ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

« Σύστημα για Επεξεργασία Λογικών Εκφράσεων »

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σπουδαστές:
Πατσάκης Νικόλαος ΑΜ: 1757
Παπαδάκης Γεώργιος ΑΜ: 3179

Επιβλέπων Καθηγητής :
Παπαδάκης Νικόλαος, Επίκουρος Καθηγητής

Ηράκλειο 2014

Πίνακας περιεχομένων

1 Περίληψη 5

2 Abstract 6

3 Εισαγωγή – Λογικές Πύλες 7

4 Λογικές Προτάσεις 14

5 Λογικοί Σύνδεσμοι 15

6 Ιδιότητες Λογικών Συνδέσμων 17

7 Συνέπεια (Consistency) 18

8 Συνέπεια και Συνεπαγωγή (Consequence and Entailment) 20

9 Εξαγωγή Συμπερασμάτων, Εγκυρότητα, Συνεπαγωγή, Ισοδυναμία (Inference, Validity, Entailment, Equivalence) 21

10 Ισοδυναμία 23

11 Προτασιακός Λογισμός (Propositional Calculus) 24

11.1 Δομή Προτάσεων 25

11.2 Ισοδυναμίες του Προτασιακού Λογισμού 28

11.3 Κανονικές Μορφές (Normal Forms) 29

11.4 Αλγόριθμος Μετατροπής σε DNF 30

11.5 Άρνηση (Negation) 33

11.6 Συνεπαγωγή στον Προτασιακό Λογισμό 36

11.7 Ταυτολογίες και Αντινομίες (Tautologies and Contradictions) 37

11.8 Ταυτολογίες και Λογική Συνεπαγωγή 39

11.9 Συνεπαγωγή και Ισοδυναμία (Material Implication and Equivalence) .. 41

11.10 Κανονικές Μορφές 44

11.11 Μετατροπή Προτάσεων του Π.Λ. σε DNF 46

11.12 Αλγόριθμος Μετατροπής Προτάσεων του Π.Λ. σε DNF 47

12 Η Γλώσσα C 49
13 Βιβλιογραφία 54
14 Παράρτημα 55

Πίνακας Εικόνων

Εικόνα 1: Σχεδιάγραμμα Πύλης NOT και Πίνακας Αληθείας 8
Εικόνα 2: Σχεδιάγραμμα Πύλης AND και Πίνακας Αληθείας..... 9
Εικόνα 3: Σχεδιάγραμμα Πύλης OR και Πίνακας Αληθείας..... 9
Εικόνα 4: Σχεδιάγραμμα Πύλης NAND και Πίνακας Αληθείας..... 10
Εικόνα 5: Σχεδιάγραμμα Πύλης NOR και Πίνακας Αληθείας..... 11
Εικόνα 6: Σχεδιάγραμμα Πύλης XOR και Πίνακας Αληθείας..... 11
Εικόνα 7: Σχεδιάγραμμα Πύλης XNOR και Πίνακας Αληθείας..... 12
Εικόνα 8: Δημοτικότητα Γλωσσών..... 50
Εικόνα 9: Hello World - το Πρώτο Πρόγραμμα κάθε Προγραμματιστή !..... 51
Εικόνα 10: Ιστορική Αναδρομή Γλωσσών Προγραμματισμού..... 53

1 Περίληψη

Η πτυχιακή εργασία μας σχετίζεται με λογικές πράξεις, πύλες και πίνακες αληθείας. Εξετάζουμε τις λογικές πράξεις: της Σύζευξης (∧), της Άρνησης (¬), της Διάζευξης (∨) και της Συνεπαγωγής (→), οι οποίες μας δίνουν ένα λογικό αποτέλεσμα (0 ή 1). Στον κώδικά μας χρησιμοποιούμε το πολύ πέντε (5) μεταβλητές. Επίσης αναφερόμαστε στον Προτασιακό Λογισμό, στους αλγόριθμους DNF, CNF και στη μετατροπή προτάσεων σε αυτούς. Η γλώσσα προγραμματισμού C έχει πολλές δυνατότητες, γι' αυτόν το λόγο έχουμε επιλέξει να γράψουμε τον κώδικάς μας με αυτήν.

2 Abstract

Our thesis relates to logical operations, gates and truth tables. We examine the logical operations of: conjunction (∧), negation (¬), disjunction (∨) and implication (→), which give us a logical result (0 or 1). In our code we use a maximum of five (5) variables. Also, refer to propositional calculus, the algorithms DNF, CNF and the conversion of proposals to them. The C programming language has many features, for this reason we have chosen to write down our code with it.

3 Εισαγωγή – Λογικές Πύλες

Τα ψηφιακά κυκλώματα αποτελούνται από λογικές πύλες, δηλαδή στοιχειώδη λογικά κυκλώματα τα οποία πραγματοποιούν τις λογικές πράξεις της άλγεβρας Boole, δηλαδή του πολλαπλασιασμού, της πρόσθεσης και του συμπληρώματος. Οι λογικές πύλες είναι λογικά κυκλώματα με πολλές εισόδους αλλά μια έξοδο.

Λογικές πράξεις και Δυαδική λογική

Πολλές φορές στα μαθηματικά και γενικότερα στις επιστήμες, έχουμε την ανάγκη να χρησιμοποιήσουμε το δυαδικό σύστημα αρίθμησης για τις εφαρμογές που έχουμε να επιλύσουμε.
Το δυαδικό σύστημα αρίθμησης περιέχει τους αριθμούς 0 (λογικό μηδέν) και 1 (λογικό ένα). Υποστηρίζει τις δύο από τις τέσσερις βασικές πράξεις, δηλαδή την πρόσθεση (Λογικό Ή (OR)) και τον πολλαπλασιασμό (λογικό ΚΑΙ (AND)). Επίσης ισχύει η έννοια του αντίστροφου αριθμού (λογικό ΟΧΙ (NOT)) και αυτό σημαίνει ότι το αντίθετο της μίας λογικής κατάστασης (είτε λογικό 0 είτε λογικό 1) είναι η άλλη κατάσταση. Για παράδειγμα, αν έχουμε το λογικό 0 μετά την λογική πράξη ΟΧΙ, το αποτέλεσμα θα είναι το λογικό 1 και αντίστροφα. Έτσι ορίζουμε άλλες δύο λογικές πράξεις (logical operations) οι οποίες είναι, η λογική αντίστροφη πρόσθεση (NOT-OR (ή συντομογραφικά NOR)) και τον λογικό αντίστροφο πολλαπλασιασμό (NOT-AND (ή συντομογραφικά NAND)).

Στο δυαδικό σύστημα (και στις πράξεις) ισχύουν τα εξής :

Όπου $\wedge$ το σύμβολο της πράξης AND	Όπου $\vee$ το σύμβολο της πράξης OR	Όπου $\neg$ το σύμβολο της πράξης NOT
$0 \wedge 0 = 0$	$0 \vee 0 = 0$	$\neg 0 = 1$
$0 \wedge 1 = 0$	$0 \vee 1 = 1$	$\neg 1 = 0$
$1 \wedge 1 = 1$	$1 \vee 1 = 1$	

Στην δυαδική λογική, υπάρχουν δύο θεωρήματα που τα απέδειξε ο μαθηματικός Augustus de Morgan τα οποία είναι τα εξής (όπου ’ εκφράζει την λογική πράξη NOT (ΟΧΙ)) :

a) (x ∧ y)’ = x’ ∨ y’
b) (x ∨ y)’ = x’ ∧ y’

Έτσι, συνδυάζοντας τις τρείς βασικές λογικές πράξεις (NOT, AND, OR) με τα θεωρήματα de Morgan, μπορούμε να εισάγουμε και να ορίσουμε τις δύο τελευταίες λογικές πράξεις. Την αποκλειστική πρόσθεση (exclusive-OR (ή συντομογραφικά XOR)) και την αντίστροφη αποκλειστική πρόσθεση (not exclusive-OR (ή συντομογραφικά XNOR)).

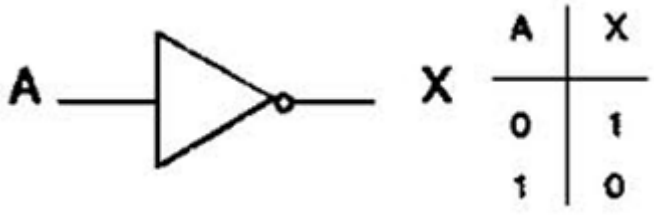
Τέλος πρέπει να αναφέρουμε ότι η προτεραιότητα των πράξεων στο δυαδικό σύστημα είναι η εξής :

a) Λογικό ΟΧΙ (NOT)
b) Λογικό ΚΑΙ (AND)
c) Λογικό Ή (OR). [1]

<u>Οι κύριες πύλες είναι οι εξής</u> :

Λογικό ΟΧΙ (NOT) :

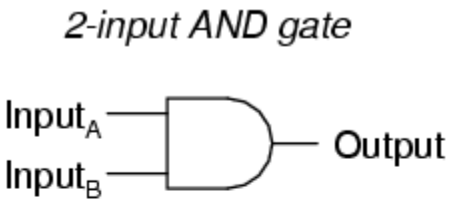
Η πρώτη λογική πράξη είναι η πράξη της αντιστροφής. Την εκφράζουμε με την λογική πύλη NOT και με αυτή μπορούμε να αντιστρέψουμε το αποτέλεσμα της εξόδου. Δηλαδή το λογικό 1 να το κάνουμε λογικό 0 και αντίστροφα. Παρακάτω βλέπουμε ότι η λογική πύλη έχει μία είσοδο (στην οποία εισάγουμε δυαδικό αριθμό (0 ή 1)) και στην έξοδο παίρνουμε το αντίστροφο από αυτό που εισάγουμε.



Εικόνα 1: Σχεδιάγραμμα Πύλης NOT και Πίνακας Αληθείας

Λογική Πύλη AND (ΚΑΙ) :

Με την πύλη AND (ΚΑΙ) πραγματοποιούμε τη λογική πράξη του πολλαπλασιασμού δύο ή περισσότερων μεταβλητών και εκφράζεται με την συνάρτηση : Output = A · B. Όπου Output είναι η έξοδος και A, B είναι οι είσοδοι.

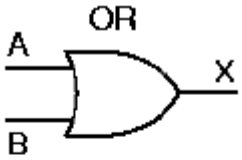


A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Εικόνα 2: Σχεδιάγραμμα Πύλης AND και Πίνακας Αληθείας

Λογική πύλη OR (Ή) :

Με την πύλη OR (Ή) πραγματοποιούμε τη λογική πράξη της πρόσθεσης και εκφράζεται από την συνάρτηση : X = A + B



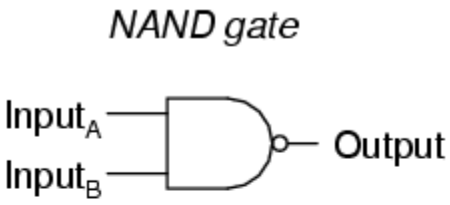
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Εικόνα 3: Σχεδιάγραμμα Πύλης OR και Πίνακας Αληθείας

Οι συμπληρωματικές πύλες είναι οι εξής :

Λογική πύλη NAND (NOT AND) :

Η λογική πύλη NAND είναι το συμπλήρωμα της λογικής πύλης AND. Μπορούμε να την δημιουργήσουμε εάν συνδέσουμε σε σειρά, μια πύλη AND και μια πύλη NOT. Την συμβολίζουμε όπως την πύλη AND με έναν κύκλο στο άκρο της, που δηλώνει την άρνηση. Η λογική συνάρτηση της πύλης NAND είναι : Output = (A · B)'

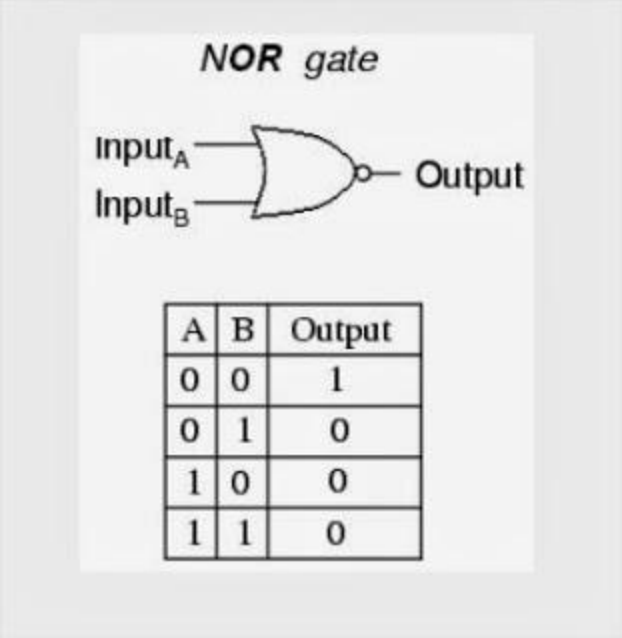


A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

Εικόνα 4: Σχεδιάγραμμα Πύλης NAND και Πίνακας Αληθείας

Λογική πράξη NOR (NOT OR) :

Η λογική πράξη NOR είναι ουσιαστικά η πράξη OR ακολουθούμενη από μια πύλη NOT στην έξοδό της. Πρακτικά όμως και σε αυτή την περίπτωση υπάρχει η πύλη NOR για να ελαχιστοποιήσουμε τις πύλες σε ένα κύκλωμα.

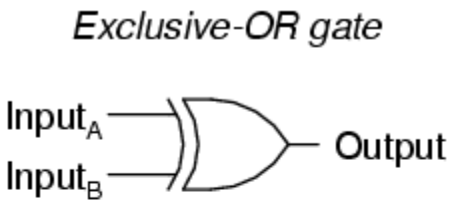


A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

Εικόνα 5: Σχεδιάγραμμα Πύλης NOR και Πίνακας Αληθείας

Αποκλειστικό Ή (XOR) :

Η λογική πύλη XOR προκύπτει από τον συνδυασμό των λογικών πράξεων AND, OR και NOT. Η λογική συνάρτηση της πύλης XOR είναι : $x \oplus y = \overline{x} \cdot y + x \cdot \overline{y}$

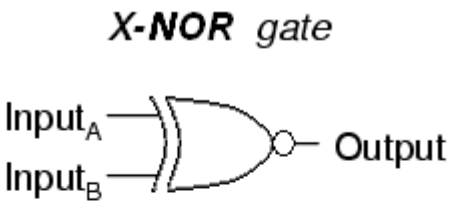


A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

Εικόνα 6: Σχεδιάγραμμα Πύλης XOR και Πίνακας Αληθείας

Αποκλειστικό NOR (XNOR) :

Η τελευταία λογική πράξη είναι η αντίστροφη αποκλειστική πρόσθεση. Είναι η αντίθετη λογική πράξη από την XOR και προκύπτει κι αυτή μέσω των τριών βασικών λογικών πυλών AND, OR και NOT. Η λογική συνάρτηση της πύλης XNOR είναι : $x \odot y = \overline{x} \cdot \overline{y} + x \cdot y$



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Εικόνα 7: Σχεδιάγραμμα Πύλης XNOR και Πίνακας Αληθείας

Συνοψίζοντας, βλέπουμε ότι με συγκεκριμένα κυκλώματα μπορούμε να υλοποιήσουμε λογικές πράξεις. Όλες οι λογικές πύλες που είδαμε παραπάνω (εκτός της πύλης NOT) υπάρχουν και με περισσότερες εισόδους.

Είναι σημαντικό να αναφέρουμε ότι οι λογικές πύλες πωλούνται στο εμπόριο σε μορφή ολοκληρωμένων κυκλωμάτων (integrated circuits (chips)). Παρακάτω βλέπουμε τα βασικότερα ολοκληρωμένα κυκλώματα της σειράς 74 για τις λογικές πύλες που αναφέραμε παραπάνω :

- 7400 → Τέσσερις λογικές πύλες NAND 2 εισόδων.
- 7402 → Τέσσερις λογικές πύλες NOR 2 εισόδων.
- 7404 → Έξι λογικές πύλες NOT.
- 7408 → Τέσσερις λογικές πύλες AND 2 εισόδων.
- 7432 → Τέσσερις λογικές πύλες OR 2 εισόδων.
- 7486 → Τέσσερις λογικές πύλες XOR 2 εισόδων.
- 74266 → Τέσσερις λογικές πύλες XNOR 2 εισόδων.

Τα ολοκληρωμένα κυκλώματα ποτέ δεν περιέχουν μία μόνο λογική πύλη. Συνήθως περιέχουν τρείς ή τέσσερις πύλες, ή στην περίπτωση που έχουμε πύλη πολλών εισόδων μπορεί να περιέχουν και μόνο μία. Ενδεικτική περίπτωση ολοκληρωμένου κυκλώματος που περιέχει μόνο μία λογική πύλη, είναι το ολοκληρωμένο κύκλωμα με κωδικό 74LS30 που περιέχει μία πύλη NAND 8 εισόδων.

Επειδή ο τρόπος συνδεσμολογίας του κάθε ολοκληρωμένου κυκλώματος διαφέρει, γι' αυτό τον λόγο υπάρχουν τα λεγόμενα φύλλα δεδομένων (data sheets). Αυτά μπορούμε να τα βρούμε πολύ εύκολα στο διαδίκτυο, πληκτρολογώντας απλά τον κωδικό του ολοκληρωμένου κυκλώματος που θέλουμε να χρησιμοποιήσουμε. [1]

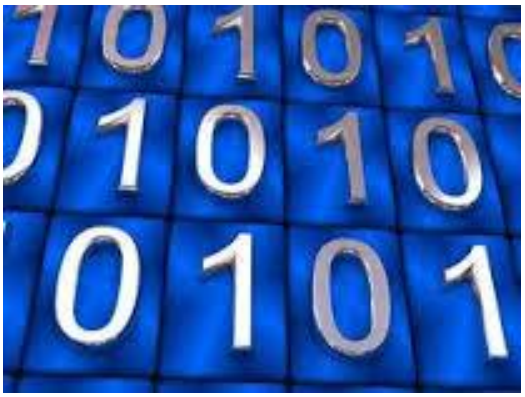
4 Λογικές Προτάσεις

Μια λογική πρόταση είναι μια γλωσσική έκφραση η οποία μπορεί να χαρακτηριστεί ως αληθής ή ως ψευδής. Υπάρχουν προτάσεις οι οποίες δεν είναι λογικές προτάσεις.

Τέτοιες, για παράδειγμα, είναι :

a) οι Ερωτηματικές προτάσεις
b) οι Προσταγές ή Ευχετικές προτάσεις
c) Μια πρόταση μπορεί να μην είναι λογική πρόταση επειδή κάποιες από τις λέξεις δεν έχουν ακριβή σημασιολογία. Π.χ., η πρόταση « ο Χ είναι φοιτητής του τμήματος ». Η πρόταση μπορεί να χαρακτηριστεί ως αληθής ή ψευδής αν γνωρίζουμε ποιος είναι ο Χ και ποιο είναι το τμήμα. Απαιτείτε δηλαδή η απόδοση ερμηνείας στους όρους αυτούς.

Στη λογική, μας ενδιαφέρουν οι προτάσεις για τις οποίες υπάρχει ερμηνεία που να τις καθιστά αληθείς ή ψευδείς.



5 Λογικοί Σύνδεσμοι

Οι λογικές προτάσεις σχηματίζονται με τη χρήση λογικών συνδέσμων :

a) **Σύζευξη** (conjunction). Συμβολίζεται με το σύμβολο $\wedge$
Παράδειγμα: Έστω ότι γνωρίζουμε δύο ιδιότητες ενός ακεραίου αριθμού x: $x > 4$ και $x < 9$. Τότε για το x γνωρίζουμε τις προτάσεις Α: $x > 4$ και Β: $x < 9$. Τότε όμως γνωρίζουμε και την πρόταση: ο x είναι μεγαλύτερος του 4 και (ο x είναι) μικρότερος του 9, δηλαδή τη σύζευξη των Α και Β. Έτσι, $A \wedge B$ δηλώνει την πρόταση « $x > 4$ και $x < 9$ » (ή όπως γράφεται εν συντομία « $4 < x < 9$ »).

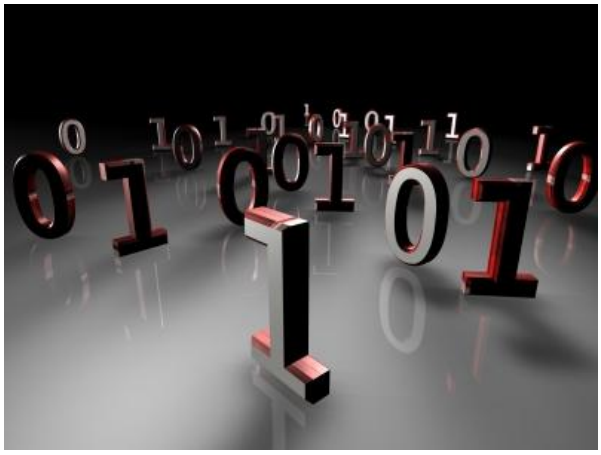
b) **Άρνηση** (negation). Συμβολίζεται με το σύμβολο $\neg$
Παράδειγμα: Έστω η πρόταση Γ: « το 50 είναι διαιρετό δια 7 ». Τότε η άρνηση της Γ ($\neg\Gamma$) είναι η πρόταση « το 50 δεν είναι διαιρετό δια 7 ».

c) **Διάζευξη** (disjunction). Συμβολίζεται με το σύμβολο $\vee$
Παράδειγμα: Έστω οι προτάσεις Δ: « το 60 είναι πολλαπλάσιο του 6 » και Ε: « το 60 είναι πολλαπλάσιο του 5 ». Τότε η διάζευξή των Δ και Ε είναι η πρόταση $\Delta \vee E$: « το 60 είναι πολλαπλάσιο του 6 ή πολλαπλάσιο του 5 ». Όπως βλέπουμε από το παράδειγμα, η διάζευξη δεν είναι αποκλειστική γιατί το 60 είναι πολλαπλάσιο και του 6 και του 5.

d) **Συνεπαγωγή (**implication**).** Συμβολίζεται με το σύμβολο $\rightarrow$
Παράδειγμα: Έστω οι προτάσεις Ζ: « ο αριθμός α είναι πολλαπλάσιο του 10 » και Η: « ο αριθμός α είναι πολλαπλάσιο του 5 ». Τότε η πρόταση $Z \rightarrow H$ (αν Ζ, τότε Η) είναι η πρόταση: « αν ο αριθμός α είναι πολλαπλάσιο του 10, τότε ο αριθμός α είναι πολλαπλάσιο του 5 ».

e) **Ισοδυναμία** (equivalence). Συμβολίζεται με το σύμβολο $\leftrightarrow$
Παράδειγμα: Έστω οι προτάσεις Θ: « το 16 είναι πολλαπλάσιο του 2 » και Κ: « το 16 είναι άρτιος αριθμός ». Τότε η πρόταση $\Theta \leftrightarrow K$ (Θ αν και μόνο αν Κ

) είναι η πρόταση « το 16 είναι πολλαπλάσιο του 2 αν και μόνο αν το 16 είναι άρτιος αριθμός ».



6 Ιδιότητες Λογικών Συνδέσμων

Για τους λογικούς συνδέσμους που ορίσαμε παραπάνω, ισχύουν οι ακόλουθες ιδιότητες:

a) Οι σύνδεσμοι $\wedge$ και $\vee$ είναι μεταθετικοί, δηλαδή, για οποιεσδήποτε προτάσεις Α και Β, οι προτάσεις $A \wedge B$ και $A \vee B$ είναι ισοδύναμες με τις προτάσεις $B \wedge A$ και $B \vee A$ αντίστοιχα.

b) Ο σύνδεσμος $\rightarrow$ δεν είναι μεταθετικός, δηλαδή, η πρόταση $A \rightarrow B$ δεν είναι ισοδύναμη με την πρόταση $B \rightarrow A$.
Παράδειγμα: Ενώ η πρόταση « αν ο α είναι πολλαπλάσιο του 10, τότε ο α είναι πολλαπλάσιο του 5 » είναι λογικά αληθής. Η πρόταση « αν ο α είναι πολλαπλάσιο του 5, τότε ο α είναι πολλαπλάσιο του 10 » δεν είναι.

Προτάσεις που συντίθενται με χρήση λογικών συνδέσμων μπορεί να είναι αληθείς, ψευδής, ή άλλοτε αληθείς και άλλοτε ψευδείς. Για παράδειγμα:

a) η πρόταση « ένας ρητός αριθμός είναι 0 ή διάφορος από το 0 » είναι αληθής
b) η πρόταση « αν $x = 3$, τότε $x = 5$ » είναι ψευδής
c) η πρόταση « το x ισούται με την απόλυτη τιμή του και το y είναι αρνητικός αριθμός » είναι άλλοτε αληθής και άλλοτε ψευδής.

Πρέπει να τυποποιηθεί η απόφαση μιας τιμής αληθείας σε προτάσεις βάσει της τιμής αληθείας των υποπροτάσεών τους. Για παράδειγμα, διαισθητικά γνωρίζουμε ότι η πρόταση « $x > 3 \rightarrow x > 0$ » είναι αληθής. Ποιος κανόνας όμως μας επιτρέπει να το συμπεράνουμε;
Πρέπει να εξεταστεί η σύνταξη των προτάσεων αλλά και οι κανόνες που καθορίζουν πότε και πώς μπορούμε να εξάγουμε έγκυρα συμπεράσματα από τα δεδομένα που διαθέτουμε.

7 Συνέπεια (Consistency)

Ένα σύνολο, το οποίο περιέχει προτάσεις οι οποίες δεν μπορούν να είναι ταυτόχρονα αληθείς λέγεται ασυνεπές (inconsistent).

Παραδείγματα:

a) Μια απλή περίπτωση, είναι όταν το σύνολο περιέχει προτάσεις οι οποίες αντικρούουν η μία την άλλη: S1 = { $x < 5$, $x \geq 5$ }, S2 = { $y = 3$, $y \neq 3$ }.
b) Μια άλλη περίπτωση, είναι όταν το σύνολο περιέχει προτάσεις οι οποίες αντικρούουν η μία την άλλη έμμεσα: S3 = { ο x είναι άρτιος αριθμός μεγαλύτερος του 2, ο x είναι πρώτος αριθμός }. Η ασυνέπεια προκύπτει από την ερμηνεία που δίνουμε στους όρους « άρτιος αριθμός » και « πρώτος αριθμός ».
c) Ασυνέπεια μπορεί να προκύψει από τρεις ή περισσότερες προτάσεις για τις οποίες δεν υπάρχει ζεύγος αντικρουόμενων προτάσεων: S4 = { $x > 2$, x άρτιος, x πρώτος }, S5 = { $x > y$, $y > z$, $z > x$ }. Εδώ παρατηρούμε ότι αν αφαιρέσουμε οποιαδήποτε πρόταση από τα σύνολα S4 και S5, τα σύνολα δεν είναι πλέον ασυνεπή.

Ένα σύνολο προτάσεων λέγεται συνεπές όταν δεν είναι ασυνεπές. Το γεγονός ότι ένα σύνολο είναι συνεπές δε σημαίνει ότι και όλες οι προτάσεις που περιέχονται σε αυτό είναι αληθείς. Σημαίνει ότι ενδέχεται να είναι αληθείς. Για παράδειγμα, το σύνολο S6 = { $x > y$, $y > z$ } είναι συνεπές παρόλο που μπορούμε να βρούμε τιμές για τα x, y, z ώστε τουλάχιστον μια από τις προτάσεις να είναι ψευδής.
Ένα σύνολο που περιέχει μόνο μια πρόταση μπορεί επίσης να είναι ασυνεπές. Η πρόταση αυτή δεν μπορεί ποτέ να είναι λογικά αληθής. Για παράδειγμα, το σύνολο S7 = { το 2 είναι διαιρέτης κάθε περιττού αριθμού } είναι ασυνεπές.
Τέτοιες προτάσεις τις λέμε λογικά ψευδείς (ή ταυτολογικά ψευδής). Οποιοδήποτε ασυνεπές σύνολο μπορεί να παράγει μια λογικά ψευδή πρόταση. Αυτό γίνετε σχηματίζοντας τη σύζευξη των μελών του συνόλου. Για παράδειγμα, από το σύνολο S5 = { $x > y$, $y > z$, $z > x$ } σχηματίζουμε την πρόταση $(x > y) \wedge (y > z) \wedge (z > x)$ η οποία είναι λογικά ψευδής.

Μια πρόταση είναι λογικά αληθής αν η άρνησή της είναι λογικά ψευδής. Για παράδειγμα, η πρόταση « ο αριθμός 2 είναι άρτιος » είναι λογικά αληθής μια και η άρνησή της (« ο αριθμός 2 είναι περιττός ») είναι λογικά ψευδής.

Επίσης, υπάρχουν προτάσεις οι οποίες δεν μπορούν να χαρακτηριστούν ούτε λογικά ψευδείς, ούτε λογικά αληθείς. Αυτές τις ονομάζουμε επαληθεύσιμες (contingent). Για παράδειγμα, η πρόταση « ο αριθμός x είναι θετικός » είναι επαληθεύσιμη.

8 Συνέπεια και Συνεπαγωγή (Consequence and Entailment)

Θεωρούμε το σύνολο προτάσεων S4 = { p1 : x > 2, p2 : x άρτιος, p3 : x πρώτος }. Αν δεχτούμε τις προτάσεις p1 και p2 τότε συμπεραίνουμε την πρόταση p4 : ο x δεν είναι πρώτος, δηλαδή την άρνηση της p3. Αν οι προτάσεις p1 και p2 είναι αληθείς τότε και η p4 πρέπει να είναι αληθής. Σε αυτή την περίπτωση λέμε ότι η p4 συνάγεται από τις p1 και p2, ή ότι είναι συνέπεια των p1 και p2. Εναλλακτικά, λέμε ότι οι p1 και p2 συνεπάγονται την p4, ή ότι μπορούμε να συμπεράνουμε την p4 από τις p1 και p2. Επίσης, από τις p2 και p3 μπορούμε να συμπεράνουμε την πρόταση p5 : x = 2 και από τις p1 και p3 μπορούμε να συμπεράνουμε την πρόταση p6 : x περιττός.

9 Εξαγωγή Συμπερασμάτων, Εγκυρότητα, Συνεπαγωγή, Ισοδυναμία (Inference, Validity, Entailment, Equivalence)

Θα εξετάσουμε τη σχέση μεταξύ της εξαγωγής ενός συμπεράσματος και της έννοιας της συνεπαγωγής. Έχουμε ένα σύνολο προτάσεων { p1, p2, … , pn }, μπορούμε να εξάγουμε ως συμπέρασμα την πρόταση c αν η c είναι συνέπεια των p1, p2, … , pn , δηλαδή αν οι p1, p2, … , pn συνεπάγονται την c. Η εξαγωγή συμπεράσματος c από υποθέσεις p1, p2, … , pn συμβολίζεται ως p1, p2, … , pn / c. Εδώ πρέπει να προσέξουμε ότι, συμπεραίνουμε ότι η c είναι αληθής υποθέτοντας ότι οι p1, p2, … , pn είναι αληθείς και ότι συνεπάγονται την c.

Αν το συμπέρασμα c πράγματι συνάγεται από τις προτάσεις p1, p2, … , pn , τότε η εξαγωγή συμπεράσματος ονομάζεται έγκυρη (valid). Αν οι p1, p2, … , pn και c είναι αληθείς προτάσεις, τότε η εξαγωγή συμπεράσματος ονομάζεται ορθή (sound). Για να δηλώσουμε ότι η εξαγωγή συμπεράσματος p1, p2, … , pn / c είναι έγκυρη χρησιμοποιούμε το συμβολισμό p1, p2, … , pn | = c.

Συχνά μας ενδιαφέρει η έγκυρη εξαγωγή συμπερασμάτων τα οποία γνωρίζουμε ότι δεν είναι ορθά. Αυτό συμβαίνει όταν ενδιαφερόμαστε για συλλογισμούς του τύπου « αν ο κόσμος ήταν …, τότε … ».

Η εξαγωγή συμπεράσματος p1, p2, … , pn / c είναι έγκυρη εφόσον δεν είναι δυνατόν για τις p1, p2, … , pn να είναι συγχρόνως αληθείς και η c να είναι ψευδής. Είναι ορθή αν είναι έγκυρη και οι p1, p2, … , pn, c είναι όλες αληθείς.

Παραδείγματα :

a) Η εξαγωγή συμπεράσματος

p_1 : q > 2
p_2 : q άρτιος
---
c : q μη πρώτος

είναι έγκυρη αλλά μη-ορθή.

b) Η εξαγωγή συμπεράσματος

p_1 : 6 > 2
p_2 : 6 άρτιος
---
c : 6 μη πρώτος

είναι έγκυρη και ορθή.

c) Η εξαγωγή συμπεράσματος

p_1 : ο αριθμός 3 είναι πρώτος
p_2 : ο αριθμός 5 είναι πρώτος
---
c : όλοι οι περιττοί είναι πρώτοι

είναι μη-έγκυρη και μη-ορθή.

Από μια έγκυρη συνεπαγωγή p1, p2, … , pn | = c μπορούμε να εξάγουμε μια λογικά αληθή πρόταση. Αυτό το πετυχαίνουμε με το να σχηματίσουμε την πρόταση p1 ∧ p2 ∧ … ∧ pn → c.

10 Ισοδυναμία

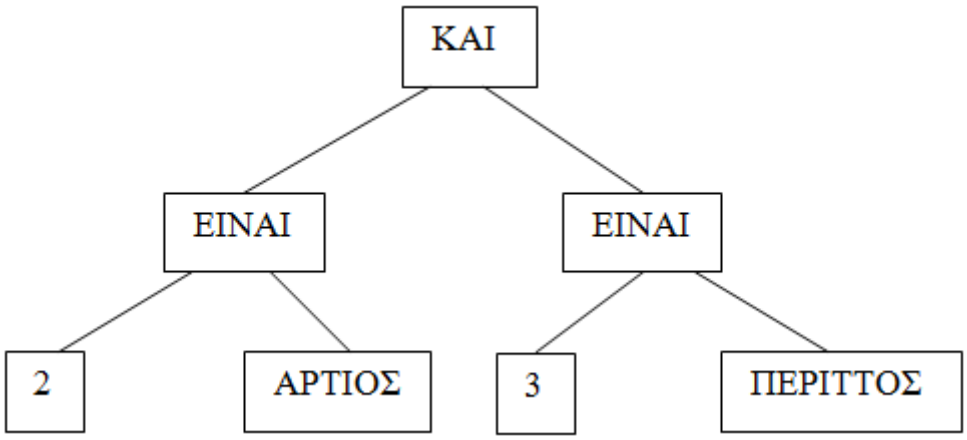
Αν για δύο προτάσεις Α και Β ισχύει ότι Α | = Β και Β | = Α, τότε οι προτάσεις Α και Β λέγονται ισοδύναμες. Δύο ισοδύναμες προτάσεις είναι συγχρόνως αληθείς ή συγχρόνως ψευδείς. Η ισοδυναμία των προτάσεων Α και Β συμβολίζεται ως Α ≡ Β.

Η σχέση Α | = Β δεν είναι συμμετρική, δηλαδή δεν ισχύει και Β | = Α. Για παράδειγμα, « ο x είναι πολλαπλάσιο του 4 | = ο x είναι άρτιος », αλλά « ο x είναι άρτιος | ≠ ο x είναι πολλαπλάσιο του 4.

Για άλλες προτάσεις ισχύει Α | = Β αν και μόνο αν Β | = Α. Για παράδειγμα, « ο x είναι πολλαπλάσιο του 2 | = ο x είναι άρτιος » και « ο x είναι άρτιος | = ο x είναι πολλαπλάσιο του 2 ».

11 Προτασιακός Λογισμός (Propositional Calculus)

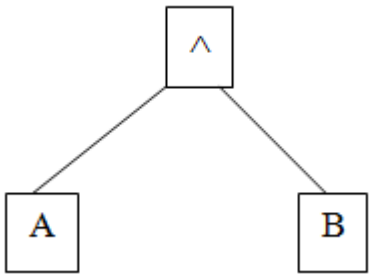
Ο Προτασιακός Λογισμός (ΠΛ) ασχολείται με τη δομή των προτάσεων και τη χρήση τους στην εξαγωγή συμπερασμάτων. Για παράδειγμα, θεωρούμαι την πρόταση s : « ο αριθμός 2 είναι άρτιος και ο αριθμός 3 είναι περιττός ». Την s μπορούμε να την θεωρήσουμε ως ερμηνεία της πρότασης « ο X είναι A και ο Y είναι B » ή ακόμα και της πρότασης « p και q ». Η δομή της πρότασης μπορεί να αναπαρασταθεί ως ένα δέντρο της μορφής :



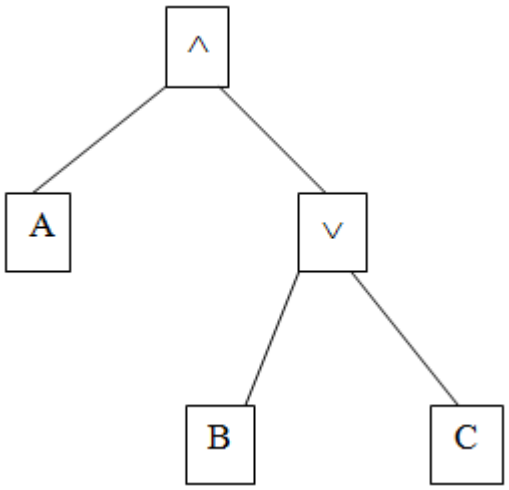
Αν η πρόταση θεωρηθεί ερμηνεία της πρότασης « p και q » τότε περιγράφεται σαν συνδυασμός δύο προτάσεων που συνδέονται με το συνδετικό « και » (σύζευξη). Η τιμή αληθείας της s εξαρτάται από τις τιμές αληθείας των p και q. Συνδετικά που χρησιμοποιούμαι για το σχηματισμό προτάσεων και για τα οποία ισχύει ότι η τιμή αληθείας της σύνθετης πρότασης εξαρτάται μόνο από τις τιμές αληθείας των προτάσεων που συντίθενται είναι αυτά που μας ενδιαφέρουν στον Προτασιακό Λογισμό. Αν το συνδετικό « διότι » ή το συνδετικό « ενώ » είχε χρησιμοποιηθεί στη θέση του συνδετικού « και », τότε αυτό δεν θα ίσχυε.

11.1 Δομή Προτάσεων

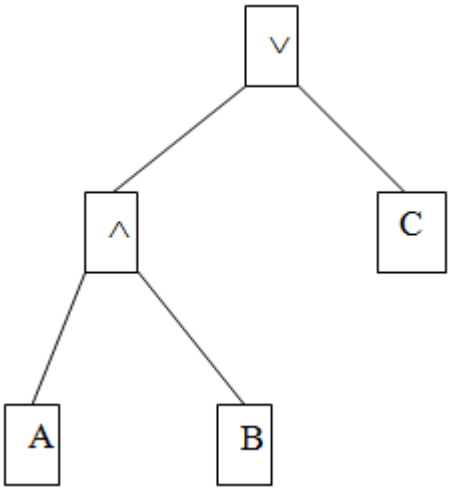
Η σύζευξη A ∧ B μπορεί να αναπαρασταθεί ως το δέντρο :



Οι A και B μπορεί να είναι σύνθετες προτάσεις. Έτσι η πρόταση A ∧ (B ∨ C) αναπαριστάται ως δέντρο :



Η χρήση των παρενθέσεων είναι απαραίτητη για να αναπαριστάται με ακρίβεια η δομή της πρότασης. Για παράδειγμα, η πρόταση A ∧ (B ∨ C) είναι διαφορετική από την πρόταση (A ∧ B) ∨ C. Η τελευταία αναπαριστάται από το δέντρο :



Πρέπει να χρησιμοποιήσουμε τις παρενθέσεις για να καταλάβουμε ποια δομή αντιστοιχεί στην πρόταση A ∧ B ∨ C. Η πρόταση A ∧ (B ∨ C) είναι μια σύζευξη προτάσεων μία από τις οποίες είναι μια διάζευξη, ενώ η πρόταση (A ∧ B) ∨ C είναι μια διάζευξη προτάσεων μία από τις οποίες είναι μία σύζευξη.

Κάθε συνδετικό έχει ένα πεδίο ή ακτίνα (scope). Το πεδίο ενός συνδετικού, είναι το σύνολο των κόμβων του δέντρου αναπαράστασης της πρότασης οι οποίοι « κρέμονται » από τον κόμβο του συνδετικού. Για παράδειγμα, το πεδίο του συνδετικού ∨ στην πρόταση A ∧ (B ∨ C) είναι το { B, C }, ενώ το πεδίο του συνδετικού ∧ είναι το { A, B ∨ C }.

Στις προτάσεις μπορούμε να κάνουμε αντικατάσταση (substitution) κάποιον συμβόλων με άλλες προτάσεις. Οποιαδήποτε σύζευξη (ή διάζευξη) μπορούμε να θεωρήσουμε ότι προέρχεται από την πρότυπη πρόταση p ∧ q (αντίστοιχα p ∨ q) όπου οι p, q αντικαθίστανται από τις κατάλληλες προτάσεις. Για παράδειγμα, η πρόταση (A ∨ (B ∧ C)) ∧ (B ∨ D) προέρχεται από την p ∧ q με αντικατάσταση του p με (A ∨ (B ∧ C)) και του q με (B ∨ D). Συχνά μάλιστα χρησιμοποιούμε τις ίδιες πρότυπες μεταβλητές στην πρότυπη πρόταση και στις εκφράσεις που αντικαθιστούν τις πρότυπες μεταβλητές. Για παράδειγμα, αν στην πρόταση A ∧ B

αντικαταστήσουμε το Α με (Α ∨ (Β ∧ C)) και το Β με (Β ∨ D) προκύπτει η πρόταση (Α ∨ (Β ∧ C)) ∧ (Β ∨ D).

Πρέπει να προσέχουμε με τις αντικαταστάσεις των πρότυπων μεταβλητών γιατί πρέπει να γίνονται συγχρόνως. Αν στο προηγούμενο παράδειγμα οι αντικαταστάσεις γίνουν διαδοχικά, η πρόταση που προκύπτει είναι η Α ∨ ((Β ∨ D) ∧ C) ∧ (Β ∨ D) η οποία δεν είναι ισοδύναμη της (Α ∨ (Β ∧ C)) ∧ (Β ∨ D).

Την αντικατάσταση του Χ με Υ την συμβολίζουμε ως Χ/Υ. Την ταυτόχρονη αντικατάσταση των Χ1, Χ2, … , Χn με Υ1, Υ2, … , Υn την συμβολίζουμε ως { Χ1/Υ1, Χ2/Υ2, … , Χn/Υn }. Αν Ε είναι μια πρόταση του ΠΛ και j μια αντικατάσταση, τότε Ej συμβολίζει την πρόταση που προκύπτει από την εφαρμογή της αντικατάσταση j στην Ε. Για παράδειγμα, αν Ε είναι η πρόταση Α ∨ Β και j = { Α / Α ∨ (Β ∧ C), Β / Β ∨ D }, τότε Ej είναι η πρόταση (Α ∨ (Β ∧ C)) ∧ (Β ∨ D).

Για να επιστρέψουμε στην αναπαράσταση της δομής των προτάσεων ως δέντρα, το βάθος μια δομής είναι το μήκος του μακρύτερου μονοπατιού από τη ρίζα του δέντρου ως τα φύλλα. Το βάθος στο οποίο μία υποπρόταση παρουσιάζεται μέσα στη δομή μιας πρότασης ορίζεται ως το μήκος του μονοπατιού από τον κόμβο του κύριου συνδετικού της υποπρότασης στην ρίζα της πρότασης.

Παραδείγματα :

- Το βάθος της πρότασης Α είναι 0.
- Το βάθος της πρότασης Α ∧ Β είναι 1.
- Το βάθος της πρότασης Α ∨ (Β ∧ C) είναι 2.
- Το βάθος της πρότασης Α ∧ ((Β ∧ (C ∨ D)) ∨ Ε είναι 4.
- Η πρόταση C ∨ D εμφανίζεται σε βάθος 3 στην πρόταση Α ∧ ((Β ∧ (C ∨ D)) ∨ Ε, ενώ η πρόταση Ε εμφανίζεται σε βάθος 2.

11.2 Ισοδυναμίες του Προτασιακού Λογισμού

Παρακάτω θα δούμε μερικές από τις βασικές ισοδυναμίες του ΠΛ για προτάσεις που χρησιμοποιούν τα συνδετικά ∧ και ∨ :

1.1 $A \wedge B \equiv B \wedge A$ (μεταθετικότητα του ∧)
1.2 $A \vee B \equiv B \vee A$ (μεταθετικότητα του ∨)
1.3 $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ (προσεταιριστικότητα του ∧)
1.4 $(A \vee B) \vee C \equiv A \vee (B \vee C)$ (προσεταιριστικότητα του ∨)
1.5 $A \wedge A \equiv A$ (αυτοπάθεια του ∧)
1.6 $A \vee A \equiv A$ (αυτοπάθεια του ∨)
1.7 $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ (επιμερισμός του ∧ πάνω στο ∨)
1.8 $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ (επιμερισμός του ∨ πάνω στο ∧)
1.9 $A \wedge (A \vee B) \equiv A$ (απορρόφηση του ∧)
1.10 $A \vee (A \wedge B) \equiv A$ (απορρόφηση του ∨)

Οι ισοδυναμίες αυτές μπορούν να επαληθευτούν με χρήση πινάκων αληθείας. Για παράδειγμα, ο πίνακας αληθείας της ισοδυναμίας (1.7) είναι ο ακόλουθος :

A	B	C	$B \vee C$	$A \wedge (B \vee C)$	$A \wedge B$	$A \wedge C$	$(A \wedge B) \vee (A \wedge C)$
α	α	α	α	α	α	α	α
α	α	ψ	α	α	α	ψ	α
α	ψ	α	α	α	ψ	α	α
α	ψ	ψ	ψ	ψ	ψ	ψ	ψ
ψ	α	α	α	ψ	ψ	ψ	ψ
ψ	α	ψ	α	ψ	ψ	ψ	ψ
ψ	ψ	α	α	ψ	ψ	ψ	ψ
ψ	ψ	ψ	ψ	ψ	ψ	ψ	ψ

Με παρόμοιο τρόπο μπορούμε να επαληθεύσουμε και τις υπόλοιπες ισοδυναμίες.

11.3 Κανονικές Μορφές (Normal Forms)

Οποιαδήποτε πρόταση χρησιμοποιεί μόνο ∧ και ∨, είναι ισοδύναμη με μια σύζευξη διαζεύξεων και μια διάζευξη συζεύξεων. Αυτές οι προτάσεις αντιστοιχούν στην συζευκτική κανονική μορφή (conjunctive normal form) και στην διαζευκτική κανονική μορφή (disjunctive normal form) αντίστοιχα. Παρακάτω υπάρχουν κάποιοι ορισμοί που χρειαζόμαστε για να ορίσουμε τις παραπάνω μορφές.

a) Ένα γράμμα (literal) είναι οποιαδήποτε πρότυπη μεταβλητή. Ένας ελάχιστος όρος (minterm) είναι ένα γράμμα ή η σύζευξη γραμμάτων. Ένας μέγιστος όρος (maxterm) είναι ένα γράμμα ή η διάζευξη γραμμάτων.
   Για παράδειγμα, το A είναι γράμμα και συγχρόνως μέγιστος και ελάχιστος όρος. Το A ∧ B ∧ C είναι ελάχιστος όρος ενώ το A ∧ A δεν είναι. Το A ∨ B είναι μέγιστος όρος ενώ το A ∨ A δεν είναι.

b) Ένας ελάχιστος (μέγιστος) όρος M1 απορροφά έναν άλλο ελάχιστο (μέγιστο) όρο M2, αν κάθε γράμμα του M1 είναι επίσης στο M2.
   Για παράδειγμα, κάθε μέγιστος (ελάχιστος) όρος απορροφά τον εαυτό του. Το A απορροφά το A ∨ C και το A ∨ C απορροφά το A ∨ B ∨ C.

c) Μια πρόταση είναι σε Διαζευκτική Κανονική Μορφή (DNF) αν είναι μια διάζευξη από ελάχιστους όρους, κανένας από τους οποίους δεν απορροφά κανέναν άλλο. Μια πρόταση είναι σε Συζευκτική Κανονική Μορφή (CNF) αν είναι μια σύζευξη από μέγιστους όρους, κανένας από τους οποίους δεν απορροφά κανένα άλλο.
   Παραδείγματα :
   - Η πρόταση (A ∧ B) ∨ (A ∧ D ∧ E) ∨ C είναι σε DNF.
   - Η πρόταση (A ∧ B) ∨ (A ∧ B ∧ C) ∨ D δεν είναι σε DNF.
   - Η πρόταση A ∧ B ∧ C είναι σε DNF και σε CNF.
   - Η πρόταση A ∨ B ∨ C είναι σε DNF και σε CNF.
   - Οι προτάσεις A, B, A ∧ B, A ∨ B είναι σε DNF και σε CNF.
   - Η πρόταση A ∨ (B ∧ C) είναι σε DNF αλλά όχι σε CNF.

11.4 Αλγόριθμος Μετατροπής σε DNF

Παρακάτω θα δούμε τα βήματα του αλγορίθμου μετατροπής σε DNF. Σαν είσοδο θα πάρουμε μια πρόταση του ΠΛ που χρησιμοποιεί μόνο τους συνδέσμους της σύζευξης ($\wedge$) και της διάζευξης ($\vee$). Σαν έξοδο θα πάρουμε την πρόταση σε DNF.

a) Για αρχή αφαιρούμε τις μη-απαραίτητες παρενθέσεις (Για παράδειγμα, η πρόταση $(A \vee B) \vee C$ γράφεται ως $A \vee B \vee C$).
b) Στη συνέχεια βρίσκουμε τη σύζευξη ($\wedge$) η οποία βρίσκεται σε μεγαλύτερο βάθος και η οποία περιέχει τουλάχιστον μια διάζευξη ($\vee$). Αν δεν υπάρχει τέτοια υποπρόταση, προχωράμε στο βήμα (c).
c) Στη σύζευξη που επιλέγουμε στο βήμα (a), εφαρμόζουμε την επιμεριστικότητα της σύζευξης (ισοδυναμία 1.7) και αφαιρούμε τις περιττές παρενθέσεις. Στη συνέχεια επιστρέφουμε στο βήμα (a).
d) Απλοποιούμε κάθε σύζευξη όσο γίνεται χρησιμοποιώντας την ισοδυναμία της αυτοπάθειας (ισοδυναμία 1.9).
e) Αν δεν υπάρχουν συζεύξεις που χρησιμοποιούν τα ίδια γράμματα, κρατάμε μόνο μία από αυτές.
f) Τέλος, παραλείπουμε κάθε σύζευξη η οποία περιέχει όλα τα γράμματα μιας άλλης σύζευξης (δηλαδή απορροφάται από μια άλλη σύζευξη).

Παράδειγμα : Ας εφαρμόσουμε τον αλγόριθμο στην πρόταση

$$(A \vee B) \wedge ((B \wedge (C \vee ((C \vee D) \wedge A))) \vee C)$$

Η πρόταση δεν περιέχει μη-απαραίτητες παρενθέσεις. Στο βήμα (a), επιλέγουμε την σύζευξη $(C \vee D) \wedge A$ σαν αυτή που βρίσκεται σε μεγαλύτερο βάθος και περιέχει τουλάχιστον μια διάζευξη. Στο βήμα (b), εφαρμόζουμε την ισοδυναμία 1.7, από την οποία προκύπτει η πρόταση

$$(A \vee B) \wedge ((B \wedge (C \vee (C \wedge A) \vee (D \wedge A))) \vee C)$$

Στην επόμενη επανάληψη, επιλέγουμε την πρόταση $B \wedge (C \vee (C \wedge A) \vee (D \wedge A))$ σαν αυτή που βρίσκεται σε μεγαλύτερο βάθος και περιέχει τουλάχιστον μια διάζευξη. Στο βήμα (b), εφαρμόζουμε την ισοδυναμία 1.7, από την οποία προκύπτει η πρόταση

$$(A \vee B) \wedge ((B \wedge C) \vee (B \wedge C \wedge A) \vee (B \wedge D \wedge A) \vee C)$$

Στην επόμενη επανάληψη, επιλέγουμε ολόκληρη την πρόταση σαν αυτή που βρίσκεται σε μεγαλύτερο βάθος και περιέχει τουλάχιστον μια διάζευξη. Στο βήμα (b), εφαρμόζουμε την ισοδυναμία 1.7, από την οποία προκύπτει η πρόταση

$$(A \wedge B \wedge C) \vee (A \wedge B \wedge C \wedge A) \vee (A \wedge B \wedge D \wedge A) \vee (A \wedge C) \vee$$
$$(B \wedge B \wedge C) \vee (B \wedge B \wedge C \wedge A) \vee (B \wedge B \wedge D \wedge A) \vee (B \wedge C)$$

Στο βήμα (c), απλοποιούμε τις συζεύξεις που προέκυψαν από τα προηγούμενα βήματα. Το αποτέλεσμα είναι η πρόταση :

$$(A \wedge B \wedge C) \vee (A \wedge B \wedge C) \vee (A \wedge B \wedge D) \vee (A \wedge C) \vee$$
$$(B \wedge C) \vee (B \wedge C \wedge A) \vee (B \wedge D \wedge A) \vee (B \wedge C)$$

Από το βήμα (d) προκύπτει η πρόταση :

$$(A \wedge B \wedge C) \vee (A \wedge B \wedge D) \vee (A \wedge C) \vee (B \wedge C)$$

Τέλος, από το βήμα (e) η πρόταση στην τελική της μορφή είναι η

$$S : (A \wedge B \wedge D) \vee (A \wedge C) \vee (B \wedge C)$$

Μπορούμε να επιβεβαιώσουμε την ισοδυναμία της αρχικής πρότασης και της πρότασης που προκύπτει από την εφαρμογή του αλγορίθμου DNF, με την κατασκευή πινάκων αληθείας για τις δύο προτάσεις. Η κατασκευή του πίνακα αληθείας για την DNF μορφή της πρότασης, είναι πολύ απλούστερη από την αντίστοιχη κατασκευή πίνακα αληθείας για την αρχική πρόταση :

A	B	C	D	A∧B∧D	A∧ C	B ∧ C		S
α	α	α	α	α	α	α		α
α	α	α	ψ	ψ	α	α		α
α	α	ψ	α	α	ψ	ψ		α
α	α	ψ	ψ	ψ	ψ	ψ		ψ
α	ψ	α	α	ψ	α	ψ		α
α	ψ	α	ψ	ψ	α	ψ		α
α	ψ	ψ	α	ψ	ψ	ψ		ψ
α	ψ	ψ	ψ	ψ	ψ	ψ		ψ
ψ	α	α	α	ψ	ψ	α		α
ψ	α	α	ψ	ψ	ψ	α		α
ψ	α	ψ	α	ψ	ψ	ψ		ψ
ψ	α	ψ	ψ	ψ	ψ	ψ		ψ

ψ	ψ	α	α	ψ	ψ	ψ		ψ
ψ	ψ	α	ψ	ψ	ψ	ψ		ψ
ψ	ψ	ψ	α	ψ	ψ	ψ		ψ
ψ	ψ	ψ	ψ	ψ	ψ	ψ		ψ

Κάθε σύζευξη της S αντιστοιχεί σε ένα σύνολο γραμμών του πίνακα.

Μετατροπή προτάσεων σε CNF : Η μετατροπή γίνεται αν στον αλγόριθμο αντικαταστήσουμε την « σύζευξη » με την « διάζευξη » και αντίστροφα.

Οι CNF και DNF μορφές είναι δυικές (dual) μορφές. Αν σε μία ισοδυναμία αντικαταστήσουμε την « σύζευξη » με την « διάζευξη » και αντίστροφα, η ισοδυναμία παραμένει.

Για παράδειγμα, γνωρίζουμε ότι ισχύει η ισοδυναμία $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$. Αν αντικαταστήσουμε $\wedge$ με $\vee$ και αντίστροφα, προκύπτει η ισοδυναμία $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ η οποία επίσης ισχύει.

11.5 Άρνηση (Negation)

Όπως έχουμε ήδη δει, για οποιαδήποτε πρόταση του ΠΛ η οποία χρησιμοποιεί τα συνδετικά της σύζευξης ($\wedge$) και της διάζευξης ($\vee$), μπορούμε να κατασκευάσουμε ένα πίνακα αληθείας. Το αντίστροφο μπορεί να γίνει. Δηλαδή αν μας δίνεται ένας πίνακας αληθείας, μπορούμε να κατασκευάσουμε μια πρόταση που αντιστοιχεί σε αυτόν.

Για παράδειγμα, θεωρούμε τον παρακάτω πίνακα αληθείας :

A	B	C		Q
α	α	α		α
α	α	ψ		ψ
α	ψ	α		α
α	ψ	ψ		ψ
ψ	α	α		α
ψ	α	ψ		ψ
ψ	ψ	α		ψ
ψ	ψ	ψ		ψ

Η Q είναι αληθής όταν η $A \wedge C$ ή $B \wedge C$ είναι αληθής. Η Q θα μπορούσε να είναι η πρόταση $(A \wedge C) \vee (B \wedge C)$ ή $(A \vee B) \wedge C$.

Υπάρχουν κάποιοι πίνακες αληθείας από τους οποίους δεν μπορούμε να βρούμε κάποια πρόταση που να χρησιμοποιεί μόνο τα συνδετικά $\wedge$ και $\vee$.

Για παράδειγμα, για τον παρακάτω πίνακα αληθείας δεν υπάρχει τέτοια πρόταση.

A	B	Q
α	α	ψ
α	ψ	ψ
ψ	α	ψ
ψ	ψ	α

Για να βρούμε την πρόταση του παραπάνω πίνακα χρειαζόμαστε ένα καινούργιο συνδετικό, αυτό της άρνησης ($\neg$).

Ο πίνακας αληθείας της Άρνησης είναι ο παρακάτω :

A		¬ A
α		ψ
ψ		α

Τώρα μπορούμε να εκφράσουμε την πρόταση Q του παρακάτω πίνακα :

A	B		Q
α	α		ψ
α	ψ		ψ
ψ	α		ψ
ψ	ψ		α

Η Q είναι $\neg A \wedge \neg B$.

Από κάθε πίνακα αληθείας μπορούμε να κατασκευάσουμε μια πρόταση χρησιμοποιώντας τα συνδετικά $\wedge$, $\vee$ και $\neg$.

Βασικές Ισοδυναμίες :

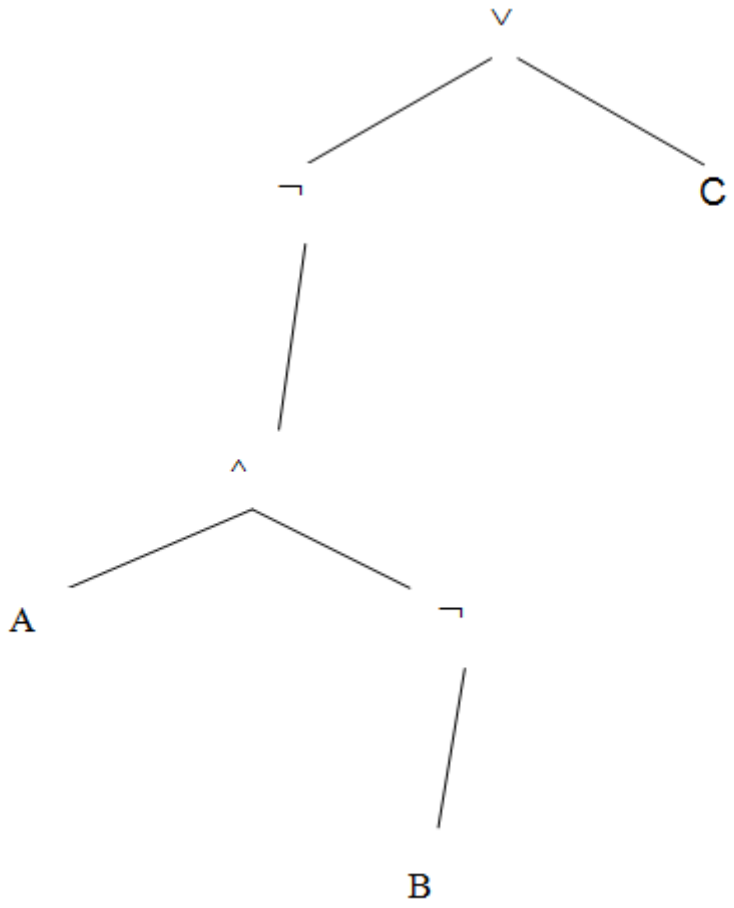
a) $\neg \neg A \equiv A$ (απαλοιφή διπλής άρνησης)
b) $\neg (A \wedge B) \equiv \neg A \vee \neg B$ (νόμος De Morgan)
c) $\neg (A \vee B) \equiv \neg A \wedge \neg B$ (νόμος De Morgan)
d) $A \oplus B \equiv (A \vee B) \wedge \neg (A \wedge B)$ (αποκλειστική διάζευξη)

Από την ισοδυναμία b) προκύπτει ότι $\neg \neg (A \wedge B) \equiv \neg (\neg A \vee \neg B)$ και από την α) έχουμε : $(A \wedge B) \equiv \neg (\neg A \vee \neg B)$. Άρα τη σύζευξη μπορούμε να την ορίσουμε χρησιμοποιώντας $\neg$ και $\vee$.

Ανάλογα από την c) έχουμε : $\neg \neg (A \vee B) \equiv \neg (\neg A \wedge \neg B)$ και από την α) έχουμε : $(A \vee B) \equiv \neg (\neg A \wedge \neg B)$. Άρα τη διάζευξη μπορούμε να την ορίσουμε χρησιμοποιώντας $\neg$ και $\wedge$.

Προτάσεις που περιλαμβάνουν άρνηση μπορούμε να τις αναπαραστήσουμε με δέντρα.

Παράδειγμα : την πρόταση ¬ (A ∧ ¬ B) ∨ C μπορούμε να την αναπαραστήσουμε από το ακόλουθο δέντρο :



11.6 Συνεπαγωγή στον Προτασιακό Λογισμό

Όπως έχουμε δει, μια πρόταση Α συνάγεται από ένα σύνολο προτάσεων Σ αν κάθε ερμηνεία που ικανοποιεί τις προτάσεις του Σ ικανοποιεί και την Α.
Με την βοήθεια πινάκων αληθείας, θα δείξουμε ότι { Α ∨ Β, ¬ Α } | = Β. Κάθε γραμμή ενός πίνακα είναι και μια ερμηνεία.

Α	Β		Α∨Β	¬Α		Β
α	α		α	ψ		α
α	ψ		α	ψ		ψ
ψ	α		α	α		α
ψ	ψ		ψ	α		ψ

Από τον πίνακα βλέπουμε ότι κάθε ερμηνεία που ικανοποιεί τις Α ∨ Β και ¬ Α ικανοποιεί και την Β. Επομένως ισχύει η { Α ∨ Β, ¬ Α } | = Β.
Με τον ίδιο τρόπο μπορούμε να εξετάσουμε την { Α ∨ Β, Α } | = ¬ Β.

Α	Β	Α∨ Β	Α		¬Β
α	α	α	α		ψ
α	ψ	α	α		α
ψ	α	α	ψ		ψ
ψ	ψ	ψ	ψ		α

Υπάρχει η ερμηνεία που καθιστά αληθείς τις υποθέσεις αλλά ψευδές το συμπέρασμα. Άρα η εξαγωγή συμπεράσματος δεν είναι έγκυρη.

11.7 Ταυτολογίες και Αντινομίες (Tautologies and Contradictions)

Με την εισαγωγή του συνδετικού της άρνησης, μπορούμε να κατασκευάσουμε προτάσεις οι οποίες είναι λογικά αληθείς ή λογικά ψευδής. Οι απλούστερες τέτοιες προτάσεις είναι οι :

- $A \vee \neg A$
- $A \wedge \neg A$

Ο πίνακας αληθείας τους είναι ο εξής :

A	$\neg A$	$A \vee \neg A$	$A \wedge \neg A$
α	ψ	α	ψ
ψ	α	α	ψ

Η πρόταση $A \vee \neg A$ είναι αληθής για κάθε ερμηνεία του Α ενώ η πρόταση $A \wedge \neg A$ είναι ψευδής. Άρα και οι δύο είναι ανεξάρτητες από τις ερμηνείες του Α.

Μια ταυτολογία είναι μια πρόταση η οποία ικανοποιείται από κάθε ερμηνεία.
Μια αντινομία είναι μια πρόταση την οποία κάθε ερμηνεία καθιστά ψευδή.

Παραδείγματα :

a) Η πρόταση S: $A \vee (\neg A \wedge B) \vee (\neg A \wedge \neg B)$ είναι ταυτολογία.
$A \vee (\neg A \wedge B) \vee (\neg A \wedge \neg B) \equiv$
$A \vee (\neg A \wedge (B \vee \neg B) \equiv$
$(A \vee \neg A) \wedge (A \vee (B \vee \neg B))$

Οι προτάσεις $(A \vee \neg A)$ και $(B \vee \neg B)$ είναι ταυτολογίες. Άρα η S είναι ισοδύναμη με μια σύζευξη δύο προτάσεων, από τις οποίες η πρώτη είναι ταυτολογία και η δεύτερη είναι μια διάζευξη μιας ταυτολογίας με την πρόταση Α. Άρα η S είναι πάντα αληθής.

b) Η πρόταση S': $A \wedge B \wedge (\neg A \vee \neg B)$ είναι αντινομία.

$A \wedge B \wedge (\neg A \vee \neg B) \equiv$

$A \wedge ((B \wedge \neg A) \vee (B \wedge \neg B)) \equiv$

$((A \wedge \neg A) \wedge B) \vee (A \wedge (B \wedge \neg B))$

Οι προτάσεις $(A \wedge \neg A)$ και $(B \wedge \neg B)$ και επομένως οι συζεύξεις στις οποίες εμφανίζονται είναι πάντα ψευδείς. Άρα και η S' είναι πάντα ψευδής.

11.8 Ταυτολογίες και Λογική Συνεπαγωγή

Μία εξαγωγή συμπεράσματος p1, p2, … , pn / c, όπου c είναι ταυτολογία, είναι έγκυρη. Γιατί αν δεν ήταν θα έπρεπε να υπάρχει μια ερμηνεία για την οποία οι προτάσεις p1, p2, … , pn να είναι αληθείς και η c ψευδής. Αλλά η c είναι ταυτολογία, επομένως πάντα αληθής. Άρα η εξαγωγή συμπεράσματος είναι έγκυρη.
Για να δηλώσουμε ότι η c είναι ταυτολογία γράφουμε | = c ενώ για να δηλώσουμε ότι είναι αντινομία γράφουμε | = ¬ c.
Αν σε κάποια εξαγωγή συμπεράσματος, κάποια από τις υποθέσεις είναι αντινομία, τότε η εξαγωγή συμπεράσματος είναι έγκυρη.
Το συμπέρασμα είναι ότι από μια αντινομία μπορούμε να συμπεράνουμε οποιαδήποτε πρόταση.
Ας υποθέσουμε ότι το σύνολο { p1, p2, … , pn } είναι μη-ικανοποιήσιμο. Τότε η πρόταση p1 ∧ p2 ∧ … ∧ pn είναι αντινομία και η πρόταση p1, p2, … , pn / c είναι έγκυρη.
Προτάσεις που δεν είναι ούτε ταυτολογίες ούτε αντινομίες τις ονομάζουμε ικανοποιήσιμες.
Οι ταυτολογίες και οι αντινομίες μας επιτρέπουν να ορίσουμε νέους κανόνες απορρόφησης. Έστω ότι το σύμβολο Τα συμβολίζει μια ταυτολογία ενώ το σύμβολο F συμβολίζει μια αντινομία. Τότε, αν η S είναι μια οποιαδήποτε πρόταση, ισχύουν οι παρακάτω ισοδυναμίες :

- S ∧ T ≡ S
- S ∧ F ≡ F
- S ∨ T ≡ T
- S ∨ F ≡ S

Παράδειγμα :

Θεωρούμε τον παρακάτω πίνακα αληθείας μιας πρότασης Q και ψάχνουμε να βρούμε μια πρόταση η οποία να περιγράφει την Q.

	A	B	C	Q
1	α	α	α	ψ
2	α	α	ψ	ψ
3	α	ψ	α	α
4	α	ψ	ψ	ψ
5	ψ	α	α	α
6	ψ	α	ψ	α
7	ψ	ψ	α	α
8	ψ	ψ	ψ	ψ

Για κάθε ερμηνεία που καθιστά την Q αληθή, μπορούμε να κατασκευάσουμε μια πρόταση που χρησιμοποιεί τα A, B, C. Έτσι για τις γραμμές 3, 5, 6, 7 του παραπάνω πίνακα θα έχουμε τις προτάσεις $A \wedge \neg B \wedge C$, $\neg A \wedge B \wedge C$, $\neg A \wedge B \wedge \neg C$, $\neg A \wedge \neg B \wedge C$ αντίστοιχα. Άρα :

$$
\begin{aligned}
Q \equiv \quad & (A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge C) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge C) && \equiv \\
& (A \wedge \neg B \wedge C) \vee ((\neg A \wedge B) \wedge (C \vee \neg C)) \vee (\neg A \wedge \neg B \wedge C) && \equiv \\
& (A \wedge \neg B \wedge C) \vee ((\neg A \wedge B) \wedge \mathbf{T}) \vee (\neg A \wedge \neg B \wedge C) && \equiv \\
& (A \wedge \neg B \wedge C) \vee (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge B) && \equiv \\
& ((A \vee \neg A) \wedge (\neg B \wedge C)) \vee (\neg A \wedge B) && \equiv \\
& (\mathbf{T} \wedge (\neg B \wedge C)) \vee (\neg A \wedge B) && \equiv \\
& (\neg B \wedge C) \vee (\neg A \wedge B)
\end{aligned}
$$

11.9 Συνεπαγωγή και Ισοδυναμία (Material Implication and Equivalence)

Τα σύμβολα | = και ≡ τα χρησιμοποιούμε για να δηλώσουμε ότι μια πρόταση του Π.Λ. είναι συνεπαγωγή ενός συνόλου προτάσεων και για να δηλώσουμε ότι δύο προτάσεις είναι ισοδύναμες αντίστοιχα. Οι εκφράσεις p1 ∧ p2 ∧ … ∧ pn | = A και A ≡ B δεν είναι προτάσεις του Π.Λ. Είναι προτάσεις για τον Π.Λ. Δεν υπάρχει τρόπος να δηλώσουμε ότι ένα σύνολο προτάσεων συνεπάγεται μια άλλη πρόταση γράφοντας μια πρόταση του Π.Λ. Δηλαδή δεν υπάρχει το ανάλογο του A | = B. Υπάρχουν όμως προτάσεις που σε πολλές περιπτώσεις είναι ισοδύναμες.
Θεωρούμε την πρόταση ¬ X ∨ Y, με τον παρακάτω πίνακα αληθείας :

X	Y	$\neg X \vee Y$
α	α	α
α	ψ	ψ
ψ	α	α
ψ	ψ	α

Άρα η πρόταση ¬ X ∨ Y είναι ψευδής μόνο όταν η X είναι αληθής και η Y είναι ψευδής. Υποθέτουμε ότι X | = Y, άρα δεν μπορεί να ισχύει ότι η X είναι αληθής και η Y είναι ψευδής.
Παράδειγμα : αν η X είναι η πρόταση ¬ (A ∨ B) και η Y είναι η πρόταση ¬ A, τότε η πρόταση ¬ ¬ (A ∨ B) ∨ ¬ A έχει τον εξής πίνακα αληθείας :

A	B	$\neg(A \vee B)$	$\neg A$	$\neg\neg(A \vee B) \vee \neg A$
α	α	ψ	ψ	α
α	ψ	ψ	ψ	α
ψ	α	ψ	α	α
ψ	ψ	α	α	α

Άρα αν X | = Y, τότε | = ¬ X ∨ Y, δηλαδή η ¬ X ∨ Y είναι ταυτολογία.

Αν δεν γνωρίζουμε ότι X | = Y αλλά ανακαλύψουμε ότι η ¬ X ∨ Y είναι ταυτολογία, τότε αυτό σημαίνει ότι δεν υπάρχει ερμηνεία που να καθιστά αυτή την πρόταση ψευδή. Αν υπήρχε, θα έπρεπε να καθιστά την ¬ X ψευδή και την Y επίσης ψευδή, δηλαδή την X αληθή και την Y ψευδή. Άρα για κάθε ερμηνεία για την οποία η Y είναι αληθής είναι και η Y αληθής, οπότε X | = Y.

Από τα παραπάνω συμπεραίνουμε ότι X | = Y αν και μόνο αν | = ¬ X ∨ Y.

Τον συνδυασμό των συνδετικών ¬, ∨ στην πρόταση ¬ X ∨ Y τον αναπαριστάμε με το σύμβολο →, το οποίο ονομάζεται συνεπαγωγή (material implication). Η συνεπαγωγή έχει τον εξής πίνακα αληθείας :

X	Y	$X \to Y$
α	α	α
α	ψ	ψ
ψ	α	α
ψ	ψ	α

Την πρόταση X → Y την διαβάζουμε ως « αν X τότε Y » ή « X μόνο αν Y » ή « Y αν X ». Την X την λέμε υπόθεση (antecedent) και την Y την λέμε συμπέρασμα (consequent).

Όσον αφορά την περίπτωση της λογικής ισοδυναμίας, υπάρχει πρόταση του Π.Λ. που να εκφράζει το A ≡ B.

Η πρόταση (A ∧ B) ∨ (¬ A ∧ ¬ B) είναι αληθής όταν η A και η B έχουν την ίδια τιμή αληθείας. Άρα A ≡ B αν και μόνο αν | = (A ∧ B) ∨ (¬ A ∧ ¬ B), όπως βλέπουμε και στον παρακάτω πίνακα αληθείας :

A	B	¬ A	¬ B	$A \wedge B$	$\neg A \wedge \neg B$	$(A \wedge B) \vee (\neg A \wedge \neg B)$
α	α	ψ	ψ	α	ψ	α
α	ψ	ψ	α	ψ	ψ	ψ
ψ	α	α	ψ	ψ	ψ	ψ
ψ	ψ	α	α	ψ	α	α

Άρα η πρόταση (A ∧ B) ∨ (¬ A ∧ ¬ B) σχετίζεται με την A ≡ B με τον ίδιο τρόπο που η A → B σχετίζεται με την A | = B. Την πρόταση (A ∧ B) ∨ (¬ A ∧ ¬ B), την γράφουμε ως A ↔ B και το σύμβολο ↔ το λέμε ισοδυναμία. Ο πίνακας αληθείας της ισοδυναμίας είναι ο παρακάτω :

X	Y	$X \leftrightarrow Y$
α	α	α
α	ψ	ψ
ψ	α	ψ
ψ	ψ	α

Το συμπέρασμα που βγάζουμε είναι ότι :

- Μια ερμηνεία ικανοποιεί την A → B αν και μόνο αν δεν ικανοποιεί την A ή ικανοποιεί τις A και B.
- Μια ερμηνεία ικανοποιεί την A ↔ B αν και μόνο αν ή ικανοποιεί τις A και B ή καθιστά και τις δύο ψευδής.

11.10 Κανονικές Μορφές

Στους αρχικούς ορισμούς των κανονικών μορφών CNF και DNF, χρησιμοποιήσαμε προτάσεις μόνο με το συνδετικό της άρνησης (¬). Στους νέους ορισμούς των κανονικών μορφών θα επιβάλλουμε την απαίτηση το πεδίο του συμβόλου της άρνησης να είναι όσο το δυνατόν μικρότερο. Βάσει αυτής της απαίτησης, μια πρόταση της μορφής ¬ (A ∧ B ∧ C) δεν θα μπορεί να συμμετέχει σε κάποια κανονική μορφή, αλλά η ισοδύναμή της πρόταση ¬ A ∨ ¬ B ∨ ¬ C θα μπορεί.

Απαραίτητοι ορισμοί :

- Ένα γράμμα είναι μια πρότυπη μεταβλητή ή η άρνηση μιας πρότυπης μεταβλητής.
- Ένας ελάχιστος όρος είναι ένα γράμμα ή η σύζευξη γραμμάτων από τα οποία κανένα δεν είναι η άρνηση κάποιου άλλου ή ένα από τα σύμβολα F και T.
- Ένας ελάχιστος όρος M1 απορροφά έναν ελάχιστο όρο M2 αν κάθε γράμμα του M1 είναι στον M2. Κάθε ελάχιστος όρος απορροφά το F και απορροφάται από το T.
- Δύο ελάχιστοι όροι M1, M2 συνενώνονται σε έναν ελάχιστο όρο M3, αν οι M1, M2 περιέχουν όλα τα γράμματα του M3 και ένα επιπλέον γράμμα το οποίο, στον ένα όρο είναι η άρνηση του επιπλέον γράμματος στον άλλο όρο. Κάθε γράμμα και η άρνησή του συνενώνονται στον όρο T.
- Μια πρόταση είναι σε DNF, αν είναι μια διάζευξη ελαχίστων όρων, κανένας από τους οποίους δεν απορροφά ή δεν συνενώνεται με κανέναν άλλο.

Παραδείγματα :

- Γράμματα : A, ¬ A, B, ¬ B
- Ελάχιστοι όροι : A, ¬ A, A ∧ ¬ B, F, T, όχι όμως A ∧ B ∧ ¬ B
- Απορρόφηση : ο κάθε ένας από τους παρακάτω όρους απορροφά όλους τους επόμενους : T, A, A ∧ C, A ∧ ¬ B ∧ C, F.
- Συνένωση : οι όροι A ∧ ¬ B ∧ ¬ C ∧ D και A ∧ ¬ B ∧ C ∧ D συνενώνονται στον όρο A ∧ ¬ B ∧ D.

- Προτάσεις σε DNF : οι προτάσεις (A ∧ ¬ B) ∨ (B ∧ C ∧ ¬ D) ∨ (A ∧ D), T, F είναι σε DNF. Οι προτάσεις (A ∧ ¬ B) ∨ (A ∧ ¬ B ∧ C) ∨ (B ∧ C ∧ D), (A ∧ ¬ B ∧ C) ∨ (A ∧ ¬ B ∧ ¬ C) ∨ (A ∧ B ∧ C), A ∨ F, B ∨ T δεν είναι σε DNF.

Μπορούν να υπάρχουν ισοδύναμες προτάσεις σε DNF που να περιέχουν διαφορετικούς ελάχιστους όρους. Για παράδειγμα, η πρόταση (A ∧ ¬ B ∧ C) ∨ (A ∧ ¬ B ∧ ¬ C) ∨ (A ∧ B ∧ C) δεν είναι σε DNF. Αν συνενώσουμε το πρώτο με το δεύτερο όρο, προκύπτει η πρόταση (A ∧ ¬ B) ∨ (A ∧ B ∧ C) ενώ αν συνενώσουμε τον πρώτο με τον τρίτο όρο, προκύπτει η πρόταση (A ∧ C) ∨ (A ∧ ¬ B ∧ C). Οι δύο προτάσεις είναι ισοδύναμες αλλά περιέχουν διαφορετικούς ελάχιστους όρους.

Τις προτάσεις T και F τις κατατάσσουμε στους ελάχιστους όρους, ώστε τις ταυτολογίες και τις αντινομίες να μπορούμε να τις εκφράσουμε σε DNF. Αν οι T και F δεν συμπεριλαμβάνονταν, τότε οι προτάσεις A ∨ ¬ A, A ∨ B ∨ ¬ B, A ∨ D ∨ ¬ D θα ήταν όλες σε DNF και όλες ισοδύναμες. Αυτές απλοποιούνται στην πρόταση T. Η πρόταση F παίζει το ρόλο της DNF μορφής μιας αντινομίας.

11.11 Μετατροπή Προτάσεων του Π.Λ. σε DNF

Η μετατροπή γίνεται σε τρία στάδια :

(α) Τα συνδετικά $\rightarrow$ και $\leftrightarrow$ τα αντικαθιστούμε από ισοδύναμες εκφράσεις, χρησιμοποιώντας τα συνδετικά $\neg$, $\wedge$, $\vee$ βάσει των ισοδυναμιών :

(1) $A \rightarrow B \equiv \neg A \vee B$

(2) $A \leftrightarrow B \equiv (A \wedge B) \vee (\neg A \wedge \neg B)$

(β) Ελαχιστοποιούμε το πεδίο της άρνησης με χρήση των κανόνων De Morgan :

(3) $\neg (A1 \wedge A2 \wedge \ldots \wedge An) \equiv \neg A1 \vee \neg A2 \vee \ldots \vee \neg An$

(4) $\neg (A1 \vee A2 \vee \ldots \vee An) \equiv \neg A1 \wedge \neg A2 \wedge \ldots \wedge \neg An$

(γ) Απαλείφουμε διπλές αρνήσεις :

(5) $\neg \neg A \equiv A$

11.12 Αλγόριθμος Μετατροπής Προτάσεων του Π.Λ. σε DNF

(1) Χρησιμοποιούμε τις ισοδυναμίες (1) και (2) για την αντικατάσταση των συνδετικών $\rightarrow$ και $\leftrightarrow$.

(2) Χρησιμοποιούμε τις ισοδυναμίες (3), (4) και (5) για να ελαχιστοποιήσουμε το πεδίο της άρνησης.

(3) Χρησιμοποιούμε την επιμεριστικότητα της σύζευξης προς τη διάζευξη, για να μετατρέψουμε διαζεύξεις που να εμφανίζονται μέσα στο πεδίο μιας σύζευξης, σε μια διάζευξη συζεύξεων.

(4) Αφαιρούμε διπλά γράμματα από ελάχιστους όρους, διπλούς ελάχιστους όρους, αντικαθιστούμε αντινομίες με F και εφαρμόζουμε την απορρόφηση και την συνένωση όπου είναι δυνατό.

Παράδειγμα : Μετατροπή της πρότασης $((A \rightarrow B) \rightarrow C) \leftrightarrow (A \leftrightarrow C)$ σε DNF.

(α) Η πρόταση είναι ισοδύναμη με :

$$((\neg A \vee B) \rightarrow C) \leftrightarrow (\neg A \vee C) \equiv$$
$$(\neg (\neg A \vee B) \vee C) \leftrightarrow (\neg A \vee C) \equiv$$
$$((\neg (\neg A \vee B) \vee C) \wedge (\neg A \vee C)) \vee (\neg (\neg (\neg A \vee B) \vee C) \wedge$$
$$\neg (\neg A \vee C)) \equiv$$

(β) Στο $2^{ο}$ βήμα η πρόταση μετατρέπεται ως εξής :

$$(((\neg \neg A \wedge \neg B) \vee C) \wedge (\neg A \vee C)) \vee ((\neg \neg (\neg A \vee B) \wedge \neg C) \wedge$$
$$(\neg \neg A \wedge \neg C)) \equiv$$
$$(((A \wedge \neg B) \vee C) \wedge (\neg A \vee C)) \vee ((\neg A \vee B) \wedge \neg C \wedge A \wedge \neg C) \equiv$$

(γ) Στο $3^{ο}$ βήμα η πρόταση μετατρέπεται ως εξής :

$$(A \wedge \neg B \wedge \neg A) \vee (A \wedge \neg B \wedge C) \vee (C \wedge \neg A) \vee (C \wedge C) \vee$$
$$(\neg A \wedge \neg C \wedge A \wedge \neg C) \vee (B \wedge \neg C \wedge A \wedge \neg C) \equiv$$

(δ) Στο $4^{ο}$ βήμα η πρόταση μετατρέπεται ως εξής :

$$(A \wedge \neg B \wedge \neg A) \vee (A \wedge \neg B \wedge C) \vee (C \wedge \neg A) \vee C \vee$$
$$(\neg A \wedge \neg C \wedge A) \vee (B \wedge \neg C \wedge A) \equiv$$
$$F \vee (A \wedge \neg B \wedge C) \vee (C \wedge \neg A) \vee C \vee F \vee (B \wedge \neg C \wedge A) \equiv$$
$$(A \wedge \neg B \wedge C) \vee (\neg A \wedge C) \vee C \vee (A \wedge B \wedge \neg C) \equiv$$
$$C \vee (A \wedge B \wedge \neg C)$$

Ο αλγόριθμος πάντα δίνει ως αποτέλεσμα μια πρόταση σε DNF, αλλά όχι πάντα την απλούστερη. Αν μάλιστα η πρόταση είναι ήδη σε DNF δεν θα αλλαχθεί καθόλου από τον αλγόριθμο.

Για παράδειγμα, η πρόταση $(A \wedge \neg B) \vee B$ είναι ισοδύναμη με την πρόταση $A \vee B$ αλλά ο αλγόριθμος δεν θα μετατρέψει την αρχική πρόταση, αλλά θα την αφήσει ως έχει.

Επίσης, η ταυτολογία $(A \rightarrow B) \rightarrow (A \rightarrow B)$ θα μετατραπεί στην DNF μορφή $(A \wedge \neg B) \vee \neg A \vee B$ αλλά όχι στην T.

Στον αλγόριθμο θα μπορούσαμε να προσθέσουμε και τη χρήση των παρακάτω ισοδυναμιών :

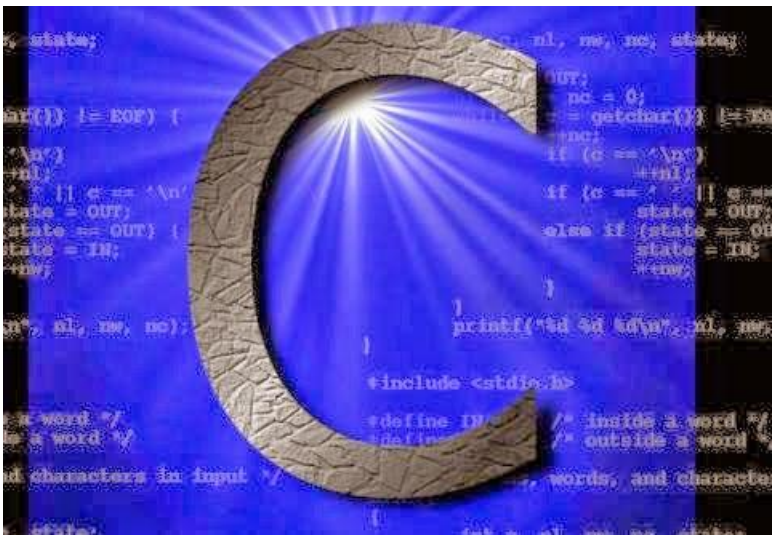
- $(A \wedge \neg B) \vee B \equiv A \vee B$
- $(A \wedge B) \vee \neg B \equiv A \vee \neg B$

Σαν ένα τελικό βήμα απλοποίησης, το οποίο μας επιτρέπει να αφαιρέσουμε από κάποιο ελάχιστο όρο κάποιο γράμμα ή κάποια άρνηση του οποίου εμφανίζεται στην πρόταση σαν ελάχιστος όρος. Και πάλι όμως δεν θα καταλήξουμε στην απλούστερη μορφή σε κάθε περίπτωση.

Δεν υπάρχει αλγόριθμος ο οποίος εγγυάται την παραγωγή της απλούστερης DNF μορφής μιας πρότασης.

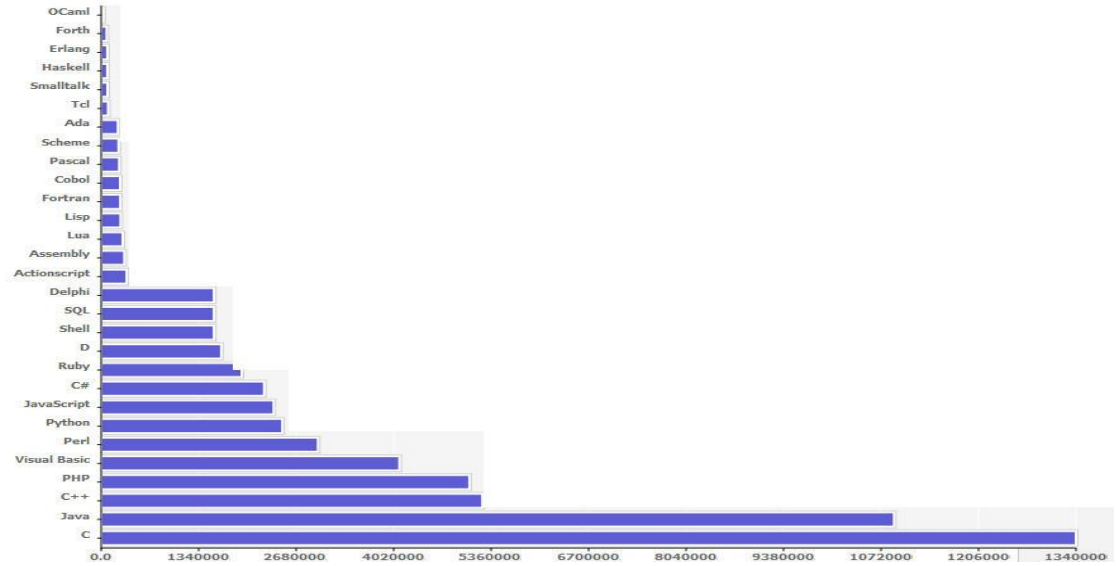
12 Η Γλώσσα C

Η C (προφέρεται « σι ») είναι μια διαδικαστική γλώσσα προγραμματισμού γενικής χρήσης, η οποία αναπτύχθηκε αρχικά, μεταξύ του 1969 και του 1973, από τον Ντένις Ρίτσι στα εργαστήρια AT&T Bell Labs για να χρησιμοποιηθεί για την ανάπτυξη του λειτουργικού συστήματος UNIX. Όπως οι περισσότερες διαδικαστικές γλώσσες προγραμματισμού που ακολουθούν την παράδοση της ALGOL, η C έχει δυνατότητες δομημένου προγραμματισμού και επιτρέπει τη χρήση αναδρομής (αλλά όχι και εμφωλευμένων συναρτήσεων), ενώ ο στατικός ορισμός του τύπου των μεταβλητών που επιβάλει, προλαμβάνει πολλά σφάλματα κατά την χρήση τους. Ο σχεδιασμός της περιλαμβάνει δομές που μεταφράζονται αποδοτικά σε τυπικές εντολές μηχανής (machine instructions) και εξ αιτίας αυτού χρησιμοποιείται συχνά σε εφαρμογές που παλιότερα γράφονταν σε συμβολική γλώσσα (assembly language). Αυτό ακριβώς το χαρακτηριστικό της, είναι που έχει σαν συνέπεια και την αυξημένη ταχύτητα εκτέλεσης των εφαρμογών που γράφονται σε αυτή, καθώς και το γεγονός ότι είναι διαθέσιμη στα περισσότερα σημερινά λειτουργικά συστήματα, συνέβαλε κατά πολύ στην καθιέρωσή της και την χρήση της για την ανάπτυξη λειτουργικών συστημάτων και λοιπών προγραμμάτων συστήματος (system software), αλλά και απλών εφαρμογών.



Την C την συγκαταλέγουμε πλέον στις πιο ευρέως χρησιμοποιούμενες γλώσσες προγραμματισμού όλων των εποχών και πολλές νεώτερες γλώσσες έχουν επηρεαστεί

άμεσα ή έμμεσα από αυτήν, συμπεριλαμβανομένου των C++, C#, D, Go, Java, JavaScript, Limbo, LPC, Perl, PHP, Python, καθώς και του κελύφους C (C sheel) του Unix. Κάποιες από αυτές τις γλώσσες έχουν επηρεαστεί κυρίως στη σύνταξη τους, με τα συστήματα τύπων, τα μοντέλα δεδομένων και το νόημα των εκφράσεων τους να διαφέρουν σημαντικά από την C. Η C++ ειδικά, ξεκίνησε σαν προεπεξεργαστής της C, αλλά έχει εξελιχθεί πλέον σε μια αντικειμενοστραφή γλώσσα, που αποτελεί υπερσύνολο της C.



Εικόνα 8: Δημοτικότητα Γλωσσών

Στην C δεν επιβάλλεται κάποια συγκεκριμένη μορφή στον πηγαίο κώδικα (όπως, για παράδειγμα, συνέβαινε στις αρχικές εκδόσεις της Fortran). Ο προγραμματιστής, χωρίς να αγνοεί φυσικά το συντακτικό της γλώσσας, είναι ελεύθερος να δώσει όποια μορφή θέλει στον κώδικα που γράφει (free-format source). Το ελληνικό ερωτηματικό (;) χρησιμοποιείται ως τερματιστής εντολών (και όχι σαν διαχωριστής, όπως στην Pascal) και τα άγκιστρα ({ }) χρησιμοποιούνται για την ομαδοποίηση εντολών (όπως τα begin/end στην Pascal).

Επίσης, στην C όλος ο εκτελέσιμος κώδικας περιέχεται σε υπορουτίνες οι οποίες ονομάζονται « συναρτήσεις ». Οι παράμετροι περνιούνται στις συναρτήσεις πάντα με τιμή (pass-by-value). Το πέρασμα με αναφορά (pass-by-reference) γίνεται έμμεσα στην ουσία, περνώντας ως παραμέτρους των συναρτήσεων, δείκτες στις μεταβλητές των οποίων θέλουμε να αλλάζουμε τις τιμές μέσα από τις συναρτήσεις.

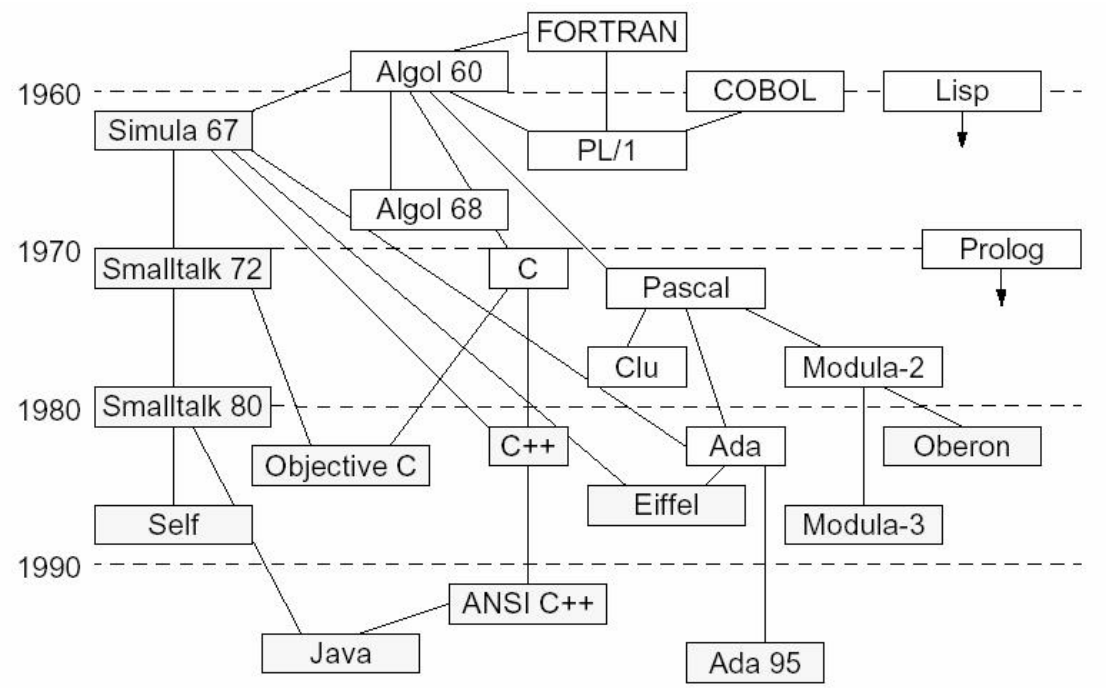
Εικόνα 9: Hello World - το Πρώτο Πρόγραμμα κάθε Προγραμματιστή !

Η C έχει ακόμη τα εξής χαρακτηριστικά :

- Έχει ένα πολύ μικρό σταθερό πλήθος λέξεων-κλειδιών (keywords), το οποίο περιλαμβάνει ένα πλήρες σύνολο δομών/εντολών ελέγχου ροής : for, if/else, while, switch, do/while και goto.
- Υπάρχει μόνο ένας χώρος ονομάτων (namespace) και τα ονόματα (μεταβλητών, συναρτήσεων, κ.τ.λ.) που ορίζονται από το χρήστη δεν διακρίνονται με κάποιο τρόπο από τις λέξεις-κλειδιά της γλώσσας.
- Υπάρχει ένα μεγάλο πλήθος αριθμητικών και λογικών τελεστών, όπως οι : +, +=, ++, -, -=, --, *, *=, /, /=, ==, &, &&, |, ||, ~, κ.ά.
- Σε μία εντολή μπορεί να γίνουν παραπάνω από μια εκχωρήσεις τιμών.
- Η τιμή που επιστρέφει μια συνάρτηση, μπορεί να αγνοηθεί εάν δεν χρειάζεται.
- Ο ορισμός των τύπων των μεταβλητών είναι στατικός και απαραίτητος, αλλά γίνονται έμμεσες μετατροπές από τη γλώσσα. Για παράδειγμα, παραστάσεις με τύπο χαρακτήρα, μπορούν να χρησιμοποιηθούν σε σημεία που απαιτείται ακέραιος.

- Η σύνταξη των δηλώσεων ονομάτων προσομοιάζει την χρήση αυτών μέσα στον εκτελέσιμο κώδικα. Η C δεν έχει ειδική λέξη-κλειδί για τον ορισμό ονομάτων ή συναρτήσεων. Μια γραμμή που ξεκινάει με το όνομα ενός τύπου, εκλαμβάνεται σαν ορισμός μεταβλητής ή συνάρτησης, ανάλογα με το αν υπάρχουν ή όχι παρενθέσεις που περικλείουν παραμέτρους συνάρτησης.
- Ο χρήστης μπορεί να ορίσει δικούς του τύπους, εάν το επιθυμεί. Μπορεί επίσης να ορίσει και τύπους εγγραφών (για παράδειγμα, struct στη C).
- Μπορούν να οριστούν πίνακες, αν και δεν υπάρχει ειδική λέξη-κλειδί για τον ορισμό τους. Η δεικτοδότηση τους γίνεται με την χρήση αγκυλών ([ ]), αν και πολύ συχνά γίνεται χρήση αριθμητικής δεικτών. Το πρώτο στοιχείο κάθε πίνακα δεικτοδοτείται πάντα από το μηδέν (0). Για παράδειγμα, το στοιχείο hello [0] είναι το πρώτο στοιχείο του πίνακα hello. Τέλος, δεν υπάρχουν τελεστές για την σύγκριση ή εκχώρηση πινάκων.
- Δεν υπάρχει ιδιαίτερος τύπος για αλφαριθμητικά, τα οποία υλοποιούνται και αντιμετωπίζονται σαν πίνακες από χαρακτήρες και έχουν έναν μηδενικό χαρακτήρα να σημαδεύει το τέλος τους (null-terminated arrays of characters).
- Είναι δυνατή η άμεση προσπέλαση χαμηλού επιπέδου στη μνήμη του υπολογιστή με τη χρήση δεικτών.
- Οι υπορουτίνες που δεν επιστρέφουν τιμή, είναι συναρτήσεις που ορίζονται να είναι τύπου « void » (είναι ψευδοτύπος που δείχνει την απουσία επιστρεφόμενης τιμής, αλλά χρησιμοποιείται και για δείκτες που δεν δείχνουν σε αντικείμενο συγκεκριμένου τύπου).
- Οι πολύπλοκες λειτουργίες, όπως οι λειτουργίες εισόδου/εξόδου, ο χειρισμός των αλφαριθμητικών, καθώς και οι μαθηματικές συναρτήσεις έχουν ανατεθεί στις αντίστοιχες βιβλιοθήκες.

Η C δεν διαθέτει κάποιες από τις δυνατότητες νεώτερων γλωσσών, όπως τον προσανατολισμό στα αντικείμενα και την συλλογή απορριμμάτων (garbage collection). [2]



Εικόνα 10: Ιστορική Αναδρομή Γλωσσών Προγραμματισμού

13 Βιβλιογραφία

[1] Λογικές Πράξεις και Δυαδική Λογική
http://masciences.blogspot.gr/2013/12/blog-post_27.html

[2] C (Γλώσσα Προγραμματισμού)
http://el.wikipedia.org/wiki/C_%28%CE%B3%CE%BB%CF%8E%CF%83%CF%83%CE%B1_%CF%80%CF%81%CE%BF%CE%B3%CF%81%CE%B1%CE%BC%CE%BC%CE%B1%CF%84%CE%B9%CF%83%CE%BC%CE%BF%CF%8D%29

[3] Σημειώσεις για Λογική από τον εισηγητή Παπαδάκη Νικόλαο.

14 Παράρτημα

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*
 * Dilwseis synarthsewn.
 */
void readVariables(char *);
void readOperators(char *,char *);
int finalCalc(char *,char *);

/*
 * Dilwseis global metavliton.
 */
int a=-1,b=-1,c=-1,d=-1,e=-1;
char x='y';

/*
 * Enarksh ths main.
 */

int main() {
	/*
	Xrisimopoioume thn while gia na mporoume na ksanakalesoume to programma, an theloume,
	gia na dwsoume ki alli parastasi.
	*/

	while(x == 'y')
	{


	/*
	 * Xreiazomaste enan pinaka gia na kataxwrisoume olokliri tin parastasi mazi me ta symvola
	 * gia tis treis logikes prakseis (&& [^] sizefksi,|| [v] diazefksi,-> sinepagwgi)
	 * Tha einai pinakas xarakthrwn giati tha pairnei symvola kai grammata.
	 * Efoson exoume maximum 5 metavlites pou pairnoun times 0-1 kai to kathe symvolo pianei
	 * xwro dyo xarakthrwn tote oi megistes theseis
	 * pou tha prepei na exei o pinakas pou tha krataei tin parastasi den tha einai parapanw
	 * apo 14 me to \0 (a&&b&&c&&d&&e).
	 */
	char parastasi[14];
```

```
        /*
         * Enas pinakas ston opoio tha apothikeuoume tin seira me tin opoia vriskoume tous logikous
         * telestes (&,|,-)
         */
        char opSequence[4];

        /*
         * Diavazoume tin parastasi.
         */
        printf("Dwste tin parastasi\n");
        gets(parastasi);

        /*
         * Xrisimopoioume mia synartisi pou tha ksexwrizei ta onomata ton metavliton apo ta symvola
         * twn logikwn praksewn kai tha tous dinei times.
         * Px. an dwsw "a&&b&&c" auto tha vriskei oti xrisimopoihsa 3 metavlites tin a tin b kai
         * tin c kai meta tha diavazei tis times tous kai tous tis ekxwrei me xrhsh deiktwn,
         * akoma vriskei tis prakseis pou tha prepei na ginoun anamesa tous kai tis pragmatopoiei.
         */
        readVariables(parastasi);

        /*
         * Ousiastika edw kanoume to idio pragma gia tous telestes me auto pou kaname gia tis
         * metavlites gia na tous exoume se mia swsth seira.++++++++++++++++++
         */
        readOperators(parastasi,opSequence);

        /*
         * Se auto to simeio exoume tis metavlites gemismenes me times kai tous operators se enan
         * pinaka opote xreiazomaste mia synartisi i opoia na kanei ton teliko ypologismo.
         */
        int result = finalCalc(parastasi,opSequence);

        printf("\n FINAL RESULT: %d \n", result);

        getchar();
        getchar();

        printf("thelete na dwsete alli parastasi ?? y or n ??\n");
        x=getch();
        }
```

```
	return 0;
}

void readVariables(char *p) {

	/*
	 * Edw vriskoume to megethos tou pinaka. ousiastika diladi an grapsw "a&&b" tha exei
	 * megethos 4 enw an grapsw "a&&b&&c" tha exei megethos 7.
	 */
	int pLength = strlen(p);

	/*
	 * Xreiazomaste enan metrhth gia na prospelasoume ton pinaka kai na vroume tis metavlites
	 * gia na tous dwsoume times.
	 */
	int i;

	/*
	 * Mia epanalipsi gia na diatreksoume olon ton pinaka.
	 */

	for(i=0; i<pLength; i++)
	{
		/*
		 * Epeidi o tropos pou dinoume tin parastasi einai sygkekrimenos diladi "a&&b" to a
		 * tha einai stin thesi miden kai meta ana treis theseis tha exoume metavlites afou
		 * to kathe symvolo apoteleitai apo dyo xaraktires opote xwrizoume tis metavlites apo
		 * ta simvola kai tous dinoume times.
		 */
		if (i%3==0) {

			/*
			 * Edw ousiastika elegxoume gia to poia metavliti prokeitai na xrisimopoihsoume
			 * etsi wste na tis dwsoume timh.
			 */

			if (p[i]=='a') {
				printf("Dwse timh gia tin metavliti %c: ",p[i]);
				scanf("%d",&a);
			}
			else if (p[i]=='b') {
				printf("Dwse timh gia tin metavliti %c: ",p[i]);
				scanf("%d",&b);
			}
```

```
                else if (p[i]=='c') {
                        printf("Dwse timh gia tin metavliti %c: ",p[i]);
                        scanf("%d",&c);
                }
                else if (p[i]=='d') {
                        printf("Dwse timh gia tin metavliti %c: ",p[i]);
                        scanf("%d",&d);
                }
                else if (p[i]=='e') {
                        printf("Dwse timh gia tin metavliti %c: ",p[i]);
                        scanf("%d",&e);
                }
        }

    }

}
void readOperators(char *p1, char *p2) {
    /*
     * Edw vriskoume to megethos tou pinaka. ousiastika diladi an grapsw "a&&b" tha exei
     * megethos 4 enw an grapsw "a&&b&&c" tha exei megethos 7.
     */
    int pLength = strlen(p1);

    /*
     * Xreiazomaste enan metrhth gia na prospelasoume ton pinaka kai na vroume tis metavlites
     * gia na tous dwsoume times.
     */
    int i;

    /*
     * Metrhths gia tin ekxwrhsh twn operators ston deftero pinaka.
     */
    int i2 = 0;

    /*
     * Mia epanalipsi gia na diatreksoume olon ton pinaka.
     */

    for(i=0; i<pLength; i++)
    {
        if (i%3==1) {
            /*
             * Vriskoume tous telestes me tin seira pou emfanizontai kai tous ekxwroume
             * se enan neo pinaka.
             */
            p2[i2] = p1[i];
```

```
                i2++;
            }
            printf("%c\n",p1[i]);
        }
}
int finalCalc(char *p1,char *p2) {

        /*
         * Se auth thn synarthsh ginontai oi telikes prakseis kai prokyptei to
apotelesma.
         */

        int pLength = strlen(p1);

        /*
         * Gia na doume poies apo tis metavlites tha xrisimopoihsoume arkei na
kseroume to
         * megethos tou pinaka diladi:
         * "a&&b"                    = megethos 4 gia 2 metavlites
         * "a&&b&&c"                 = megethos 7 gia 3 metavlites
         * "a&&b&&c&&d"     = megethos 10 gia 4 metavlites
         * "a&&b&&c&&d&&e"  = megethos 13 gia 5 metavlites
         */

        /*
         * To sum einai i metavliti pou tha krataei to teliko apotelesma
         */
        int sum;

        if (pLength == 4) {
            //a - p2[0] - b

            if (p1[1] == '&') {
               if (a == 1 && b == 1)
                    return 1;
                    else
                    return 0;
            }
            else if (p1[1] == '|') {
                    if (a == 0 && b == 0)
                    return 0;
                    else
                    return 1;
            }
            else if (p1[1] == '-') {
                    if (a == 0)
                    return 1;
               else if (b == 0)
                    return 0;
                    else if (b == 1)
```

```
                return 1;
        }
}
else if (pLength == 7) {
        //a - p2[0] - b - p2[1] - c

        if (p1[1] == '&' && p1[4] == '&')
        {
                if (a == 0)
                return 0;
                else if (b == 0)
                return 0;
                else if (c == 0)
                return 0;
                else
                return 1;
        }
        else if (p1[1] == '&' && p1[4] == '|')
        {
                if (c == 1)
                return 1;
                else if (a == 1 && b == 1)
                return 1;
                else
                return 0;
        }
        else if (p1[1] == '|' && p1[4] == '&')
        {
                if (a == 0 && b == 0)
                return 0;
                else if (c == 1)
                return 1;
                else
                return 0;
        }
        else if (p1[1] == '|' && p1[4] == '|')
        {
                if (a == 0 && b == 0 && c == 0)
                return 0;
                else
                return 1;
        }
        else if (p1[1] == '&' && p1[4] == '-')
        {
                if (a == 1 && b == 1 && c == 0)
                return 0;
                else
                return 1;
        }
        else if (p1[1] == '|' && p1[4] == '-')
```

```
        {
                if (a == 0 && b == 1 && c == 0)
                return 0;
                if (a == 1 && b == 0 && c == 0)
                return 0;
                if (a == 1 && b == 1 && c == 0)
                return 0;
                else
                return 1;
        }

}
else if (pLength == 10) {
        //a - p2[0] - b - p2[1] - c - p2[2] - d

        if(p1[1] == '&' && p1[4] == '&' && p1[7] == '&')
        {
                if(a == 1 && b == 1 && c == 1 && d == 1)
                return 1;
                else
                return 0;
        }
        if(p1[1] == '|' && p1[4] == '|' && p1[7] == '|')
        {
                if(a == 0 && b == 0 && c == 0 && d == 0)
                return 0;
                else
                return 1;
        }
        if(p1[1] == '&' && p1[4] == '&' && p1[7] == '|')
        {
                if(d == 1)
                return 1;
                else if(a == 1 && b == 1 && c == 1)
                return 1;
                else
                return 0;
    }
    if(p1[1] == '&' && p1[4] == '|' && p1[7] == '&')
        {
                if(c == 1 && d == 1)
                return 1;
                else if(a == 1 && b == 1 && c == 0 && d == 1)
                return 1;
                else
                return 0;
    }
    if(p1[1] == '&' && p1[4] == '|' && p1[7] == '|')
        {
                if(b == 0 && c == 0 && d == 0)
```

```
                return 0;
                else if(a == 0 && b == 1 && c == 0 && d == 0)
                return 0;
                else
                return 1;
}
if(p1[1] == '|' && p1[4] == '|' && p1[7] == '&')
        {
                if(a == 0 && b == 0 && c == 0 && d == 1)
                return 0;
                else if(d == 1)
                return 1;
                else
                return 0;
}
if(p1[1] == '|' && p1[4] == '&' && p1[7] == '|')
        {
                if(c == 0 && d == 0)
                return 0;
                else if(a == 0 && b == 0 && c == 1 && d == 0)
                return 0;
                else
                return 1;
}
if(p1[1] == '|' && p1[4] == '&' && p1[7] == '&')
        {
                if(a == 0 && b == 0 && c == 1 && d == 1)
                return 0;
                else if(c == 1 && d == 1)
                return 1;
                else
                return 0;
}
if(p1[1] == '&' && p1[4] == '&' && p1[7] == '-')
        {
                if(a == 1 && b == 1 && c == 1 && d == 0)
                return 0;
                else
                return 1;
        }
        if(p1[1] == '|' && p1[4] == '|' && p1[7] == '-')
        {
                if(a == 0 && b == 0 && c == 0)
                return 1;
                else if(d == 1)
                return 1;
                else
                return 0;
        }
        if(p1[1] == '&' && p1[4] == '|' && p1[7] == '-')
```

```
		{
			if(a == 1 && b == 1 && c == 0 && d == 0)
			return 0;
			else if (c == 1 && d == 0)
			return 0;
			else
			return 1;
		}
		if(p1[1] == '|' && p1[4] == '&' && p1[7] == '-')
		{
			if(a == 0 && b == 1 && c == 1 && d == 0)
			return 0;
			if(a == 1 && b == 0 && c == 1 && d == 0)
			return 0;
			if(a == 1 && b == 1 && c == 1 && d == 0)
			return 0;
			else
			return 1;
		}


	}

	return -1000;
}
```