Πανεπιστήμιο Θεσσαλίας
Τμήμα Πληροφορικής

Μεταγλωττιστές

Παραδείγματα Ενοτήτων 1-2

Ενότητα 1: Εισαγωγή

Άσκηση 1-1:

Θεωρήστε μια υποθετική γλώσσα προγραμματισμού και την παρακάτω γραμματική (χωρίς συμφραζόμενα):

```
assign → var = expr
var    → ID paren
paren  → ( expr ) | ε
expr   → expr + term | expr – term | term
term   → term * var | CONST / term | var | CONST | ( expr )
```

η οποία περιγράφει τις αναθέσεις τιμών σε μεταβλητές σε αυτήν τη γλώσσα.

Α. Χρησιμοποιώντας κάποιο γενικό σχήμα μετάφρασης οδηγούμενης από τη σύνταξη, δώστε μια παραγωγή και ένα δέντρο συντακτικής ανάλυσης για την παρακάτω συμβολοσειρά:

```
y2(i+1) = 101 / (x-33*j) + 2
```

υποδεικνύοντας παράλληλα τις λεκτικές μονάδες που κάποιος λεκτικός αναλυτής επιστρέφει στο συντακτικό, όταν αυτές ζητηθούν.

Β. Δώστε ένα πιθανό αφηρημένο συντακτικό δέντρο για την ίδια εντολή, καθώς και έναν πιθανό τελικό κώδικα MIPS που να μπορεί να προκύψει από το δέντρο αυτό.

Απάντηση

Α. Όπως έχουμε δει, η μετάφραση κάποιου προγράμματος της αρχικής γλώσσας σε κώδικα τελικής γλώσσας περνάει από διαδοχικές φάσεις. Κάθε φάση όμως δεν είναι απαραίτητο να έχει ολοκληρωθεί, πριν ξεκινήσει η επόμενη, αλλά μπορεί να επικαλύπτεται με αυτήν. Αυτό σημαίνει ότι κάποια φάση μπορεί να διακοπεί προσωρινά μετά την εφαρμογή της σε τμήμα του προγράμματος, επιτρέποντας την επόμενη φάση να εφαρμοστεί στο ίδιο τμήμα, πριν η ίδια φάση προχωρήσει σε επόμενο τμήμα.
Στα σχήματα μετάφρασης οδηγούμενης από τη σύνταξη, κεντρική φάση της μετάφρασης είναι η συντακτική ανάλυση. Ο συντακτικός αναλυτής (ΣΑ) αναλύει το αρχικό πρόγραμμα, καλώντας το λεκτικό αναλυτή (ΛΑ) όποτε χρειάζεται κάποια λεκτική μονάδα, και κατά διαστήματα εκτελεί και σημασιολογικές ρουτίνες για σημασιολογική ανάλυση και παραγωγή ενδιάμεσου κώδικα. Έτσι, οι φάσεις μετάφρασης που καταλήγουν στην παραγωγή ενδιάμεσου κώδικα επικαλύπτονται, με το ΛΑ να διακόπτεται μετά την αναγνώριση μιας λεκτικής μονάδας, και το ΣΑ να διακόπτεται σε επιλεγμένα σημεία των συντακτικών κανόνων που εφαρμόζει. Στα σημεία αυτά γίνεται σημασιολογική ανάλυση και παραγωγή ενδιάμεσου κώδικα για το νέο τμήμα του προγράμματος που αναλύθηκε από τις δύο προηγούμενες φάσεις. Οι δύο επόμενες φάσεις, η βελτιστοποίηση και η παραγωγή τελικού κώδικα, μπορούν να γίνουν είτε σε τμήματα είτε στο σύνολο του ενδιάμεσου κώδικα.
Για απλούστευση, θεωρούμε ότι ο ΣΑ λειτουργεί με βάση τη διαδικασία παραγωγής του προγράμματος εισόδου από επάνω προς τα κάτω, ξεκινώντας με το αρχικό σύμβολο της γραμματικής assign, και επιλέγοντας κανόνες με ιδανικό τρόπο. Όπως θα δούμε σε επόμενο κεφάλαιο,

η συντακτική ανάλυση μπορεί να γίνεται και από κάτω προς τα επάνω, *ελαττώνοντας* το πρόγραμμα εισόδου στο αρχικό σύμβολο της γραμματικής. Στην πραγματικότητα, η δεδομένη γραμματική δεν αντιμετωπίζεται με τον πρώτο τρόπο, θα χρησιμοποιήσουμε όμως μια απλοϊκή παραλλαγή «ιδανικής επιλογής» του τρόπου αυτού, η οποία δίνει καλύτερα τη γενική ιδέα της μετάφρασης, χωρίς να προχωρήσουμε σε λεπτομέρειες.

Για να βρούμε μια παραγωγή για τον δεδομένο κώδικα, θα προχωρήσουμε – χωρίς αυτό να είναι απαραίτητο – αντικαθιστώντας το αριστερότερο κάθε φορά μη τερματικό σύμβολο στους προτασιακούς τύπους που θα συναντάμε. Έτσι, ξεκινώντας με το σύμβολο assign, έχουμε τις εξής κινήσεις:

```
assign ⇒ var = expr ⇒ ID paren = expr
```

που είναι αναγκαστικές, εφ' όσον δεν έχουμε επιλογές στους δύο συντακτικούς κανόνες που χρησιμοποιούμε.
Στο σημείο αυτό, κι έχοντας φτάσει σε τερματικό σύμβολο, ο ΣΑ καλεί το ΛΑ, ώστε να πάρει την επόμενη λεκτική μονάδα και να διαπιστώσει αν αυτή αντιστοιχεί στο σύμβολο αυτό. Αν δεν αντιστοιχεί στο σύμβολο αυτό – και αν βέβαια δεν έχουμε λεκτικό σφάλμα, θα έχουμε συντακτικό σφάλμα. Έτσι, για το συγκεκριμένο πρόγραμμα που έχουμε, ο ΛΑ αναγνωρίζει τη συμβολοσειρά "y2" ως αναγνωριστικό (όνομα μεταβλητής) και επιστρέφει στο ΣΑ το αναμενόμενο σύμβολο ID, επιτρέποντας σε αυτόν να συνεχίσει. Η ίδια η συμβολοσειρά "y2" διατηρείται στον πίνακα συμβόλων (ΠΣ), ώστε οι επόμενες φάσεις της μετάφρασης να γνωρίζουν το πραγματικό όνομα της μεταβλητής. Για τους σκοπούς του ΣΑ, το σύμβολο ID είναι αρκετό, καθώς μόνο αυτό εμφανίζεται στους συντακτικούς κανόνες της γλώσσας, και όχι το πραγματικό όνομα της μεταβλητής. Επειδή όμως η μετάφραση είναι οδηγούμενη από τη σύνταξη, ο ΛΑ επιστρέφει στο ΣΑ *και* τον αντίστοιχο δείκτη στον ΠΣ, ώστε αυτός να χρησιμοποιηθεί από τις επόμενες φάσεις που θα μεσολαβήσουν αργότερα.
Η παραγωγή τώρα μπορεί να συνεχιστεί για το σύμβολο paren, όπου όμως υπάρχουν δύο επιλογές. Ο ΣΑ καλεί το ΛΑ, ώστε με την επόμενη λεκτική μονάδα να μπορέσει να αποφασίσει ποια κίνηση θα κάνει. Έτσι, για την πρώτη επιλογή απαιτείται το σύμβολο '(', ενώ για τη δεύτερη επιλογή απαιτείται το '=', το οποίο είναι αυτό που ακολουθεί στην παραγωγή, αν το paren παράγει την κενή συμβολοσειρά. Ο ΛΑ επιστρέφει τη λεκτική μονάδα που αντιστοιχεί στο σύμβολο '(', κι επομένως η παραγωγή μας γίνεται:

```
assign ⇒+ ID ( expr ) = expr
```

όπου τα ενδιάμεσα βήματα παραγωγής που παραλείψαμε είναι αυτά που περιγράψαμε πιο πάνω.
Αν ο ΛΑ επέστρεφε λεκτική μονάδα που δεν αντιστοιχούσε ούτε στο σύμβολο '(' ούτε στο σύμβολο '=', θα είχαμε συντακτικό σφάλμα.

Για το σύμβολο expr υπάρχουν τρεις επιλογές. Όμως, ο ΣΑ δε μπορεί να αποφασίσει με μία μόνο λεκτική μονάδα στη διάθεσή του, επειδή οι αντίστοιχοι κανόνες εμφανίζουν αναδρομή, και πρέπει να καλέσει το ΛΑ διαδοχικά, μέχρι να μπορεί να επιλέξει κάποιον κανόνα[1]. Εδώ οι διαδοχικές κλήσεις σταματάνε όταν εμφανιστεί το σύμβολο ')', και λόγω της παρουσίας του '+' ο ΣΑ καταλήγει στην παραγωγή:

```
assign ⇒+ ID ( expr + term ) = expr
```

Οι ενδιάμεσες λεκτικές μονάδες που επέστρεψε ο ΛΑ, και οι οποίες αντιστοιχούν στα σύμβολα ID, '+', CONST και ')', αποθηκεύονται για μετέπειτα χρήση. Η συμβολοσειρά "i" εισάγεται στον ΠΣ. Ο αντίστοιχος δείκτης στον ΠΣ, αλλά και η τιμή της σταθεράς που αντιστοιχεί στη

[1] Στην πραγματικότητα αυτός είναι ο λόγος που το μοντέλο ανάλυσης από πάνω προ τα κάτω δε μπορεί να χρησιμοποιηθεί για τη συγκεκριμένη γραμματική, μια που επιθυμούμε να μπορούμε να αποφασίζουμε ντετερμινιστικά την κίνηση του ΣΑ με πεπερασμένο πλήθος – ιδανικά μίας – λεκτικών μονάδων. Πιο σύνθετα μοντέλα συντακτικής ανάλυσης αντιμετωπίζουν επιτυχώς το πρόβλημα της αναδρομής με ντετερμινιστικό τρόπο. Εναλλακτικά, με μη ντετερμινιστική ανάλυση, επιλέγουμε κάποιον κανόνα στην τύχη, και μόλις καταλήξουμε σε συντακτικό σφάλμα, επιστρέφουμε στο ίδιο σημείο και επιλέγουμε άλλον κανόνα. Μια τέτοια ανάλυση όμως αυξάνει την πολυπλοκότητα του ΣΑ εκθετικά με τον αριθμό των διαθέσιμων κινήσεων.

συμβολοσειρά “1” επιστρέφονται επίσης στο ΣΑ και αποθηκεύονται μαζί με τις αντίστοιχες λεκτικές μονάδες.
Στη συνέχεια, κι επειδή μέχρι το σύμβολο ‘+’ δε μεσολαβεί άλλο ‘+’ ή ‘-’, επιλέγουμε τον τρίτο εναλλακτικό κανόνα για το σύμβολο expr, και η παραγωγή γίνεται:

```
assign ⇒+ ID ( term + term ) = expr
```

Τώρα ο ΣΑ πρέπει να επιλέξει κανόνα για το πρώτο από τα δύο σύμβολα term. Με διαθέσιμη επόμενη λεκτική μονάδα αυτή που αντιστοιχεί στο σύμβολο ID, αποκλείονται οι κανόνες που αρχίζουν με τα σύμβολα CONST και ‘(’. Παραμένουν έτσι δύο κανόνες, αλλά για άλλη μια φορά δε μπορούμε να επιλέξουμε με μία μόνο λεκτική μονάδα, λόγω της αναδρομής που έχει το term. Μπορούμε εύκολα να διαπιστώσουμε ότι για να επιλέξουμε τον πρώτο κανόνα απαιτείται η παρουσία του συμβόλου ‘*’ πριν το ‘+’, που όμως δεν υπάρχει. Ο ΣΑ επιλέγει τον τρίτο κανόνα για το σύμβολο term, και στη συνέχεια το μοναδικό κανόνα για το var, οπότε η παραγωγή γίνεται:

```
assign ⇒+ ID ( var + term ) = expr
       ⇒ ID ( ID paren + term ) = expr
```

Το νέο τερματικό σύμβολο ID που εμφανίστηκε στην παραγωγή αναμενόταν, αφού ταυτίζεται με την επόμενη λεκτική μονάδα που ο ΣΑ έχει αποθηκεύσει και χρησιμοποιήσει για τις παραπάνω επιλογές. Ο ΣΑ μπορεί να συνεχίσει με το σύμβολο paren, για το οποίο επιλέγει το δεύτερο κανόνα, εφ’ όσον δεν υπάρχει ‘(’ πριν το ‘+’:

```
assign ⇒+ ID ( ID + term ) = expr
```

Μετά το ‘+’ που επαληθεύει τη μέχρι τώρα ανάλυση, αφού αντιστοιχεί στην επόμενη αποθηκευμένη λεκτική μονάδα, ο ΣΑ πρέπει για άλλη μια φορά να επιλέξει κανόνα για το term. Λόγω απουσίας του συμβόλου ‘*’ πριν το ‘)’ αποκλείεται ο πρώτος κανόνας. Παρόμοια, λόγω απουσίας του συμβόλου ‘/’ πριν το ‘)’ αποκλείεται ο δεύτερος κανόνας, παρ’ όλο που η παρουσία του CONST θα μπορούσε να τον δικαιολογήσει. Ο τρίτος κανόνας απαιτεί την παρουσία του συμβόλου ID, ενώ ο πέμπτος κανόνας απαιτεί την παρουσία του ‘(’ ως επόμενη λεκτική μονάδα, κι επομένως και οι δύο αποκλείονται. Έτσι, ο τέταρτος κανόνας θα δώσει:

```
assign ⇒+ ID ( ID + CONST ) = expr
```

Με το σύμβολο CONST που εμφανίστηκε επαληθεύεται η ανάλυση. Στο σημείο αυτό πρέπει να κληθεί ο σημασιολογικός αναλυτής, για να επαληθεύσει τη σημασιολογική ορθότητα της έκφρασης “i+1” που μόλις αναγνωρίστηκε συντακτικά, για παράδειγμα την ορθότητα της εφαρμογής του τελεστή της πρόσθεσης στη μεταβλητή i και στη σταθερά 1. Επίσης μπορεί να γίνει παραγωγή ενδιάμεσου κώδικα για τον υπολογισμό της τιμής της έκφρασης αυτής.
Η συντακτική ανάλυση επαληθεύεται περαιτέρω με το σύμβολο ‘)’. Και στο σημείο αυτό ο σημασιολογικός αναλυτής είναι απαραίτητος, για να επαληθεύσει την ορθότητα της δεικτοδότησης της μεταβλητής y2. Όπως και προηγουμένως, μπορεί να παραχθεί ενδιάμεσος κώδικας για τον υπολογισμό της διεύθυνσης του στοιχείου της διανυσματικής μεταβλητής y2 που δεικτοδοτείται από την παραπάνω έκφραση.

Τώρα έχουμε φτάσει σε σημείο που απαιτείται ξανά η κλήση του ΛΑ. Αυτός επιστρέφει τη λεκτική μονάδα που αντιστοιχεί στο σύμβολο ‘=’, όπως και αναμενόταν για αποφυγή συντακτικού σφάλματος. Στη συνέχεια, και για επιλογή κανόνα για το expr, καλούμε διαδοχικά το ΛΑ μέχρι το τέλος του αρχικού προγράμματος, ώστε να έχουμε στη διάθεσή μας όλες τις εμφανίσεις των συμβόλων ‘+’ και ‘-’, αν υπάρχουν, κάτι που είναι αναγκαίο για αντιμετώπιση της αναδρομής που έχει το expr. Έτσι, ο ΣΑ αποθηκεύει για μετέπειτα ανάλυση τις λεκτικές μονάδες που αντιστοιχούν στα σύμβολα CONST, ‘/’, ‘(’, ID, ‘-’, CONST, ‘*’, ID, ‘)’, ‘+’ και CONST, αποθηκεύοντας παράλληλα και τους όποιους δείκτες στον ΠΣ ή τις όποιες τιμές σταθερών. Το σύμβολο ‘+’ που συναντάμε μας οδηγεί στην επιλογή του αντίστοιχου κανόνα για το expr:

```
assign ⇒+ ID ( ID + CONST ) = expr + term
```

Επειδή δε μεσολαβεί άλλο σύμβολο ‘+’ ή ‘-’, επιλέγουμε στη συνέχεια τον τελευταίο κανόνα για το expr, και παίρνουμε την παραγωγή:

```
assign ⇒+ ID ( ID + CONST ) = term + term
```

Παραβλέποντας τα σύμβολα μεταξύ των παρενθέσεων ‘(’ και ‘)’ που ο ΣΑ έχει αποθηκεύσει, και τα οποία δε μπορούν να χρησιμοποιηθούν πριν η παραγωγή συναντήσει την αριστερή παρένθεση, η παρουσία των συμβόλων CONST και ‘/’ και η ταυτόχρονη απουσία του ‘*’ πριν το σύμβολο ‘+’ μας οδηγεί στην επιλογή του δεύτερου κανόνα για το πρώτο από τα δύο term, επαληθεύοντας ταυτόχρονα τη συντακτική ορθότητα μέχρι και το σύμβολο ‘/’:

```
assign ⇒+ ID ( ID + CONST ) = CONST / term + term
```

Η απουσία του συμβόλου ‘*’ μεταξύ των ‘)’ και ‘+’ αποκλείει την επιλογή του πρώτου κανόνα για το πρώτο από τα δύο term, οπότε η παρουσία του συμβόλου ‘(’ ως επόμενη λεκτική μονάδα μας αφήνει μόνο μία επιλογή, αυτή του τελευταίου κανόνα για το term:

```
assign ⇒+ ID ( ID + CONST ) = CONST / ( expr ) + term
```

Αφού επαληθεύσουμε τη συντακτική ορθότητα με το σύμβολο ‘(’, μπορούμε να προχωρήσουμε στην ανάλυση της έκφρασης που είναι μέσα στις παρενθέσεις. Έτσι, επιλέγουμε το δεύτερο κανόνα για το σύμβολο expr, εξαιτίας της παρουσίας του ‘-’ πριν το ‘)’:

```
assign ⇒+ ID ( ID + CONST ) = CONST / ( expr - term ) + term
```

Επειδή δεν υπάρχει άλλο ‘+’ ή ‘-’ πριν το ‘-’, επιλέγουμε τον τελευταίο κανόνα για το expr, και στη συνέχεια, επειδή δεν υπάρχει ‘*’ ή ‘/’ , αλλά και επειδή επόμενο σύμβολο είναι το ID, επιλέγουμε τον τρίτο κανόνα για το πρώτο από τα τρία τώρα σύμβολα term, και το μοναδικό κανόνα για το var:

```
assign ⇒+ ID ( ID + CONST ) = CONST / ( term - term ) + term
       ⇒ ID ( ID + CONST ) = CONST / ( var - term ) + term
       ⇒ ID ( ID + CONST ) = CONST / ( ID paren - term ) + term
```

Τώρα ο ΣΑ μπορεί να επαληθεύσει την ορθότητα της παρουσίας του συμβόλου ID, και να επιλέξει το δεύτερο κανόνα για το σύμβολο paren, αφού δεν ακολουθεί ‘(’:

```
assign ⇒+ ID ( ID + CONST ) = CONST / ( ID - term ) + term
```

Στη συνέχεια επαληθεύει το σύμβολο ‘-’ και επιλέγει τον πρώτο κανόνα για το πρώτο από τα δύο term, λόγω της παρουσίας του ‘*’ πριν το ‘)’:

```
assign ⇒+ ID ( ID + CONST ) = CONST / ( ID - term * var ) + term
```

Το επόμενο σύμβολο είναι το CONST, χωρίς να ακολουθεί ‘/’, οπότε επιλέγεται ο τέταρτος κανόνας για το πρώτο term:

```
assign ⇒+ ID ( ID + CONST ) = CONST / ( ID - CONST * var ) + term
```

Το πρόγραμμα είναι συντακτικά ορθό για τα δύο επόμενα σύμβολα, και ο ΣΑ εφαρμόζει το μοναδικό κανόνα για το var, ενώ στη συνέχεια επιλέγει το δεύτερο κανόνα για το paren, εφ’ όσον ακολουθεί το σύμβολο ‘)’:

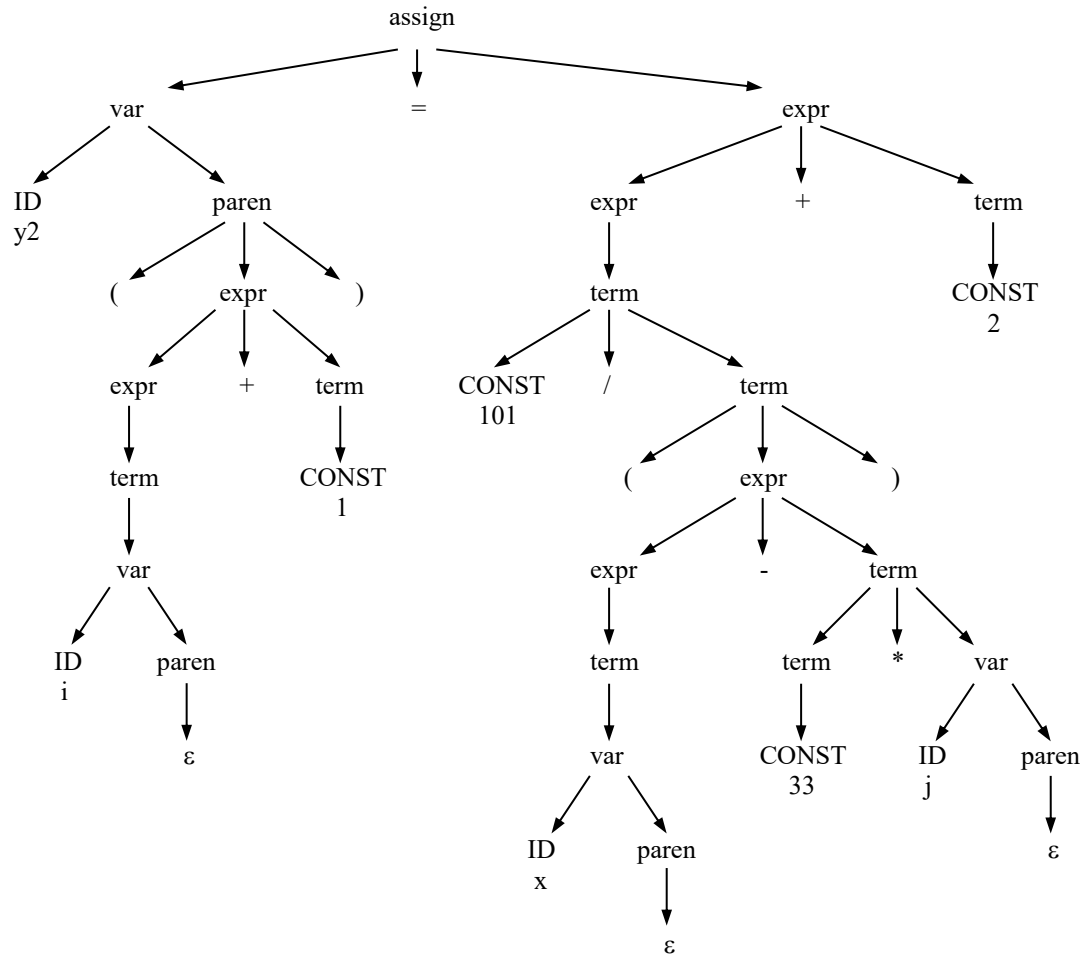
```
assign ⇒+ ID ( ID + CONST ) = CONST / ( ID - CONST * ID paren ) + term
       ⇒ ID ( ID + CONST ) = CONST / ( ID - CONST * ID ) + term
```

Με την επαλήθευση των συμβόλων ID και ‘)’, ο ΣΑ καλεί το σημασιολογικό αναλυτή για έλεγχο της έκφρασης του πολλαπλασιασμού και της αφαίρεσης. Τώρα μπορεί να παραχθεί ο ενδιάμεσος κώδικας για τον υπολογισμό της τιμής της έκφρασης “x-33*j”. Στη συνέχεια, με την επαλήθευση του ‘+’, καλείται πάλι ο σημασιολογικός αναλυτής για έλεγχο της έκφρασης της διαίρεσης, και μπορεί να παραχθεί ο αντίστοιχος ενδιάμεσος κώδικας. Για την υπόλοιπη παραγωγή, επιλέγεται ο τέταρτος κανόνας για το σύμβολο term, εφ’ όσον το επόμενο σύμβολο από το πρόγραμμα είναι το CONST, ενώ δεν ακολουθεί άλλο:

```
assign ⇒+ ID ( ID + CONST ) = CONST / ( ID - CONST * ID ) + CONST
```

Εδώ ολοκληρώνεται η επαλήθευση της συντακτικής ορθότητας του προγράμματος με το σύμβολο CONST, καλείται ο σημασιολογικός αναλυτής για έλεγχο της έκφρασης της πρόσθεσης, αλλά και της ανάθεσης, και μπορεί να παραχθεί ο ενδιάμεσος κώδικας για το υπόλοιπο πρόγραμμα.

Το δέντρο συντακτικής ανάλυσης (ΔΣΑ) που αντιστοιχεί στην πιο πάνω παραγωγή φαίνεται στην επόμενη σελίδα, όπου κάτω από τα τερματικά σύμβολα ID και CONST γράφονται οι πληροφορίες που ο ΛΑ παρέχει για σημασιολογική ανάλυση. Στην πραγματικότητα, αντί για τα ονόματα των μεταβλητών που εδώ γράφονται για απλούστευση, παρέχονται οι αντίστοιχοι



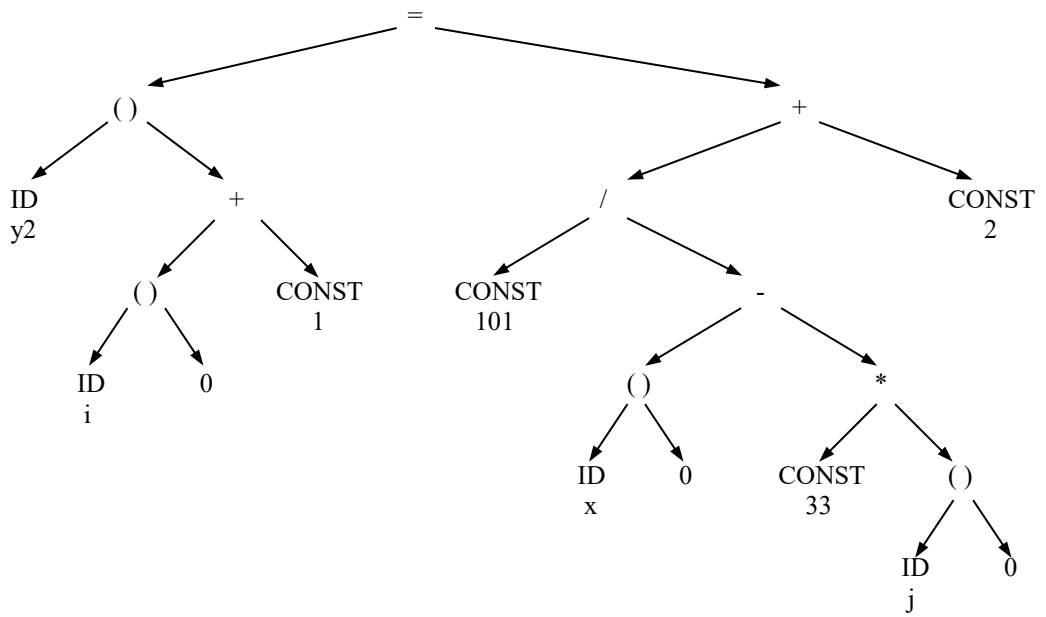
Δέντρο συντακτικής ανάλυσης για την έκφραση "`y2(i+1) = 101 / (x-33*j) + 2`"

δείκτες στον ΠΣ. Το ΔΣΑ αποτελεί μια εναλλακτική απεικόνιση της παραγωγής, με τη διαδικασία παραγωγής να προκύπτει από το ΔΣΑ με διαπέραση αυτού από πάνω προς τα κάτω και από αριστερά προς τα δεξιά.

Β. Ένα αφηρημένο συντακτικό δέντρο (ΑΣΔ) μπορεί να προκύψει από το ΔΣΑ, με όσο το δυνατό περισσότερες αφαιρέσεις μη τερματικών συμβόλων, και προωθήσεις στη θέση αυτών τερματικών συμβόλων τελεστών από το επόμενο επίπεδο, καθώς και με αφαιρέσεις άλλων τερματικών συμβόλων που δεν παρέχουν καμία πληροφορία για το σημασιολογικό αναλυτή ή για την παραγωγή ενδιάμεσου κώδικα. Εναλλακτικά, το ΑΣΔ μπορεί να κατασκευαστεί κατ' ευθείαν μέσα από το ΣΑ, μια που οι παραπάνω απλοποιήσεις μπορούν να ενσωματωθούν στην κατασκευή του, ενώ από την άλλη μεριά το ΔΣΑ δε χρειάζεται στην πραγματικότητα να κατασκευαστεί. Έτσι για παράδειγμα, ο ΣΑ μπορεί να κατασκευάσει κόμβους τελεστών με παιδιά τα τελούμενα αυτών, ή ακόμα μπορεί να αποφύγει την κατασκευή κόμβου όταν κάτι τέτοιο δεν είναι απαραίτητο.

Το ΑΣΔ που αντιστοιχεί στο παραπάνω ΔΣΑ δίνεται στην επόμενη σελίδα. Παρατηρούμε ότι:

1. Από το αρχικό δέντρο διατηρήσαμε τα τερματικά σύμβολα ID, CONST και όλους τους τελεστές, συμπεριλαμβανομένου του τελεστή ανάθεσης. Ακόμα, αφαιρέσαμε τις παρενθέσεις που αντιστοιχούν στον κανόνα "term → (expr)", επειδή δε χρειάζονται στη συνέχεια. Πράγματι, η προτεραιότητα που σημασιολογικά δίνουν στην εκτέλεση κάποιων πράξεων παραμένει στη δομή του δέντρου και μετά την αφαίρεσή τους από αυτό, δεδομένου ότι ο τελικός κώδικας που εκτελεί τις πράξεις παράγεται με διαπέραση του ΑΣΔ από κάτω προς τα πάνω.



Ένα αφηρημένο συντακτικό δέντρο για την έκφραση "*y2(i+1) = 101 / (x-33*j) + 2*"

2. Από τα μη τερματικά σύμβολα, αφαιρέσαμε τα assign, expr και term, επειδή προωθήσαμε τα αντίστοιχα σύμβολα τελεστών στη θέση αυτών. Ειδικότερα για παραγωγές όπου δε συμμετέχουν τελεστές (όπως για παράδειγμα από τον κανόνα "expr → term") το μη τερματικό σύμβολο του αριστερού μέλους αφαιρέθηκε εντελώς από το δέντρο, και στη θέση του ανέβηκε το σύμβολο του δεξιού μέλους, που πιθανά αφαιρέθηκε κι αυτό με τη σειρά του.
3. Το μη τερματικό σύμβολο var αφαιρέθηκε και αντικαταστάθηκε με το σύμβολο '( )', το οποίο λειτουργεί ως τελεστής αναφοράς σε πίνακα με τη σημασιολογία που περιγράφει τον υπολογισμό μιας διεύθυνσης μνήμης από την πρόσθεση της αρχικής διεύθυνσης μιας μεταβλητής και της τιμής ενός δείκτη. Ας σημειωθεί ότι το σύμβολο αυτό παριστάνει μια διεύθυνση, και όχι απαραίτητα την τιμή της αντίστοιχης μεταβλητής.
4. Το μη τερματικό σύμβολο paren αφαιρείται, με την προώθηση του αντίστοιχου κόμβου προς τα επάνω. Ειδικότερα, η κενή συμβολοσειρά αντικαθίσταται με το 0 σαν τιμή του αντίστοιχου δείκτη. Με τον τρόπο αυτό χειριζόμαστε βαθμωτές μεταβλητές ως πίνακες με δείκτη 0.

Όπως αναφέραμε παραπάνω, ο τελικός κώδικας μπορεί να παραχθεί με διαπέραση του ΑΣΔ από κάτω προς τα πάνω και από αριστερά προς τα δεξιά. Πριν όμως ο μεταγλωττιστής μπορέσει να παράγει τελικό κώδικα, θα πρέπει να γνωρίζει την οργάνωση του χώρου δεδομένων που απαιτεί η γλώσσα που μεταφράζεται, και τον τρόπο με τον οποίο η οργάνωση αυτή θα υποστηριχτεί στην τελική γλώσσα. Επίσης, θα πρέπει να γνωρίζει το μοντέλο συνόλου εντολών της τελικής γλώσσας, ώστε να ξέρει πού ο τελικός κώδικας θα μπορεί να αποθηκεύει εν-διάμεσα αποτελέσματα κάποιων πράξεων, και φυσικά τις ίδιες τις διαθέσιμες εντολές της τελικής γλώσσας.
Κάνοντας κάποιες συγκεκριμένες υποθέσεις για την οργάνωση του χώρου δεδομένων, τις οποίες θα αποφύγουμε να αναλύσουμε εδώ, αλλά και θεωρώντας το μοντέλο συνόλου εντολών και τις εντολές MIPS, ένας τελικός κώδικας που θα μπορούσε να παραχθεί από το παραπάνω ΑΣΔ δίνεται στη συνέχεια:

Κώδικας	Σχόλια
`addiu $10,$gp,192`	Αρχική διεύθυνση διανύσματος y2
`lw $11,20($sp)`	Φόρτωση τιμής μεταβλητής i
`addiu $11,$11,1`	Αύξηση κατά 1
`sll $11,$11,2`	Υπολογισμός διεύθυνσης του στοιχείου y2(i+1)
`addu $10,$10,$11`	

`ori $11,$0,101`	Σταθερά 101
`lw $12,180($gp)`	Φόρτωση τιμής μεταβλητής x
`ori $13,$0,33`	Σταθερά 33
`lw $14,36($sp)`	Φόρτωση τιμής μεταβλητής j
`multu $13,$14`	Πολλαπλασιασμός
`mflo $13`	
`subu $12,$12,$13`	Αφαίρεση
`divu $11,$12`	Διαίρεση
`mflo $11`	
`addiu $11,$11,2`	Αύξηση κατά 2
`sw $11,0($10)`	Αποθήκευση του στοιχείου y2(i+1)

Οι υποθέσεις που οδήγησαν στον παραπάνω κώδικα ξεφεύγουν από τους εισαγωγικούς σκοπούς της άσκησης, αλλά αναλύονται εκτενώς σε επόμενα κεφάλαια του μαθήματος.

<u>Ενότητα 2:</u> Γλώσσες, γραμματικές και αυτόματα

Άσκηση 2-1:

Να αποδείξετε την ισοδυναμία ενός ΜΠΑ-ε με το ΝΠΑ που κατασκευάζεται από αυτό με τη μέθοδο κατασκευής του δυναμοσυνόλου των καταστάσεων του πρώτου.

Απάντηση

Δύο ΠΑ είναι ισοδύναμα, όταν αναγνωρίζουν την ίδια γλώσσα. Άρα, για να αποδείξουμε το ζητούμενο, αρκεί να αποδείξουμε ότι κάθε συμβολοσειρά που αναγνωρίζεται από το ένα ΠΑ αναγνωρίζεται και από το άλλο.
Έστω λοιπόν τυχαία συμβολοσειρά $\alpha \equiv x_1x_2\ldots x_n$, με $x_1,x_2,\ldots,x_n$ σύμβολα του αλφαβήτου των δύο ΠΑ.
Έστω κατ' αρχήν ότι η α αναγνωρίζεται από το αρχικό ΜΠΑ-ε. Αφού η συμβολοσειρά αυτή αναγνωρίζεται από το ΜΠΑ-ε, θα υπάρχει κάποια ακολουθία $S_{0,1}$, $S_{0,2}$, …, $S_{0,m0}$, $S_{1,1}$, $S_{1,2}$, …, $S_{1,m1}$, …, $S_{n,1}$, $S_{n,2}$, …, $S_{n,mn}$ διαδοχικών καταστάσεών του, όχι αναγκαστικά διαφορετικών μεταξύ τους, με $S_{0,1}$ την αρχική κατάσταση και $S_{n,mn}$ μια τελική κατάστασή του, και με m0, m1, …, mn ≥ 1, τέτοιων ώστε οι μεταβάσεις $S_{0,1} \rightarrow S_{0,2} \rightarrow \ldots \rightarrow S_{0,m0}$, $S_{1,1} \rightarrow S_{1,2} \rightarrow \ldots \rightarrow S_{1,m1}$, ..., $S_{n,1} \rightarrow S_{n,2} \rightarrow \ldots \rightarrow S_{n,mn}$ να είναι ε-μεταβάσεις, και οι μεταβάσεις $S_{0,m0} \rightarrow S_{1,1}$, $S_{1,m1} \rightarrow S_{2,1}$, ..., $S_{n-1,m(n-1)} \rightarrow S_{n,1}$ να είναι μεταβάσεις για τα σύμβολα x_1, x_2, …, x_n αντίστοιχα. Τότε μπορούμε εύκολα να βρούμε την ακολουθία S_0', S_1', …, S_n' διαδοχικών καταστάσεων του ΝΠΑ, η οποία να αναγνωρίζει την α, ως εξής:

- S_0' = ε-κλείσιμο($\{S_{0,1}\}$). Η S_0' είναι η αρχική κατάσταση του ΝΠΑ, όπως αυτή ορίζεται από τον αλγόριθμο μετατροπής.
- S_1' = ε-κλείσιμο($\overline{\delta}$ (S_0', x_1)). Η S_1' προκύπτει άμεσα από τον αλγόριθμο μετατροπής, και επομένως ακολουθεί την S_0' στο ΝΠΑ με είσοδο x_1, ενώ περιέχει την $S_{1,1}$, αφού η S_0' περιέχει την $S_{0,m0}$ και η $S_{1,1}$ ακολουθεί την $S_{0,m0}$ στο αρχικό ΠΑ με είσοδο x_1.
- Όμοια συνεχίζοντας βρίσκουμε τις καταστάσεις S_i', με $2 \le i \le n$. Για τον ίδιο λόγο όπως προηγουμένως, η S_n' περιέχει την κατάσταση $S_{n,1}$.
- Η S_n' περιέχει τότε και την $S_{n,mn}$ ως εφαρμογή της συνάρτησης ε-κλείσιμο. Αφού η $S_{n,mn}$ είναι τελική κατάσταση στο ΜΠΑ-ε, η S_n' θα είναι τελική κατάσταση στο ΝΠΑ, κι επομένως η α θα αναγνωρίζεται στο ΝΠΑ.

Έστω τώρα ότι η συμβολοσειρά α αναγνωρίζεται από το ΝΠΑ. Αυτό σημαίνει ότι υπάρχει μια ακολουθία S_0', S_1', …, S_n' διαδοχικών καταστάσεων του ΝΠΑ, όχι αναγκαστικά διαφορετικών μεταξύ τους, με S_0' την αρχική κατάσταση και S_n' μια τελική κατάστασή του, τέτοιων ώστε οι μεταβάσεις $S_0' \rightarrow S_1'$, $S_1' \rightarrow S_2'$, ..., $S_{n-1}' \rightarrow S_n'$ να είναι μεταβάσεις για τα σύμβολα x_1, x_2, …, x_n αντίστοιχα. Η κατασκευή του ΝΠΑ μας λέει ότι κάθε μία από τις μεταβάσεις αυτές είναι μεταβάσεις μεταξύ συνόλων καταστάσεων του αρχικού ΜΠΑ-ε. Έτσι, η μετάβαση $S_{i-1}' \rightarrow S_i'$ που γίνεται για το σύμβολο x_i ($1 \le i \le n$) υποδηλώνει ότι υπάρχει τουλάχιστον μία μετάβαση μεταξύ δύο καταστάσεων του ΜΠΑ-ε, μία από το σύνολο S_{i-1}' και μία από το σύνολο S_i', που γίνεται για το x_i, και όλες οι καταστάσεις του S_i' που δεν είναι άμεσα προσιτές από καταστάσεις του S_{i-1}' είναι αναγκαστικά προσιτές από άλλες καταστάσεις του S_i' μέσω ε-μεταβάσεων. Με βάση την παρατήρηση αυτή μπορούμε να βρούμε μια ακολουθία $S_{0,1}$, $S_{0,2}$, …, $S_{0,m0}$, $S_{1,1}$, $S_{1,2}$, …, $S_{1,m1}$, …, $S_{n,1}$, $S_{n,2}$, …, $S_{n,mn}$ διαδοχικών καταστάσεων του αρχικού ΠΑ, με m0, m1, …, mn ≥ 1, η οποία να αναγνωρίζει την α, δουλεύοντας από το τέλος της συμβολοσειράς προς την αρχή, ως εξής:

- Εφόσον η S_n' είναι τελική κατάσταση του ΝΠΑ, θα περιέχει τουλάχιστον μία τελική κατάσταση του αρχικού ΜΠΑ-ε. Θεωρούμε μία τέτοια τελική κατάσταση ως την κατάσταση $S_{n,mn}$.
- Σύμφωνα με την παραπάνω παρατήρηση, υπάρχει ένα ζεύγος καταστάσεων $S_{n-1,m(n-1)}$ και $S_{n,1}$ του αρχικού ΠΑ, με την πρώτη να ανήκει στο σύνολο S_{n-1}' και τη δεύτερη να ανήκει στο σύνολο S_n', με μετάβαση από την πρώτη στη δεύτερη με το σύμβολο x_n, τέτοιων ώστε,

αν η $S_{n,1}$ δεν ταυτίζεται με την $S_{n,mn}$, τότε να υπάρχει ακολουθία $S_{n,1}$, $S_{n,2}$, …, $S_{n,mn}$ διαδοχικών καταστάσεων του ΜΠΑ-ε που να ανήκουν στο σύνολο S_n', με τις μεταβάσεις $S_{n,1} \rightarrow S_{n,2} \rightarrow \ldots \rightarrow S_{n,mn}$ να είναι όλες ε-μεταβάσεις.

- Επαναλαμβάνουμε τα παραπάνω ξεκινώντας με την $S_{n-1,m(n-1)}$ και για όλα τα προηγούμενα σύμβολα της *α*, μέχρι να καταλήξουμε σε κάποια κατάσταση $S_{0,m0}$ του αρχικού ΠΑ, η οποία να περιέχεται στην αρχική κατάσταση S_0' του ΝΠΑ.
- Από την κατασκευή της κατάστασης S_0' γνωρίζουμε ότι αν η $S_{0,m0}$ δεν ταυτίζεται με την αρχική κατάσταση $S_{0,1}$ του αρχικού ΠΑ, τότε θα υπάρχει ακολουθία $S_{0,1}$, $S_{0,2}$, …, $S_{0,m0}$ διαδοχικών καταστάσεών του που να ανήκουν στο σύνολο S_0', με τις μεταβάσεις $S_{0,1} \rightarrow S_{0,2} \rightarrow \ldots \rightarrow S_{0,m0}$ να είναι όλες ε-μεταβάσεις. Έχοντας καταλήξει στην αρχική κατάσταση του ΜΠΑ-ε, έχουμε βρει την ακολουθία καταστάσεών του που αναγνωρίζει την *α*.

Για την ειδική περίπτωση που η *α* είναι η κενή συμβολοσειρά, η απόδειξη δεν αλλάζει, αρκεί να μη θεωρήσουμε καθόλου μεταβάσεις για σύμβολα του αλφαβήτου των δύο ΠΑ. Τότε η $S_{0,m0}$ θα είναι τελική κατάσταση του αρχικού ΜΠΑ-ε, ενώ η S_0' θα είναι αναγκαστικά και τελική κατάσταση του ΝΠΑ.□

Άσκηση 2-2:

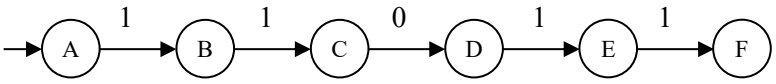
Θεωρήστε τη γλώσσα των συμβολοσειρών ψηφίων 0 και 1, στις οποίες υπάρχουν τουλάχιστον δύο 1 πριν από κάθε 0, εκτός αν το 0 είναι το πρώτο ψηφίο, και τουλάχιστον δύο 1 μετά από κάθε 0, εκτός αν το 0 είναι το τελευταίο ψηφίο.

Α. Να δώσετε ένα ΠΑ που να αναγνωρίζει τις συμβολοσειρές της γλώσσας. Αν δεν είναι ήδη ΝΠΑ, να το μετατρέψετε σε ΝΠΑ, και αν δεν είναι ήδη ελάχιστο, να το ελαχιστοποιήσετε. Στη συνέχεια να ανάγετε το τελικό ΝΠΑ σε ΚΕ.

Β. Να δώσετε μια άλλη ΚΕ που να περιγράφει τις συμβολοσειρές της γλώσσας. Χρησιμοποιήστε την κατασκευή Thompson για να ανάγετε την ΚΕ σε ΜΠΑ-ε. Να μετατρέψετε το ΜΠΑ-ε σε ΝΠΑ και να το ελαχιστοποιήσετε. Επαληθεύστε ότι προέκυψε το ίδιο ελάχιστο ΝΠΑ με προηγουμένως!

Απάντηση

Α. Σύμφωνα με την περιγραφή των συμβολοσειρών που πρέπει να αναγνωρίζονται από αυτό, το ζητούμενο ΠΑ θα πρέπει να περιέχει υποχρεωτικά το πιο κάτω τμήμα:



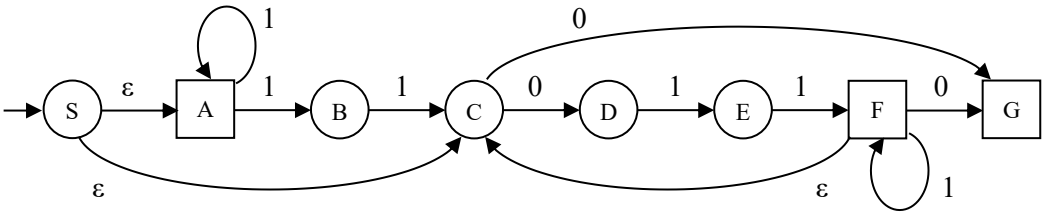
ώστε να αναγνωρίζει μια συμβολοσειρά από δύο 1 ακολουθούμενα από 0 και πάλι από δύο 1.

Για τις συμβολοσειρές που δεν ξεκινάνε με 0 ούτε τελειώνουν σε 0, η κατάσταση Α θα αποτελεί αρχική κατάσταση, ενώ η κατάσταση F θα αποτελεί τελική κατάσταση. Επίσης, η Α μπορεί να είναι και τελική κατάσταση, ώστε να αναγνωρίζονται συμβολοσειρές χωρίς κανένα ψηφίο 0, συμπεριλαμβανομένης και της κενής συμβολοσειράς.
Μετάβαση από την Α στον εαυτό της με το ψηφίο 1 επιτρέπει την αναγνώριση συμβολοσειράς με περισσότερα από δύο συνεχόμενα 1 πριν το 0, ενώ μετάβαση με 1 από την F στον εαυτό της επιτρέπει πρόσθετα ψηφία 1 μετά τα δύο 1 που υποχρεωτικά ακολουθούν ένα 0.
Ακόμα, μια ε-μετάβαση από την κατάσταση F πίσω στην C εξασφαλίζει την αναγνώριση συμβολοσειρών με περισσότερα από ένα ψηφία 0 που να συμφωνούν με τη δεδομένη περιγραφή.

Για να συμπεριλάβουμε την περίπτωση η συμβολοσειρά να αρχίζει με 0, εισάγουμε μια νέα αρχική κατάσταση, έστω S, από την οποία να μεταβαίνουμε αυθόρμητα είτε στην Α είτε στην C. Τέλος, για να συμπεριλάβουμε την περίπτωση η συμβολοσειρά να τελειώνει σε 0, αρκεί να

προσθέσουμε μια ακόμα τελική κατάσταση, έστω G, στην οποία να μεταβαίνουμε με 0 είτε από την C είτε από την F.

Κατασκευάσαμε έτσι το πιο κάτω ΠΑ, το οποίο είναι ΜΠΑ-ε:



Για να μετατρέψουμε το αυτόματο αυτό σε ΝΠΑ, εφαρμόζουμε τον αλγόριθμο μετατροπής, δημιουργώντας μία-μία τις καταστάσεις του νέου αυτόματου, εξετάζοντας στη συνέχεια τις μεταβάσεις της.
Έτσι, η αρχική κατάσταση του νέου αυτόματου, έστω A', θα είναι το ε-κλείσιμο του συνόλου {S}:

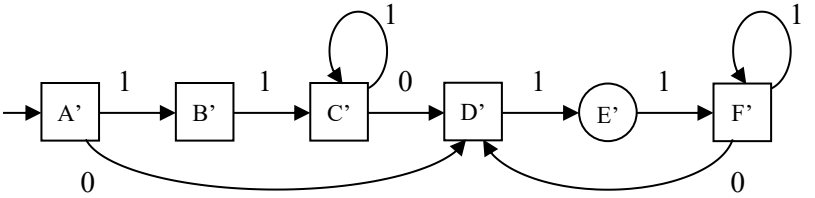
A' ≡ ε-κλείσιμο({S}) = {S,A,C}

Οι υπόλοιπες καταστάσεις και οι μεταβάσεις μεταξύ αυτών για το νέο αυτόματο βρίσκονται ως εξής:

B' ≡ δ(A', 1) = ε-κλείσιμο({A,B}) = {A,B}
D' ≡ δ(A', 0) = ε-κλείσιμο({D,G}) = {D,G}
C' ≡ δ(B', 1) = ε-κλείσιμο({A,B,C}) = {A,B,C}
E' ≡ δ(D', 1) = ε-κλείσιμο({E}) = {E}
δ(C', 1) = ε-κλείσιμο({A,B,C}) = C'
δ(C', 0) = ε-κλείσιμο({D,G}) = D'
F' ≡ δ(E', 1) = ε-κλείσιμο({F}) = {F,C}
δ(F', 1) = ε-κλείσιμο({F}) = F'
δ(F', 0) = ε-κλείσιμο({D,G}) = D'

ενώ άλλες μεταβάσεις δεν ορίζονται, ή αλλιώς θεωρούμε ότι οδηγούν σε μια υπονοούμενη κατάσταση παγίδευσης, κάτι που ισοδυναμεί με απόρριψη.
Οι τελικές καταστάσεις του νέου αυτόματου είναι οι A', B', C', D' και F', εφ' όσον περιέχουν μία τουλάχιστον από τις τελικές καταστάσεις του αρχικού αυτόματου A, F και G.

Το ισοδύναμο ΝΠΑ είναι επομένως το εξής:



Για να ελαχιστοποιήσουμε το παραπάνω ΝΠΑ, εφαρμόζουμε τον αλγόριθμο ελαχιστοποίησης, που εξετάζει επαναληπτικά όλα τα δυνατά ζεύγη πιθανά ισοδύναμων καταστάσεων.
Έτσι, διαχωρίζοντας τις τελικές από τις μη τελικές καταστάσεις, και συνυπολογίζοντας την υπονοούμενη κατάσταση παγίδευσης, έστω Π, στην οποία καταλήγουν όλες οι μη ορισμένες μεταβάσεις, λαμβάνουμε το ακόλουθο αρχικό σύνολο ζευγών πιθανά ισοδύναμων καταστάσεων S:

S = {(E',Π), (A',B'), (A',C'), (A',D'), (A',F'), (B',C'), (B',D'), (B',F'), (C',D'), (C',F'), (D',F')}

Στη συνέχεια θα εξετάσουμε το S όσες φορές χρειάζεται, διαγράφοντας ζεύγη καταστάσεων που προκύπτουν μη ισοδύναμες.

Κατ' αρχήν παρατηρούμε ότι η Ε' δε μπορεί να είναι ισοδύναμη με την Π, εφ' όσον έχει μετάβαση προς τελική κατάσταση και επομένως δεν είναι κατάσταση παγίδευσης. Άρα το ζεύγος (Ε',Π) διαγράφεται από το σύνολο S.
Μια ικανή – αλλά όχι και αναγκαία – συνθήκη διαγραφής ζεύγους καταστάσεων που είναι εύκολο να ελεγχθεί όταν στο ΝΠΑ δεν απεικονίζονται καταστάσεις παγίδευσης, είναι αν για κάποιο σύμβολο η μια κατάσταση από το ζεύγος δεν ορίζει μετάβαση, ενώ η άλλη ορίζει. Το ζεύγος τότε διαγράφεται, αφού οι δύο καταστάσεις διακρίνονται για τουλάχιστον μία συμβολοσειρά, αυτή που με το σύμβολο αυτό οδηγεί σε αναγνώριση μέσω της μετάβασης που είναι ορισμένη. Τέτοια συμβολοσειρά πρέπει να υπάρχει, αλλιώς η κατάσταση στην οποία γίνεται μετάβαση θα έπρεπε να είναι κατάσταση παγίδευσης.
Τα ζεύγη που διαγράφονται με βάση την παραπάνω συνθήκη, και από τη στιγμή που η Ε' δεν είναι κατάσταση παγίδευσης, είναι τα (A',B'), (A',D'), (B',C'), (B',F'), (C',D') και (D',F'), επειδή οι καταστάσεις B' και D' δεν ορίζουν μετάβαση για το ψηφίο 0.
Έτσι, καταλήγουμε στο ακόλουθο σύνολο S:

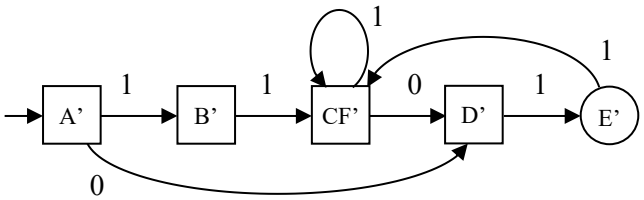
S = { (A',C'), (A',F'), (B',D'), (C',F')}

Οι καταστάσεις του πρώτου ζεύγους δε διακρίνονται για το ψηφίο 0, αφού και οι δύο με αυτό οδηγούν στην ίδια κατάσταση D'. Διακρίνονται όμως για το ψηφίο 1, αφού με αυτό οδηγούν στις καταστάσεις B' και C', και το ζεύγος (B',C') δεν ανήκει στο S. Επομένως, το πρώτο ζεύγος διαγράφεται από το S. Για τον ίδιο λόγο, οι καταστάσεις του δεύτερου ζεύγους διακρίνονται για το ψηφίο 1, και επομένως, και το δεύτερο ζεύγος διαγράφεται από το S. Το τρίτο ζεύγος επίσης διαγράφεται από το S, αφού οι καταστάσεις του διακρίνονται για το ψηφίο 1, με το οποίο η μία οδηγεί σε τελική και η άλλη σε μη τελική κατάσταση.
Το τελευταίο ζεύγος (C',F') είναι το μόνο που θα παραμείνει στο S, επειδή οι καταστάσεις του οδηγούν για 0 στην ίδια κατάσταση και για 1 στον εαυτό τους. Πιο συγκεκριμένα, οι καταστάσεις του ζεύγους δε διακρίνονται ούτε για το ψηφίο 0, αφού αυτό οδηγεί και τις δύο στην D', αλλά ούτε και για το ψηφίο 1, αφού αυτό τις οδηγεί στο ζεύγος (C',F') που είναι αυτό που εξετάζουμε και ακόμα ανήκει στο S.
Έτσι, το S θα γίνει:

S = {(C',F')}

Εξετάζοντας το S για άλλη μια φορά, θα δούμε ότι το ζεύγος (C',F') δε διαγράφεται, κι επομένως θα μας δώσει και τις μοναδικές ισοδύναμες καταστάσεις του αυτόματου.

Μετά από σύμπτυξη των ισοδύναμων καταστάσεων, το ζητούμενο ΝΠΑ θα προκύψει ως εξής:

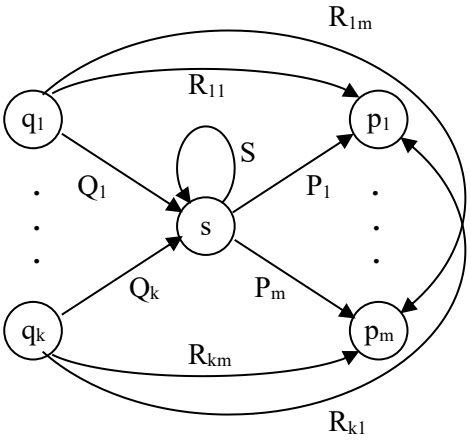


όπου οι δύο ισοδύναμες καταστάσεις έχουν αντικατασταθεί από την κατάσταση CF'.

Για να βρούμε την κανονική έκφραση (ΚΕ) που αντιστοιχεί στο παραπάνω ελαχιστοποιημένο ΝΠΑ, θα χρησιμοποιήσουμε τον αλγόριθμο που περιγράφεται στη συνέχεια. Ο αλγόριθμος αυτός βασίζεται σε δύο ορισμούς:
(α) Το *γενικευμένο πεπερασμένο αυτόματο* είναι το ΠΑ, το οποίο αντί για μεμονωμένα σύμβολα, έχει ως ετικέτες ΚΕ πάνω στο αλφάβητό του, έτσι ώστε μια μετάβαση να μπορεί να γίνει, εάν και μόνο εάν η αντίστοιχη ΚΕ αναγνωρίζεται στη συμβολοσειρά εισόδου, και
(β) Η *κενή ΚΕ*, που συμβολίζουμε ως ∅, είναι η ΚΕ που περιγράφει την κενή γλώσσα, με ιδιότητες: $\varnothing|A = A$, $\varnothing A = \varnothing$, $\varnothing^* = \varepsilon$, όπου Α άλλη ΚΕ.

Έτσι, για κάθε τελική κατάσταση του αυτόματου:

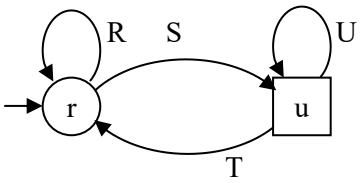
1. Θεωρούμε το αυτόματο που προκύπτει από το αρχικό αν διατηρηθεί μόνο αυτή η τελική κατάσταση και οι υπόλοιπες μετατραπούν σε μη τελικές. Απαλείφουμε όσες καταστάσεις γίνονται ή προκύπτουν ισοδύναμες με καταστάσεις παγίδευσης. Γενικεύουμε το αυτόματο, θεωρώντας τα σύμβολα μετάβασης ως ΚΕ.
2. Αφαιρούμε διαδοχικά από το αυτόματο καταστάσεις και τις αντίστοιχες συνδέσεις γύρω από αυτές, δημιουργώντας νέες συνδέσεις με ετικέτες κατάλληλες ΚΕ. Έστω το πιο κάτω τμήμα ενός γενικευμένου ΝΠΑ:



Το σχήμα αυτό παρουσιάζει όλες τις μεταβάσεις που θα μπορούσαν να σχετίζονται με κάποια τυχαία μη τελική κατάσταση s. Οι υπόλοιπες καταστάσεις μπορούν να είναι και τελικές. Μια κατάσταση p_i είναι δυνατό να ταυτίζεται με μια κατάσταση q_j. Αν η απ' ευθείας μετάβαση μεταξύ των p_i και q_j δεν ορίζεται, ο όρος R_{ij} γίνεται ∅. Παρόμοια για τον όρο S, αν δεν υπάρχει κυκλική μετάβαση στην s.
Η αφαίρεση της κατάστασης s από το αυτόματο αυτό οδηγεί σε ένα αυτόματο όπου κάθε κατάσταση q_i συνδέεται απ' ευθείας με κάθε κατάσταση p_j με ετικέτα την ΚΕ:

$R_{ij}|Q_iS^*P_j$

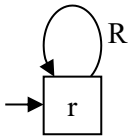
Η αφαίρεση καταστάσεων γίνεται για όλες τις καταστάσεις πλην της αρχικής και της τελικής.
3. Όταν δε μπορούν να αφαιρεθούν άλλες καταστάσεις, υπάρχουν δύο δυνατές περιπτώσεις. Είτε έχουμε ένα αυτόματο της μορφής:



όπου R, S, T και U είναι επιμέρους ΚΕ (πιθανά όλες ∅ εκτός της S), οπότε μια ΚΕ που δίνει τις συμβολοσειρές που αναγνωρίζονται από το αυτόματο μπορεί να είναι η:

$(R|SU^*T)^*SU^*$

είτε έχουμε ένα αυτόματο της μορφής:

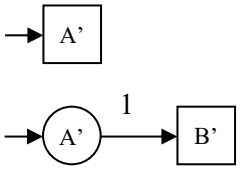


όπου R είναι επιμέρους ΚΕ (πιθανά ∅), οπότε η ΚΕ των συμβολοσειρών του αυτόματου είναι η:

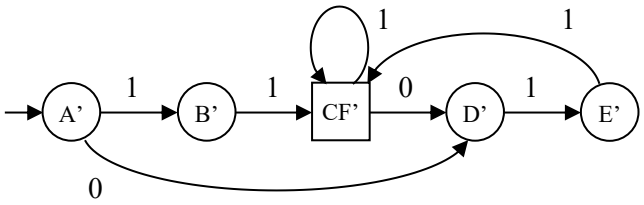
R^*

Η ζητούμενη ΚΕ είναι η διάζευξη όλων των ΚΕ που βρίσκονται για κάθε τελική κατάσταση του ΝΠΑ ξεχωριστά.

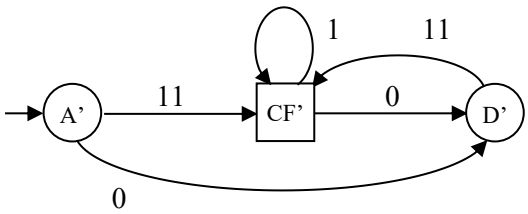
Για το ΝΠΑ που μας δίνεται, μπορούμε εύκολα να πάρουμε 4 επιμέρους αυτόματα, ένα για κάθε μία από τις 4 τελικές του καταστάσεις.
Έτσι, για τις τελικές καταστάσεις A’ και B’ παίρνουμε τα αυτόματα:



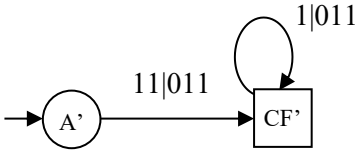
τα οποία αναγνωρίζουν τις συμβολοσειρές ε και 1 αντίστοιχα. Παρατηρήστε ότι στα δύο πιο πάνω αυτόματα όλες οι υπόλοιπες καταστάσεις έχουν αφαιρεθεί, επειδή ισοδυναμούν με καταστάσεις παγίδευσης, καθώς δε μπορούν να οδηγήσουν στην A’ ή στη B’, αντίστοιχα.
Για την τελική κατάσταση CF’ παίρνουμε το αυτόματο:



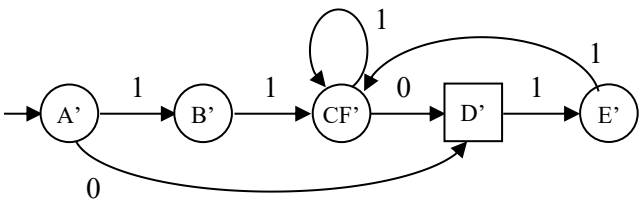
στο οποίο μπορούμε να εφαρμόσουμε τον πιο πάνω αλγόριθμο, κατ’ αρχήν αφαιρώντας την κατάσταση B’, ώστε να πάρουμε μετάβαση από την A’ στη CF’ με ετικέτα 11, καθώς και την κατάσταση E’, ώστε να πάρουμε μια νέα μετάβαση από την D’ στη CF’ με ετικέτα 11. Έτσι, προκύπτει το ακόλουθο γενικευμένο αυτόματο:



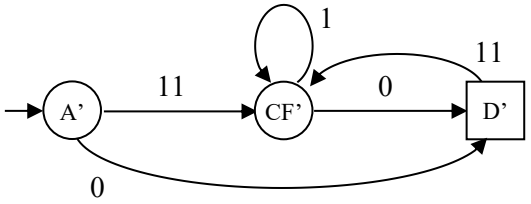
Η κατάσταση D’ αφαιρείται με ταυτόχρονη δημιουργία μιας εναλλακτικής μετάβασης από την A’ στη CF’ με ετικέτα 011 και μιας εναλλακτικής κυκλικής μετάβασης από την CF’ στον εαυτό της, επίσης με ετικέτα 011.
Οι δύο μεταβάσεις από την A’ στη CF’ συμπτύσσονται σε μία, με ετικέτα τη διάζευξη των ετικετών των αρχικών μεταβάσεων, δηλαδή την 11|011. Παρόμοια, οι δύο εναλλακτικές κυκλικές μεταβάσεις της CF’ συμπτύσσονται σε μία, με ετικέτα 1|011. Έτσι προκύπτει το ακόλουθο αυτόματο:



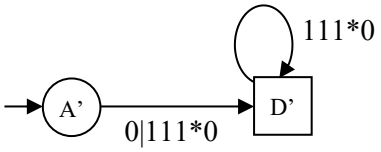
οπότε η ΚΕ που αντιστοιχεί στο αυτόματο αυτό είναι η (11|011)(1|011)*.
Τέλος, για την τελική κατάσταση D’ το αντίστοιχο αυτόματο θα είναι:



Αφαιρώντας όπως παραπάνω τις καταστάσεις Β' και Ε' προκύπτει το γενικευμένο αυτόματο:



Τώρα αφαιρούμε την CF', οπότε δημιουργείται εναλλακτική μετάβαση από την A' στην D' με ετικέτα 111*0, και κυκλική μετάβαση από την D' στον εαυτό της, επίσης με ετικέτα 111*0. Οι δύο μεταβάσεις από την A' στην D' συμπτύσσονται σε μία, με ετικέτα 0|111*0. Καταλήγουμε έτσι στο αυτόματο:



με αντίστοιχη ΚΕ την (0|111*0)(111*0)*.
Συνολικά, η ζητούμενη ΚΕ για το ΝΠΑ που βρήκαμε θα είναι η:

ε|1|(11|011)(1|011)*| (0|111*0)(111*0)*

Παρατηρήστε ότι:
(1) Το ΜΠΑ που κατασκευάσαμε αρχικά δεν είναι το μοναδικό για την περιγραφή που μας δίνεται στην άσκηση. Για παράδειγμα, η δεύτερη κυκλική μετάβαση θα μπορούσε να ήταν σε μια από τις καταστάσεις D, E, αντί στην F.
(2) Η πιο πάνω μέθοδος εύρεσης της ΚΕ μπορεί να εφαρμοστεί απ' ευθείας σε ένα ΜΠΑ, μια που σε κανένα στάδιο της κατασκευής αυτής δε χρησιμοποιήσαμε το γεγονός ότι το αυτόματο είναι ντετερμινιστικό.
(3) Στη μέθοδο αναγωγής ΠΑ σε ΚΕ που είδαμε, θεωρήσαμε ξεχωριστά κάθε τελική κατάσταση. Θα μπορούσαμε εναλλακτικά να μετατρέψουμε όλες τις τελικές καταστάσεις σε μη τελικές, εισάγοντας παράλληλα ε-μεταβάσεις από αυτές προς μια νέα μοναδική τελική κατάσταση. Έτσι δεν χρειάζεται να δουλεύουμε ξεχωριστά για κάθε τελική κατάσταση.
(4) Η ΚΕ που περιγράφει μια κανονική γλώσσα δεν είναι μοναδική. Αν για παράδειγμα εφαρμόσουμε τον ίδιο αλγόριθμο στο μη ελαχιστοποιημένο ΝΠΑ, θα καταλήξουμε στην ΚΕ:
ε|1|111*|(0|111*0)(111*0)*|(0|111*0)11(1|011)*

Β. Επειδή η ΚΕ περιγραφής της γλώσσας δεν είναι μοναδική, μπορούμε να βρούμε κάποια άλλη ΚΕ, η οποία όμως να προκύπτει με φυσικό τρόπο από την περιγραφή της γλώσσας, ακριβώς όπως κάναμε στο προηγούμενο ερώτημα για να βρούμε το αρχικό ΜΠΑ-ε.
Έτσι, μια επιμέρους ΚΕ που να επιβάλλει τουλάχιστον δύο 1 πριν και μετά από κάθε 0 είναι η εξής:

1*110111*

η οποία για να επιτρέπει πολλαπλά 0, διαμορφώνεται με τον τελεστή '*' στην ΚΕ:

1*11(0111*)*

Για να επιτρέψουμε πιθανό αρχικό και τελικό 0, τροποποιούμε την ΚΕ με τον τελεστή '?' σε:

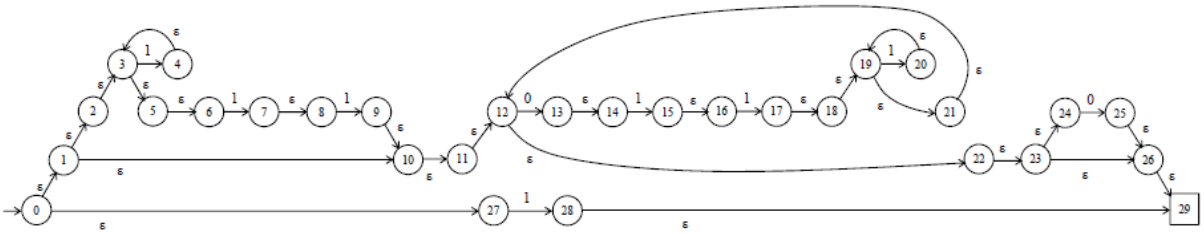
(1*11)?(0111*)*0?

όπου αντί να βάλουμε όρο 0? στην αρχή, κάναμε τα αρχικά 1 προαιρετικά, ώστε να συμπεριλαμβάνεται και η κενή συμβολοσειρά.
Τέλος, παρατηρώντας ότι η πιο πάνω έκφραση δεν καλύπτει την περίπτωση της συμβολοσειράς "1", την επεκτείνουμε σε:

(1*11)?(0111*)*0?|1

που είναι και η τελική μας ΚΕ.

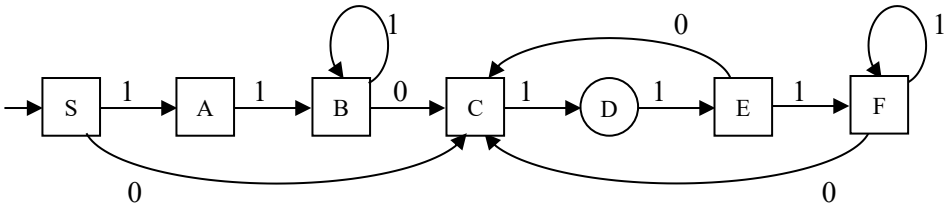
Στη συνέχεια εφαρμόζουμε την κατασκευή Thompson στην ΚΕ που βρήκαμε – χωρίς να κάνουμε απλοποιήσεις, και λαμβάνουμε το πιο κάτω ΜΠΑ-ε:



Η μετατροπή αυτού σε ΝΠΑ γίνεται με το γνωστό τρόπο, για αρχική κατάσταση S:

S ≡	ε-κλείσιμο({0}) =		{0,1,2,3,5,6,10,11,12,22,23,24,26,27,29}
A ≡	δ(S, 1) =	ε-κλείσιμο({4,7,28}) =	{3,4,5,6,7,8,28,29}
C ≡	δ(S, 0) =	ε-κλείσιμο({13,25}) =	{13,14,25,26,29}
B ≡	δ(A, 1) =	ε-κλείσιμο({4,7,9}) =	{3,4,5,6,7,8,9,10,11,12,22,23,24,26,29}
D ≡	δ(C, 1) =	ε-κλείσιμο({15}) =	{15,16}
	δ(B, 0) =	ε-κλείσιμο({13,25}) =	C
	δ(B, 1) =	ε-κλείσιμο({4,7,9}) =	B
E ≡	δ(D, 1) =	ε-κλείσιμο({17}) =	{12,17,18,19,21,22,23,24,26,29}
	δ(E, 0) =	ε-κλείσιμο({13,25}) =	C
F ≡	δ(E, 1) =	ε-κλείσιμο({20}) =	{12,19,20,21,22,23,24,26,29}
	δ(F, 0) =	ε-κλείσιμο({13,25}) =	C
	δ(F, 1) =	ε-κλείσιμο({20}) =	F

Οι τελικές καταστάσεις του ΝΠΑ είναι οι S, A, B, C, E και F, εφ' όσον περιέχουν την τελική κατάσταση 29 του αρχικού ΜΠΑ-ε.
Το ΝΠΑ που προέκυψε είναι:



Μπορούμε εύκολα να δούμε ότι οι καταστάσεις E και F του ΝΠΑ που βρήκαμε είναι ισοδύναμες, οπότε το ΝΠΑ ελαχιστοποιείται στο ίδιο ΝΠΑ που βρήκαμε και νωρίτερα.

Άσκηση 2-3:

Δίνεται η παρακάτω διφορούμενη γραμματική χωρίς συμφραζόμενα:

$$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid + E \mid - E \mid * E \mid (E) \mid ID$$

που αντιστοιχεί σε αριθμητικές παραστάσεις μεταξύ αναγνωριστικών μιας γλώσσας προγραμματισμού. Οι τελεστές έχουν τη συνήθη προτεραιότητα και προσεταιριστικότητα, ενώ ειδικά ο τελεστής '' όταν εφαρμόζεται σε μοναδικό τελούμενο εκτελεί αποδεικτοδότηση και έχει την υψηλότερη προτεραιότητα.*

Α. Να δείξετε πώς με εφαρμογή κανόνων προτεραιότητας και προσεταιριστικότητας μπορεί να παραχθεί μονοσήμαντο δέντρο συντακτικής ανάλυσης για τη συμβολοσειρά:

```
-ID-(ID*ID/*ID/ID-ID)*(*ID-ID**ID-ID)*ID-(ID+*ID)
```

Β. Να δώσετε μια ισοδύναμη μη διφορούμενη της παραπάνω γραμματικής, η οποία να είναι σύμφωνη με τις προτεραιότητες και προσεταιριστικότητες των τελεστών της.

Απάντηση

Μια διφορούμενη γραμματική μπορεί να παράγει την ίδια συμβολοσειρά με περισσότερους από έναν τρόπους, είτε με αριστερότερη, είτε με δεξιότερη παραγωγή. Για παράδειγμα, η συμβολοσειρά

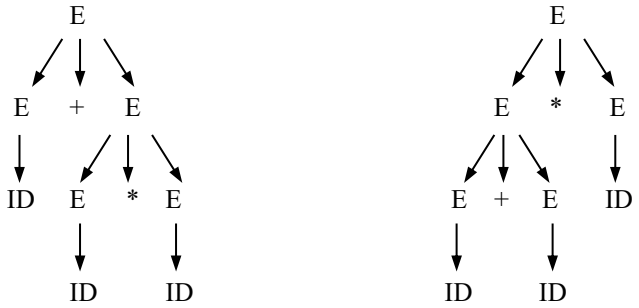
```
ID+ID*ID
```

μπορεί να παραχθεί από την παραπάνω γραμματική με αριστερότερη παραγωγή με τους εξής δύο τρόπους:

$$E \Rightarrow E+E \Rightarrow_L ID+E \Rightarrow ID+E*E \Rightarrow_L ID+ID*E \Rightarrow ID+ID*ID$$

$$E \Rightarrow E*E \Rightarrow_L E+E*E \Rightarrow_L ID+E*E \Rightarrow_L ID+ID*E \Rightarrow ID+ID*ID$$

Στους δύο αυτούς τρόπους αντιστοιχούν δύο διαφορετικά δέντρα συντακτικής ανάλυσης, τα εξής:



Παρόλο που και τα δύο αυτά δέντρα είναι συντακτικά αποδεκτά, δηλαδή κατασκευάζονται από δύο ορθές παραγωγές, στο βήμα παραγωγής τελικού κώδικα θα δούμε ότι το δεύτερο παράγει "λάθος" κώδικα, με την έννοια ότι αυτός δεν εκτελεί τον επιθυμητό αλγεβρικό υπολογισμό. Πράγματι, ο κώδικας του δεύτερου δέντρου εκτελεί πρώτα την πρόσθεση και μετά τον πολλαπλασιασμό, όπως ακριβώς θα γινόταν αν η παράσταση είχε παρενθέσεις, ως (ID+ID)*ID. Αυτό το λάθος δεν είναι συντακτικό, αλλά σημασιολογικό, δηλαδή δίνεται λάθος ερμηνεία στην εφαρμογή των τελεστών πρόσθεσης και πολλαπλασιασμού, και ειδικότερα, λάθος προτεραιότητα μεταξύ των δύο τελεστών.

Για να αποφύγουμε τη λάθος ερμηνεία στην εφαρμογή των τελεστών της γραμματικής μας, πρέπει να βρούμε τρόπο, ώστε χρησιμοποιώντας τις ιδιότητές τους, να κατασκευάζουμε πάντα ένα μοναδικό, και μάλιστα το "σωστό" δέντρο, για οποιαδήποτε τυχαία συμβολοσειρά της γλώσσας. Αυτό ισοδυναμεί με μετασχηματισμό της διφορούμενης γραμματικής σε μη διφορούμενη, στην ουσία ενσωματώνοντας τις σημασιολογικές ιδιότητες των τελεστών στη σύνταξη της γλώσσας. Κάτι τέτοιο είναι εφικτό για γραμματικές αριθμητικών παραστάσεων, οι οποίες δεν είναι *εγγενώς* διφορούμενες.

Α. Οι ιδιότητες τελεστών που μας ενδιαφέρουν για την αντιμετώπιση του παραπάνω προβλήματος και τη μετατροπή της διφορούμενης γραμματικής σε μη διφορούμενη είναι η προτεραιότητα και η προσεταιριστικότητα. Οι δύο αυτές ιδιότητες μας λένε ποιος από δύο διαδοχικούς τελεστές θα εφαρμοστεί πρώτος. Για παράδειγμα, με τον πολλαπλασιασμό να έχει υψηλότερη προτεραιότητα από την πρόσθεση, η παραπάνω συμβολοσειρά παράγεται με τον πρώτο από τους δύο τρόπους που δίνονται, ώστε στον τελικό κώδικα η δεύτερη πράξη να γίνει πριν από την πρώτη. Επίσης, με την πράξη της αφαίρεσης να έχει αριστερή προσεταιριστικότητα, η συμβολοσειρά

```
ID-ID-ID
```

μπορεί να αντιστοιχηθεί στην αριστερότερη παραγωγή

$$E \Rightarrow E-E \Rightarrow_L E-E-E \Rightarrow_L ID-E-E \Rightarrow_L ID-ID-E \Rightarrow ID-ID-ID$$

και όχι στην

$$E \Rightarrow E-E \Rightarrow_L ID-E \Rightarrow ID-E-E \Rightarrow_L ID-ID-E \Rightarrow ID-ID-ID$$

που θα προέκυπτε αν η πράξη της αφαίρεσης είχε δεξιά προσεταιριστικότητα. Παρατηρήστε ότι κώδικας που παράγεται με δεξιά προσεταιριστικότητα για τη συμβολοσειρά αυτή οδηγεί στον υπολογισμό του αριθμού `ID-(ID-ID)` και όχι του `(ID-ID)-ID` που εννοείται με αυτήν. Γενικά, η προσεταιριστικότητα έχει νόημα μόνο μεταξύ τελεστών ίσης προτεραιότητας.
Επομένως, ακόμα και χωρίς άμεσο μετασχηματισμό της διφορούμενης γραμματικής σε μη διφορούμενη, μπορούμε να βρίσκουμε το επιθυμητό δέντρο συντακτικής ανάλυσης, τοποθετώντας ψηλότερα σε αυτό τελεστές χαμηλότερης προτεραιότητας, κατεβαίνοντας προς τα αριστερά για αριστερή προσεταιριστικότητα και προς τα δεξιά για δεξιά προσεταιριστικότητα. Και αντίστοιχα σκεφτόμαστε για να βρούμε τη μοναδική αριστερότερη (ή δεξιότερη) παραγωγή που οδηγεί σε αυτό το δέντρο.
Αξίζει να σημειωθεί πως αν σε μια παράσταση συμμετέχουν παρενθέσεις, το γεγονός ότι αυτές επιβάλλουν υψηλότερη προτεραιότητα σε τελεστές που περικλείουν είναι ιδιότητα που είναι ήδη ενσωματωμένη στη γραμματική, για όλα τα πιθανά φωλιάσματα παρενθέσεων. Έτσι, ψηλότερα στο δέντρο θα βρίσκονται εφαρμογές τελεστών σε εξωτερικά ζευγάρια ή εκτός παρενθέσεων, και χαμηλότερα θα βρίσκονται τελεστές σε φωλιασμένες παρενθέσεις μεγαλύτερου βάθους φωλιάσματος.

Έτσι λοιπόν, για τη γραμματική που μας δίνεται, η προτεραιότητα των τελεστών ορίζεται με την ακόλουθη αύξουσα σειρά:

- `+,-` (πρόσημο, πρόσθεση, αφαίρεση)
- `*,/` (πολλαπλασιασμός, διαίρεση)
- `*` (αποδεικτοδότηση)

ενώ για όλους τους τελεστές ορίζουμε αριστερή προσεταιριστικότητα. Παρατηρήστε ότι η προσεταιριστικότητα δεν επηρεάζει το αποτέλεσμα του κώδικα που πιθανά παράγεται για τις πράξεις της πρόσθεσης και του πολλαπλασιασμού[2], παρά μόνο για τις πράξεις της αφαίρεσης (όπως και του αρνητικού προσήμου) και της διαίρεσης.

Με προτεραιότητα και προσεταιριστικότητα τελεστών όπως παραπάνω, η αριστερότερη παραγωγή που αντιστοιχεί στη συμβολοσειρά που μας δίνεται είναι η ακόλουθη:

```
E ⇒  E-E ⇒L
  ⇒L E-E-E ⇒L
  ⇒L -E-E-E ⇒L
  ⇒L -ID-E-E ⇒L
  ⇒L -ID-E*E-E ⇒L
  ⇒L -ID-E*E*E-E ⇒L
  ⇒L -ID-(E)*E*E-E ⇒L
  ⇒L -ID-(E-E)*E*E-E ⇒L
  ⇒L -ID-(E/E-E)*E*E-E ⇒L
  ⇒L -ID-(E/E/E-E)*E*E-E ⇒L
  ⇒L -ID-(E*E/E/E-E)*E*E-E ⇒L
  ⇒L -ID-(ID*E/E/E-E)*E*E-E ⇒L
  ⇒L -ID-(ID*ID/E/E-E)*E*E-E ⇒L
  ⇒L -ID-(ID*ID/*E/E-E)*E*E-E ⇒L
  ⇒L -ID-(ID*ID/*ID/E-E)*E*E-E ⇒L
  ⇒L -ID-(ID*ID/*ID/ID-E)*E*E-E ⇒L
  ⇒L -ID-(ID*ID/*ID/ID-ID)*E*E-E ⇒L
  ⇒L -ID-(ID*ID/*ID/ID-ID)*(E)*E-E ⇒L
  ⇒L -ID-(ID*ID/*ID/ID-ID)*(E-E)*E-E ⇒L
  ⇒L -ID-(ID*ID/*ID/ID-ID)*(E-E-E)*E-E ⇒L
  ⇒L -ID-(ID*ID/*ID/ID-ID)*(*E-E-E)*E-E ⇒L
```

[2] Αγνοώντας τις συνέπειες της πεπερασμένης ακρίβειας των αναπαραστάσεων που χρησιμοποιούνται.

$\Rightarrow_L$ -ID-(ID*ID/*ID/ID-ID)*(*ID-E-E)*E-E $\Rightarrow_L$
$\Rightarrow_L$ -ID-(ID*ID/*ID/ID-ID)*(*ID-E*E-E)*E-E $\Rightarrow_L$
$\Rightarrow_L$ -ID-(ID*ID/*ID/ID-ID)*(*ID-ID*E-E)*E-E $\Rightarrow_L$
$\Rightarrow_L$ -ID-(ID*ID/*ID/ID-ID)*(*ID-ID**E-E)*E-E $\Rightarrow_L$
$\Rightarrow_L$ -ID-(ID*ID/*ID/ID-ID)*(*ID-ID**ID-E)*E-E $\Rightarrow_L$
$\Rightarrow_L$ -ID-(ID*ID/*ID/ID-ID)*(*ID-ID**ID-ID)*E-E $\Rightarrow_L$
$\Rightarrow_L$ -ID-(ID*ID/*ID/ID-ID)*(*ID-ID**ID-ID)*ID-E $\Rightarrow$
$\Rightarrow$ -ID-(ID*ID/*ID/ID-ID)*(*ID-ID**ID-ID)*ID-(E) $\Rightarrow$
$\Rightarrow$ -ID-(ID*ID/*ID/ID-ID)*(*ID-ID**ID-ID)*ID-(E+E) $\Rightarrow_L$
$\Rightarrow_L$ -ID-(ID*ID/*ID/ID-ID)*(*ID-ID**ID-ID)*ID-(ID+E) $\Rightarrow$
$\Rightarrow$ -ID-(ID*ID/*ID/ID-ID)*(*ID-ID**ID-ID)*ID-(ID+*E) $\Rightarrow$
$\Rightarrow$ -ID-(ID*ID/*ID/ID-ID)*(*ID-ID**ID-ID)*ID-(ID+*ID)

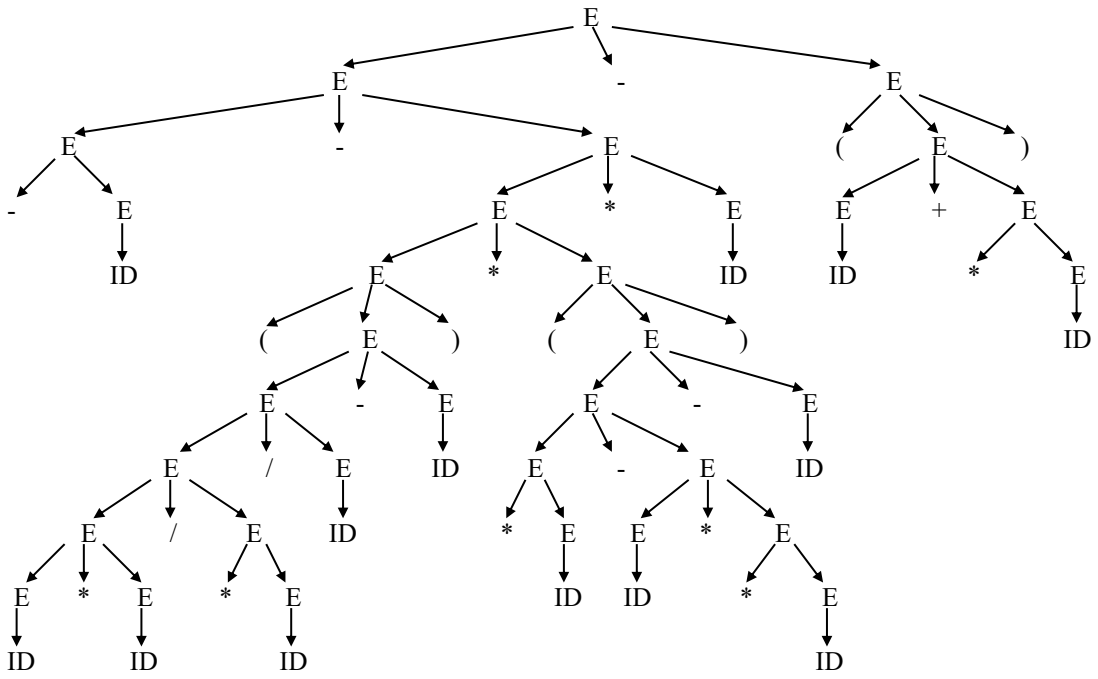
Παρατηρήστε ότι σύμφωνα με όσα είπαμε παραπάνω, η προτεραιότητα των τελεστών μας υποχρεώνει να εφαρμόζουμε πρώτα κανόνες που αντιστοιχούν σε πράξεις χαμηλότερης προτεραιότητας, και ύστερα κανόνες που αντιστοιχούν σε πράξεις υψηλότερης προτεραιότητας, σε περιπτώσεις που αν δεν είχαμε ορισμένη προτεραιότητα, θα μας έδιναν το δικαίωμα επιλογής και κατά συνέπεια παραπάνω από μία αριστερότερες παραγωγές. Για παράδειγμα, τα τρία πρώτα βήματα της παραπάνω παραγωγής γίνονται επειδή η αφαίρεση και το πρόσημο έχουν χαμηλότερη προτεραιότητα από τον πολλαπλασιασμό, ενώ στο 14ο βήμα η αποδεικτοδότηση εφαρμόζεται, αφού έχουν εφαρμοστεί η αφαίρεση, ο πολλαπλασιασμός και οι διαιρέσεις που βρίσκονται μέσα στην πρώτη παρένθεση.
Η αριστερή προσεταιριστικότητα, σε συνδυασμό με το γεγονός ότι παίρνουμε την αριστερότερη παραγωγή, μας υποχρεώνει να εφαρμόσουμε όλους τους κανόνες που αντιστοιχούν σε πράξεις ίσης προτεραιότητας *από τα δεξιά προς τα αριστερά*, πριν προχωρήσουμε σε κανόνες που αντιστοιχούν σε πράξεις υψηλότερης προτεραιότητας, σε περιπτώσεις διαδοχικής εφαρμογής τελεστών ίσης προτεραιότητας. Για παράδειγμα, το πρώτο βήμα της πιο πάνω παραγωγής αντιστοιχεί στη δεξιά, το δεύτερο αντιστοιχεί στην αριστερή αφαίρεση, και το τρίτο αντιστοιχεί στο αρνητικό πρόσημο της παράστασης έξω από τις παρενθέσεις. Παρόμοια, στα βήματα 9-11 εφαρμόζουμε πρώτα τις διαιρέσεις και ύστερα τον πολλαπλασιασμό που βρίσκεται αριστερά των διαιρέσεων, για την παράσταση μέσα στο πρώτο ζεύγος παρενθέσεων.
Αν πάντως επιθυμούσαμε τη δεξιότερη παραγωγή, τότε με αριστερή προσεταιριστικότητα θα εφαρμόζαμε κανόνες που να αντιστοιχούν σε πράξεις υψηλότερης προτεραιότητας, πριν συνεχίσουμε με κανόνες που να αντιστοιχούν σε πράξεις ίσης προτεραιότητας. Τέλος, για τελεστές με δεξιά προσεταιριστικότητα, θα ενεργούσαμε αντίστροφα.

Το *μοναδικό* ΔΣΑ της δεδομένης συμβολοσειράς που αντιστοιχεί στην πιο πάνω παραγωγή δίνεται στην επόμενη σελίδα.

Β. Αν δεν είχαμε τους τελεστές προσήμου, η προτεραιότητα και η προσεταιριστικότητα των τελεστών της γραμματικής θα μας οδηγούσαν εύκολα στη νέα ισοδύναμη γραμματική:

E → E + T | E – T | T
T → T * F | T / F | F
F → * F | (E) | ID

όπου τα τρία επίπεδα προτεραιότητας οδήγησαν στο διαχωρισμό του ενός αρχικού επιπέδου κανόνων σε τρία, που το καθένα αντιστοιχεί σε ξεχωριστό επίπεδο προτεραιότητας. Η αριστερή προσεταιριστικότητα των τελεστών με δύο τελούμενα εισόδου μεταφράστηκε σε αριστερή αναδρομή στους κανόνες. Έτσι, όπως αναλύσαμε και πιο πάνω, οι κανόνες με τελεστές χαμηλότερης προτεραιότητας χρησιμοποιούνται σε μια παραγωγή πριν από εκείνους με τελεστές υψηλότερης προτεραιότητας, ενώ για διαδοχική εφαρμογή τελεστών ίσης προτεραιότητας, εκείνοι που είναι δεξιότερα παράγονται πρώτοι από τους κανόνες τους, ώστε ο τελικός κώδικας να παραχθεί με την αντίστροφη, σωστή σειρά. Η παραπάνω γραμματική δεν είναι διφορούμενη, εφόσον δεν υπάρχει τρόπος να πάρουμε την ίδια συμβολοσειρά με περισσότερα από ένα ΔΣΑ.



Δέντρο συντακτικής ανάλυσης για τη συμβολοσειρά
"`-ID-(ID*ID/*ID/ID-ID)*(*ID-ID**ID-ID)*ID-(ID+*ID)`"

Αν τώρα εισάγουμε τελεστές προσήμου με ίση προτεραιότητα με τον τελεστή αποδεικτοδότησης και αριστερή προσεταιριστικότητα, τότε η γραμματική διαμορφώνεται ως εξής:

E → E + T | E – T | T
T → T * F | T / F | F
F → + F | – F | * F | (E) | ID

Η γραμματική αυτή είναι ισοδύναμη της αρχικής, με την έννοια ότι παράγει τις ίδιες ακριβώς συμβολοσειρές, και δεν είναι διφορούμενη.
Αν όμως θελήσουμε οι τελεστές προσήμου να έχουν την προτεραιότητα των τελεστών πρόσθεσης/αφαίρεσης, τότε, σε περίπτωση που τοποθετήσουμε τους κανόνες προσήμου στο πρώτο επίπεδο:

E → + E | – E | E + T | E – T | T

η γραμματική θα είναι διφορούμενη, αφού το πρόσημο μπορεί να παραχθεί είτε πριν είτε μετά την υποέκφραση πρόσθεσης/αφαίρεσης. Επομένως, θα πρέπει οι κανόνες προσήμου να τοποθετηθούν σε ένα νέο επίπεδο κάτω από αυτό των κανόνων πρόσθεσης/αφαίρεσης, αλλά πάνω από εκείνο των κανόνων πολλαπλασιασμού/διαίρεσης. Έτσι, διατηρείται η προτεραιότητα χαμηλότερη από αυτή των τελεστών πολλαπλασιασμού/διαίρεσης και αποδεικτοδότησης, και επιβάλλεται η αριστερή προσεταιριστικότητα. Τα τρία πρώτα επίπεδα θα διαμορφωθούν:

E → E + T' | E – T' | T'
T' → + T' | – T' | T
T → T * F | T / F | F

Στη συνέχεια όμως, αν διατηρήσουμε κανόνες προσήμου και στο τελευταίο επίπεδο, η γραμματική θα είναι πάλι διφορούμενη, αλλά αν τους αφαιρέσουμε, δε θα παράγονται συμβολοσειρές στις οποίες το πρόσημο – ή μια ακολουθία προσήμων – ακολουθεί τελεστή πολλαπλασιασμού/διαίρεσης ή αποδεικτοδότησης. Διαχωρίζουμε έτσι αυτή την περίπτωση, εισάγοντας ένα σύμβολο που να αντιστοιχεί σε υποέκφραση που ακολουθεί τέτοιον τελεστή, και αφαιρούμε τους κανόνες προσήμου από τη γενική περίπτωση. Καταλήγουμε έτσι στη μη διφορούμενη ισοδύναμη γραμματική:

E → E + T' | E – T' | T'
T' → + T' | – T' | T
T → T * F' | T / F' | F
F' → + F' | – F' | * F' | (E) | ID
F → * F' | (E) | ID

όπου οι τελεστές προσήμου έχουν την προτεραιότητα των τελεστών πρόσθεσης/αφαίρεσης. Αν το πρόσημο – ή η ακολουθία προσήμων – ακολουθεί τελεστή πολλαπλασιασμού/διαίρεσης ή αποδεικτοδότησης, θα παράγεται από το σύμβολο F', διαφορετικά από το σύμβολο T'.

Άσκηση 2-4:

Να δοθεί (μη διφορούμενη) γραμματική χωρίς συμφραζόμενα που να παράγει τις συμβολοσειρές:
$\{1^n0^n\}$, *όπου* $n \geq 0$

Στη συνέχεια να κατασκευάσετε ένα ΑΣ που να αναγνωρίζει τις συμβολοσειρές αυτές, και να δείξετε τις κινήσεις του στις συμβολοσειρές "111000" και "110".

Απάντηση

Η ζητούμενη γραμματική πρέπει να περιέχει κάποιο συντακτικό κανόνα που παράγει ταυτόχρονα ένα 1 στα αριστερά και ένα 0 στα δεξιά του δεξιού του μέλους:

A → 1B0

Έτσι μόνο θα μπορεί να παράγει συμβολοσειρές με τον ίδιο αριθμό 1 και 0, ώστε τα 1 να προηγούνται των 0.
Ο κανόνας αυτός πρέπει ακόμα να είναι αναδρομικός, αφού ο αριθμός n των 1 και 0 είναι μεταβλητός:

A → 1A0

επειδή δε μεταξύ των 1 και 0 στις συμβολοσειρές που παράγονται δεν παρεμβάλλεται άλλο σύμβολο, το A μπορεί να παράγει εναλλακτικά την κενή συμβολοσειρά:

A → ε

Ακόμα, επειδή πριν του πρώτου 1 και μετά το τελευταίο 0 δεν υπάρχει άλλο σύμβολο, το A μπορεί να είναι το αρχικό σύμβολο της γραμματικής, γεγονός που ικανοποιεί και την απαίτηση η τιμή του n να μπορεί να είναι 0.

Σύμφωνα με τα παραπάνω, μια μη διφορούμενη γραμματική χωρίς συμφραζόμενα για την παραγωγή των συμβολοσειρών της άσκησης είναι η G = (T, N, P, S) όπου:

T = {0, 1}
N = {S}
P = {S → 1S0 S → ε}

Ένα ΑΣ που μπορεί να χρησιμοποιηθεί για την αναγνώριση των συμβολοσειρών της G θα εισάγει στη στοίβα του όσα σύμβολα 1 βλέπει στην είσοδό του, και στη συνέχεια, για κάθε είσοδο 0, θα αφαιρεί ένα 1 από αυτήν. Αναγνώριση θα μπορεί να γίνεται όταν η στοίβα περιέχει το αρχικό σύμβολο στοίβας.
Έτσι, ένα τέτοιο ΑΣ είναι το M = (T, Q, H, δ, q_0, h_0, F) όπου:

T = {0, 1}
Q = {S, A, Z}
H = {X, 1}
q_0 = S
h_0 = X
F = {Z}

και η συνάρτηση μετάβασης περιγράφεται από το ακόλουθο πίνακα μετάβασης:

	X	1	⊣
S	read(1) ⇒ push(1) keep ⇒ move(Z)	read(1) ⇒ push(1) read(0) ⇒ pop, move(A)	
A	keep ⇒ move(Z)	read(0) ⇒ pop	
Z			accept

Παρατηρήστε ότι η κατάσταση A είναι απαραίτητη για να υποδεικνύει ότι έχει ολοκληρωθεί η είσοδος των 1. Έτσι αποφεύγεται η αναγνώριση συμβολοσειρών που έχουν απλά *ταιριασμένα* 1 και 0, όπως για παράδειγμα η "11101000".

Για την πρώτη από τις δύο συμβολοσειρές που δίνονται, οι διαδοχικές κινήσεις του M φαίνονται παρακάτω. Στην περίπτωση αυτή η συμβολοσειρά θα αναγνωριστεί.

κίνηση	στοίβα	είσοδος	κατάσταση	κίνηση που ακολουθεί
0	X	111000	S	read(1) ⇒ push(1)
1	X1	11000	S	read(1) ⇒ push(1)
2	X11	1000	S	read(1) ⇒ push(1)
3	X111	000	S	read(0) ⇒ pop, move(A)
4	X11	00	A	read(0) ⇒ pop
5	X1	0	A	read(0) ⇒ pop
6	X	ε	A	keep ⇒ move(Z)
7	X	ε	Z	αποδοχή

Για τη δεύτερη συμβολοσειρά, οι κινήσεις του M ακολουθούν, ενώ αυτή θα απορριφτεί.

κίνηση	στοίβα	είσοδος	κατάσταση	κίνηση που ακολουθεί
0	X	110	S	read(1) ⇒ push(1)
1	X1	10	S	read(1) ⇒ push(1)
2	X11	0	S	read(0) ⇒ pop, move(A)
3	X1	ε	A	οπισθοδρόμηση
4	X	110	S	keep ⇒ move(Z)
5	X	110	Z	απόρριψη

Παρατηρήστε ότι το ΑΣ είναι μη ντετερμινιστικό λόγω της ε-μετάβασης στο κελί (S,X) του πίνακα μεταβάσεων. Έτσι, για να απορρίψουμε οριστικά τη συμβολοσειρά "110", πρέπει να ελέγξουμε και τις δύο επιλογές που έχουμε στο συγκεκριμένο κελί, οπότε μόλις η μία αποτύχει, οπισθοδρομούμε και δοκιμάζουμε και τη δεύτερη.