γλώσσας Uni-C. Το τελικό διάγραμμα βρίσκεται εδώ.

6.8 Ρητή αναφορά ελλείψεων

Η αναφορά περιλαμβάνει όλα τα ζητούμενα που απαιτούνται:

- Πρότυπο αναγνώρισης
- Κανονική έκφραση με περιγραφή
- Αυτόματο πεπερασμένης κατάστασης με περιγραφή
- Πίνακας μεταβάσεων με πίνακα καταστάσεων
- Κωδικοποίηση μέσω FSM με σχολιασμό κώδικα
- Περιπτώσεις ελέγχου με εξαντλητικές δοκιμαστικές εκτελέσεις και σχολιασμό αποτελεσμάτων
- Προβλήματα που αντιμετωπίσαμε και τρόποι αντιμετώπισης

Ο κώδικας εκτελέστηκε σε περιβάλλον Windows με εγκατάσταση Windows Subsystem for Linux (WSL). Το τερματικό που έτρεξε η ομάδα τον κώδικα ήταν σε Ubuntu 22.04 LTS, συνεπώς οι εντολές που χρησιμοποιήσαμε για την εκτέλεση του κώδικα είναι οι εξής:

```
gcc -o fsm fsm.c

./fsm 6_white_spaces.fsm

./fsm -list 6_white_spaces.fsm

./fsm -trace 6_white_spaces.fsm
```

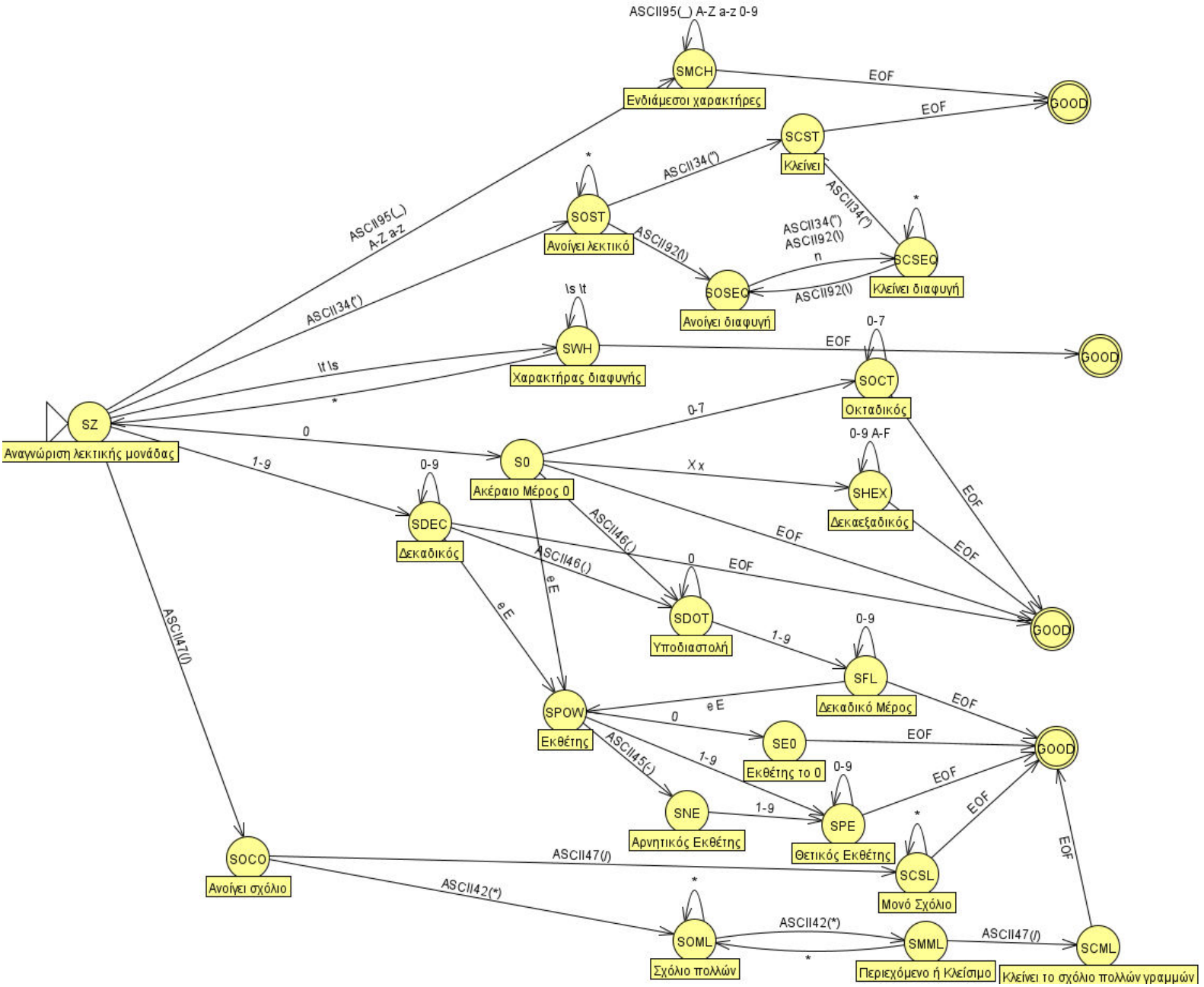
ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

Ενιαίο

Την προσομοίωση την ανέλαβαν οι φοιτητές όλης της ομάδας

Αθανασίου Βασίλειος Ευάγγελος (ΠΑΔΑ-19390005) – Θεοχάρης Γεώργιος (ΠΑΔΑ-19390283) – Τάτσης Παντελής (ΠΑΔΑ-20390226) – Ηλιού Ιωάννης (ΠΑΔΑ-19390066) – Δομινάρης Βασίλειος (ΠΑΔΑ-21390055)

Ενιαίο αυτόματο πεπερασμένης κατάστασης ή ΔΜ



Εικόνα 1. Ενιαίο πεπερασμένο αυτόματο των λεκτικών μονάδων της γλώσσας Uni-C

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

Το ενιαίο αυτόματο αναγνώρισης των λεκτικών μονάδων της γλώσσας Uni-C είναι σχεδιασμένο να αναγνωρίζει και να διαχωρίζει διάφορα είδη λεκτικών, αριθμών και σχολίων από ένα κείμενο. Εδώ είναι πώς λειτουργεί:

Αρχική Κατάσταση (START = SZ):

Αναγνωρίζει λατινικούς χαρακτήρες, κάτω παύλα και ψηφία.

Αναγνωρίζει διπλά εισαγωγικά ως ένδειξη αρχής λεκτικού.

Αναγνωρίζει αριθμούς που ξεκινούν με 0 ως οκταδικούς, δεκαεξαδικούς ή με υποδιαστολή.

Αναγνωρίζει σχόλια.

Αναγνωρίσματα (1_Αναγνωρίσματα - SMCH):

Συνεχίζει να αναγνωρίζει λατινικούς χαρακτήρες, κάτω παύλες, και ψηφία.

Μόλις συναντήσει διπλά εισαγωγικά, μπαίνει σε κατάσταση λεκτικού (SOST).

Λεκτικά (2_Λεκτικά - SOST):

Αναγνωρίζει τα λεκτικά μέχρι να συναντήσει δεύτερο διπλό εισαγωγικό ή χαρακτήρα διαφυγής.

Όταν συναντήσει χαρακτήρα διαφυγής, μπαίνει σε κατάσταση για ανάγνωση χαρακτήρων διαφυγής (SOSEQ).

Ακέραιοι & Αριθμοί (3_Ακέραιοι & 4_Αριθμοί κινούμενης υποδιαστολής):

Αναγνωρίζει ακέραιους αριθμούς, δεκαεξαδικούς, οκταδικούς, και αριθμούς κινούμενης υποδιαστολής.

Μπορεί να αναγνωρίσει αρνητικούς αριθμούς, δυνάμεις και εκθέτη σε επιστημονική μορφή.

Σχόλια (5_Σχόλια - SOCO):

Αναγνωρίζει τα σχόλια, είτε μίας γραμμής είτε πολλαπλών γραμμών.

White Spaces (6_White_spaces - SWH):

Αναγνωρίζει τους χαρακτήρες λευκού χώρου (κενό, tab, νέα γραμμή).

Κάθε κατάσταση έχει προκαθορισμένη συμπεριφορά για κάθε χαρακτήρα που διαβάζει. Ανάλογα με τον χαρακτήρα που διαβάζει και την τρέχουσα κατάσταση, το αυτόματο αποφασίζει που να μεταβεί στη συνέχεια. Στην τελική κατάσταση (GOOD) ολοκληρώνεται η ανάγνωση επιτυχώς.

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

Ενιαίος πίνακας μεταβάσεων

Ο ενιαίος πίνακας μεταβάσεων βρίσκεται σε ξεχωριστό φύλλο στο αρχείο **Final_Array.xlsx** για λόγους αναγνωσιμότητας.

Ο ενιαίος πίνακας καταστάσεων βρίσκεται σε ξεχωριστό φύλλο στο αρχείο **Final_Array.xlsx** για λόγους αναγνωσιμότητας.

Κωδικοποίηση μέσω FSM

Η κωδικοποίηση μέσω FSM του αυτομάτου αναγνώρισης των λεκτικών μονάδων της γλώσσας Uni-C βρίσκεται στο πηγαίο αρχείο **Final.fsm**

```
START = SZ


// ================== 0_Αρχική κατάσταση ==================


SZ:                    // --- [State Z] Κατάσταση αναγνώρισης λεκτικής μονάδας ---
    A-Z -> SMCH              // 1_Αναγνωρίσματα
    _   -> SMCH              // 1_Αναγνωρίσματα
    a-z -> SMCH              // 1_Αναγνωρίσματα


    "   -> SOST              // 2_Λεκτικά


    0   -> S0                // 3_Ακέραιοι ή 4_Αριθμοί κινούμενης υποδιαστολής
    1-9 -> SDEC              // 3_Ακέραιοι ή 4_Αριθμοί κινούμενης υποδιαστολής


    /   -> SOCO              // 5_Σχόλια


    \t  -> SWH               // 6_White_spaces χαρακτήρες
    \s  -> SWH               // 6_White_spaces χαρακτήρες


    EOF -> GOOD              // Τέλος αρχείου


    *   -> BAD               // Άκυρος χαρακτήρας
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

```
// ================== 1_Αναγνωρίσματα ==================


SMCH:                   // --- [State Mid Characters] Κατάσταση για τους ενδιάμεσους
                                                        χαρακτήρες ---

    A-Z a-z -> SMCH         // Πεζά ή κεφαλαία λατινικά γράμματα

    _       -> SMCH         // Κάτω παύλα

    0-9     -> SMCH         // Ψηφία

    \n      -> SZ           // Το αναγνώρισμα είναι έγκυρο και συνεχίζεται η ανάγνωση

    EOF     -> GOOD         // Το αναγνώρισμα είναι έγκυρο και τερματίζεται η ανάγνωση

    *       -> BAD          // Τα αναγνώρισμα είναι μη έγκυρο


// ================== 2_Λεκτικά ==================


SOST:                   // --- [State Open String] Κατάσταση για το περιεχόμενο του
                                                    λεκτικού ---

    "   -> SCST             // Το λεκτικό κλείνει με διπλό εισαγωγικό

    \\  -> SOSEQ            // Ακολουθεί σειρά διαφυγής

    *   -> SOST             // Ακολουθεί οποιοσδήποτε άλλος χαρακτήρας


SOSEQ:                  // --- [State Open Sequence] Κατάσταση για χαρακτήρες με σειρά
                                                       διαφυγής ---

    \\  -> SCSEQ           // Ακολουθεί ο χαρακτήρας slash (\)

    "   -> SCSEQ           // Ακολουθεί διπλό εισαγωγικό

    n   -> SCSEQ           // Ακολουθεί το n για την αλλαγή γραμμής

    *   -> BAD             // Το λεκτικό είναι μη έγκυρο


SCSEQ:                  // --- [State Close Sequence] Κατάσταση για κλείσιμο λεκτικού ή
                                                       σειρά διαφυγής ---

    "   -> SCST            // Το λεκτικό κλείνει με διπλό εισαγωγικό

    \\  -> SOSEQ           // Επιστροφή στην κατάσταση για χαρακτήρες με σειρά διαφυγής

    *   -> SCSEQ           // Ακολουθεί οποιοσδήποτε άλλος χαρακτήρας
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

```
SCST:                   // --- [State Close String] Κατάσταση για το το κλείσιμο του
                                                      λεκτικού ---

    \n  -> SZ              // Το λεκτικό είναι έγκυρο και συνεχίζεται η ανάγνωση

    EOF -> GOOD            // Το λεκτικό είναι έγκυρο και τερματίζεται η ανάγνωση

    *   -> BAD             // Το λεκτικό είναι άκυρο


// ================== 3_Ακέραιοι & 4_Αριθμοί κινούμενης υποδιαστολής ==================


S0:                     // --- [State 0] Κατάσταση για το 0 ως ακέραιο μέρος ---

    0-7 -> SOCT             // Οκταδικός ακέραιος

    x X -> SHEX             // Δεκαεξαδικός ακέραιος

    .   -> SDOT             // Το ακέραιο μέρος είναι το 0 και ακολουθεί υποδιαστολή για
                               το δεκαδικό μέρος

    E e -> SPOW             // Η βάση της δύναμης είναι το 0 και ακολουθεί η
                               επιστημονική μορφή του εκθέτη

    \n  -> SZ              // Ο ακέραιος 0 είναι έγκυρος και συνεχίζεται η ανάγνωση

    EOF -> GOOD            // Ο ακέραιος 0 είναι έγκυρος και τερματίζεται η ανάγνωση

    *   -> BAD             // Ο ακέραιος είναι μη έγκυρος


SDEC:                   // --- [State Decimal] Κατάσταση θετικού δεκαδικού ακεραίου ---

    0-9 -> SDEC             // Ακολουθούν κι άλλα ψηφία από 0-9

    .   -> SDOT             // Ακολουθεί υποδιαστολή για το δεκαδικό μέρος

    E e -> SPOW             // Ακολουθεί η επιστημονική μορφή του εκθέτη

    \n  -> SZ              // Ο δεκαδικός ακέραιος είναι έγκυρος και συνεχίζεται η
                               ανάγνωση

    EOF -> GOOD            // Ο δεκαδικός ακέραιος είναι έγκυρος και τερματίζεται η
                               ανάγνωση

    *   -> BAD             // Ο ακέραιος είναι μη έγκυρος


SHEX:                   // --- [State Hexadecimal] Κατάσταση δεκαεξαδικού ακεραίου ---

    0-9 A-F -> SHEX         // Ακολουθούν κι άλλα ψηφία από 0-9 και γράμματα από A-F

    \n      -> SZ           // Ο δεκαεξαδικός ακέραιος είναι έγκυρος και συνεχίζεται η
                               ανάγνωση

    EOF     -> GOOD         // Ο δεκαεξαδικός ακέραιος είναι έγκυρος και τερματίζεται η
                               ανάγνωση
```

```
    *       -> BAD          // Ο ακέραιος είναι μη έγκυρος


SOCT:                   // --- [State Octal] Κατάσταση οκταδικού ακεραίου ---
    0-7 -> SOCT             // Ακολουθούν κι άλλα ψηφία από 0-7
    \n  -> SZ               // Ο οκταδικός ακέραιος είναι έγκυρος και συνεχίζεται η
                               ανάγνωση
    EOF -> GOOD             // Ο οκταδικός ακέραιος είναι έγκυρος και τερματίζεται η
                               ανάγνωση
    *   -> BAD              // Ο ακέραιος είναι μη έγκυρος


// ================= 4_Αριθμοί κινούμενης υποδιαστολής ==================


SDOT:                    // --- [State Dot] Κατάσταση για την υποδιαστολή ---
    0   -> SDOT             // Ακολουθεί το ψηφίο 0
    1-9 -> SFL              // Ακολουθούν ψηφία για το δεκαδικό μέρος από 1-9
    *   -> BAD              // Ο αριθμός είναι μη έγκυρος


SFL:                     // --- [State Float] Κατάσταση για το δεκαδικό μέρος ---
    0-9 -> SFL              // Ακολουθούν κι άλλα ψηφία για το δεκαδικό μέρος από 0-9
    E e -> SPOW             // Ακολουθεί η επιστημονική μορφή του εκθέτη
    \n  -> SZ               // Ο αριθμός είναι έγκυρος και συνεχίζεται η ανάγνωση
    EOF -> GOOD             // Ο αριθμός είναι έγκυρος και τερματίζεται η ανάγνωση
    *   -> BAD              // Ο αριθμός είναι μη έγκυρος


SPOW:                    // --- [State Power] Κατάσταση για τις δυνάμεις ---
    -   -> SNE              // Ακολουθεί αρνητικό πρόσημο για τον εκθέτη
    0   -> SE0              // Ο εκθέτης είναι 0
    1-9 -> SPE              // Ακολουθεί θετικός εκθέτης
    *   -> BAD              // Ο αριθμός είναι μη έγκυρος


SNE:                     // --- [State Negative Exponential] Κατάσταση για αρνητικό
                                                          εκθέτη ---
    1-9 -> SPE              // Ακολουθούν ψηφία από 1-9 για τον αρνητικό εκθέτη
```

```
    *   -> BAD              // Ο αριθμός είναι μη έγκυρος


SPE:                    // --- [State Positive Exponential] Κατάσταση για θετικό εκθέτη
                                                          ---

    0-9 -> SPE              // Ακολουθούν ψηφία από 1-9 για τον θετικό εκθέτη

    \n  -> SZ               // Ο αριθμός είναι έγκυρος και συνεχίζεται η ανάγνωση

    EOF -> GOOD             // Ο αριθμός είναι έγκυρος και τερματίζεται η ανάγνωση

    *   -> BAD              // Ο αριθμός είναι μη έγκυρος


SE0:                    // --- [State Exponential 0] Κατάσταση για το ψηφίο 0 για
                                                     εκθέτης στην δύναμη ---

    \n  -> SZ               // Ο αριθμός είναι έγκυρος και συνεχίζεται η ανάγνωση

    EOF -> GOOD             // Ο αριθμός είναι έγκυρος και τερματίζεται η ανάγνωση

    *   -> BAD              // Ο αριθμός είναι μη έγκυρος


// ================== 5_Σχόλια ==================


SOCO:                   // --- [State Open Comment] Κατάσταση ανοιχτού σχολίου ---

    /  -> SCSL              // Ανοίγει σχόλιο μίας γραμμής

    \* -> SOML              // Ανοίγει σχόλιο πολλών γραμμών

    *  -> BAD               // Το σχόλιο είναι μη έγκυρο


SOML:                   // --- [State Open Multiple Line (Comment)] Κατάσταση σχολίου
                                                                    πολλών γραμμών ---

    \*  -> SMML            // Πρόθεση για κλείσιμο του σχολίου

    *   -> SOML            // Περιεχόμενο σχολίου

    EOF -> BAD             // Το σχόλιο είναι μη έγκυρο


SMML:                   // --- [State Mid Multiple Line (Comment)] Κατάσταση σχολίου
                                                                   πολλών γραμμών ---

    /   -> SCML            // Κλείνει το σχόλιο πολλών γραμμών

    *   -> SOML            // Επιστροφή στην κατάσταση για περιεχόμενο σχολίου

    EOF -> BAD             // Το σχόλιο είναι μη έγκυρο
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

```
SCML:                   // --- [State Close Multiple Line (Comment)] Κατάσταση έγκυρου
                                                                      σχολίου πολλών
                                                                      γραμμών ---

    \n  -> SZ              // Το σχόλιο είναι έγκυρο και συνεχίζεται η ανάγνωση

    EOF -> GOOD            // Το σχόλιο είναι έγκυρο και τερματίζεται η ανάγνωση

    *   -> BAD             // Το σχόλιο είναι μη έγκυρο


SCSL:                   // --- [State Close Single Line (Comment)] Κατάσταση έγκυρου
                                                                    σχολίου μίας γραμμής
                                                                    ---

    \n  -> SZ              // Το σχόλιο είναι έγκυρο και συνεχίζεται η ανάγνωση

    EOF -> GOOD            // Το σχόλιο είναι έγκυρο και τερματίζεται η ανάγνωση

    *   -> SCSL            // Το σχόλιο είναι μη έγκυρο


// ================== 6_White_spaces ==================


SWH:                    // --- [State Whitespace] Κατάσταση για ανάγνωση χαρακτήρα
                                                   διαφυγής ---

    *     -> SZ          // Αναγνωρίζει οποιοδήποτε άλλο χαρακτήρα διαβάζεται,
                             επιστροφή στην αρχική κατάσταση

    \n    -> GOOD        // Τελειώνει με χαρακτήρα διαφυγής και ακολουθεί newline

    \s \t -> SWH         // Ακολουθεί χαρακτήρας διαφυγής, παραμονή στην ίδια κατάσταση


// ================== Κατάσταση εξόδου: ΟΧΙ ==================


BAD:

    * -> BAD


// ================== Κατάσταση εξόδου: ΝΑΙ ==================


GOOD(OK):
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

Περιπτώσεις ελέγχου

#1 Το αυτόματο δέχεται τις ακόλουθες εισόδους, καθώς η πρώτη είσοδος είναι αναγνώρισμα, η δεύτερη string χαρακτήρων, η τρίτη είναι ακέραιος αριθμός, η τέταρτη είναι δεκαδικός αριθμός, η πέμπτη είναι σχόλιο και η τελευταία είσοδος είναι αναγνώρισμα παρόλο που μεσολαβεί whitespace χαρακτήρας (\t).

```
./fsm -trace Final.fsm
x
sz x -> smch
smch \n -> sz
"y"
sz " -> sost
sost y -> sost
sost " -> scst
scst \n -> sz
1
sz 1 -> sdec
sdec \n -> sz
1.1
sz 1 -> sdec
sdec . -> sdot
sdot 1 -> sfl
sfl \n -> sz
// z
sz / -> soco
soco / -> scsl
scsl \s -> scsl
scsl z -> scsl
scsl \n -> sz
        hi
```

```
sz \t -> swh

swh h -> sz

sz i -> smch

smch \n -> sz

sz EOF -> good

YES
```

#2 Το αυτόματο απορρίπτει την ακόλουθη συμβολοσειρά καθώς ένα αναγνώρισμα δεν πρέπει να ξεκινάει με κάποιον αριθμό.

```
./fsm -trace Final.fsm

1_user

sz 1 -> sdec

sdec _ -> bad

bad u -> bad

bad s -> bad

bad e -> bad

bad r -> bad

bad \n -> bad

bad EOF -> bad

NO
```

#3 Το ακόλουθο string γίνεται αποδεκτό από το αυτόματο καθώς οι χαρακτήρες περικλείονται από τους ειδικούς χαρακτήρες (“ ”).

```
./fsm Final.fsm

"user

one

"

YES
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

#4 Κατά το αυτόματο η ακολουθία χαρακτήρων είναι έγκυρη. Η συμβολοσειρά είναι ένα αναγνώρισμα που πληροί τους κανόνες της γλώσσας Uni-C, που ορίζουν ότι ένα αναγνώρισμα πρέπει να ξεκινάει με κάποιο γράμμα [A-Za-z].

```
 ./fsm -trace Final.fsm

E012

sz E -> smch

smch 0 -> smch

smch 1 -> smch

smch 2 -> smch

smch \n -> sz

sz EOF -> good

YES
```

#5 Το αυτόματο αποδέχεται τα σχόλια που ακολουθούν. Αυτό συμβαίνει διότι τα σχόλια ορίζονται με τους χαρακτήρες ‘//’.

```
./fsm Final.fsm

// hi

// mark

// how

// are

// you?

YES
```

#6 Είναι έγκυρο από το αυτόματο. Με τους χαρακτήρες “/*” δηλώνεται η αρχή των σχολίων και με τους χαρακτήρες “*/” το τέλος τους.

```
./fsm Final.fsm

/*


this is a comment
```

```
*/

YES
```

```
#7   Το αυτόματο απορρίπτει την ακόλουθη είσοδο, καθώς στην γλώσσα Uni-C οι προσημασμένοι ακέραιοι αριθμοί δεν ορίζονται.

./fsm Final.fsm

-0

NO
```

```
#8   Ο χαρακτήρας ‘“‘ που βρίσκεται στην αρχή και στο τέλος του λεκτικού καθιστά την συμβολοσειρά ένα έγκυρο string που γίνεται δεκτό από το αυτόματο.

./fsm Final.fsm

"-0"

YES
```

```
#9 Οι χαρακτήρες “//” δηλώνουν το ξεκίνημα του σχολίου. Για το λόγο η παρακάτω συμβολοσειρά γίνεται δεκτή ως σχόλιο από το αυτόματο.

./fsm Final.fsm

//-0

YES
```

```
#10   Η ακόλουθη συμβολοσειρά είναι λάθος, διότι δεν αποτελεί σχόλιο στην γλώσσα Uni-C. Αυτό συμβαίνει διότι το “this is a string” βρίσκεται εκτός των ειδικών χαρακτήρων (“ ”). Η συμβολοσειρά δεν γίνεται αποδεκτή από το αυτόματο.

./fsm Final.fsm

"\"this is a string""

NO
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

#11 Η παρακάτω συμβολοσειρά δεν είναι αποδεκτή από το αυτόματο γιατί η γλώσσα Uni-C αποδέχεται την χρήση του χαρακτήρα backslash (\) ως χαρακτήρα μέσα στο λεκτικό, μόνο με χρήση σειράς διαφυγής (\\).

```
./fsm Final.fsm

"this is a string\0"

NO
```

#12 Οι παρακάτω είσοδοι δεν είναι αποδεκτές ως αναγνωριστικά και απορρίπτονται από το αυτόματο, καθώς ο ειδικός χαρακτήρας "-" δεν μπορεί να χρησιμοποιηθεί για τον ορισμό ενός αναγνωριστικού.

```
./fsm Final.fsm

1-Id

2-Str

3-Int

4-Fl

5-Com

6-Wh

NO
```

#13 Οι παρακάτω είσοδοι ειναι whitespace χαρακτήρες της γλώσσας Uni-C και άρα γίνονται αποδεκτές από το αυτόματο.

```
./fsm -trace Final.fsm

        tab

sz \t -> swh

swh t -> sz

sz a -> smch

smch b -> smch

smch \n -> sz

 space

sz \s -> swh
```

```
swh s -> sz

sz p -> smch

smch a -> smch

smch c -> smch

smch e -> smch

smch \n -> sz

newline

sz n -> smch

smch e -> smch

smch w -> smch

smch l -> smch

smch i -> smch

smch n -> smch

smch e -> smch

smch \n -> sz

sz EOF -> good

YES
```

#14 Οι πρώτες τρεις δοκιμές είναι έγκυρες. Η τελευταία δοκιμή με τους χαρακτήρες "0xAf" είναι λάθος, επειδή ο χαρακτήρας "f" είναι πεζό γράμμα, αν ήταν κεφαλαίο θα ήταν έγκυρο.

```
./fsm -trace Final.fsm

007

sz 0 -> s0

s0 0 -> soct

soct 7 -> soct

soct \n -> sz

0

sz 0 -> s0
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

```
s0 \n -> sz

123

sz 1 -> sdec

sdec 2 -> sdec

sdec 3 -> sdec

sdec \n -> sz

0xAf

sz 0 -> s0

s0 x -> shex

shex A -> shex

shex f -> bad

bad \n -> bad

bad EOF -> bad

NO
```

```
#15  Η πρώτη είσοδος είναι αποδεκτή ως ακέραιος, η δεύτερη είσοδος είναι αποδεκτή ως αριθμός του 8-δικου συστήματος αρίθμησης, η τρίτη είναι αποδεκτή ως ακέραιος αριθμός, η τέταρτη ως αριθμός του  16-δικου συστήματος αρίθμησης. Η πέμπτη είσοδος είναι αποδεκτή ως δεκαδικός αριθμός, η έκτη είσοδος είναι αποδεκτή ως εκθετικός αριθμός, η έβδομη επίσης είναι αποδεκτή ως εκθετικός αριθμός όπως και η τελευταία είσοδος.

./fsm Final.fsm

0

007

123

0xAF01

3.14

1E-1

1.45E0

0e0
```

```
YES
```

#16 Η τρίτη είσοδος δεν είναι αποδεκτή από το αυτόματο, καθώς τα σχόλια στην Uni-C δεν ξεκινάνε με τον χαρακτήρα “#”.

```
./fsm Final.fsm

// C

/* C */

# Python

NO
```

#17 Είναι σωστές οι ακόλουθες είσοδοι. Η πρώτη και η δεύτερη αποτελούν σχόλια ενώ η τρίτη είσοδος είναι string.

```
./fsm Final.fsm

// C

/* C */

"# Python"

YES
```

#18 Η ακόλουθη συμβολοσειρά δεν είναι αποδεκτή από το αυτόματο, διότι τα σχόλια δεν πρέπει γράφονται μεταξύ των ειδικών χαρακτήρων “* *\” αλλα μεταξύ των χαρακτήρων “/* */”.

```
./fsm Final.fsm

\*

*

*\

NO
```

#19 Το αυτόματο απορρίπτει την πρώτη είσοδο, καθώς το -0 δεν είναι αποδεκτό ως εκθέτης. Οι υπόλοιπες είσοδοι είναι σωστές.

```
./fsm Final.fsm
```

```
4e-0

1.5

0

4e0

NO
```

```
#20 Η πρώτη είσοδος είναι αποδεκτή γιατί χρησιμοποιείται κατάλληλα ο χαρακτήρας διαφυγής (\") ώστε να αγνοηθούν κατάλληλα τα (“ ”). Η δεύτερη είσοδος δεν κάνει ορθή χρήση του χαρακτήρα διαφυγής (\"), διότι το backslash (\) δεν βρίσκεται εντός του string. Η είσοδος δεν είναι αποδεκτή από το αυτόματο. Η τρίτη είσοδος δεν είναι αποδεκτή από το αυτόματο καθώς το 6\3=2 δεν βρίσκεται εντός διπλών εισαγωγικών (“).

./fsm Final.fsm

"he said \"this is a string\"\n"

"he said "\"this is a string\"\""

"he said "6\3=2""

NO
```

```
#21 Τα λεκτικά είναι έγκυρα, διότι οι χαρακτήρες “//” δηλώνουν σχόλιο, το tab είναι whitespace χαρακτήρας και τα “/* */” χρησιμοποιούνται επίσης για σχόλια.

./fsm Final.fsm

// comment?     tab

//

/*

*/

YES
```

```
#22 Η δεύτερη συμβολοσειρά είναι αποδεκτή ως εκθετικός αριθμός. Η πρώτη συμβολοσειρά δεν είναι αποδεκτή επειδή μετά το e οι μόνοι αποδεκτοί χαρακτήρες είναι οι πραγματικοί αριθμοί.

./fsm Final.fsm
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

4exp0

4e0

NO

#23 Η τελεία στο τέλος δεν αντιστοιχεί στο πρότυπο του αυτομάτου όσον αφορά τα αναγνωριστικά στη γλώσσα Uni-C. Οι 2 πρώτες είσοδοι είναι σωστές ως αναγνωριστικά και γίνονται αποδεκτές.

./fsm Final.fsm

_us_

er_

_service.

NO

#24 Δεν είναι αποδεκτή η συμβολοσειρά, διότι έχει μονά εισαγωγικά (‘ ’) αντί για διπλά εισαγωγικά (“ ”) οπότε είναι λάθος ορισμός string.

./fsm Final.fsm

"\n\n \'This is a string\'"

NO

#25 Οι παρακάτω είσοδοι είναι αποδεκτές από το αυτόματο καθώς τα strings περικλείονται εντός των διπλών εισαγωγικών (“ ”).

./fsm Final.fsm

"\n\n"

"

"

YES

#26 Η συμβολοσειρά δεν είναι αποδεκτή καθώς ο whitespace χαρακτήρας μετά το “Hello” δεν καθιστά το hello world ως ένα αναγνωριστικό στη γλώσσα Uni-C.

./fsm Final.fsm

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

```
Hello World

NO
```

```
#27  Τα πρώτα 2 λεκτικά είναι μη αποδεκτά από το αυτόματο, καθώς τα σχόλια πρέπει να ξεκινάνε με (//) αντί (/). Η τελευταία είσοδος είναι σωστό σχόλιο.

./fsm Final.fsm

/

/

// yes

NO
```

```
#28  Το πρώτο λεκτικό είναι αποδεκτό ως αριθμός του 8-δικού συστήματος αρίθμησης. Το δεύτερο λεκτικό δεν είναι αποδεκτό ως αριθμός του 8-δικου συστήματος γιατί οι αριθμοί 8 και 9 δεν βρίσκονται εντός του πεδίου αριθμών [0-7].

./fsm Final.fsm

0017

089

NO
```

```
#29  Τα λεκτικά είναι αποδεκτά ως αριθμοί στη γλώσσα Uni-C, το πρώτο είναι αριθμός του 8-δικου συστήματος και το δεύτερο είναι ακέραιος.

./fsm Final.fsm

0017

0

YES
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

#30 Το πρώτο λεκτικό είναι αποδεκτό από το αυτόματο ως 16-δικος αριθμός. Το δεύτερο είναι μη αποδεκτό διότι οι 16-δικοι αριθμοί πρέπει να ξεκινούν με "0x" ή "0X". Το τρίτο λεκτικό είναι αποδεκτό ως αναγνωριστικό.

```
./fsm Final.fsm

0xAF

01EF

A356

NO
```

#31 Τα παρακάτω λεκτικά είναι αποδεκτά από το αυτόματο, καθώς είναι 16-δικοί αριθμοί. Όλα τα λεκτικά ξεκινάνε σωστά με "0x" και περιέχουν αριθμούς εντός του πεδίου [0-F].

```
./fsm Final.fsm

0xAF

0X01EF

0xA356

YES
```

#32 Το παρακάτω λεκτικό είναι σωστό ως string καθώς περικλείεται από διπλά εισαγωγικά (" "), για τον λόγο αυτό είναι αποδεκτό από το αυτόματο.

```
/fsm Final.fsm

"-0x1135

1-0

001

1.56

1,56

"

YES
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

#33 Ο χαρακτήρας "\" δεν είναι αποδεκτός για την παράθεση σχολίων, συνεπώς η δεύτερη είσοδος δεν είναι αποδεκτή από το αυτόματο.

```
./fsm Final.fsm

//////////////////////

\\\\\\\\\\\\\\\\\\\\\\

NO
```

#34 Ο χαρακτήρας "-" δεν είναι αποδεκτός για τον ορισμό αναγνωριστικού στη γλώσσα Uni-C. Για τον λόγο αυτό το πρώτο λεκτικό δεν είναι αποδεκτό από το αυτόματο, ενώ το δεύτερο λεκτικό είναι αποδεκτό.

```
./fsm Final.fsm

uni-c

uni_c

NO
```

#35 Οι whitespace χαρακτήρες είναι αποδεκτοί από το αυτόματο όπως και το αναγνωριστικό "uni_c".

```
./fsm Final.fsm

        tab

        tab

        tab

uni_c

        tab

YES
```

#36 Η χρήση του χαρακτήρα διαφυγής ("\) είναι σωστή στο πρώτο λεκτικό που γίνεται αποδεκτό από το αυτόματο. Το δεύτερο λεκτικό επίσης είναι αποδεκτό γιατί είναι string που περικλείεται από διπλά εισαγωγικά.

```
./fsm Final.fsm

"\\\"\"\n"
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

```
"

\\\\"

YES
```

#37 Είναι λάθος τα παρακάτω λεκτικά διότι αντί για τον χαρακτήρα (/) χρησιμοποιείται το backslash (\). Το backslash δεν χρησιμοποιείται για τον ορισμό σχολίων.

```
./fsm Final.fsm

\\ hey

\* *\

NO
```

#38 Το λεκτικό δεν είναι έγκυρο, καθώς για να ήταν αποδεκτό ως σχόλιο θα έπρεπε στο τέλος αντί για τους ειδικούς χαρακτήρες "/*" να χρησιμοποιούνται οι χαρακτήρες "*/".

```
./fsm Final.fsm

/*

this

is

a

comment

/*

NO
```

#39 Το πρώτο λεκτικό δεν είναι αποδεκτό, καθώς χρησιμοποιείται το backslash (\) για σχολιασμό. Το δεύτερο λεκτικό είναι γραμμένο εντός εισαγωγικών και άρα είναι μη αποδεκτό, επειδή τα εισαγωγικά δεν χρησιμοποιούνται στον ορισμό κάποιου λεκτικού στη γλώσσα Uni-C.

```
./fsm Final.fsm

\*._.

123456910
```

```
*\

(.)

NO
```

```
#40  Τα παρακάτω λεκτικά είναι αποδεκτά από το αυτόματο, εκτός του πέμπτου λεκτικού διότι το ερωτηματικό (?) δεν χρησιμοποιείται για τον ορισμό αναγνωριστικών στη γλώσσα Uni-C.

./fsm Final.fsm

OK

12345

3.45

3E-1

SURE?

YES

NO
```

Προβλήματα και τρόποι αντιμετώπισης

Πρόβλημα αντιμετωπίσαμε στο ενιαίο αυτόματο και στην κωδικοποίηση μέσω FSM, καθώς, εξαρχής, είχαμε αποφασίσει να δέχεται μόνο μία έγκυρη λεκτική μονάδα σε μία εκτέλεση κώδικα. Συνεπώς, στην αρχική έκδοση του κώδικα οι ενδεικτικές περιπτώσεις ελέγχου ήταν οι εξής:

```
./fsm Final.fsm

identifier

YES
```

```
./fsm Final.fsm

identifier

34

NO
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

Από το αρχικό διάγραμμα μετάβασης, όταν φτάναμε σε μετάβαση που με είσοδο τον χαρακτήρα newline, μεταβαίναμε στην κατάσταση GOOD, δεν βρίσκαμε τρόπο το αυτόματο να συνεχίζει να αναγνωρίζει κι άλλες λεκτικές μονάδες με μία εκτέλεση κώδικα.

Εν κατακλείδι, προγραμματίσαμε στο FSM αλλά δεν σχεδίασαμε στο αυτόματο για λόγους αναγνωσιμότητας, μία έξτρα μετάβαση για κάθε τέτοια κατάσταση που απείχε μία είσοδο για την μετάβαση στην κατάσταση εξόδου GOOD. Η μετάβαση παίρνει για είσοδο τον χαρακτήρα αλλαγής γραμμής (\n) και επιστρέφει πίσω στην αρχική κατάσταση SZ, για την συνέχεια της ανάγνωσης συμβολοσειρών. Είχαμε εξασφαλίσει ότι η προηγούμενη συμβολοσειρά ήταν έγκυρη καθώς, στην μετάβαση πριν την GOOD το μόνο που έλειπε ήταν ο χαρακτήρας αυτός για την μετάβαση στην GOOD. Ο χαρακτήρας εισόδου για την μετάβαση στην GOOD αντικαταστάθηκε με το EOF (End-of-File).

```
./fsm Final.fsm

identifier

YES
```

```
./fsm Final.fsm

identifier

34

YES
```

Ο προβληματισμός για τους whitespaces χαρακτήρες παρέμενε έντονος μέχρι την ολοκλήρωση του ενιαίου, ωστόσο το πόρισμα αναγράφεται εδώ.

Για λόγους αναγνωσιμότητας αποφασίσαμε τον ενιαίο πίνακα μεταβάσεων να τον γράψουμε σ' ένα excel αρχείο.

Ρητή αναφορά ελλείψεων

Η αναφορά περιλαμβάνει όλα τα ζητούμενα που απαιτούνται:

- Αυτόματο πεπερασμένης κατάστασης με περιγραφή
- Πίνακας μεταβάσεων με πίνακα καταστάσεων
- Κωδικοποίηση μέσω FSM με σχολιασμό κώδικα
- Περιπτώσεις ελέγχου με εξαντλητικές δοκιμαστικές εκτελέσεις και σχολιασμό αποτελεσμάτων
- Προβλήματα που αντιμετωπίσαμε και τρόποι αντιμετώπισης

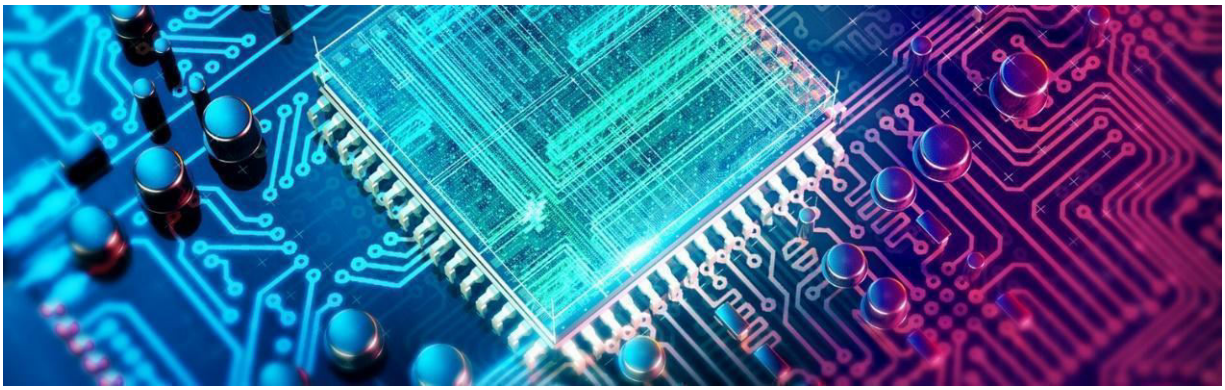
ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

Ο κώδικας εκτελέστηκε σε περιβάλλον Windows με εγκατάσταση Windows Subsystem for Linux (WSL). Το τερματικό που έτρεξε η ομάδα τον κώδικα ήταν σε Ubuntu 22.04 LTS, συνεπώς οι εντολές που χρησιμοποιήσαμε για την εκτέλεση του κώδικα είναι οι εξής:

```
gcc -o fsm fsm.c

./fsm Final.fsm

./fsm -list Final.fsm

./fsm -trace Final.fsm
```

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

Σας ευχαριστούμε για την προσοχή σας.



ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ