Blue text on a black background

Description automatically generated

***Σχολή Μηχανικών***

***Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών***

***Εργαστήριο «Μεταγλωττιστές»***

***Τμήμα Β2 – Μέρος Α***

***Ομάδα 2***

*ΚΟΝΤΟΥΛΗΣ ΔΗΜΗΤΡΙΟΣ 21390095*

*ΑΓΓΕΛΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ ΜΕΝΤΖΕΛΟΣ 21390132*

*ΕΥΦΡΟΣΥΝΗ ΒΑΡΣΟΥ 21390021*

*ΓΚΙΟΖΙ ΕΝΤΕΡΙΣΑ* *21390041*

*ΑΛΕΞΟΠΟΥΛΟΣ ΛΕΩΝΙΔΑΣ 21390006*

Περιεχόμενα

[1. Εισαγωγή 3](#_Toc165301076)

[2. Τεκμηρίωση 3](#_Toc165301077)

[2.1 Κανονικές Εκφράσεις 3](#_Toc165301078)

[2.2 Κώδικας FSM 5](#_Toc165301079)

[2.3 Διαγράμματα Πεπερασμένων Καταστάσεων 8](#_Toc165301080)

[2.4 Πίνακες Μετάβασής 9](#_Toc165301081)

[3. Ενιαίο Πεπερασμένο Αυτόματο 10](#_Toc165301082)

[3.1 Κώδικας FSM 10](#_Toc165301083)

[3.2 Διάγραμμα Πεπερασμένων Καταστάσεων 10](#_Toc165301084)

[3.3 Πίνακας Μετάβασης 10](#_Toc165301085)

[4. Περιπτώσεις Ελέγχου 11](#_Toc165301086)

[5. Ανάλυση αρμοδιοτήτων 11](#_Toc165301087)

# 1. Εισαγωγή

Το παρών έγγραφό αποτελεί την κωδικοποίησή και ανάλυση των λεκτικών μονάδων και αναγνωριστικών της γλώσσας Uni-C σε πεπερασμένα αυτόματα. Ειδικότερα ως λεκτικές μονάδες, εννοούμε τις πηγαίες λέξεις που αναγνωρίζονται από ένα πρότυπο αναγνώρισης μέσω ενός λεκτικού αναλυτή. Οι λεκτικές μονάδες χρειάζεται να αναγνωρίζονται για την έπειτα χρήση τους από στον συντακτικό αναλυτή ενώ τα αναγνωριστικά αναγνωρίζονται από τον λεκτικό αναλυτή αλλά παραλείπονται από τον συντακτικό.

Στην γλώσσα Uni-C, ως λεκτικές μονάδες έχουμε τα αναγνωριστικά (identifiers), τις λέξεις κλειδιά (keywords), τα κυριολεκτικά (λεκτικά και αριθμητικά) και τους τελεστές (operators). Επιπλέον, ως αναγνωριστικά έχουμε τους διαχωριστές χαρακτήρων (ακολουθίες κενών και tab) και τα σχόλια που, όπως αναφέραμε, δεν αποτελούν λεκτικές μονάδες αλλά αναγνωρίζονται και αυτά από πρότυπα και επομένως πρέπει ο λεκτικός αναλυτής να τα αναγνωρίζει.

Επομένως, στην τεκμηρίωση της εργασία μας αναλύουμε αρχικά σε κανονικές εκφράσεις, κωδικοποιούμε και σχεδιάζουμε τα πεπερασμένα αυτόματα και τους πίνακες μεταβάσεις τους των παραπάνω λεκτικών μονάδων (εκτός των keywords) και των σχολίων. Τους διαχωριστές χαρακτήρων τους ενσωματώνουμε στο ενιαίο πεπερασμένο αυτόματο που αναγνωρίζει όλες τις λεκτικές μονάδες και τα σχόλια.

Ειδικότερα, το κεφάλαιο της τεκμηρίωσης αναλύει τα επιμέρους (πέντε) πεπερασμένα αυτόματα παραθέτοντας όλα τους κώδικες, σχήματα και πίνακες μετάβασης. Το κεφάλαιο με το ενιαίο πεπερασμένο αυτόματο, ενσωματώνει όλα τα επιμέρους αυτόματα σε ένα ενιαίο παραθέτοντας αντίστοιχα την κωδικοποίηση του, το σχήμα του και τον γενικό πίνακα μετάβασης του. Στο κεφάλαιο με τις περιπτώσεις ελέγχου παραθέτονται ορισμένοι έλεγχοι λειτουργίας των επιμέρους αυτόματων και του ενιαίου αυτόματου σε εικόνες, επιβεβαιώνοντας την σωστή τους λειτουργία. Στο τελευταίο κεφάλαιο αναλύονται η αρμοδιότητες κάθε μέλους της ομάδας μας στην υλοποίηση της εργασίας.

# 2. Τεκμηρίωση

Ακολουθεί η ανάλυση και κωδικοποίηση κατά σειρά των αναγνωριστικών (identifiers), των λεκτικών κυριολεκτικών (string literals), των αριθμητικών κυριολεκτικών (numerical literals) και των σχολίων (comments).

Αρχικά, για το καθένα παραθέτουμε την κανονική έκφραση που το περιγράφει σύμφωνα με το πρότυπο αναγνώρισης που αναγράφεται στο έγγραφο ανάλυσης της γλώσσας Uni-C. Στη συνέχεια, σύμφωνα με την κάθε κανονική έκφραση την κωδικοποιούμε σε πεπερασμένο αυτόματο, με χρήση του εργαλείου fsm, που αναγνωρίζει την λεκτική μονάδα ή αναγνωριστικό καταλήγοντας δηλαδή σε μια κατάσταση αποδοχής ή όχι. Σχεδιάζουμε το σχήμα σύμφωνα με τον κώδικα του πεπερασμένου αυτόματου (Microsoft Visio) και τέλος σύμφωνα με το σχήμα κατασκευάζουμε τον πίνακα μετάβασης που περιέχει πληροφορίες σχετικά με τις μεταβάσεις που μπορεί να πραγματοποιήσει το αυτόματο από μία κατάσταση σε μία άλλη.

## 2.1 Κανονικές Εκφράσεις

Ακολουθεί η περιγραφή και διατύπωση της κάθε κανονικής έκφρασης των λεκτικών μονάδων και των σχολίων.

Αναγνωριστικά/Identifiers

[a-zA-Z\_]+[a-zA-Z\_0-9]\*

Το πρώτο σύνολο μέσα στις αγκύλες ταιριάζει κάθε λατινικό χαρακτήρα (μικρό ή κεφάλαιο) και την \_ (κάτω παύλα) και το + περιγράφει ότι πρέπει να εμφανίζεται πάνω 1 ή περισσότερες φορές. Έτσι, υποχρεώνουμε ο πρώτος χαρακτήρας να είναι λατινικό γράμμα ή κάτω παύλα (πρέπει να υπάρξει μια φόρα τουλάχιστον). Το δεύτερο σύνολο στις δεύτερες αγκύλες ταιριάζει το προηγούμενο σύνολο που αναφέραμε με προσθήκη των αριθμητικών ψηφίων (0-9) και το \* περιγράφει ότι πρέπει να εμφανίζεται 0 ή περισσότερες φορές. Επομένως, καταφέραμε να μην μπορέσουν τα αριθμητικά ψηφία να εμφανιστούν ως πρώτοι χαρακτήρες σε ένα αναγνωριστικό, γιατί αναγκαστικά πρέπει να γραφτεί ένας χαρακτήρας ή μια κάτω παύλα για να γραφτεί μετά ένας αριθμός.

Λεκτικά Κυριολεκτικά/String Literals

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

Τα Strings πρέπει να αρχίζουν και να τελειώνουν με “ (quotation marks) οπότε για αυτό τα βάζουμε στην αρχή και στο τέλος του regex. To πρώτο σύνολο (αγκύλες) μέσα στην παρένθεση περιγράφει όλα τα σύμβολα εκτός (^) από το “ (quotation mark) και τo \ (backslash) 0 ή περισσότερες φορές (\*). Έτσι, μπορούμε να έχουμε οποιονδήποτε χαρακτήρα, αριθμό ή σύμβολο μέσα στο string ή και κανένα διατυπώνοντας δηλαδή το κενό string. Το δεύτερο σύνολο περιγράφει την δυνατότητα να έχουμε escape sequences μέσα στο string, δηλαδή υποχρεώνοντας το πρώτο \ και μετά μπορεί να ακολουθήσει μόνο \, n ή “ και μετά πάλι μπορεί να ακολουθήσει οποιοσδήποτε χαρακτήρας, αριθμός ή σύμβολο όπως προηγουμένως. Επομένως, καταφέραμε να έχουμε empty strings, escape sequences και οποιονδήποτε χαρακτήρα μέσα σε ένα λεκτικό κυριολεκτικό.

Αριθμητικά Κυριολεκτικά/Numerical Literals

(0[xX][\dA-F]+)|(0[0-7]+)|(([1-9]\d\*|0)(\.\d+)?([eE][-]?([1-9][0-9]\*|0))?)

Τελεστές/Operators

([\+\-\\*\/\=\!\<\>]=)|[\\*\/\=\!\<\>%]|[&\+\-]{1,2}|(\|\|)

Η πρώτη περίπτωση μέσα σε παρένθεση εξασφαλίζει ότι το παραπάνω σύνολο από τελεστές πρέπει να ακολουθούνται από το ίσων (=). Η δεύτερη περίπτωση (OR |) απλώς εξασφαλίζει να τυπώνονται οι τελεστές που δεν αποτελούν με έναν άλλον τελεστή κάποιο ζευγάρι. Η τρίτη περίπτωση εξασφαλίζει τους τελεστές που μπορούν να εμφανιστούν μία φορά μόνοι τους αλλά και αποτελούν ζευγάρι με τον εαυτό τους. Η τελευταία περίπτωση εξασφαλίζει να ταιριάξει ο || (LOGICAL OR) τελεστής. Με τις παραπάνω περιπτώσεις, εξασφαλίσαμε το ταίριασμα όλων των τελεστών που αναγράφονται στο πρότυπο.

Σχόλια/Comments

\/\\*(.|\n)\*\\*\/|\/\/.\*

Τα σχόλια μπορούν να διατυπωθούν με δύο τρόπους: με δύο slash (//) και έπειτα οποιονδήποτε άλλο χαρακτήρα (εκτός whitespace characters) ή αρχίζοντας και τελειώνοντας αντίστοιχα με /\* \*/ και εμπεριέχοντας οποιονδήποτε χαρακτήρα αναμεσά τους (συμπεριλαμβανομένων των whitespace characters). H πρώτη OR εξασφαλίζει τον δεύτερο τρόπο που περιγράψαμε με . (τελεία) να εννοούμε οποιονδήποτε χαρακτήρα εκτός του newline (οπότε για αυτό βάλαμε σε OR το \n γιατί τα σχόλια μπορούν υπάρχουν και σε άλλη γραμμή). Η δεύτερη OR εξασφαλίζει τον πρώτο τρόπο που απλώς ξεκινάει με // και έχει έπειτα οποιονδήποτε χαρακτήρα (.).

## 2.2 Κώδικας FSM

Ακολουθεί η κωδικοποίηση των επιμέρους πεπερασμένων αυτομάτων. Ο σχολιασμός σε τι αποσκοπεί κάθε κατάσταση υπάρχουν σαν σχόλια μέσα στα .fsm αρχεία. Γενικά, το \n το χρησιμοποιούμε για να τερματίσει το fsm πρόγραμμα και δεν αντικραπτοτίζεται στις κανονικές εκφράσεις ή στα σχήματα/πίνακες μετάβασης των επιμέρους πεπερασμένων αυτόματων. Οπότε για αυτό στους περισσότερους κώδικες στην κατάσταση GOOD δεχόμαστε το \n και ξαναπάμε στην κατάσταση αποδοχής.

Αναγνωριστικά/Identifiers

START=SZ

// the SZ state is the state that checks for the start of the identifier

SZ: A-Z a-z \_ -> S0 // if it starts with any letter, uppercase or lowercase, or \_, go to S0

\* -> BAD // anything else like digits (0-9) for example is not accepted

S0: a-z A-Z \_ 0-9 -> S0 // if the start of the name is correct, check the rest of the identifer

\n -> GOOD // if its just a single letter, or just one underscore (\_), its accetable

\* -> BAD // anything else is bad

GOOD(OK):

Λεκτικά Κυριολεκτικά/String Literals

START=SZ

SZ: " -> S0 // string is opened

\* -> BAD // if its anything else it's not a string

// when we are in this state (S0) we are inside a string

S0: \* -> S1 // anything inside the string is accepted, and we go to S1

\\ -> S2 // if its a slash go to S2 to check for \, " or n

" -> GOOD // string is closed, go to GOOD (Accepting State)

S1: \\ -> S2 // if there is a \ and then go to S2 to check for another one (\\ translates to \ in a string)

" -> GOOD // in this case the string is closed, so we go to GOOD (Accepting State)

\* -> S1 // if you receive anything (inside the string) keep looping to S1

S2: \\ " n -> S1 // check for a second \ (\), if the string closes ("), or if there is a new line character (\n)

\* -> BAD // anything else is not accepted

GOOD(OK): \n -> GOOD

Λεκτικά Κυριολεκτικά/String Literals

Αριθμητικά Κυριολεκτικά/String Literals

START=S0

S0: 0 -> S1

1-9 -> S2

\* -> BAD

S1: . -> S3

e E -> S4

X x -> S6

0-7 -> S7

\n -> GOOD

\* -> BAD

// Integer

S2: 0-9 -> S2

. -> S3

e E -> S4

\n -> GOOD

\* -> BAD

// Float

S3: 0-9 -> S10

\* -> BAD

S10: 0-9 -> S10

e E -> S4

\n -> GOOD

\* -> BAD

// Exponential

S4: 1-9 -> S8

- -> S5

0 -> GOOD

\* -> BAD

S5: 1-9 -> S8

\* -> BAD

// Hexadecimal

S6: A-F 0-9 -> S9

\* -> BAD

S9: A-F 0-9 -> S9

\n -> GOOD

\* -> BAD

// Octal

S7: 0-7 -> S7

\n -> GOOD

\* -> BAD

S8: 0-9 -> S8

\n -> GOOD

\* -> BAD

GOOD(OK) : \n -> GOOD

Τελεστές/Operators

START=SZ

SZ: \\* / \= ! < > -> S1 // These operators can be written alone, or followed by an equal sign (=)

| -> S2 // if | is written then it's about an OR logical operation, so we go to S2

+ -> S3 // the plus sign, depending on the following sign, can mean 3 different opeartions.

- -> S4 // the minus sign can also mean 3 different operations like the plus sign.

& -> S5 // if & is written then it's about a logical AND operation, so we go to S5

% -> GOOD // this is the modulus operation which is only one percentage sign (%)

\* -> BAD // anything else is not accepted because it's not an operator

// state regarding \* / = ! < >

S1: \n \= -> GOOD // if its just a signle operator (\* / ...), or followed by an equal sign (=)

\* -> BAD // anything else is not accepted

// state regarding | sign

S2: | -> GOOD // check if there is another |. If there is then it's an OR operation (||)

\* -> BAD // if it's only one | and anything else, then it's not valid

// state regarding plus sign(+)

S3: \n \+ \= -> GOOD // the operations can be addition (+), increment (++) or +=

\* -> BAD // if it's anything else after the plus sign then it's not accepted

// state regarding minus sign(-)

S4: \n \- \= -> GOOD // the operations can be subtraction (-), increment (--) or -=

\* -> BAD // it it's just one minus and anything else after then it's not accepted.

//state regarding ampersand (&)

S5: \n & -> GOOD // if there is another & sign then it's a logical AND operation (&&)

\* -> BAD // if it's only one & sign followed by anything else it's not valid

GOOD(OK): \n -> GOOD

Σχόλια/Comments

START = SZ

SZ: / -> S0 // comment is started

\* -> BAD // if it's anything else is not a comment

S0: / -> S1 // if there is another bracket, then it's a one line comment

\\* -> S2 // if there is an asterisk followed, then it's about a multi-line comment

\* -> BAD // if it's anything else then it's not valid

S1: \n -> GOOD // in this case we have a single line comment (//)

\* -> S1 // here we are inside the comment, so pretty much everything is accepted

S2: \\* -> S3 // in this case we have /\* which is a multi-line comment and we check if there is an asterisk

\* -> S2 // everything else is not accepted

S3: / -> GOOD // if there is a slash (/) after the asterisk (from S2), the multi-line comment closes

\* -> BAD // anything else does not regard the comment

GOOD(OK): \n ->GOOD

## 2.3 Διαγράμματα Πεπερασμένων Καταστάσεων

Ακολουθούν τα επιμέρους σχήματα των πεπερασμένων αυτομάτων. H κατάσταση SB είναι η BAD κατάσταση που περιγράφουμε στον κώδικα και η SG είναι η GOOD κατάσταση. Με την μετάβαση (\*) εννοούμε τον οποιαδήποτε άλλο χαρακτήρα που δεν έχει αναφερθεί μετάβαση. Επιπλέον, στον κώδικα τα έχουμε όλα να καταλήγουνε στην κατάσταση αποδοχής GOOD γιατί όπως είπαμε χρειαζόμαστε σαν τερματισμό το \n. Στα σχήματα όμως, ορίζουμε και καταστάσεις αποδοχής αυτές που στον κώδικα με \n πάνε στην κατάσταση GOOD. Αυτό το κάνουμε έτσι ώστε τα σχήματα να φαίνονται πιο περιεκτικά και σύντομα.

*Σημείωση:* όλα τα επιμέρους σχήματα καταλήγουν στην κατάσταση SB (BAD) εκτός από το σχήμα για τα αριθμητικά κυριολεκτικά. Αυτό έγινε λόγω μεγάλης πολυπλοκότητας του σχήματος αν όλες οι καταστάσεις κατέληγαν εκεί. Εννοείται ότι όποια άλλη μετάβαση υπάρξει καταλήγει στην κατάσταση SB (BAD).

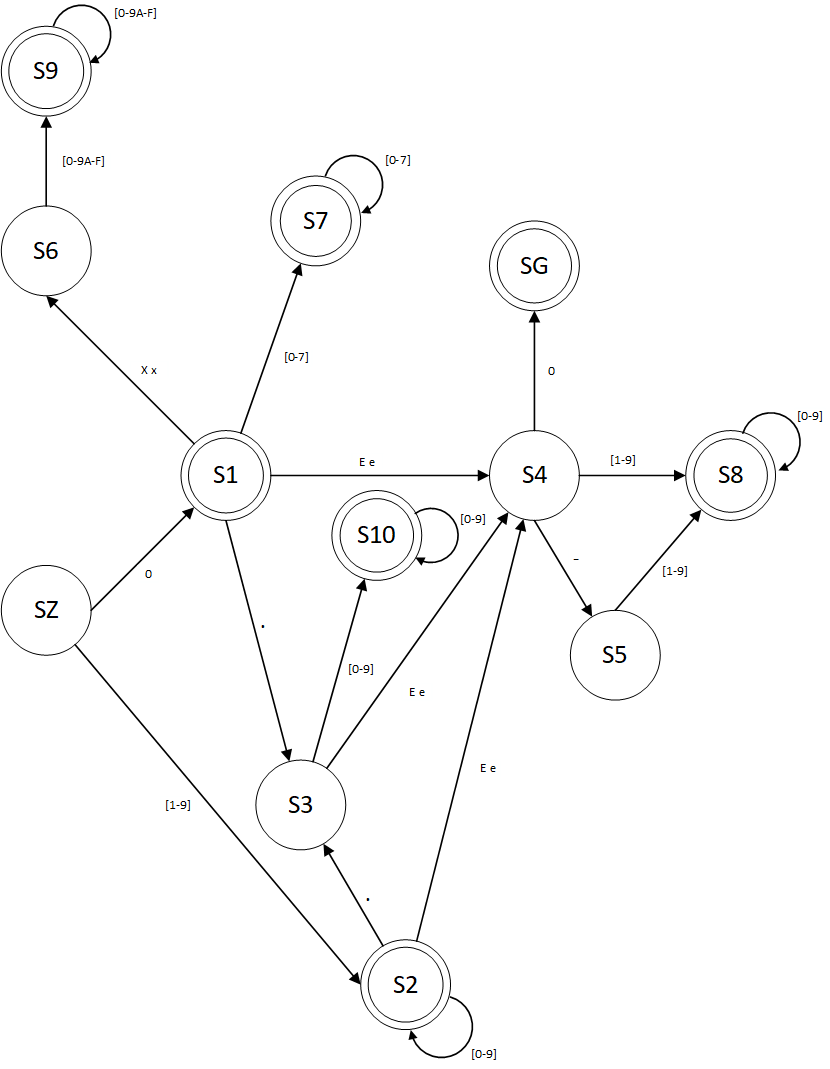
Αναγνωριστικά/Identifiers



Λεκτικά Κυριολεκτικά/String Literals



Αριθμητικά Κυριολεκτικά/Numerical Literals



Τελεστές/Operators

## 

Σχόλια/Comments



## 2.4 Πίνακες Μετάβασής

Ακολουθούν οι πίνακες μεταβάσεις για κάθε επιμέρους πεπερασμένο αυτόματο.

Αναγνωριστικά/Identifiers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A-Z | a-z | \_ | 0-9 | \* / OTHER |
| SZ | S0 | S0 | S0 | BAD | BAD |
| S0 | S0 | S0 | S0 | S0 | BAD |
| SB / BAD | BAD | BAD | BAD | BAD | BAD |

Λεκτικά Κυριολεκτικά/String Literals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | “ | \ | n | \* / OTHER |
| SZ | S0 | BAD | BAD | BAD |
| S0 | S3 | S2 | S1 | S1 |
| S1 | S3 | S2 | S1 | S1 |
| S2 | S1 | S1 | S1 | BAD |
| S3 | BAD | BAD | BAD | BAD |
| SB / BAD | BAD | BAD | BAD | BAD |

Αριθμητικά Κυριολεκτικά/Numerical Literals

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1-9 | X x | 1-7 | E e | . | - | A-F | \* / OTHER |
| SZ | S1 | S2 | BAD | BAD | BAD | BAD | BAD | BAD | BAD |
| S1 | S7 | BAD | S6 | S7 | S4 | S3 | BAD | BAD | BAD |
| S2 | S2 | S2 | BAD | S2 | S4 | S3 | BAD | BAD | BAD |
| S3 | S10 | S10 | BAD | S10 | S4 | BAD | BAD | BAD | BAD |
| S4 | SG | S8 | BAD | S8 | BAD | BAD | S5 | BAD | BAD |
| S5 | BAD | S8 | BAD | S8 | BAD | BAD | BAD | BAD | BAD |
| S6 | S9 | S9 | BAD | S9 | BAD | BAD | BAD | S9 | BAD |
| S7 | S7 | BAD | BAD | S7 | BAD | BAD | BAD | BAD | BAD |
| S8 | S8 | S8 | BAD | S8 | BAD | BAD | BAD | BAD | BAD |
| S9 | S9 | S9 | BAD | S9 | BAD | BAD | BAD | S9 | BAD |
| S10 | S10 | S10 | BAD | S10 | BAD | BAD | BAD | BAD | BAD |

Τελεστές/Operators

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | & | - | + | % | | | \ | \* | / | = | ! | < | > | \* / OTHER |
| SZ | S5 | S4 | S3 | SG | S2 | S1 | S1 | S1 | S1 | S1 | S1 | S1 | BAD |
| S1 | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | SG | BAD | BAD | BAD | BAD |
| S2 | BAD | BAD | BAD | BAD | SG | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD |
| S3 | BAD | BAD | SG | BAD | BAD | BAD | BAD | BAD | SG | BAD | BAD | BAD | BAD |
| S4 | BAD | SG | BAD | BAD | BAD | BAD | BAD | BAD | SG | BAD | BAD | BAD | BAD |
| S5 | SG | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD |
| SG | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD |
| SB / BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD | BAD |

Σχόλια/Comments

|  |  |  |  |
| --- | --- | --- | --- |
|  | / | \* | \\* |
| SZ | S0 | SB | BAD |
| S0 | S1 | SB | S2 |
| S1 | BAD | S1 | BAD |
| S2 | BAD | S2 | S3 |
| S3 | SG | SB | BAD |
| SG | BAD | BAD | BAD |
| SB | BAD | BAD | BAD |

# 3. Ενιαίο Πεπερασμένο Αυτόματο

## 3.1 Κώδικας FSM

## 3.2 Διάγραμμα Πεπερασμένων Καταστάσεων

## 3.3 Πίνακας Μετάβασης

# 4. Περιπτώσεις Ελέγχου

# 5. Ανάλυση αρμοδιοτήτων