ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

Διπλωματική Εργασία

Επεξεργαστής Ασαφούς Λογικής Για Συστήματα Ελέγχου

Γιάννης Αβρίθης
Επιβλέπων Καθηγητής: Γ. Τσιβίδης

ΑΘΗΝΑ, ΣΕΠΤΕΜΒΡΙΟΣ 1993

Πίνακας Περιεχομένων

Πρόλογος

Η *ασαφής λογική (fuzzy logic)* είναι ένας σχετικά πρόσφατος κλάδος των μαθηματικών που δίνει τη δυνατότητα στους υπολογιστές να μοντελοποιούν τον πραγματικό κόσμο με ένα τρόπο παρόμοιο με αυτό που χρησιμοποιούν οι άνθρωποι. Σε αντίθεση με τους υπολογιστές, οι άνθρωποι σκέφτονται χρησιμοποιώντας περισσότερο γλωσσικούς όρους όπως "ζεστός" ή "γρήγορος", παρά ακριβείς αριθμητικές έννοιες όπως "40 βαθμοί Κελσίου" ή "100 χιλιόμετρα ανά ώρα". Ετσι, σε αντίθεση με την κλασσική λογική, ο κύριος σκοπός της ασαφούς λογικής είναι να μοντελοποιήσει τον *ανακριβή* τρόπο σκέψης που παίζει σημαντικό ρόλο στην ικανότητα του ανθρώπου να λαμβάνει ορθολογικές αποφάσεις μέσα σε ένα αβέβαιο και διαρκώς μεταβαλλόμενο περιβάλλον. Αυτή η ικανότητα, με τη σειρά της, εξαρτάται από την ικανότητά μας να εξάγουμε συμπεράσματα βασιζόμενοι στη συσσώρευση γνώσης και εμπειρίας που είναι ανακριβής, ελλιπής, ασαφής ή αναξιόπιστη.

Η ασαφής λογική είναι στην πραγματικότητα ένα υπερσύνολο της συμβατικής ή "δίτιμης" λογικής που βασίζεται στην άλγεβρα Boole. Η επέκταση έγκειται στο γεγονός ότι η ασαφής λογική έχει τη δυνατότητα να μεταχειρίζεται την έννοια της *μερικής αλήθειας* – δηλαδή τις τιμές αλήθειας που βρίσκονται μεταξύ του "απόλυτα αληθούς" και του "απόλυτα ψευδούς". Πρώτος ο Πολωνός Jan Lukasiewicz [29] στις αρχές της δεκαετίας του '30 εισήγαγε την έννοια της "τρίτιμης" λογικής και στην συνέχεια επέκτεινε το διάστημα των τιμών αλήθειας από το {0, 1/2, 1} στο σύνολο όλων των πραγματικών αριθμών που περιέχονται στο [0,1]. Το 1937, ο Max Black [49] εφάρμοσε τη "συνεχή" λογική του Lukasiewicz σε σύνολα, ενώ το 1965 ο Lofti Zadeh [4] ανέπτυξε τη θεωρία των ασαφών συνόλων και εισήγαγε τον όρο *ασαφής (fuzzy)* στην τεχνική ορολογία. Γι'αυτό και ο Zadeh θεωρείται ο "πατέρας" της ασαφούς λογικής.

Το πραγματικό όμως ενδιαφέρον για την ασαφή λογική δεν εκδηλώθηκε παρά μόλις την τελευταία δεκαετία, όταν πλέον τα οφέλη της άρχισαν να γίνονται αισθητά όχι μόνο στη διεθνή επιστημονική κοινότητα, αλλά και στις βιομηχανίες και στο ευρύτερο κοινό σε ολόκληρο τον κόσμο. Μετά την πρώτη της βιομηχανική εφαρμογή στον έλεγχο μιάς ατμομηχανής το 1973 από τον E. Mamdani [12], η ασαφής λογική χρησιμοποιείται σήμερα σε μία πληθώρα περιοχών έρευνας, όπως τα έμπειρα συστήματα, ο αυτόματος έλεγχος, η τεχνητή νοημοσύνη, η επεξεργασία σημάτων και εικόνων, η όραση υπολογιστών, η αναγνώριση προτύπων, τα ευφυή ρομποτικά συστήματα, η ιατρική διάγνωση, τα οικονομικά συστήματα, τα συστήματα

αποφάσεων και αμέτρητα άλλα. Η εφαρμογή όμως που έδωσε τη μεγαλύτερη ώθηση στην έρευνα πάνω στην ασαφή λογική είναι αναμφισβήτητα ο αυτόματος έλεγχος, αφού είναι η πρώτη εφαρμογή που έδωσε χειροπιαστά αποτελέσματα ως προς την ανωτερότητα της ασαφούς λογικής σε σχέση με τις παραδοσιακές μεθόδους ελέγχου. Παραδείγματα τέτοιων εντυπωσιακών εφαρμογών είναι οι βιομηχανικές εφαρμογές όπως ο έλεγχος χημικών διεργασιών (π.χ. παραγωγή τσιμέντου) [1, 7, 9, 45], η αυτόματη λειτουργία τρένων [16] και ο έλεγχος ανελκυστήρων [25], κινητήρων [55] και ρομποτικών συστημάτων [31], οι εφαρμογές στην αυτοκινητοβιομηχανία [26], όπως τα συστήματα ψεκασμού, η ενεργητική ανάρτηση, τα αντιολισθητικά φρένα, ακόμα και η αυτόματη οδήγηση αυτοκινήτου [32, 34], καθώς και η παραγωγή "έξυπνων" καταναλωτικών προϊόντων [25, 26] από βιντεοκάμερες και συστήματα κλιματισμού μέχρι πλυντήρια και ηλεκτρικές σκούπες που λειτουργούν με ασαφή λογική[1].

Σε όλα τα παραπάνω παραδείγματα η ασαφής λογική ξεπερνά σε επιδόσεις τις συμβατικές μεθόδους από πλευράς ακρίβειας, ευελιξίας, οικονομίας και ομαλής απόκρισης σε γρήγορα μεταβαλλόμενες συνθήκες. Η επιτυχία των συστημάτων αυτών οφείλεται σε δύο κυρίως λόγους. Ο πρώτος είναι ότι η λειτουργία τους βασίζεται σε *κανόνες* [11] και όχι σε κάποιο μαθηματικό μοντέλο. Ετσι, σε πολύπλοκες βιομηχανικές διαδικασίες ελέγχου για παράδειγμα, ένας ειδικός που χειρίζεται τη διαδικασία μπορεί ευκολότερα να εκφράσει λεκτικά τι κάνει, παρά με μαθηματικούς όρους, και η ασαφής λογική δίνει τη δυνατότητα κωδικοποίησης της γνώσης και της εμπειρίας του χειριστή σε γλωσσικούς κανόνες. Ο δεύτερος λόγος είναι ότι αυτό που μετράει περισσότερο σε τέτοιες εφαρμογές δεν είναι η *ακρίβεια*, αλλά η δυνατότητα λήψης αποφάσεων μέσα σε αβεβαιότητα. Οταν, για παράδειγμα, κάποιος μαθαίνει να οδηγεί, ο δάσκαλος του ποτέ δεν του λέει "άρχισε να φρενάρεις *30 μέτρα* πριν το φανάρι", αλλά "άρχισε να φρενάρεις *έγκαιρα*".

Η επιτυχής εφαρμογή της ασαφούς λογικής στα έμπειρα συστήματα και τα συστήματα ελέγχου έχει δημιουργήσει τα τελευταία χρόνια την ανάγκη ανάπτυξης ενός νέου τύπου hardware για την υλοποίηση των μεθόδων της ασαφούς λογικής. Η ασαφής λογική είναι από τη φύση της *"αναλογική"* - αφού χρησιμοποιεί τιμές αλήθειας στο συνεχές διάστημα [0,1] - και επομένως η υλοποίηση με ψηφιακούς υπολογιστές και μικροεπεξεργαστές δεν είναι η καταλληλότερη, σε αντίθεση με τα αναλογικά κυκλώματα VLSI που είναι αποτελεσματικότερα όσον αφορά την ταχύτητα των υπολογισμών, το μέγεθος και την πολυπλοκότητα των κυκλωμάτων, την κατανάλωση και το κόστος σχεδίασης και κατασκευής. Γι'αυτό το λόγο έχει ξεκινήσει από την ομάδα μικροηλεκτρονικής του Ε.Μ.Π. ένα ερευνητικό πρόγραμμα πάνω στην ασαφή λογική με τελικό στόχο την κατασκευή σε ολοκληρωμένο κύκλωμα ενός προγραμματιζόμενου "ασαφούς" μικροεπεξεργαστή *(fuzzy microprocessor)* γενικής χρήσης που θα εκτελεί πράξεις ασαφούς λογικής.

---

[1] Περισσότερες λεπτομέρειες σχετικά με τις εφαρμογές της ασαφούς λογικής μπορουν να βρεθούν στα [1], [9], [16], [22], [25], [26], [32], [34], [38], [45], [47], [53], [55] και [68].

Παράλληλα με τη σχεδίαση του μικροεπεξεργαστή, έχει ήδη δημιουργηθεί στα πλαίσια μιας διπλωματικής εργασίας [3] το απαραίτητο software που δίνει τη δυνατότητα προσομοίωσης συστημάτων κανόνων έτσι ώστε να βοηθηθεί η ανάπτυξη εφαρμογών σε hardware ή software. Πρόκειται για το *"Fuzzy expert"* ένα ολοκληρωμένο πρόγραμμα που λειτουργεί σε περιβάλλον X Windows, είναι πολύ φιλικό προς το χρήστη, έχει σύστημα παραθύρων, pull-down menus κλπ., και επιτρέπει στο χρήστη την εξαγωγή συμπερασμάτων για τις διάφορες παραλλαγές των μεθόδων που χρησιμοποιούνται και την εύκολη ανάπτυξη του καταλληλότερου ασαφούς συστήματος για την κάθε εφαρμογή.

Αυτό που έγινε σαφές από τη χρήση του προγράμματος Fuzzy Expert ήταν ότι η πληθώρα των τύπων των πράξεων που είναι διαθέσιμες σε κάθε στάδιο των υπολογισμών και η απειρία σχεδόν των δυνατών συνδυασμών όλων των τύπων πράξεων επιβάλλει μία θεωρητική μελέτη για το πώς πρέπει να συνδυάζονται οι πράξεις αυτές μεταξύ τους ώστε να προκύπτει ένα "συνεπές" λογικό σύστημα. Αυτή η θεωρητική μελέτη, καθώς και η αναζήτηση νέων, απλούστερων μεθόδων υπολογισμού και η διερεύνηση των προϋποθέσεων κάτω από τις οποίες οι μέθοδοι αυτές μπορούν να εφαρμοστούν, έγινε στα πλαίσια της παρούσας διπλωματικής εργασίας και παρουσιάζεται αναλυτικά στο Μέρος Α.

Το πιό εντυπωσιακό όμως τμήμα της δουλειάς που έγινε στη διπλωματική αυτή εργασία, και που παρουσιάζεται στο Μέρος Β, είναι η ανάλυση, σχεδίαση και πλήρης πρακτική κατασκευή ενός συστήματος ελέγχου που λειτουργεί με ασαφή λογική. Συγκεκριμένα, πρόκειται για έναν ασαφή, βασισμένο σε κανόνες ρυθμιστή, που μετακινεί μια σφαίρα (μπάλα του ping-pong) στο εσωτερικό ενός κατακόρυφου, πλαστικού και διάφανου σωλήνα. Η μετακίνηση γίνεται μεταξύ δύο προκαθορισμένων σημείων μεταβάλλοντας την τάση που εφαρμόζεται σε έναν ανεμιστήρα που βρίσκεται στερεωμένος στο κάτω μέρος του σωλήνα. Σκοπός του πειράματος είναι η μετακίνηση της μπάλας μεταξύ των δύο σημείων να γίνεται στο μικρότερο δυνατό χρόνο και με τις λιγότερες δυνατές ταλαντώσεις. Η υλοποίηση του κυκλώματος ελέγχου, δηλαδή του *επεξεργαστή ασαφούς λογικής (fuzzy processor)* έγινε με διακριτά στοιχεία του εμπορίου ενώ όλοι οι υπολογισμοί γίνονται εξ'ολοκλήρου με αναλογικές μεθόδους χωρις να μεσολαβεί μετατροπή A/D ή D/A σε κανένα στάδιο της επεξεργασίας. Η μέθοδος που ακολουθήθηκε είναι παρόμοια με εκείνη που ακολουθείται στη σχεδίαση του ασαφούς μικροεπεξεργαστή που γίνεται στα πλαίσια μιας διδακτορικής διατριβής στο Εργαστήριο Ηλεκτρονικής Ε.Μ.Π.

Το πείραμα αυτό, όπως είναι προφανές, δεν επιλέχθηκε με κριτήριο τις πιθανές εφαρμογές που θα μπορούσε να έχει. Αντίθετα, είχε καθαρά χαρακτήρα επίδειξης και η σκοπιμότητά του ήταν διπλή: από τη μία να γίνει φανερή η δυνατότητα υλοποίησης μεθόδων της ασαφούς λογικής με πλήρως αναλογικά κυκλώματα, και από την άλλη να κατασκευαστεί ένα απλό σχετικά σύστημα ελέγχου έτσι ώστε όταν τελικά υλοποιηθεί ο μικροεπεξεργαστής να μπορεί να δοκιμαστεί αμέσως στην πράξη και νά πάρουμε συγκριτικά αποτελέσματα.

Με μεγάλη ευχαρίστηση θα ήθελα να αναγνωρίσω την προσφορά του Αγαμέμνωνα Βρεττού και ιδιαίτερα του αδερφού μου Παναγιώτη Αβρίθη. Η βοήθειά τους υπήρξε πολύτιμη στη συγγραφή αυτής της διπλωματικής αυτής εργασίας. Επίσης θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στη Μαρία Πίκη και στους γονείς μου για την πολύτιμη συμπαράστασή τους στο δύσκολο αυτό έργο.

ΜΕΡΟΣ Α

ΘΕΩΡΙΑ ΑΣΑΦΟΥΣ ΛΟΓΙΚΗΣ

Εισαγωγή

Οι "μύθοι" περί ασαφούς λογικής

Στη σύντομη σχετικά ιστορία της η ασαφής λογική έχει συναντήσει εμπόδια, έχει δεχθεί επιθέσεις, έχει παρεξηγηθεί και αγνοηθεί όσο καμμία άλλη θεωρία. Τις περισσότερες φορές οι επιθέσεις γίνονται από ανθρώπους που γνωρίζουν ελάχιστα γι'αυτήν. Ενας από τους κυριότερους λόγους για την αντίσταση που δέχθηκε μέχρι να καθιερωθεί στη διεθνή επιστημονική κοινότητα, είναι ότι έρχεται σε αντίθεση με την κλασσική Δυτική λογική της "απόλυτης αλήθειας" και του "απόλυτου ψεύδους". Γι'αυτό το λόγο κρίθηκε σκόπιμο στο σημείο αυτό και πριν αναπτυχθεί η θεωρία της ασαφούς λογικής να ξεκαθαριστούν ορισμένοι *"μύθοι"* [23] που επικρατούν σχετικά με την ασαφή λογική.

Κατ'αρχήν, ο μύθος που σχετίζεται με τ'όνομά της : *Η ασαφής λογική είναι "ασαφής"*. Οταν κανείς ακούει για ένα "ασαφές" σύστημα, το μυαλό του αμέσως πάει σε κάτι το θολό, το ακατανόητο, το μυστήριο. Αντίθετα όμως, παρά το γεγονός ότι σκοπός της Α.Σ. είναι να μοντελοποιήσει τον ανακριβή τρόπο σκέψης του ανθρώπου, οι μέθοδοι που χρησιμοποιεί είναι απολύτως σαφείς και καθορισμένες και εκφράζονται με μαθηματική ακρίβεια, όπως θα γίνει αμέσως κατανοητό στα επόμενα δύο κεφάλαια.

Ο δεύτερος μύθος : *"Η ασαφής λογική είναι μία ακόμα μορφή πιθανότητας"* είναι εκείνος που έχει προξενήσει τη μεγαλύτερη αντίδραση προς την ασαφή λογική. Σύμφωνα με τον M.Gupta [40] υπάρχουν δύο είδη αβεβαιότητας: Η αβεβαιότητα *U-Type One*, η οποία έχει να κάνει με φαινόμενα που προκύπτουν από την τυχαία συμπεριφορά των φυσικών συστημάτων (όπως π.χ. η τυχαία κίνηση των μορίων του αέρα), και η οποία έχει μελετηθεί επιτυχώς επί αιώνες με βάση τη θεωρία πιθανοτήτων και τη στατιστική, και η αβεβαιότητα *U-Type Two* που σχετίζεται με φαινόμενα όπως η ανθρώπινη σκέψη και αντίληψη (όπως π.χ. "αυτό το λουλούδι είναι όμορφο"), η οποία έχει ελάχιστα μελετηθεί και στην οποία η θεωρία πιθανοτήτων δε θα μπορούσε να δώσει κανένα αξιόλογο αποτέλεσμα. Αυτό ακριβώς το κενό έρχεται να καλύψει η Α.Σ.

Ενα παράδειγμα θα δείξει καλύτερα τη διαφορά. Ας θεωρήσουμε ότι υπάρχει 50% πιθανότητα να υπάρχει ένα μήλο στο ψυγείο, και ας θεωρήσουμε ότι υπάρχει μισό μήλο στο ψυγείο. Οι δύο καταστάσεις είναι επιφανειακά ισοδύναμες ως προς την αριθμητική τους αβεβαιότητα – και στις δύο περιπτώσεις η αβεβαιότητα εκφράζεται με αριθμούς στο μοναδιαίο διάστημα [0,1]. Ομως οντολογικά οι δύο καταστάσεις διαφέρουν. Η μία είναι τυχαία, η άλλη είναι ασαφής.

Κεφάλαιο 1

ΑΣΑΦΗ ΣΥΝΟΛΑ

1.1 Κλασσικά και ασαφή σύνολα

Στην κλασσική θεωρία συνόλων, ένα *σύνολο (set)* ορίζεται ως συλλογή ορισμένων αντικειμένων, δηλαδή αποτελείται από ένα αριθμό - άπειρο ή πεπερασμένο - *στοιχείων (elements)*, ενώ τα στοιχεία όλων των συνόλων υπό μελέτη ανήκουν σε ένα σταθερό *υπερσύνολο αναφοράς (universe of discourse)*, το οποίο μπορεί να είναι είτε διακριτό είτε συνεχές. Η παραδοσιακή θεωρία συνόλων επιβάλλει αυστηρά κριτήρια συμμετοχής των στοιχείων σε ένα σύνολο. Ενα στοιχείο δηλαδή, είτε ανήκει, είτε δεν ανήκει σε ένα σύνολο χωρίς να υπάρχουν ενδιάμεσες καταστάσεις. Ετσι ένα σύνολο Α μπορεί να περιγραφεί με τη χαρακτηριστική του συνάρτηση μ_A:

$$\mu_A(x) = \begin{cases} 1, & \alpha\nu \quad x \in A \\ 0, & \alpha\nu \quad x \notin A \end{cases} \tag{1.1}$$

Για ένα κλασσικό σύνολο λοιπόν, η χαρακτηριστική συνάρτηση είναι της μορφής:

$$\mu_A(x) : X \rightarrow \{0, 1\} \tag{1.2}$$

όπου Χ είναι το υπερσύνολο αναφοράς.

Στη θεωρία ασαφών συνόλων ο *βαθμός συμμετοχής* $\mu_A(x)$ ενός στοιχείου x στο σύνολο Α μπορεί να παίρνει άπειρο αριθμό τιμών μέσα από το συνεχές κλειστό διάστημα [0,1]. Ετσι ένα *ασαφές σύνολο* μπορεί να παρασταθεί σαν ένα σύνολο διατεταγμένων ζευγών

$$A = \{\mu_A(x)/x \mid x \in X\} \tag{1.3}$$

όπου πλέον η μ_A (x) καλείται *συνάρτηση συμμετοχής (membership function)* του x στο σύνολο A και είναι της μορφής :

$$\mu_A(x) : X \rightarrow [0,1] \qquad (1.4)$$

Στην κλασσική λογική η αλήθεια μιας πρότασης συνήθως ανάγεται στον υπολογισμό ενός βαθμού συμμετοχής ενός στοιχείου σε κάποιο (κλασσικό) σύνολο. Ετσι, για παράδειγμα, η πρόταση " ο Γιάννης είναι ψηλός" είναι αληθής αν και μόνο αν το άτομο που συμβολίζεται "Γιάννης" ανήκει στο σύνολο των "ψηλών ανθρώπων". Αντίθετα στην ασαφή λογική η παραπάνω πρόταση επιτρέπεται να είναι *μερικώς* αληθής και η *τιμή αλήθειας* της (truth value) ορίζεται σαν το βαθμό συμμετοχής του ατόμου "Γιάννης" στο ασαφές πλέον σύνολο "ψηλοί" άνθρωποι. *Επομένως η επέκταση των κλασσικών συνόλων σε ασαφή είναι ισοδύναμη με την επέκταση της κλασσικής λογικής σε ασαφή.* Η ισοδυναμία θα γίνει περισσότερο φανερή παρακάτω με τον ορισμό πράξεων με ασαφή σύνολα.

Στο σχήμα 1.1 φαίνονται οι συναρτήσεις συμμετοχής των παραδοσιακών και των κλασσικών συνόλων που αντιστοιχούν στις έννοιες ΚΟΝΤΟΣ και ΨΗΛΟΣ. Με A, B συμβολίζονται τα ασαφή σύνολα "ψηλοί άνθρωποι" και "κοντοί άνθρωποι" αντίστοιχα. Παρατηρούμε την απότομη μετάβαση που συμβαίνει στα παραδοσιακά σύνολα στο ύψος 1,80 m., η οποία σύμφωνα με τα κριτήρια της καθημερινής ζωής είναι πολλές φορές αφύσικη και δημιουργεί αμφισβητούμενες καταστάσεις σχετικά με το αν π.χ. ενας άνθρωπος με ύψος 1.79 m είναι ψηλός ή όχι.

Αντίθετα στα ασαφή σύνολα παρατηρούμε μια *βαθμιαία μετάβαση*, ένα φαινόμενο τόσο συνηθισμένο στην καθημερινή ζωή που μοντελοποιείται μαθηματικά χάρη στην έννοια των ασαφών συνόλων. Αμφισβητούμενες καταστάσεις δεν υπάρχουν στα ασαφή σύνολα, αφού ένας άνθρωπος επιτρέπεται πλέον να είναι *ταυτόχρονα μερικώς* ΚΟΝΤΟΣ και *μερικώς* ΨΗΛΟΣ. Πρέπει να τονίσουμε ότι η μορφή των καμπυλών που παριστάνουν τις συναρτήσεις συμμετοχής των ασαφών συνόλων εκφράζει ουσιαστικά μία *υποκειμενική* έννοια (π.χ. την έννοια ΨΗΛΟΣ) και επομένως πρέπει να επιλέγεται κατάλληλα ώστε να ανταποκρίνεται στην πραγματικότητα. Ανάλογα με την κάθε εφαρμογή, έχουν προταθεί διάφοροι τύποι συναρτήσεων που παρουσιάζονται σε επόμενο κεφάλαιο.

1.2 Πράξεις πάνω σε ασαφή σύνολα

Εστω A, B δύο ασαφή σύνολα με συναρτήσεις συμμετοχής $\mu_A(x)$, $\mu_B(x)$ αντίστοιχα, όπου $x \in X$. Ορίζονται τότε οι παρακάτω πράξεις :

1) **Ενωση (union)** των A και B είναι το ασαφές σύνολο $A \cup B$ με συνάρτηση συμμετοχής:

$$\mu_{A\cup B}(x) = \mu_A(x) \vee \mu_B(x) = \max(\mu_A(x), \mu_B(x)) \quad \forall x \in X \tag{1.5}$$

όπου με το σύμβολο $\vee$ παριστάνουμε τον συντελεστή ***maximum***. Η πράξη αυτή αντιστοιχεί στο *λογικό OR (διαζευκτικό Η)* της ασαφούς λογικής και είναι μία γενίκευση του λογικού OR της άλγεβρας Boole, στην οποία $\mu_A(x)$, $\mu_B(x) \in \{0,1\}$.

2) **Τομή (intersection)** των A και B είναι το ασαφές σύνολο $A \cap B$ με συνάρτηση συμμετοχής:

$$\mu_{A\cap B}(x) = \mu_A(x) \wedge \mu_B(x) = \min(\mu_A(x), \mu_B(x)) \quad \forall x \in X \tag{1.6}$$

όπου με το σύμβολο "$\wedge$" παριστάνουμε τον τελεστή ***minimum***. Η πράξη αυτή αντιστοιχεί στο λογικό "*AND (ΚΑΙ)*" της ασαφούς λογικής και είναι επίσης μία γενίκευση του λογικού AND της άλγεβρας Boole. Οι πράξεις AND και OR δύο τυχαίων συνόλων A και B φαίνονται στο σχήμα 1.2.

3) **Συμπλήρωμα (complement)** του A είναι το ασαφές σύνολο A^c με συνάρτηση συμμετοχής:

$$\mu_{A^c}(x) = 1 - \mu_A(x) \quad \forall x \in X \tag{1.7}$$

Το συμπλήρωμα A^c του A εικονίζεται στο σχήμα 1.3.

4) Το σύνολο A λέγεται **κενό** αν και μόνο αν η συνάρτηση συμμετοχής του είναι μηδέν παντού στο X:

$$A = \varnothing \leftrightarrow \mu_A(x) = 0 \quad \forall x \in X \tag{1.8}$$

5) Τα A,B είναι **ίσα** αν και μόνο αν οι συναρτήσεις συμμετοχής τους είναι ίσες:

$$A = B \leftrightarrow \mu_A(x) = \mu_B(x) \quad \forall x \in X \tag{1.9}$$

6) **Μέτρο ομοιότητας (consistency)** των A και B. Σε αντίθεση με την ισότητα δύο ασαφών συνόλων που είναι μία αυστηρή έννοια (είτε ισχύει είτε όχι) μπορεί να οριστεί ένα *ασαφές* μέτρο ομοιότητας των δύο συνόλων σαν μία συνάρτηση των $\mu_A(x)$, $\mu_B(x)$ που παίρνει τιμές στο διάστημα [0,1]. Η συνάρτηση αυτή παίρνει τιμές κοντά στη μονάδα όταν τα A,B "μοιάζουν" μεταξύ τους και κοντά στο μηδέν όταν τα A,B "διαφέρουν". Διάφορες συναρτήσεις έχουν οριστεί για το σκοπό αυτό και παρουσιάζονται σε επόμενο κεφάλαιο.

7) Το A λέγεται **υποσύνολο (subset)** του B αν και μόνο η συνάρτηση συμμετοχής του είναι μικρότερη ή ίση αυτής του B παντού στο X :

$$A \subset B \Leftrightarrow \mu_A(x) \leq \mu_B(x) \quad \forall x \in X \tag{1.10}$$

8) **Ασαφή υποσύνολα.** Οπως και προηγουμένως, η έννοια του υποσυνόλου που μόλις ορίστηκε είναι *σαφής* (είτε ισχύει είτε όχι). Γι'αυτό ορίζεται η έννοια του *ασαφούς υποσυνόλου (fuzzy subsethood)* [8] ως εξής : Το A είναι ασαφές υποσύνολο του B σε βαθμό S(A,B):

$$S(A,B) = \frac{M(A \cap B)}{M(A)} \tag{1.11}$$

όπου M(A) το *μέτρο* του ασαφούς συνόλου A :

$$M(A) = \sum_{x_i \in X} \mu_A(x_i) \tag{1.12}$$

για διακριτό υπερσύνολο αναφοράς X και

$$M(A) = \int_{x \in X} \mu_A(x) dx \tag{1.13}$$

για συνεχές X. Σύμφωνα με τον ορισμό του, το S(A,B) παίρνει κι αυτό τιμές στο διάστημα [0,1], είναι πάντα ίσο με 1 όταν $A \subset B$ και μικρότερο από 1 όταν $A \not\subset B$. Παρατηρούμε ότι όταν $B \subset A$, τότε $S(A,B) \neq 0$, δηλ. ένα υπερσύνολο του B είναι ταυτόχρονα και υποσύνολό του σε κάποιο βαθμό, γεγονός που αντιτίθεται στην κλασσική λογική.

9) **Εντροπία (Entropy)** [8] του A ορίζεται η συνάρτηση E(A) : X → [0,1]:

$$E(A) = \frac{M(A \cap A^c)}{M(A \cup A^c)} \tag{1.14}$$

Η εντροπία είναι ένα μέτρο της "ασάφειας" ενός ασαφούς συνόλου και ισούται με μηδέν όταν το A είναι σαφές ($A \cap A^c = \emptyset$, $M(A \cap A^c) = 0$) και 1 όταν το A είναι τελείως ασαφές ($\mu_A(x) = 1/2$ $\forall x \in X$, $A = A^c$). Οπως προκύπτει από τις (1.11) και (1.14),

$$E(A) = S(A \cup A^c, A \cap A^c) \tag{1.15}$$

(Entropy - Subethood Theorem, [8]). Δηλαδή η εντροπία του A ισούται με το βαθμό στον οποίο το υπερσύνολο $A \cup A^c$ είναι υποσύνολο του υποσυνόλου του $A \cap A^c$, μία σχέση που η κλασσική λογική απαγορεύει.

10) **Συμπύκνωση (concentration)** του A ονομάζεται το ασαφές σύνολο CON(A) με συνάρτηση συμμετοχής:

$$\mu_{CON(A)}(x) = [\mu_A(x)]^2 \quad \forall x \in X \tag{1.16}$$

11) **Διαστολή (dilation)** του A ονομάζεται το ασαφές σύνολο DIL(A) με συνάρτηση συμμετοχής:

$$\mu_{DIL(A)}(x) = \sqrt{\mu_A(x)} \quad \forall x \in X \tag{1.17}$$

Οι πράξεις της συμπύκνωσης και της διαστολής εικονίζονται στο σχήμα 1.4. Η συμπύκνωση έχει την ιδιότητα να "στενεύει" την έννοια που παριστάνει ένα ασαφές σύνολο ενώ η διαστολή τη "γενικεύει". Έτσι π.χ. αν το A παριστάνει την έννοια "ΨΗΛΟΣ" τότε το CON(A) παριστ. την έννοια "ΠΟΛΥ ΨΗΛΟΣ" ενώ το DIL(A) την έννοια "ΣΧΕΤΙΚΑ ΨΗΛΟΣ"

12) **Αύξηση αντίθεσης (intensification)** του A συμβολίζεται INT(A) και ορίζεται ως:

$$\mu_{INT(A)} = \begin{cases} 2[\mu_A(x)]^2, & 0 \le \mu_A(x) \le 1/2 \\ 1-2[1-\mu_A(x)]^2, & 1/2 \le \mu_A(x) \le 1 \end{cases} \tag{1.18}$$

Η πράξη της αύξησης αντίθεσης, που εικονίζεται στο σχήμα 1.5, έχει την ιδιότητα να κάνει μία έννοια πιο "αυστηρή". Αν π.χ. A σημαίνει "ΨΗΛΟΣ" τότε INT(A) θα σημαίνει " ΟΠΩΣΔΗΠΟΤΕ ΨΗΛΟΣ".

13) **Ασαφοποίηση (fuzzification)**. Η ασαφοποίηση έχει σαν αποτέλεσμα τη μετατροπή ενος κλασσικού (σαφούς) συνόλου σε ασαφές ή την αύξηση του βαθμού ασάφειας (της *εντροπίας*) ενός ασαφούς συνόλου. Χαρακτηρίζεται από ενα σταθερό ασαφές σύνολο K(x) που ονομάζεται *πυρήνας (kernel)*, συμβολίζεται F(A;K), και ορίζεται μόνο στην περίπτωση που το X είναι διακριτό. Η πράξη της ασαφοποίησης απεικονίζεται στο σχήμα 1.6. Ο ορισμός της, καθώς και παραδείγματα εφαρμογής της, δίνονται στα [1], [28].

Πρέπει να τονίσουμε ότι οι περισσότερες από τις πράξεις που ορίστηκαν στην παράγραφο αυτή ορίζονται στη σχετική βιβλιογραφία και με πολλούς ακόμη διαφορετικούς τρόπους. Οι τρόποι αυτοί, καθώς και οι στρατηγικές συνδυασμού τους, μελετώνται σε επόμενο κεφάλαιο.

1.3 Ασαφής Λογική και Αλγεβρα Boole

Με βάση τους ορισμούς που δόθηκαν στην προηγούμενη παράγραφο για την ένωση, την τομή και το συμπλήρωμα των ασαφών συνόλων μπορούμε εύκολα να δείξουμε ότι οι περισσότερες από τις ιδιότητες που ισχύουν στην άλγεβρα Boole - που αντιπροσωπεύει μαθηματικά την κλασσική "δίτιμη" λογική [67] - ισχύουν επίσης και για τα ασαφή σύνολα, και επομένως και για την ασαφή λογική. Λόγω της *"αρχής του δυϊσμού" (duality principle)* οι ιδιότητες παρουσιάζονται σε ζεύγη συμπληρωματικών ή "δυϊκών" μεταξύ τους προτάσεων.

Για κάθε $A,B,C \subset X$ ισχύει :

1) α) **Κλειστότητα** ως προς πράξη $\cap$.
   b) **Κλειστότητα** ως προς πράξη $\cup$.

2) **Ουδέτερο στοιχείο:** α) $A \cup \emptyset = \emptyset \cup A = A$
   b) $A \cap X = X \cap A = A$

3) **Αντιμεταθετικότητα:** α) $A \cup B = B \cup A$
   b) $A \cap B = B \cap A$

4) **Επιμεριστικότητα:** α) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
   b) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

5) **Αδύναμη πράξη:** α) $A \cup A = A$
   b) $A \cap A = A$

6) α) $A \cup X = X$
   b) $A \cap \emptyset = \emptyset$

7) **Διπλή άρνηση:** $(A^c)^c = A$

8) **Προσεταιριστικότητα:** α) $A \cup (B \cup C) = (A \cup B) \cup C$
   b) $A \cap (B \cap C) = (A \cap B) \cap C$

9) **Θεώρημα De Morgan:** α) $(A \cup B)^c = A^c \cap B^c$
   b) $(A \cap B)^c = A^c \cup B^c$

10) **Απορρόφηση:** α) $A \cup (A \cap B) = A$
    b) $A \cap (A \cup B) = A$

Υπενθυμίζουμε ότι με $\emptyset$ συμβολίζουμε το κενό σύνολο και X το υπερσύνολο αναφοράς. Οι τέσσερις πρώτες ιδιότητες είναι αξιώματα της άλγεβρας Boole ενώ

οι υπόλοιπες έξι θεωρήματα. Ουσιαστικά η μόνη διαφορά μεταξύ κλασσικής και ασαφούς λογικής είναι ένα αξίωμα της ασαφούς λογικής που εκφράζει το *"νόμο της μη - αντίφασης" (law of noncontradiction)* $A \cap A^c = \emptyset$ και το *"νόμο του αποκλειόμενου μέσου" (law of excluded middle)* $A \cup A^c = X$. Στην ασαφή λογική αυτοί οι δύο νόμοι καταπατούνται [8]:

$$A \cap A^c \neq \emptyset \qquad (1.19)$$

$$A \cup A^c \neq X \qquad (1.20)$$

Το γεγονός αυτό δεν είναι καθόλου τυχαίο και στην πραγματικότητα εκφράζει την ουσία της ασαφούς λογικής : Η ασαφής λογική ξεκινάει όταν $A \cap A^c = \emptyset$, όταν το Α και το αντίθετο του δεν αποκλείονται αμοιβαία, αλλά μπορούν να ισχύουν ταυτόχρονα σε κάποιο βαθμό, όταν δηλαδή η μετάβαση από το Α στο αντίθετο του δεν είναι απότομη αλλά *βαθμιαία* και το μεταξύ τους σύνορο δεν είναι πλέον αυστηρά καθορισμένο αλλά ασαφές (σχήμα 1.7).

Οι δύο νόμοι που αναφέρθηκαν παραπάνω δείχνουν μία ακόμη διαφορά μεταξύ της ασαφούς λογικής και της θεωρίας πιθανοτήτων, αφού η τελευταία συμμορφώνεται με τους νόμους αυτούς :

$$P(A \cap A^c) = P(\emptyset) = 0 \qquad (1.21)$$

$$P(A \cup A^c) = P(X) = 1 \qquad (1.22)$$

δηλαδή σύμφωνα με τη θεωρία πιθανοτήτων το $A \cap A^c$ είναι ένα απίθανο συμβάν, ενώ το $A \cup A^c$ είναι ένα σίγουρο συμβάν.

Συμπερασματικά λοιπόν αναφέρουμε ότι η ασαφής λογική είναι μία επέκταση της κλασσικής, όπως διαπιστώθηκε και στον πρόλογο, ενώ στην περίπτωση που οι τιμές αλήθειας περιορίζονται στο {0,1}, οι δύο λογικές ταυτίζονται.

Κεφάλαιο 2

ΑΣΑΦΗ ΣΥΣΤΗΜΑΤΑ ΕΛΕΓΧΟΥ

2.1 Περιγραφή λειτουργίας

Τα συστήματα αυτόματου ελέγχου που βασίζονται στην ασαφή λογική *(fuzzy controllers)* χρησιμοποιούν *κανόνες (rules)* της μορφής :

IF x IS	A	THEN	z IS C	(2.1a)
IF x IS	A AND y IS B	THEN	z IS C	(2.1b)
IF x IS	A OR y IS B	THEN	z IS C	(2.1c)

όπου τα x,y είναι οι είσοδοι του συστήματος και z η έξοδος. Ο κανόνας (2.1a) έχει μία *υπόθεση (antecedent)* και ένα *επακόλουθο (consequent)* ενώ οι (2.1b) και (2.1c) έχουν από δύο υποθέσεις. Στην πράξη οι κανόνες μπορούν να έχουν περισσότερες υποθέσεις (ή και επακόλουθα) συνδυασμένες μεταξύ τους με τους τελεστές AND, OR και NOT (άρνηση ή συμπλήρωμα). Ετσι, σε ένα σύστημα ελέγχου π.χ. για την οδήγηση ενός αυτοκινήτου, ένας κανόνας θα μπορούσε να έχει τη μορφή:

```
IF ((velocity IS big) AND (turn IS close)) OR (obstacle IS near)
THEN  (force-to-break-pedal IS very big)
```

Στην περίπτωση αυτή, τα "velocity", "turn" και "obstacle" είναι οι είσοδοι του συστήματος και το "force-to-break-pedal" είναι η έξοδος. Οι παραπάνω εκφράσεις συνήθως ονομάζονται *ασαφείς μεταβλητές (fuzzy variables)*, ενώ οι εκφράσεις "big", "close" και "near" ονομάζονται *ετικέττες (labels)*. Οι τιμές που μπορεί να πάρει μία ασαφής μεταβλητή είναι "ετικέττες" από ασαφή σύνολα, π.χ. στο παράδειγμά μας η ταχύτητα αυτοκινήτου μπορεί να είναι "μικρή", "μέση", "μεγάλη" κλπ. Οι τιμές αυτές περιγράφονται με τη βοήθεια ασαφών συνόλων. Γενικότερα οι τιμές μιας ασαφούς μεταβλητής μπορούν να είναι προτάσεις σε κάποια ειδικά προδιαγραμμένη γλώσσα (π.χ. με συνδυασμό ασαφών μεταβλητών και *γλωσσικών περιγραμμάτων (hedges)* ή *τροποποιητών (modifiers)*). Στην περίπτωση αυτή οι ασαφείς μεταβλητές ονομάζονται *γλωσσικές μεταβλητές*. Τα περιγράμματα ή τροποποιητές είναι εκφράσεις (όπως το "very" στο παράδειγμά μας) που χρησιμοποιούνται για να μετατρέπουν την τιμή (το ασαφές σύνολο) της

"ετικέττας" στην οποία εφαρμόζονται. Ετσι, πχ. η δύναμη που βάζει κανείς στο φρένο, μπορεί να είναι "μεγάλη", "πολύ μεγάλη", "σχετικά μεγάλη", "όχι πολύ μεγάλη" κλπ. Η υλοποίηση των τροποποιητών γίνεται με συνδυασμό των πράξεων της συμπύκνωσης, διαστολής κλπ. που είδαμε στο προηγούμενο κεφάλαιο. Το πρόγραμμα "Fuzzy Expert" έχει τη δυνατότητα να προσομοιώνει οσοδήποτε πολύπλοκα συστήματα κανόνων της μορφής αυτής. Εχει ενσωματωμένο interpreter και δέχεται αρχεία εισόδου σε μία γλώσσα ειδικά προδιαγραμμένη για την περιγραφή κανόνων ασαφούς λογικής.

Για να καταλάβουμε όμως καλύτερα πως λειτουργεί η ασαφής λογική στα συστήματα ελέγχου που είναι *βασισμένα σε κανόνες (rule-based controllers)* ας δούμε ένα συγκεκριμένο παράδειγμα που προέρχεται από μία βιομηχανική εφαρμογή. Πρόκειται για ένα αυτόματο ρυθμιστή μιας διαδικασίας παραγωγής τσιμέντου που αναπτύχθηκε το 1980 στη Δανία [7]. Ο τρόπος λειτουργίας του συστήματος απεικονίζεται στο σχήμα 2.1. Στο σχήμα αυτό φαίνονται 4 από τους 30 ή 40 κανόνες του συστήματος (κάθε κανόνας αντιστοιχεί σε μία σειρά καμπυλών στο σχ. 2.1). Οι κανόνες είναι οι εξής :

Κανόνας 1: **IF** CT **IS** ZE **AND** LC **IS** LOW **THEN** FA **IS** MN
Κανόνας 2: **IF** CT **IS** ZE **AND** LC **IS** OK **THEN** FA **IS** NONE
Κανόνας 3: **IF** CT **IS** NEG **AND** LC **IS** LOW **THEN** FA **IS** SP
Κανόνας 4: **IF** CT **IS** NEG **AND** LC **IS** OK **THEN** FA **IS** MP

όπου CT : Change in Kiln-drive Torque (μεταβολή στη ροπή του κινητήρα)
LC : Free-Lime Content (περιεχόμενο σε ασβέστη)
FA : Fuel Adjustment (ρύθμιση τροφοδοσίας καυσίμων)
ZE : Zero (μηδέν)
NEG: Negative (αρνητικό)
SP : Small Positive (ελαφρώς θετικό)
MP : Medium Positive (μετρίως θετικό)
MN : Medium Negative (μετρίως αρνητικό)

Οπως είναι προφανές, οι εκφράσεις που περιέχονται σε κάθε κανόνα κωδικοποιούνται με κατάλληλες συντομογραφίες ώστε το σύνολο των κανόνων (*rulebase*) να είναι σύντομο και να μπορούμε να το επεξεργαζόμαστε ευκολότερα. Το σύστημα έχει 2 εισόδους, το CT - που μπορεί να πάρει τις τιμές ("ετικέττες") Zero και Negative - και το LC - που μπορεί να πάρει τις τιμές LOW και OK - και μία έξοδο, το FA - που παίρνει τις τιμές Medium Negative, None, Small Positive και Medium Positive.

Στο σημείο αυτό πρέπει να διευκρινίσουμε ότι εφόσον πρόκειται για ένα πραγματικό σύστημα οι έξοδοι παίρνουν *σαφείς (crisp)* τιμές, είναι δηλαδή πραγματικοί αριθμοί. Οπως θα δούμε και παρακάτω, αυτό συμβαίνει στά περισσότερα ασαφή συστήματα, ενώ ασαφείς εισόδους και εξόδους έχουμε μόνο στα ασαφή έμπειρα συστήματα. Επομένως *η ασάφεια υπεισέρχεται μόνο στη διατύπωση των κανόνων.*

Αναφερόμενοι στο σχήμα 2.1, οι είσοδοι στο σύστημα σε μία συγκεκριμένη χρονική στιγμή είναι CT = −1.2 % ανά ώρα, και LC = 0.54 % . Ας δούμε τώρα πώς το σύστημα υπολογίζει την έξοδο, FA. Αρχικά για κάθε είσοδο και για κάθε κανόνα υπολογίζεται ο βαθμός στον οποίο ισχύει η συγκεκριμένη υπόθεση του κανόνα. Για τον 3ο κανόνα π.χ., υπολογίζονται οι τιμές

$$\mu_{NEG}(CT) = \mu_{NEG}(-1.2) = 0.29 \ , \quad \text{και}$$
$$\mu_{LOW}(LC) = \mu_{LOW}(0.54) = 0.63$$

όπου $\mu_{NEG}(x)$ είναι η συνάρτηση συμμετοχής του ασαφούς συνόλου που περιγράφει την "ετικέττα" NEG και $\mu_{LOW}(x)$ η αντίστοιχη συνάρτηση για την "ετικέττα" LOW. Στη συνέχεια, εφόσον οι 2 υποθέσεις του κανόνα συνδέονται μεταξύ τους με AND, παίρνουμε την ελάχιστη από τις δύο παραπάνω τιμές :

$$\mu_3 = \min \ (\ \mu_{NEG}(CT), \ \mu_{LOW}(LC) \ ) = 0.29$$

όπου μ_3 είναι ο "*βαθμός εκπλήρωσης*" (degree of fulfilment, DOF) του κανόνα 3. Το επιμέρους συμπέρασμα του κανόνα 3 λοιπόν (δηλ. FA IS SP) εφαρμόζεται κατά 0.29. Αυτό γίνεται πολλαπλασιάζοντας τη συνάρτηση συμμετοχής $\mu_{SP}(x)$ που εκφράζει την "ετικέττα" SP επί 0.29. Ετσι το επιμέρους συμπέρασμα SP δίνεται από τη σχέση SP'=0.29 · SP ή

$$\mu_{SP'}(x) = 0.29 \cdot \mu_{SP}(x) \qquad \forall x$$

Η πράξη αυτή ονομάζεται *συνεπαγωγή (implication).*

Με τον ίδιο ακριβώς τρόπο υπολογίζουμε και τους βαθμούς εκπλήρωσης ή "βάρη" των άλλων κανόνων, $\mu_1 = 0.98$, $\mu_2 = 0.63$, $\mu_3 = \mu_4 = 0.29$, και το τελικό συμπέρασμα προκύπτει από το συνδυασμό των επιμέρους με την πράξη *EITE (ELSE)* ή *aggregation* (που συνήθως είναι μέσος όρος ή maximum). Ετσι προκύπτει το ασαφές σύνολο που φαίνεται κάτω δεξιά στο σχήμα 3.1 και που παριστάνει το συνολικό συμπέρασμα των 4 κανόνων. Τέλος, για να υπολογιστεί μια σαφής (crisp) τιμή για την έξοδο του συστήματος, χρειάζεται να γίνει *απο-ασαφοποίηση* (*defuzzification*) του τελικού συμπεράσματος. Αυτή υπολογίζεται ως το κέντρο βάρους της τελικής καμπύλης και η έξοδος προκύπτει FA = −0.048 m^3/h.

Παρατηρούμε ότι σε αντίθεση με τα κλασσικά συστήματα κανόνων, στην περίπτωση αυτή στο τελικό συμπέρασμα συνεισφέρουν *όλοι* οι κανόνες *ταυτόχρονα*, κατά ένα ποσοστό που είναι ανάλογο με το βαθμό εκπλήρωσης της συνθήκης τους. Ετσι, ακόμα και στην περίπτωση κανόνων με αντιφατικά συμπεράσματα, το σύστημα είναι ικανό να πάρει μια λογική απόφαση "ζυγίζοντας" κατάλληλα τα συμπεράσματα αυτά. Επιπλέον, εφόσον οι υποθέσεις των κανόνων εκφράζονται με ασαφή σύνολα που δίνουν μη μηδενικές τιμές DOF για σχετικά μεγάλες περιοχές διακύμανσης των εισόδων, *χρειάζονται πολύ λιγότεροι κανόνες σε σχέση με τα συμβατικά συστήματα*, όπου ένας κανόνας ισχύει μόνο για μια συγκεκριμένη τιμή της εισόδου.

Δύο ακόμη σημεία είναι αξιοσημείωτα στο συγκεκριμένο παράδειγμα. Πρώτον, το σύνολο των 30 ή 40 κανόνων μπορεί κάλλιστα να περιγραφεί από μία *μη - γραμμική, πολυδιάστατη συνάρτηση* που καθορίζει ακριβώς την έξοδο του συστήματος για κάθε συνδυασμό εισόδων. Ομως σ'αυτή την περίπτωση οι απαιτήσεις μνήμης και το κόστος προγραμματισμού είναι πολύ μεγαλύτερα από την περίπτωση των 40 κανόνων και των 10 - 20 συναρτήσεων συμμετοχής που καθορίζουν τα χρησιμοποιούμενα ασαφή σύνολα.

Δεύτερον, το συγκεκριμένο παράδειγμα αναφέρεται σε μία χημική διαδικασία που είναι εξαιρετικά δύσκολο να περιγραφεί από ένα μαθηματικό μοντέλο. Παρόλα αυτά ένας πεπειραμένος χειριστής καταφέρνει να το ελέγξει χρησιμοποιώντας μόνο μερικούς εμπειρικούς κανόνες και "κοινή λογική". Οπως τονίστηκε και στον πρόλογο, η ασαφής λογική είναι ένα μαθηματικό εργαλείο που επιτρέπει τη μοντελοποίηση αυτών των κανόνων και τη μεταφορά της "εμπειρίας" από τον άνθρωπο χειριστή στο αυτοματοποιημένο σύστημα ελέγχου. Το ασαφές σύστημα που παρουσιάσαμε ήταν το πρώτο αυτοματοποιημένο σύστημα που χρησιμοποιήθηκε σε μία τέτοια εφαρμογή, και μάλιστα είχε σαν αποτέλεσμα *την εξοικονόμηση ενέργειας* και *τη βελτίωση της ποιότητας* του παραγόμενου προϊόντος [7].

2.2 Η γενική μέθοδος υπολογισμού

Μετά από τη σύντομη παρουσίαση που έγινε στην προηγούμενη παράγραφο, ας δούμε τώρα πιο αναλυτικά τις μεθόδους που ακολουθούνται σε ένα ασαφές σύστημα ελέγχου και τις παραμέτρους που επηρεάζουν τους υπολογισμούς. Στην πιό γενική περίπτωση, ένα τέτοιο σύστημα αποτελείται από κανόνες της μορφής:

$$R_i: \text{ IF } f(x_1 \text{ IS } A_{i1}, \ldots, x_k \text{ IS } A_{ik}) \text{ THEN } y_1 \text{ IS } B_{i1} \text{ AND } \ldots \text{ AND } y_m \text{ IS } B_{im} \qquad (2.2)$$

όπου $1 \le i \le N$,
$x_1 \in X_1, \ldots, x_k \in X_k$
$y_1 \in Y_1, \ldots, y_m \in Y_m$

Δηλαδή το σύστημα αυτό έχει N κανόνες, k εισόδους και m εξόδους. Οι είσοδοι $x_1,..,x_k$ είναι όλες *σαφείς (crisp)* αριθμοί και αντιστοιχούν στις *"ετικέττες" (labels)* ή ασαφή σύνολα $A_{i1},..,A_{ik}$ που είναι προφανώς διαφορετικές για κάθε κανόνα και που ορίζονται στα υπερσύνολα αναφοράς $X_1,..,X_k$ αντίστοιχα. Ομοια και οι έξοδοι $y_1,..,y_m$ είναι σαφείς και αντιστοιχούν στα ασαφή σύνολα $B_{i1},..,B_{im}$ που ορίζονται στα υπερσύνολα αναφοράς $Y_1,..,Y_m$. Τα $X_1,..,X_k$ και $Y_1,..,Y_m$ μπορούν να είναι είτε διακριτά είτε συνεχή.

Η συνάρτηση f στη 2.2 δηλώνει οποιοδήποτε συνδυασμό των ορισμάτων με τις λογικές πράξεις AND και OR (η πράξη NOT συνήθως υπεισέρχεται σαν *τροποποιητής (modifier)* στον ορισμό των ασαφών συνόλων). Τις περισσότερες φορές όμως οι

υποθέσεις (antecedents) x_1 IS A_{i1},..,x_k IS A_{ik} συνδυάζονται μεταξύ τους μονο με AND, ενώ οι κανόνες που περιέχουν OR "σπάνε" σε περισσότερους κανόνες (όπως θα αναλύσουμε σε επόμενο κεφάλαιο, αυτό το "σπάσιμο" δεν ισχύει πάντα). Για παράδειγμα, ένας κανόνας της μορφής

IF x_1 IS A_1 AND (x_2 IS A_2 OR x_3 IS A_3) THEN y IS B

μπορεί να αναλυθεί σε δύο κανόνες

IF x_1 IS A_1 AND x_2 IS A_2 THEN y IS B
IF x_1 IS A_1 AND x_3 IS A_3 THEN y IS B

Επίσης από δώ και στο εξής για απλότητα θα θεωρούμε κανόνες με ένα μόνο *επακόλουθο (consequent)*, αφού και για περισσότερα επακόλουθα οι υπολογισμοί είναι ακριβώς ίδιοι. Με βάση τις παρατηρήσεις αυτές η (2.2) γίνεται

R_i: IF x_1 IS A_{i1} AND....AND x_k IS A_{ik}
THEN y IS B_i (2.3)

όπου $y \in Y$.

Οι υπολογισμοί που απαιτούνται για να εξαχθεί το συμπέρασμα, έστω y_0, από τις εισόδους $x_1,\ldots,x_k$ είναι οι εξής :

1) Για κάθε κανόνα, υπολογίζεται ο βαθμός στον οποίο ισχύουν οι υποθέσεις x_1 IS A_{i1},..,x_k IS A_{ik}. Δηλαδή υπολογίζονται οι αριθμοί $\mu_{Ai1}(x_1),..,\mu_{Aik}(x_k)$ για κάθε $i\in\{1,..,N\}$.

2) Οι βαθμοί αυτοί συνδυάζονται μεταξύ τους με την πράξη AND, δηλ. υπολογίζεται ο αριθμός
$$\mu_i = \mu_{Ai1}(x_1) \text{ AND....AND } \mu_{Aik}(x_k) \qquad (2.4)$$
για κάθε $i\in\{1,\ldots,N\}$. Οπως θα δούμε παρακάτω, εκτός από τους τελεστές mimimum και maximum έχουν προταθεί ακόμα πολλοί τρόποι υλοποίησης των πράξεων AND και OR.

3) Υπολογίζεται το επιμέρους συμπέρασμα του κάθε κανόνα, δηλ. το ασαφές σύνολο B'_i τέτοιο ώστε
$$\mu_{B'i}(y) = \mu_i \rightarrow \mu_{Bi}(y) \qquad \forall\, y\in Y \qquad (2.5)$$
για κάθε $i\in\{1,\ldots,N\}$. Η πράξη " $\rightarrow$ " ονομάζεται *συνεπαγωγή (implication)* και υπάρχουν επίσης πολλοί τρόποι ορισμού της.

4) Τα επιμέρους συμπεράσματα B'_i συνδυάζονται με την πράξη της *άθροισης (aggregation)* ή *ΕΙΤΕ (ELSE)* για να δώσουν ένα ολικό συμπέρασμα (ασαφές σύνολο) B':
$$\mu_{B'}(y) = \mu_{B'1}(y) \text{ ELSE....ELSE } \mu_{B'N}(y) \quad \forall\, y\in Y \qquad (2.6)$$
όπου συνήθως η πράξη ELSE υλοποιείται σαν maximum.

5) Τέλος , η σαφής έξοδος y_0 προκύπτει από την *απο-ασαφοποίηση (defuzzification)* του ασαφούς συνόλου B':

$$y_0 = defuzzifier\ (B') \tag{2.7}$$

όπου ο τελεστής defuzzifier θα οριστεί παρακάτω.

Μια ειδική περίπτωση των κανόνων της μορφής (2.3) είναι το ασαφές σύνολο B_i του επακόλουθου να είναι *μοναδιαίο (singleton)*, δηλ.

$$\mu_{Bi}(y)= \begin{cases} 1, & y=y_i \\ 0, & y\neq y_i \end{cases} \tag{2.8}$$

Ετσι ουσιαστικά ο κανόνας i προτείνει μία συγκεκριμένη, σαφή έξοδο y_i και η (2.3) γίνεται

$$R_i:\ \text{IF } x_1 \text{ IS } A_{i1} \text{ AND....AND } x_k \text{ IS } A_{ik} \quad \text{THEN } y \text{ IS } Y_i \tag{2.9}$$

Αυτή η μέθοδος είναι πολύ σημαντική γιατί απλοποιεί σε μεγάλο βαθμό τις πράξεις implication, aggregation & deffuzification, έχοντας μάλιστα σε πολλές περιπτώσεις καλύτερα πρακτικά αποτελέσματα. Αυτή η μέθοδος χρησιμοποιήθηκε και στην εφαρμογή που παρουσιάζεται στο μέρος Β.

Πιο γενικά, η έξοδος y_i μπορεί να είναι γραμμικός συνδυασμός των εισόδων [6], [13], [31], [34] δίνοντας κανόνες της μορφής

$$R_i:\ \text{IF } x_1 \text{ IS } A_{i1} \text{ AND....AND } x_k \text{ IS } A_{ik} \quad \text{THEN } y = a_{i0} + a_{i1}x_1 + \ldots + a_{ik}x_k \tag{2.10}$$

ενώ ακόμη γενικότερα, το y_i μπορεί να είναι οποιαδήποτε συνάρτηση των εισόδων [31]:

$$R_i:\ \text{IF } x_1 \text{ IS } A_{i1} \text{ AND....AND } x_k \text{ IS } A_{ik} \quad \text{THEN } y = h_i\ (x_1,\ldots,x_k) \tag{2.11}$$

όπου h_i οποιαδήποτε συνάρτηση των $x_1,\ldots,x_k$ (διαφορετική για κάθε κανόνα).

2.3 Παράμετροι που διαφοροποιούν τη μέθοδο υπολογισμού.

Η γενική μέθοδος που εξετάσαμε στην προηγούμενη παράγραφο μπορεί να πάρει πολλές διαφορετικές μορφές ανάλογα με την επιλογή του τρόπου υλοποίησης των πράξεων AND, OR, NOT, implication, aggregation και defuzzification, των

συναρτήσεων συμμετοχής, των τροποποιητών (modifiers) καθώς και πολλών άλλων παραμέτρων. Οι επιλογές αυτές, καθώς και οι δυνατοί τρόποι συνδυασμών τους, μελετώνται στο κεφάλαιο αυτό. Αξίζει να σημειώσουμε στο σημείο αυτό ότι για την πρακτική επιβεβαίωση πολλών από τα θεωρητικά αποτελέσματα που παρουσιάζουμε στο κεφάλαιο αυτό, αλλά και για την απεικόνιση των αποτελεσμάτων αυτών χρησιμοποιήσαμε το μαθηματικό πακέτο MATLAB. Το MATLAB είναι ένα πρόγραμμα ιδανικό για την προσομείωση μεθόδων της ασαφούς λογικής γιατί έχει μεγάλες δυνατότητες χειρισμού διανυσμάτων και πινάκων οποιουδήποτε μεγέθους, προγραμματίζεται σε ιδιαίτερα απλή γλώσσα και διαθέτει πολλές ευκολίες για την επικοινωνία με το χρήστη και την απεικόνιση των αποτελεσμάτων.

Ουσιαστικά αυτό που δημιουργήθηκε ήταν 55 αρχεία εισόδου για το MATLAB που υλοποιούν ισάριθμες *συναρτήσεις* και που παρουσιάζονται αναλυτικά στο Παράρτημα Α. Οι συναρτήσεις αυτές προσομοιώνουν όλες τις μεθόδους υπολογισμού που παρουσιάζονται στη διπλωματική αυτή εργασία, και είναι πολύ χρήσιμες για τη συγκριτική μελέτη των μεθόδων αυτών. Η ανωτερώτητα του MATLAB φαίνεται από το γεγονός ότι μέσα σε πολύ σύντομο χρονικό διάστημα αναπτύχθηκε ένα *σύνολο συναρτήσεων* που *ξεπερνά* σε υπολογιστικές δυνατότητες το πρόγραμμα "Fuzzy Expert" (χωρίς βέβαια να διαθέτει ενσωματωμένο interpreter ούτε user interface σαν αυτά του "Fuzzy Expert", αφού για τους σκοπούς αυτούς χρησιμοποιήθηκαν έτοιμες ρουτίνες του MATLAB).

2.3.1 Επιλογή των λειτουργικών χαρακτηριστικών

Με τον όρο λειτουργικά χαρακτηριστικά εννοούμε τη γενική αρχιτεκτονική του ασαφούς συστήματος, τον τρόπο υλοποίησής του (δηλ. με software ή hardware) και κυρίως ποια δεδομένα εισέρχονται στο σύστημα, τι είδους επεξεργασία υφίστανται τα δεδομένα αυτά, και ποιά δεδομένα εξέρχονται από το σύστημα στον εξωτερικό κόσμο [15]. Στην περίπτωση που το μαθηματικό μοντέλο της συγκεκριμένης διαδικασίας δεν είναι γνωστό, ο σχεδιαστής πρέπει να έχει βαθειά κατανόηση των τριών τελευταίων σημείων. Δεν πρέπει να ξεχνάμε ότι το ασαφές σύστημα ελέγχου είναι ένα *υποσύστημα* του οποίου σκοπός είναι ο προσδιορισμός μιας (*μη-γραμμικής*, στη γενική περίπτωση) συνάρτησης ελέγχου. Ο ρόλος του σχεδιαστή επομένως είναι να επιλέξει τον καλύτερο τρόπο με τον οποίο το υποσύστημα αυτό συνδέεται με τη *συνολική αρχιτεκτονική* του συστήματος. Το στάδιο αυτό της σχεδίασης είναι το σημαντικότερο και απαιτεί μεγάλη πείρα.

Ουσιαστικό ζητούμενο στο στάδιο αυτό είναι ο καθορισμός των μεταβλητών εισόδου και εξόδου του συστήματος και των αντιστοίχων υπερσυνόλων αναφοράς, δηλ. των διαστημάτων μέσα στα οποία κυμαίνονται οι μεταβλητές αυτές. Αξίζει να σημειώσουμε ότι σε μία πρόσφατη δημοσίευση [36] αναφέρεται μία αποτελεσματική μέθοδος για την αυτόματη επιλογή των καταλληλότερων μεταβλητών εισόδου με βάση στατιστικά στοιχεία εισόδου - εξόδου του συστήματος.

2.3.2 Επιλογή των κανόνων

Η επιλογή των κανόνων που περιγράφουν καλύτερα την επιθυμητή συμπεριφορά του ασαφούς συστήματος είναι ένα σημείο - κλειδί στη σχεδίασή του. Στο στάδιο αυτό αρχικά το υπερσύνολο αναφοράς κάθε μεταβλητής (εισόδου και εξόδου) *διαμερίζεται* (*clustering*) σε ένα πλήθος *ασαφών περιοχών*. Σε κάθε τέτοια περιοχή δίνουμε ένα μοναδικό όνομα ή ετικέττα (label). Επίσης, ανάλογα με την έννοια που αντιπροσωπεύει η ασαφής περιοχή, της αντιστοιχίζουμε ένα ασαφές σύνολο, του οποίου η ακριβής μορφή θα εξεταστεί στην επόμενη παράγραφο. Το πλήθος των ασαφών περιοχών είναι συνήθως ένας μονός αριθμός μεταξύ πέντε και εννέα (αυτό θα γίνει περισσότερο κατανοητό στην πρακτική εφαρμογή του Μέρους Β) ενώ η πυκνότητα των ασαφών περιοχών θα πρέπει να είναι μέγιστη γύρω από το βέλτιστο σημείο ελέγχου (σημείο ισορροπίας) ώστε κοντά στο σημείο αυτό να έχουμε μεγαλύτερη ευαισθησία και αποτελεσματικότερο έλεγχο.

Το πλήθος των κανόνων συνήθως σχετίζεται απλά με το πλήθος των μεταβλητών εισόδου και το πλήθος των ασαφών περιοχών. Αν π.χ. ένα σύστημα έχει δύο εισόδους κάθε μία απ'τις οποίες χωρίζεται σε 5 ασαφείς περιοχές (π.χ. Μεγάλο Αρνητικό, Μικρό αρνητικό, περίπου Μηδέν, Μικρό θετικό, Μεγάλο θετικό) τότε οι δυνατοί συνδυασμοί εισόδων είναι 25, οπότε απαιτούνται 25 συνολικά κανόνες. Ενα τέτοιο σχήμα ακολουθήθηκε και στην πρακτική εφαρμογή.

Η διατύπωση των εμπειρικών κανόνων ελέγχου αλλά και η επιλογή των σημαντικότερων απ'αυτούς (π.χ. στο προηγούμενο παράδειγμα οι 7 μόνο από τους 25 κανόνες μπορεί να είναι αρκετοί για κάποια εφαρμογή) είναι μία δύσκολη εργασία και χρειάζεται πείρα. Αν όμως υπάρχουν διαθέσιμα δεδομένα εισόδου - εξόδου του συστήματος με βάση κάποιο άλλο σύστημα ελέγχου ή έναν έμπειρο χειριστή, τότε μπορεί να χρησιμοποιηθεί μία από τις πολλές μεθόδους που έχουν προταθεί για την αυτόματη εξαγωγή των κανόνων από τα δεδομένα αυτά. Οι μέθοδοι αυτές βασίζονται κυρίως στα *νευρωνικά δίκτυα* [8, 14, 38, 39], τους *γενετικούς αλγόριθμους* [18, 19, 38, 68], και άλλες μεθόδους εκμάθησης και βελτιστοποίησης [6, 13, 14, 34]. Η μέθοδος μάλιστα που παρουσιάζεται στο [36], είναι ικανή να βρίσκει με μεγάλη ακρίβεια τις πιό κατάλληλες μεταβλητές εισόδου, την καταλληλότερη διαμέριση του υπερσυνόλου αναφοράς σε ασαφείς περιοχές, το πλήθος και το περιεχόμενο των κανόνων, ακόμα και τις συναρτήσεις συμμετοχής. Παρόμοιες μελέτες υπάρχουν στα [51, 52]. Επίσης έχουν προταθεί *προσαρμοζόμενα (adaptive)* ασαφή συστήματα [8, κεφ. 8], [14] και *αυτορυθμιζόμενα* συστήματα ελέγχου (*self-organizing controllers, SOC*) [13], [46], [47] που έχουν τη δυνατότητα να αξιολογούν ανά πάσα στιγμή την ποιότητα των κανόνων τους και να προσαρμόζονται ανάλογα με τις εκάστοτε εξωτερικές συνθήκες.

Ενα σημείο που χρειάζεται προσοχή στην *εκμάθηση* των κανόνων από κάποιο έμπειρο χειριστή είναι ότι οι άνθρωποι μπορεί πολλές φορές να κάνουν λάθη κατά το χειρισμό μιας διαδικασίας, τα οποία όμως μπορούν εύκολα να επανορθώνουν. Τότε όμως τα εσφαλμένα δεδομένα εισόδου - εξόδου του συστήματος είναι δυνατό να υπονομεύσουν τη διαδικασία εκμάθησης [11].

2.3.3 Επιλογή των συναρτήσεων συμμετοχής

Οι συναρτήσεις συμμετοχής μπορούν να είναι είτε *διακριτές* (Σχήμα 2.2) είτε *συνεχείς* (Σχήματα 2.3, 2.4). Η πρώτη περίπτωση εφαρμόζεται όταν το υπερσύνολο αναφοράς X είναι διακριτό, όταν δηλ. γίνεται ψηφιακή υλοποίηση του συστήματος, και στην περίπτωση αυτή η συνάρτηση συμμετοχής καθορίζεται από τις συντεταγμένες των σημείων που το αποτελούν, δηλ. από το βαθμό συμμετοχής των στοιχείων του. Για παράδειγμα, χρησιμοποιώντας το συμβολισμό της σχέσης (1.3), το ασαφές σύνολο του σχήματος 2.2 ορίζεται ως

{0/1, 0.1/2, 0.4/3, 0.7/4, 0.9/5, 1/6, 0.8/7, 0.6/8, 0.5/9, 0.3/10}

(στην περίπτωση αυτή έχουμε X={1,2,...,10}).

Η δεύτερη περίπτωση εφαρμόζεται όταν το υπερσύνολο αναφοράς είναι συνεχές, δηλ. όταν η υλοποίησή είναι αναλογική και τότε η συνάρτηση συμμετοχής καθορίζεται και πάλι από τις συντεταγμένες κάποιων σημείων με τη χρήση κάποιας μεθόδου *παρεμβολής (interpolation)* (Σχήμα 2.3) όπου X = [1,10] ή *παραμετρικά* με τη χρήση κάποιας συνεχούς συνάρτησης, όπως αυτή του Σχήματος 2.4 που δίνεται από τη σχέση

$\mu(x)$ = gauss (x;0.6, 0.2)

όπου η συνάρτηση gauss θα οριστεί αμέσως παρακάτω.

Στη βιβλιογραφία έχουν προταθεί μία πληθώρα τέτοιων συναρτήσεων. Μια συγκριτική μελέτη υπάρχει στο [43]. Οι κυριότερες απ'αυτές είναι:

1) **Τριγωνική (triangular):**

$$\mathbf{triang}(x;a,s) = \begin{cases} 1+(x-a)/s, & a-s<x\le a \\ (a+s-x)/s, & a<x<a+s \\ 0, & \text{αλλού} \end{cases} \qquad (2.12)$$

2) **Τραπεζοειδής (trapezoidal):**

$$\mathbf{trapez}(x;a,s,p) = \begin{cases} 1, & a-p/2<x\le a+p/2 \\ \dfrac{x-a+s}{a-p/2}, & a-s<x\le a-p/2 \\ \dfrac{a+s-x}{a-p/2}, & a+p/2<x\le a+s \\ 0, & \text{αλλού} \end{cases} \qquad (2.13)$$

3) **Gaussian:**

$$\text{gauss}(x;a,s) = e^{-\left(\frac{x-a}{s/2}\right)^2 \ln 2} \tag{2.14}$$

4) **Lorentzian:**

$$\text{lorentz}(x;a,s) = \frac{1}{1+\left(\frac{x-a}{s/2}\right)^2} \tag{2.15}$$

ή πιο γενικά

$$\text{lorentz}(x;a,s,p) = \frac{1}{1+\left(\frac{|x-a|}{s/2}\right)^p} \tag{2.16}$$

5) **Σίγμα (Sigma):**

$$S(x;a,s) = \begin{cases} 0, & x \le a-s \\ f\left(1+\frac{x-a}{s}\right), & a-s < x \le a \\ 1, & a < x \end{cases} \tag{2.17}$$

6) **Ζήτα (Zeta):**

$$Z(x;a,s) = 1-S(x;a+s,s) \tag{2.18}$$

7) **Πι (Pi):**

$$\Pi(x;a,s) = \begin{cases} S(x;a,s), & x \le a \\ Z(x;a,s), & x > a \end{cases} \tag{2.19}$$

όπου

$$f(x) = \begin{cases} 2x^2, & 0 \le x \le 1/2 \\ 1-2(1-x)^2, & 1/2 < x \le 1 \end{cases} \tag{2.20}$$

Οι συναρτήσεις *triang* και *trapez* φαίνονται στο Σχήμα 2.5, οι *gauss*, *lorentz* και *Π* στο Σχήμα 2.6, η γενικευμένη *lorentz* στο Σχήμα 2.7, και οι *S* και *Z* στα Σχήματα 2.8 και 2.9 αντίστοιχα. Οσο αφορά τις 5 πρώτες συναρτήσεις, είναι αύξουσες συναρτήσεις του x για x≤a και φθίνουσες για x>a. Η παράμετρος a καθορίζει τη *μέση τιμή* και η s το *εύρος* της κάθε καμπύλης. Ειδικότερα για τις triang, gauss, lorentz και Π, το s καθορίζεται έτσι ώστε μ(a ± S) = 1/2, ενώ για την trapez η πάνω βάση του τραπεζίου ισούται με p και η κάτω βάση με 2s. Για τη συνάρτηση lorentz που ορίζεται από τη γενική σχέση (2.16), η

παράμετρος p καθορίζει την *αντίθεση (contrast)*. Τέλος οι συναρτήσεις S και Z έχουν διαφορετική μορφή από τις προηγούμενες: η πρώτη είναι παντού αύξουσα συνάρτηση του x, ενώ η δεύτερη φθίνουσα.

Συνήθως στην πράξη χρησιμοποιούνται τριγωνικές ή τραπεζοειδείς συναρτήσεις λόγω της ευκολότερης υλοποίησής τους σε hardware. Πάντως οι υπόλοιπες "καμπύλες" συναρτήσεις είναι πιο "φυσικές" και παρουσιάζουν καλύτερη συμπεριφορά στα συστήματα ελέγχου λόγω της ομαλότερής τους μορφής. Το γεγονός αυτό επιβεβαιώθηκε και στο πειραματικό μέρος, όπου υλοποιήθηκαν με αναλογικά κυκλώματα συναρτήσεις τριγωνικές, τραπεζοειδείς και τύπου Π. Οι συναρτήσεις τύπου Π έδωσαν τα καλύτερα αποτελέσματα. Μεταξύ των "καμπύλων" συναρτήσεων gauss, lorentz και Π μπορούμε να αναφέρουμε ότι η υλοποίηση της Π είναι ευκολότερη (κυρίως με αναλογικά κυκλώματα), αλλά η καλύτερη μορφή εξαρτάται από τη συγκεκριμένη εφαρμογή. Δυστυχώς δεν υπάρχει μία γενικά αποδεκτή μέθοδος ακριβούς καθορισμού της μορφής των συναρτήσεων συμμετοχής και γι'αυτό στις περιπτώσεις που αναζητείται βελτιστοποίηση ως προς τον παράγοντα αυτό, είναι σκόπιμο να χρησιμοποιηθεί κάποια μέθοδος *εκμάθησης* όπως αυτή που παρουσιάζεται στο [36].

Πάντως αξίζει να σημειωθεί ότι η ακριβής μορφή των συναρτήσεων συμμετοχής δεν είναι ιδιαίτερα σημαντική ως προς τη συμπεριφορά ενός ασαφούς συστήματος ελέγχου. Πολύ σημαντικότερες είναι οι διαδικασίες που περιγράφτηκαν στις παραγράφους 2.3.1 και 2.3.2. Αυτό έγινε φανερό με το πείραμα του Μέρους Β, στο οποίο για σχετικά μεγάλες μεταβολές των συναρτήσεων συμμετοχής, το σύστημα εξακολουθούσε να λειτουργεί. Τέλος, μια σύγκριση μεταξύ των διακριτών και των συνεχών συναρτήσεων συμμετοχής γίνεται επίσης στο Μέρος Β, όπου γίνονται σαφή τα πλεονεκτήματα της αναλογικής υλοποίησης ως προς την ψηφιακή.

2.3.4 Επιλογή των πράξεων AND, OR και NOT

Για την υλοποίηση των πράξεων AND (ΚΑΙ), OR (διαζευκτικό `Η) και NOT (άρνηση) στην ασαφή λογική έχουν προταθεί στη βιβλιογραφία πολλές μέθοδοι ([1], [3], [8, κεφ. 8], [32], [42]). Σύμφωνα με το [42], αν ορίσουμε $I = [0,1]$, τότε μια πράξη τύπου AND είναι μια απεικόνιση $* : I \times I \to I$, ονομάζεται *triangular norm* ή απλά *T-norm* και ικανοποιεί τις ακόλουθες συνθήκες:

i) $a * 1 = a$
ii) $a * b = b * a$
iii) $a * (b * c) = (a * b) * c$
iv) $a \leq b \Rightarrow a * c \leq b * c$

για κάθε a, b, c ∈ I. Δηλ. η πράξη * έχει ουδέτερο στοιχείο το 1, είναι αντιμεταθετική, προσεταιριστική και αύξουσα ως προς τα δύο ορίσματά της. Αντίστοιχα μία πράξη τύπου OR είναι επίσης μια απεικόνιση * : I x I → I, ονομάζεται *T-conorm* και ικανοποιεί τις συνθήκες (ii), (iii), (iv) και τη συνθήκη

i') $a * 0 = a$

για κάθε a ∈ I, δηλ. έχει για ουδέτερο στοιχείο το 0 αντί για το 1. Τέλος η πράξη NOT είναι μια απεικόνιση c : I → I που ορίζεται ως $x^c = 1 - x \quad \forall x \in I$. Αποδεικνύεται τότε ότι αν T είναι ένα συνεχές T-norm, τότε η πράξη T^c που ορίζεται ως

$$a\ T^c\ b = (\ a^c\ T\ b^c\ )^c \qquad (2.21)$$

είναι ένα συνεχές T-conorm. Η σχέση (2.21) εκφράζει το νόμο του *De Morgan*. Για να ισχύει ένα "συνεπές" λογικό σύστημα, οι πράξεις AND και OR θα πρέπει να ορίζονται σε "ζευγάρια" έτσι ώστε να ισχύει ο νόμος αυτός. Πράγματι, όλοι οι τύποι AND & OR που έχουν προταθεί στη βιβλιογραφία ορίζονται κατ'αυτό τον τρόπο. Οι κυριότεροι απ'αυτούς είναι:

1) Για την πράξη AND :

i) Minimum: $a\ T\ b = \min(a, b)$

ii) Product: $a\ T\ b = a \cdot b$

iii) Bounded Product: $a\ T\ b = \max(a+b-1, 0)$

iv) Drastic Product:
$$a\ T\ b = \begin{cases} 0, & \text{αν } a, b < 1 \\ \min(a, b), & \text{αλλού} \end{cases}$$

v) Yager T-norm: $a\ T\ b = \max(0, 1 - [(1-a)^p + (1-b)^p]^{1/p}), \quad p>0$

vi) Gamma Product: $a\ T\ b = (a \cdot b)^{1-\gamma}, \quad \gamma<1$

2) Για την πράξη OR :

i) Maximum: $a\ T^c\ b = \max(a, b)$

ii) Probabilistic Sum: $a\ T^c\ b = a + b - ab$

iii) Bounded Sum: $a\ T^c\ b = \min(a+b, 1)$

iv) Drastic Sum:
$$a\ T^c\ b = \begin{cases} 1, & \text{αν } a, b > 0 \\ \max(a, b), & \text{αλλού} \end{cases}$$

v) Yager T-norm: $a\ T^c\ b = \min(1, (a^p + b^p)^{1/p})$

vi) Gamma Sum: $a\ T^c\ b = 1 - [(1-a)(1-b)]^{1-\gamma}, \quad \gamma<1$

Είναι πολύ εύκολο να δείξουμε ότι οι πράξεις minimum & maximum πληρούν τη συνθήκη (2.21) (De Morgan). Το ίδιο συμβαίνει με τις product & probabilistic sum, bounded product & bounded sum κ.λ.π. Επίσης είναι προφανές

ότι όλες οι παραπάνω πράξεις είναι αντιμεταθετικές και αύξουσες ως προς τα a και b, ενώ όλες οι πράξεις AND έχουν ουδέτερο στοιχείο το 1 και οι πράξεις OR το 0. Την προσεταιριστικότητα είναι εύκολο να τη δείξουμε χρησιμοποιώντας και την ιδιότητα De Morgan :

$$(a\ T\ b)\ T\ c = a\ T\ (b\ T\ c) \Leftrightarrow$$
$$(a^c\ T^c\ b^c)^c\ T\ c = a\ T\ (b^c\ T^c\ c^c)^c \Leftrightarrow$$
$$(a^c\ T^c\ b^c)^c\ T\ c^c = a^c\ T^c\ (b^c\ T^c\ c^c) \qquad (2.22)$$

Δηλ. αν η προσεταιριστικότητα ισχύει για κάποιο τύπο AND, τότε ισχύει και για τον αντίστοιχο τύπο OR & αντίστροφα. Ετσι έχουμε :

1) Τα **minimum**, **maximum** είναι προφανώς προσεταιριστικά.
2) Το ίδιο ισχύει για το **product**, επομένως λόγω της (2.22) και για το **probabilistic sum**.
3) Για το **bounded product** έχουμε
$$(a\ T\ b)\ T\ c = \max\ (\ \max(a+b-1,\ 0\ ) + c - 1,\ 0) =$$
$$= \max\ (\ \max(a+b+c-2,\ c-1),\ 0\ )$$
$$= \max\ (a+b+c-2,\ 0)$$

Ομοια $a\ T\ (b\ T\ c) = \max\ (a+b+c-2,\ 0)$, άρα ισχύει η προσεταιριστικότητα, οπότε λόγω της (2.22) ισχύει και για το **bounded sum**. Γενικότερα, για την περίπτωση N εισόδων, το bounded product γίνεται :

$$a_1\ T\ a_2\ T\ \ldots\ T\ a_N = \max\ (a_1+\ldots+a_N-N+1,\ 0) \qquad (2.23)$$

ενώ το bounded sum :

$$a_1\ T^c\ a_2\ T^c\ \ldots\ T^c\ a_N = \min\ (a_1+\ldots+a_N,\ 1) \qquad (2.24)$$

4) Για το **drastic product** έχουμε :

$$(a\ T\ b)\ T\ c = \begin{cases} 0, & \text{αν } (aTb),\ c<1 \\ \min(aTb,c), & \text{αν } (aTb)=1 \text{ ή } c=1 \end{cases}$$

$$= \begin{cases} 0, & \text{αν } (\ a<1 \text{ ή } b<1\ ) \text{ και } c<1 \\ x_3, & \text{αν } (aTb)=1 \Leftrightarrow a=b=1 \\ 0, & \text{αν } (a,b < 1) \text{ και } c=1 \\ \min(a,b), & \text{αν } (a=1 \text{ ή } b=1) \text{ και } c=1 \end{cases}$$

$$= \begin{cases} 0, & \text{αν 2 τουλάχιστο απ'τα a,b,c είναι } <1 \\ \min(a,b,c), & \text{αλλιώς} \end{cases}$$

Ομοιο ακριβώς προκύπτει και το αT(bTc) οπότε ισχύει η προσεταιριστικότητα. Επομένως ισχύει και για το **drastic sum**.

Γενικότερα, για N εισόδους, το drastic product γίνεται:

$$a_1 T a_2 T \ldots T a_N = \begin{cases} 0, & \text{αν 2 τουλ. απ'τα } a_1,\ldots,a_N \text{ είναι } <1 \\ \min(a_1\ldots,a_N) & ,\text{αλλιώς} \end{cases} \qquad (2.25)$$

και το drastic sum:

$$a_1 T^c a_2 T^c \ldots T^c a_N = \begin{cases} 1, & \text{αν 2 τουλ. απ'τα } a_1,\ldots,a_N \text{ είναι } >0 \\ \max(a_1,\ldots,a_N), & \text{αλλιώς} \end{cases} \qquad (2.26)$$

5) Για το **Yager T-conorm** έχουμε:

$$(aT^c b)T^c c = \min(1,\ (\ (\min(1,(a^p+b^p)^{1/p}))^p + c^p\ )^{1/p}) =$$

$$= \begin{cases} \min(1,\ (a^p+b^p+c^p)^{1/p}), & \text{αν } (a^p+b^p)^{1/p} \le 1 \\ \min(1,\ (1+c^p)^{1/p}) = 1, & \text{αν } (a^p+b^p)^{1/p} > 1 \end{cases}$$

$$= \min(1,\ (a^p+b^p+c^p)^{1/p})$$

Ομοιο προκύπτει και το aT(bTc), οπότε το Yager T-conorm (επομένως και το **Yager T-norm**) είναι προσεταιριστικό. Γενικά, για N εισόδους, το Yager T-norm γίνεται:

$$a_1 T a_2 T \ldots T a_N = \max(0,\ 1-[(1-a_1)^p+\ldots+(1-a_N)^p]^{1/p}) \qquad (2.27)$$

και το T-conorm:

$$a_1 T^c a_2 T^c \ldots T^c a_N = \min(1,\ [a_1^p+a_2^p+\ldots+a_N^p]^{1/p}) \qquad (2.28)$$

6) Τέλος, για τις πράξεις **gamma product & gamma sum** η προσεταιριστικότητα δεν ισχύει.

Οι γραφικές παραστάσεις των παραπάνω πράξεων φαίνονται στα Σχήματα 2.10 (AND) και 2.11 (OR). Στα σχήματα αυτά παριστάνεται το a T b (για το AND) ή το a T^c b (για το OR) συναρτήσει του $a\in[0,1]$ για διάφορες τιμές του b (b = 0, 0.1, 0.2,..., 1). Ως προς την επιλογή των πράξεων αυτών μπορούμε να κάνουμε τα παρακάτω σχόλια:

1) Οι πράξεις AND και OR μπορούν να επιλέγονται αυθαίρετα, αρκεί να ισχύει η συνθήκη (2.22) (De Morgan).

2) Οι πράξεις min & max είναι οι μόνες για τις οποίες ισχύουν *όλες* οι ιδιότητες οι αντίστοιχες της άλγεβρας Boole που αναφέρθηκαν στην παράγραφο 1.3. Επομένως σε ένα λογικό σύστημα που πρέπει να είναι επέκταση της κλασσικής λογικής (ιδίως στα *έμπειρα συστήματα*) η επιλογή min - max είναι απαραίτητη. Πάντως στην περίπτωση που τα a, b περιορίζονται στο {0, 1},

όλες οι πράξεις AND & OR ανάγονται στις αντίστοιχες της κλασσικής λογικής:

AND

a	b	a T b
0	0	0
0	1	0
1	0	0
1	1	1

OR

a	b	a T b
0	0	0
0	1	1
1	0	1
1	1	1

3) Στα συστήματα ελέγχου αντίθετα, αυτό που κυρίως μας ενδιαφέρει είναι η σωστή συμπεριφορά του συστήματος. Στην περίπτωση αυτή, συναρτήσεις όπως οι min, bounded product και drastic product (και οι αντίστοιχες για το OR) που παρουσιάζουν "*επίπεδα*" τμήματα στις γραφικές παραστάσεις τους, θα πρέπει να αποφεύγονται. Οπως εξηγείται στο [32] οι συναρτήσεις που είναι *γνησίως αύξουσες* ως προς τα a και b, που παριστάνονται δηλ. από συνεχώς μεταβαλλόμενες καμπύλες (product, Yager T-norm, gamma product και οι αντίστοιχες OR), εκφράζουν πιο σωστά την ανθρώπινη κρίση σε μια διαδικασία ελέγχου.

4) Το drastic product (και drastic sum) ελάχιστα χρησιμοποιείται σε συστήματα ελέγχου γιατί τότε ένας κανόνας λαμβάνεται υπόψη μόνο αν κάποια υπόθεσή του έχει βαθμό συνέπειας 1. Ετσι ουσιαστικά προκύπτουν σαφή (crisp) συστήματα. Σε μικρότερο βαθμό ισχύει η παρατήρηση αυτή και για το bounded product (και bounded sum).

5) Τα product και probabilistic sum μπορούν να χρησιμοποιηθούν σε διαδικασίες που παρουσιάζουν *πιθανοκρατική* συμπεριφορά, καθώς υλοποιούν το γινόμενο και το άθροισμα πιθανοτήτων αντίστοιχα.

6) Αξίζει να παρατηρήσουμε ότι για $p \to 0$, το Yager T-norm [T-conorm] ταυτίζεται με το drastic product [sum]. Για $p=1$ ταυτίζεται με το bounded product [sum], ενώ για $p \to \infty$ με το minimum [maximum]. Η ιδιότητα αυτή, που απεικονίζεται και στα σχήματα 2.8 και 2.9 είναι πολύ χρήσιμη γιατί μας επιτρέπει να εκφράσουμε τις πράξεις AND και OR *παραμετρικά*. Δηλ. για διάφορες τιμές μιας μόνο παραμέτρου μπορούμε να έχουμε διαφορετικούς τύπους AND/OR. Κάτι ανάλογο συμβαίνει με την παράμετρο γ των gamma product/sum.

6) Τελικά στην πράξη στα συστήματα ελέγχου χρησιμοποιούνται μόνο οι πράξεις min/max, λόγω της παρατήρησης (1) αλλά και λόγω της εύκολης υλοποίησής τους σε hardware. Σε ορισμένες περιπτώσεις επίσης χρησιμοποιούνται τα product/probabilistic sum σε συνδυασμό με implication τύπου product [1]. Στην πρακτική μας εφαρμογή χρησιμοποιήσαμε AND τύπου minimum.

Τέλος, μπορούμε να αναφέρουμε ότι για την πράξη NOT έχει προταθεί ένας μόνο εναλλακτικός τρόπος υλοποίησης [8, κεφ. 7]:

$$N_a(x) = (1-x) / (1+ax) , \quad a>-1 \tag{2.29}$$

που όμως ισοδυναμεί με ένα συνδυασμό του κλασσικού NOT και ενός τύπου concentration/dilation (που παρουσιάζεται σε επόμενο κεφάλαιο) και δεν έχει χρησιμοποιηθεί στην πράξη.

2.3.5 Επιλογή του implication

Για την πράξη της *συνεπαγωγής (implication)* έχουν προταθεί στη βιβλιογραφία πάνω από 15 διαφορετικοί τρόποι υλοποίησης ([1], [6], [8, κεφ.8], [12], [27], [48], και κυρίως στο [33]). Οι κυριότεροι απ'αυτούς είναι:

i) Minimum: $a \rightarrow b = \min(a, b)$

ii) Product: $a \rightarrow b = a \cdot b$

iii) Lukaciewicz: $a \rightarrow b = \min(1-a+b, 1)$

iv) Lee: $a \rightarrow b = \max(1-a, b)$

v) Gödel: $a \rightarrow b = \begin{cases} 1, & \text{αν } a \le b \\ b, & \text{αλλού} \end{cases}$

vi) Goguen: $a \rightarrow b = \begin{cases} 1, & \text{αν } a \le b \\ b/a, & \text{αλλού} \end{cases}$

Μία πρώτη παρατήρηση που θα κάνουμε είναι ότι οι τύποι implication μπορούν να χωριστούν σε 2 ομάδες : τα (i), (ii) που ονομάζουμε implication **τύπου I** και τα υπόλοιπα που ονομάζουμε **τύπου II**. Ολα τα implication είναι συναρτήσεις της μορφής "→" : I x I → I, και είναι αύξουσες συναρτήσεις ως προς το b. Ομως, ενώ τα implication τύπου I είναι επίσης *αύξουσες* συναρτήσεις και ως προς το a, τα implication τύπου II είναι *φθίνουσες* συναρτήσεις ως προς το a. Η διαφορετική συμπεριφορά των 2 τύπων implication φαίνεται στο Σχήμα 2.12, όπου για το ασαφές σύνολο A = trapez (0.5, 0.4, 0.4) βρίσκουμε για κάθε τύπο implication το σύνολο A':

$$\mu_{A'}(x) = a \rightarrow \mu_A(x) \qquad \forall\, x \in X$$

όπου a=0.6 και X=[0,1]. Γίνεται εδώ προφανές ότι όσο το a μικραίνει, το A' "μικραίνει" για τα implications τύπου I και "μεγαλώνει" για τα implications τύπου II.

Επίσης, αν περιορίσουμε τα a,b στο {0,1}, τα implication τύπου I δίνουν:

a	b	a → b
0	0	0
0	1	0
1	0	0
1	1	1

Πίνακας 2.1: *Implication τύπου I*

δηλ. ισοδυναμούν με πράξη AND. Αντίθετα τα implication τύπου II δίνουν :

a	b	a → b
0	0	1
0	1	1
1	0	0
1	1	1

Πίνακας 2.2: *Implication τύπου II*

δηλ. ισοδυναμούν με τον τύπο implication που χρησιμοποιείται στην κλασσική λογική ([2], [8, κεφ.7]), ο οποίος "*είναι ψευδής αν και μόνο αν η υπόθεση (antecedent) είναι αληθής και το επακόλουθο (consequent) ψευδές*" (Ορισμός 1).

Το γεγονός αυτό μας κάνει να αναρωτηθούμε : τι σημαίνει "a → b = c" σύμφωνα με τον ορισμό που δόθηκε παραπάνω, και τί σημαίνει "a → b = c" όταν η πράξη αυτή χρησιμοποιείται σε ένα ασαφές σύστημα, όπως στην παράγραφο 2.2;

Στην 1η περίπτωση, το a υποδηλώνει την υπόθεση, το b το επακόλουθο, και το c δηλώνει πόσο η πρόταση "το a συνεπάγεται το b" είναι αληθής. Δηλ. γνωρίζουμε πόσο ισχύει η υπόθεση και το επακόλουθο (τα a και b) και ζητάμε το c δηλ. προσπαθούμε να αξιολογήσουμε τη συνεπαγωγή. Με τη λογική αυτή ο ορισμός 1 είναι σωστός και ένα implication τύπου II μπορεί να χρησιμοποιηθεί. Πιο γενικά στην περίπτωση της ασαφούς λογικής ο ορισμός 1 μπορεί να επεκταθεί και να γίνει: "*η συνεπαγωγή είναι απόλυτα αληθής αν και μόνο αν ο βαθμός αλήθειας της υπόθεσης είναι μικρότερος ή ίσος με το βαθμό αλήθειας του επακόλουθου*" (Ορισμός 2). Ο ορισμός αυτός επιβεβαιώνεται με το θεώρημα 2.7 του [42], σύμφωνα με το οποίο για κάθε implication τύπου II ισχύει ότι a → b = 1 αν και μόνο αν $a \leq b$.

Στη 2η περίπτωση όμως, τα πράγματα είναι διαφορετικά: γνωρίζουμε πόσο ισχύει η υπόθεση (a) ενός κανόνα, γνωρίζουμε πόσο ισχύει ο κανόνας (c) και ζητάμε το επακόλουθο (b) του κανόνα. Αρα λοιπόν στην περίπτωση αυτή χρειάζεται μια πράξη αντίθετη από το implication. Σύμφωνα με το θεώρημα 3.2 του [42] (Modus Ponens), ένας λογικός τρόπος για να βρούμε το b γνωρίζοντας

το a και το a → b είναι η πράξη *AND*, αφού [a T (a→b)]→b. Επομένως τα implications τύπου I, που ισοδυναμούν ουσιαστικά με AND, είναι καταλληλότερα για την περίπτωση αυτή. Με τη λογική αυτή, θα μπορούσαν να χρησιμοποιηθούν σαν implication και άλλοι τύποι AND, όπως bounded product, Yager T-norm κλπ. Στην πρακτική εφαρμογή του μέρους Β επιλέξαμε σαν implication το product.

Στην πράξη στα ασαφή συστήματα ελέγχου χρησιμοποιούνται σχεδόν πάντα τα implications τύπου I (minimum ή γινόμενο). Γίνεται δηλ. η σιωπηλή παραδοχή ότι τα implications αυτά έχουν καλύτερη συμπεριφορά (όντως έχουν) χωρίς όμως να εξηγείται το γιατί. Μια λογική εξήγηση για το θέμα αυτό δίνεται στο [8, κεφ. 8] όπου αναφέρεται ότι το implication τύπου Lukaciewicz (τύπου II) τείνει να είναι τις περισσότερες φορές κοντά στη μονάδα - αφού min(1-a+b, 1) < 1 μόνο αν a>b - και επομένως τα επιμέρους συμπεράσματα των κανόνων είναι σχεδόν παντού ίσα με τη μονάδα. Αυτό θα γίνει πιο φανερό αργότερα που θα ασχοληθούμε με το συνδυασμό implication και aggregation.

Ας δούμε τώρα πώς μπορούμε να συνδυάσουμε κάθε τύπο implication με κάποιο τύπο AND (και συνεπώς με κάποιο τύπο OR, αφού τα AND & OR συνδυάζονται μεταξύ τους σύμφωνα με το νόμο De Morgan). Για το σκοπό αυτό θα χρησιμοποιήσουμε το επιχείρημα ότι πρέπει να πληρούται η σχέση

$$a \rightarrow (b \rightarrow c) = (a \; T \; b) \rightarrow c \qquad (2.30)$$

Η σχέση αυτή ισχύει προφανώς στην κλασσική λογική, ενώ το παραπάνω επιχείρημα αναφέρεται επίσης και στα [1], [42]. Για κάθε τύπο implication, λοιπόν, αρκεί να βρούμε ένα τύπο AND ώστε να ικανοποιείται η (2.30).

i,ii) Για τα *minimum & product,* που είναι τύποι AND ουσιαστικά, μπορούμε να επιλέξουμε το AND ίδιο με το implication, οπότε λόγω της προσεταιριστικότητας του AND η (2.30) ισχύει.

iii) Για το *Lukaciewicz* έχουμε:

$$\begin{aligned} a \rightarrow (b \rightarrow c) &= \min\,(1 - a + \min(1-b+c,\ 1),\ 1) = \\ &= \min\,(\ \min(1 - (a+b-1) + c,\ 2-a),\ 1) = \\ &= \min\,(1 - (a+b-1) + c,\ 2-a,\ 1) = \\ &= \min\,(1 - (a+b-1) + c,\ 1) = \\ &= [\ \max\,(a+b-1)\ ] \rightarrow c = \\ &= (a \; T \; b) \rightarrow c \end{aligned}$$

αρκεί να επιλέξουμε σαν AND το *bounded product.*

iv) Για το *Lee* έχουμε:

$$
\begin{aligned}
a \to (b \to c) &= \max\,(1-a,\ \max(1-b,\ c)) = \\
&= \max\,(\ \max(1-a,\ 1-b),\ c) = \\
&= \max\,(1 + \max(-a,\ -b),\ c) = \\
&= \max\,(1 - \min(a,\ b),\ c) = \\
&= \min\,(a,\ b) \to c = \\
&= (a\ T\ b) \to c
\end{aligned}
$$

αρκεί να επιλέξουμε σαν AND το minimum.

v) Για το *Gödel* έχουμε:

$$
a \to (b \to c) = a \to \begin{cases} 1, & \text{αν } b \le c \\ c, & \text{αν } b > c \end{cases}
$$

$$
= \begin{cases} 1, & \text{αν } b \le c \text{ και } a \le 1 \\ 1, & \text{αν } b \le c \text{ και } a \ge 1 \text{ (που δεν ισχύει)} \\ 1, & \text{αν } b > c \text{ και } a \le c \\ c, & \text{αν } b > c \text{ και } a > c \end{cases}
$$

$$
= \begin{cases} 1, & \text{αν } \min(a,\ b) \le c \\ c, & \text{αν } \min(a,\ b) > c \end{cases}
$$

$$
= \min(a,\ b) \to c = (a\ T\ b) \to c
$$

αρκεί να επιλέξουμε σαν AND το minimum.

vi) Τέλος, για το *Goguen* έχουμε:

$$
a \to (b \to c) = a \to \begin{cases} 1, & \text{αν } b \le c \\ c/b, & \text{αν } b > c \end{cases}
$$

$$
= \begin{cases} 1, & \text{αν } b \le c \text{ και } a < 1 \\ 1/a, & \text{αν } b \le c \text{ και } a > 1 \text{ (που δεν ισχύει)} \\ 1, & \text{αν } b > c \text{ και } a \le c/b \\ c/(ab), & \text{αν } b > c \text{ και } a > c/b \end{cases}
$$

$$
= \begin{cases} 1, & \text{αν } ab \le c \\ c/(ab), & \text{αν } ab > c \end{cases}
$$

$$
= (ab) \to c = (a\ T\ b) \to c
$$

αρκεί σαν AND να επιλέξουμε το product.

Συνοπτικά λοιπόν έχουμε :

Implication →	AND
minimum	minimum
product	product
Lukeciewicz	bounded product
Lee	minimum
Gödel	minimum
Goguen	product

Πίνακας 2.3: *Δυνατοί συνδυασμοί AND - Implication.*

Στο ίδιο ακριβώς αποτέλεσμα μπορούσαμε να φτάσουμε με το επιχείρημα που χρησιμοποιείται στο [42]. Εκεί για κάθε τύπο AND (T-norm) ορίζεται ένα implication (τύπου II) "→" : I x I → I σύμφωνα με τη σχέση

$$a \rightarrow b = \sup \{x \mid a \; T \; x \leq b\} \quad \forall \, a,b \in I \tag{2.31}$$

Η σχέση αυτή δίνει σχεδόν τα ίδια αποτελέσματα με αυτά του Πίνακα 2.3 για τα implication τύπου II (εκτός από το Lee). Στη συνέχεια με βάση αυτό τον ορισμό αποδεικνύεται ότι ισχύει η (3.30).

2.3.6 Επιλογή του aggregation.

Οι διάφοροι τύποι aggregation που έχουν προταθεί [1], [6], [8, κεφ.10], [27] είναι :

i) OR (δηλ. ίδιο με τον τύπο του OR)
ii) AND (δηλ. ίδιο με τον τύπο του AND)
iii) Αθροισμα (Sum) (δηλ. "ELSE" = "+")

Ο τύπος (i) χρησιμοποιείται πάντα σε συνδυασμό με τα implication τύπου I ενώ ο τύπος (ii) με τα implication τύπου II ([1], [27], [29], [70]). Αυτό θα γίνει πιο κατανοητό με ένα παράδειγμα. Στο παράδειγμα αυτό θα χρησιμοποιήσουμε το minimum σαν αντιπροσωπευτικό του τύπου I και το Lee σαν αντιπροσωπευτικό του τύπου II. Ας θεωρήσουμε λοιπόν ένα σύστημα που έχει τους εξής 2 κανόνες :

R_1 : IF x IS A_1 THEN y IS B_1
R_2 : IF x IS A_2 THEN y IS B_2

όπου τα ασαφή σύνολα B_1, B_2 φαίνονται στο Σχήμα 2.13, και ας υποθέσουμε ότι σε κάποια συγκεκριμένη χρονική στιγμή έχουμε $\mu_1 = \mu_{A1}(x) = 0.3$ και $\mu_2 = \mu_{A2}(x) = 0.7$.

Στο Σχήμα 2.14 χρησιμοποιήσαμε implication minimum και aggregation maximum (το αντίστοιχο OR). Φαίνεται εκεί ότι συνάρτηση συμμετοχής του τελικού συμπεράσματος B παίρνει μεγαλύτερες τιμές κοντά στο B_2, του οποίου η υπόθεση ισχύει περισσότερο. Έτσι όταν γίνει defuzzification (που συνήθως επιλέγεται το center of gravity, όπως θα δούμε στη συνέχεια και δίνει σαν τελικό αποτέλεσμα την αριστερή κατακόρυφη διακεκομμένη γραμμή) η τελική έξοδος θα εξαρτάται περισσότερο από τη θέση του B_2, παρά από τη θέση του B_1, όπως είναι αναμενόμενο. Αν αντίθετα επιλέγαμε σαν aggregation το minimum (δηλ. AND), τότε θα είχαμε $\mu_B(x) = 0$ παντού και δε θα μπορούσαμε να πάρουμε κανένα λογικό αποτέλεσμα.

Αντίστοιχα στο Σχήμα 2.15 χρησιμοποιήσαμε implication Lee και aggregation minimum (το αντίστοιχο AND). Και στην περίπτωση αυτή ισχύουν ανάλογα συμπεράσματα, ενώ βλέπουμε ότι αν χρησιμοποιούσαμε σαν aggregation το maximum (δηλ. OR) θα προέκυπτε το παράλογο αποτέλεσμα η έξοδος να εξαρτάται περισσότερο από τον κανόνα που ισχύει λιγότερο. Στο παράδειγμα αυτό βλέπουμε επίσης ότι αν είχαμε $\mu_1 = \mu_2 = \mu$ τότε το $\mu_B(x)$ θα ήταν σταθερό και ίσο με $1-\mu$ παντού. Έτσι δε θα μπορούσαμε να εξάγουμε κανένα λογικό συμπέρασμα. Γενικότερα, οι τύποι II για το implication αντιμετωπίζουν προβλήματα όταν περισσότεροι από έναν αντιφατικοί μεταξύ τους κανόνες ισχύουν στον ίδιο βαθμό.

Τέλος αναφέρουμε ότι το aggregation τύπου SUM χρησιμοποιείται μόνο στην περίπτωση του implication τύπου I, αντί για το OR. Έχει καλύτερα αποτελέσματα από το OR, γιατί αν π.χ. 2 κανόνες ισχύουν στον ίδιο βαθμό και έχουν το ίδιο επακόλουθο, το OR αγνοεί το 2ο κανόνα, ενώ αντίθετα με το SUM ο 2ος κανόνας λαμβάνεται υπόψη (αφού στην περίπτωση αυτή όλες οι συναρτήσεις συμμετοχής προστίθενται). Για το λόγο αυτό στην πρακτική εφαρμογή του Μέρους B χρησιμοποιήσαμε aggregation τύπου SUM.

2.3.7 Επιλογή του defuzzification.

Για το defuzzification έχουν επίσης προταθεί αρκετοί τρόποι υλοποίησης, από τους οποίους παρουσιάζουμε τέσσερις.

1) **Center of gravity** ή **center of area (COA)**. Ο γενικός τύπος της συνάρτησης defuzzifier της σχέσης (2.7), που δίνει το τελικό (σαφές) αποτέλεσμα εξόδου, είναι :

$$y_o = \frac{\int_Y \mu_{B'}(y) y \, dy}{\int_Y \mu_{B'}(y) \, dy} \qquad (2.32)$$

για συνεχές υπερσύνολο αναφοράς Y, και

$$y_o = \frac{\sum_{y_i \in Y} \mu_B(y_i)\, y_i}{\sum_{y_i \in Y} \mu_B(y_i)} \qquad (2.33)$$

για διακριτό υπερσύνολο αναφοράς Y. Ο τύπος αυτός λαμβάνει υπόψη του τα επιμέρους συμπεράσματα όλων των κανόνων και είναι ο πιο συχνά χρησιμοποιούμενος.

2) **Mean of Maxima (MOM)**, ή *μέσος όρος μεγίστων*. Η μέθοδος αυτή είναι ίδια με την παραπάνω, με τη διαφορά ότι η τελική τιμή προκύπτει από το κέντρο βάρους της περιοχής που βρίσκεται κάτω από τις *μέγιστες* τιμές της καμπύλης. Έτσι λαμβάνονται υπόψη μόνο οι κανόνες με τον υψηλότερο βαθμό συνέπειας.

Η μέθοδος MOM είναι η καταλληλότερη για το implication τύπου II και χρησιμοποιείται σε συνδυασμό με aggregation τύπου AND. Αυτό γίνεται περισσότερο κατανοητό από την καμπύλη του Σχήματος 2.15. Στο Σχήμα αυτό η δεξιά κατακόρυφη διακεκομμένη γραμμή δείχνει το αποτέλεσμα του deffuzification με τη μέθοδο MOM, ενώ η αριστερή με τη μέθοδο COA. Παρατηρούμε όμως ότι το "οροπέδιο", που σχηματίζεται στην καμπύλη B' λόγω του implication Lee δεν περιέχει ουσιαστικά καμία απολύτως πληροφορία, και επομένως η μέθοδος COA εσφαλμένα επηρεάζεται από το "οροπέδιο" αυτό και δίνει y_0 μικρότερο από το κανονικό (αριστερή γραμμή). Αντίθετα η μέθοδος MOM δεν επηρεάζεται και δίνει την έξοδο που αντιστοιχεί στον κανόνα με τον υψηλότερο βαθμό συνέπειας, το R_2.

Αντίστοιχα στο σχήμα 2.14 η μέθοδος MOM δίνει το ίδιο ακριβώς αποτέλεσμα (δεξιά γραμμή), και η COA δίνει και πάλι μικρότερο y_0 (αριστερή γραμμή). Αυτή τη φορά όμως το αποτέλεσμα αυτό είναι σωστότερο γιατί η μέθοδος COA επηρεάζεται από τον *κανόνα 1* και όχι από μια καμπύλη που δεν περιέχει πληροφορία. Με τον τρόπο αυτό ο κανόνας 1 "ζυγίζεται" κατά 0.3 και ο κανόνας 2 κατά 0.7.

Συμπερασματικά, ο συνδυασμός [implication: τύπου I, aggregation: OR ή SUM, defuzzification : COA] δίνει πιο "συνεπή" αποτελέσματα σε σχέση με το συνδυασμό [implication: τύπου II, aggregation: AND, defuzzification: MOM] και φυσικά χρησιμοποιείται πολύ συχνότερα στην πράξη.

3) Μια ειδική περίπτωση του COA στην περίπτωση που τα επακόλουθα B_i είναι *μοναδιαία (singletons)* (βλ. σχέση (2.8)) είναι το **weighted average** (μέσος όρος με βάρη) που έχει προταθεί στα [6], [8, κεφ.10], [17], [31], [59] και που είναι ιδιαίτερα ελκυστικό λόγω της απλότητας υλοποίησής του. Συγκεκριμένα, στην ειδική περίπτωση που επιλέγουμε implication : product και aggregation : Sum, η μέθοδος αυτή δίνει αποτέλεσμα

$$y_o = \frac{\mu_1 y_1 + \cdots + \mu_N y_N}{\mu_1 + \cdots + \mu_N} \qquad (2.34)$$

όπου $\mu_1,\ldots,\mu_N$ είναι οι βαθμοί ικανοποίησης των N κανόνων και $y_1,\ldots,y_N$ οι προτεινόμενες έξοδοι. Παρά το γεγονός ότι η εξίσωση (2.34) είναι πολύ εύκολο να υλοποιηθεί σε hardware - γι'αυτό εξάλλου και επιλέχθηκε στην εφαρμογή του Μέρους Β - τα αποτελέσματά της είναι εξίσου ικανοποιητικά, όπως διαπιστώθηκε και στην πράξη. Αποδεικνύεται μάλιστα ότι η μέθοδος αυτή είναι ισοδύναμη με τη μέθοδο COA στην περίπτωση που έχουμε implication : product και aggregation : Sum. Στην περίπτωση αυτή η σχέση (2.32) γίνεται ([8, θεωρ.11])

$$y_o = \frac{\sum_{i=1}^{N} \mu_i c_i I_i}{\sum_{i=1}^{N} \mu_i I_i} \qquad (2.35)$$

όπου I_i είναι το εμβαδό της καμπύλης B_i (βλ. § 2.2) και c_i είναι η μέση τιμή της. Αν θεωρήσουμε ότι $I_1 = I_2 = \ldots = I_N$ και ονομάσουμε $y_i = C_i$, i=1,...,N η (2.35) καταλήγει στην (2.34).

4) Τέλος, αναφέρουμε τη μέθοδο **SLIDE** (*S*emi-*LI*near *De*fuzzification) που προτείνεται στο [37]. Με τη γενικευμένη αυτή μέθοδο δίνεται η δυνατότητα να μεταβάλλεται ο τύπος του defuzzification ανάλογα με την τιμή μιας παραμέτρου δ. Η μέθοδος SLIDE περιέχει τις μεθόδους COA και **MOM** σαν ειδικές περιπτώσεις. Συγκεκριμένα, η πρώτη προκύπτει για $\delta = 1$ και η δεύτερη για $\delta \to \infty$. Επίσης αναπτύσσεται στο [37] ένας αλγόριθμος "εκμάθησης" της παραμέτρου δ ώστε τα αποτελέσματα να είναι όσο πιο κοντά γίνεται στα αναμενόμενα.

Κεφάλαιο 3

ΕΜΠΕΙΡΑ ΣΥΣΤΗΜΑΤΑ

Μια δεύτερη πολύ σημαντική εφαρμογή της ασαφούς λογικής είναι τα έμπειρα συστήματα. Σαν επέκταση της κλασσικής λογικής, η ασαφής λογική δίνει στα έμπειρα συστήματα τη δυνατότητα εξαγωγής συμπερασμάτων μέσα σε αβεβαιότητα. Η μέθοδος της ασαφούς λογικής διαφέρει σε μεγάλο βαθμό από τη μέθοδο των *"συντελεστών βεβαιότητας" (certainty factors)* που χρησιμοποιείται στα κλασσικά έμπειρα συστήματα [2]. Ενώ δηλ. οι συντελεστές βεβαιότητας λειτουργούν "έξω" από το μηχανισμό εξαγωγής συμπερασμάτων του συστήματος και απλά αξιολογούν την αξιοπιστία των συμπερασμάτων, η ασαφής λογική είναι ένα μαθηματικό "εργαλείο" που είναι στενά συνδεδεμένο με τις μεθόδους συνεπαγωγής και τις τεχνικές εξαγωγής συμπερασμάτων.

Συστήματα στα οποία έχει χρησιμοποιηθεί επιτυχώς η ασαφής λογική είναι το MYCIN, το REVEAL [7] και άλλα. Στα συστήματα αυτά είναι δυνατή η διατύπωση κανόνων της μορφής "οι τιμές θα πρέπει να είναι *χαμηλές*, *περίπου* διπλάσιες από το κόστος, και *περίπου* ίσες με τις τιμές των ανταγωνιστών μας, αρκεί να μην είναι *υπερβολικά χαμηλές*". Συστήματα αυτής της μορφής έχουν πολλές εφαρμογές στην ιατρική διάγνωση, στο μάρκετινγκ, στην λήψη αποφάσεων ως προς τη διοίκηση επιχειρήσεων, στην πρόβλεψη πωλήσεων κ.λ.π.

3.1. Περίπτωση κανόνων με μία υπόθεση

Ενας από τους βασικούς κανόνες του κατηγορικού λογισμού, που χρησιμεύει σαν μηχανισμός εξαγωγής συμπερασμάτων στην κλασσική λογική είναι ο *modus ponens* ("τρόπος του θέτειν") [2], [5] που έχει ως εξής :

Κανόνας : IF X IS A THEN Y IS B
Υπόθεση : X IS A

Συμπέρασμα : Y IS B

όπου "X IS A" και "Y IS B" είναι λογικές προτάσεις και παίρνουν τιμή στο {0, 1}. Οπως είναι φανερό, ένας τέτοιος κανόνας δε μπορεί να δώσει συμπέρασμα στην περίπτωση που η υπόθεση δεν ισχύει *ακριβώς*.

Στην ασαφή λογική, αντίθετα, χρησιμοποιείται ο *γενικευμένος* κανόνας *generalized modus ponens* (GMP) :

Κανόνας : IF X IS A THEN Y IS B
Υπόθεση : X IS A'

Συμπέρασμα : Y IS B'

Βλέπουμε δηλ. ότι ο κανόνας αυτός μπορεί να αντιμετωπίσει προβλήματα της καθημερινής ζωής όπως π.χ. "Αν ο ρυθμός πωλήσεων αυξάνεται τότε οι τιμές πρέπει να είναι υψηλές", γιατί έχει τη δυνατότητα να δίνει "απάντηση" ακόμα και για περιπτώσεις που δεν έχουν προβλεφθεί.

Ενα έμπειρο σύστημα που βασίζεται στην ασαφή λογική λειτουργεί με κανόνες της ίδιας μορφής με την (2.2) με τη διαφορά ότι οι είσοδοι και οι έξοδοι του συστήματος μπορούν να είναι *ασαφείς*. Η εξαγωγή συμπερασμάτων από ασαφείς υποθέσεις βασίζεται στον GMP, η μαθηματική μοντελοποίηση του οποίου έγινε από τον Zadeh το 1973 [5] με τον *συνθετικό συμπερασματικό κανόνα (compositional rule of inference)*:

$$B' = A' \circ R(A,B) \tag{3.1}$$

όπου A' είναι η υπόθεση, B' το συμπέρασμα, "$\circ$" η πράξη της *σύνθεσης (composition)* και R(A,B) ένα δισδιάστατο ασαφές σύνολο που εκφράζει τη *σχέση* μεταξύ των A και B, δηλ. τον κανόνα "IF X IS A THEN Y IS B" :

$$\mu_{R(A,B)}(x,y) = \mu_A(x) \rightarrow \mu_B(y) \quad \forall\, x\in X,\ y\in Y \tag{3.2}$$

όπου X το υπερσύνολο αναφοράς των συνόλων A, A' και Y των συνόλων B, B', ενώ το " $\rightarrow$ " συμβολίζει την πράξη του implication. Η υλοποίηση της πράξης "$\circ$" (composition), όπως ορίστηκε από τον Zadeh, έχει ως εξής :

$$\mu_{B'}(y) = \sup_{x\in X} \{ \mu_{A'}(x) \circ \mu_{R(A,B)}(x,y) \} \quad \forall\, y\in Y \tag{3.3}$$

η οποία, αντικαθιστώντας την (3.2) στην (3.3) γίνεται

$$\mu_{B'}(y) = \sup_{x\in X} \{ \mu_{A'}(x) \circ [\mu_A(x) \rightarrow \mu_B(y)] \} \quad \forall\, y\in Y \tag{3.4}$$

Η παραπάνω σχέση ισχύει για συνεχή υπερσύνολα X,Y. Αν τα X,Y είναι διακριτά (όπως συνήθως γίνεται στην υλοποίηση με hardware), η (3.4) γίνεται

$$\mu'_B(y_i) = \max_j \{ \mu_{A'}(x_j) \circ [\mu_A(x_j) \rightarrow \mu_B(y_i)] \} \quad \forall i \tag{3.5}$$

όπου i=1,...,N, j=1,...,M και M,N είναι το πλήθος των στοιχείων των X,Y αντίστοιχα. Στο εξής θα συμβολίζουμε

$$a_j = \mu_A(x_j), \quad a'_j = \mu_{A'}(x_j), \quad j=1,\ldots,M \tag{3.6}$$
$$b_i = \mu_A(y_j), \quad b'_i = \mu_{B'}(y_j), \quad i=1,\ldots,N \tag{3.7}$$

Ετσι η (3.5) γίνεται

$$b'_i = \max \{a'_j \circ (a_j \rightarrow b_i)\} \quad \forall i \tag{3.8}$$

Τη σχέση αυτή την ονομάζουμε **ΜΟΝΤΕΛΟ I.**

Για την υλοποίηση του " → " (implication) χρησιμοποιείται κάποιος από τους τύπους της παραγράφου 2.3.5, ενώ το " ∘ " είναι ουσιαστικά κάποιος τύπος AND, οπότε υλοποιείται με κάποιο από τους τύπους της παραγράφου 2.3.4. Π.χ. μερικοί τύποι composition είναι

i) **max − min** για AND : minimum
ii) **max − *** για AND : product
iii) **max − θ** για AND : bounded product
iv) **max − Λ** για AND : drastic product

Παρατηρούμε ότι στην ειδική περίπτωση *σαφούς (crisp)* εισόδου x_o, η είσοδος αυτή περιγράφεται με ένα *singleton* (παρ. 2.2) δηλ. με μία συνάρτηση *δέλτα του Kronecker* :

$$\mu_{A'}(x_j) = \begin{cases} 1, & x_j = x_o \\ 0, & x_j \neq x_o \end{cases} \tag{3.10}$$

Και η (3.5) γίνεται

$$\mu_{B'}(y_i) = 1 \circ [\mu_A(x_o) \rightarrow [\mu_B(y_i)] = \mu_A(x_o) \rightarrow \mu_B(y_i) \quad \forall i \tag{3.11}$$

Αντίστοιχα για συνεχή X,Y, το singleton παριστάνεται από μία συνάρτηση *δέλτα του Dirac* και η (3.4) γίνεται :

$$\mu_{B'}(y) = \mu_A(x_o) \rightarrow \mu_B(y) \quad \forall\, y \in Y \tag{3.12}$$

Βλέπουμε λοιπόν ότι η μέθοδος που χρησιμοποιείται στον ασαφή έλεγχο είναι ειδική περίπτωση της μεθόδου που παρουσιάζεται εδώ. Στον ασαφή έλεγχο οι υπολογισμοί είναι πολύ απλούστεροι.

Ακριβώς λόγω της αυξημένης πολυπλοκότητας του Μοντέλου I θα προσπαθήσουμε να βρούμε άλλα, απλοποιημένα μοντέλα που προκύπτουν από το γενικό κάτω από ορισμένες προϋποθέσεις. Παρατηρώντας τη σχέση (3.8) βλέπουμε

ότι θα ήταν πολύ απλούστερο τα a'_j να "συγκριθούν" πρώτα με τα a_j για να δώσουν ένα *μονοδιάστατο βαθμό εκπλήρωσης* (DOF) του κανόνα και στη συνέχεια να εφαρμοστεί η μέθοδος του ασαφούς ελέγχου. Για το σκοπό αυτό εισάγουμε την **ΥΠΟΘΕΣΗ Α:**

$$a \circ (b \rightarrow c) = (a \circ b) \rightarrow c \qquad \forall\, a,b,c \in I \tag{3.13}$$

όπου $I = [0,1]$. Ετσι η (3.8) γίνεται

$$b'_i = \max \{(a'_j \circ a_j) \rightarrow b_i\} \qquad \forall i \tag{3.14}$$

Στη συνέχεια εισάγουμε την **ΥΠΟΘΕΣΗ Β :**

$$a \geq b \Leftrightarrow (a \rightarrow c) \geq (b \rightarrow c) \qquad \forall\, a,b,c \in I \tag{3.15}$$

δηλ. το implication είναι αύξουσα συνάρτηση της υπόθεσης. Ετσι παίρνουμε

$$b'_i = \max_j \{a'_j \circ a_j\} \rightarrow b_i = \mu \rightarrow b \qquad \forall i \tag{3.16}$$

(**ΜΟΝΤΕΛΟ ΙΙ**), που είναι η επιθυμητή σχέση, όπου μ είναι ο βαθμός εκπλήρωσης του κανόνα. Παρατηρούμε ότι το

$$\mu = \max_j \{a'_j \circ a_j\} \tag{3.17}$$

είναι ένα *μέτρο ομοιότητας* των ασαφών συνόλων Α και Α'. Παρόμοια μέτρα ομοιότητας έχουν προταθεί στα [44], [48] και μια συγκριτική μελέτη υπάρχει στο [50]. Χρησιμοποιώντας άλλα μέτρα ομοιότητας η (3.8) δεν ισχύει πλέον, αλλά στην πράξη παίρνουμε καλύτερα αποτελέσματα.

Το Μοντέλο ΙΙ είναι πολύ απλούστερο από το Ι γιατί οι υπολογισμοί που εμπλέκονται σ'αυτό είναι μονοδιάστατοι, ενώ στο Ι δισδιάστατοι. Η ισχύς του επιβεβαιώνεται στο Σχήμα 3.1 για το οποίο χρησιμοποιήθηκε το πρόγραμμα **comp.m** του MATLAB. Στην περίπτωση αυτή επιλέξαμε implication : product και composition : product. Οπως παρατηρούμε, πράγματι ισχύει ότι το αποτέλεσμα της (3.8) (που υπολογίζεται από το **comp.m**) ισούται με εκείνο της (3.16).

Η υπόθεση Α ισχύει όταν ο τύπος του implication επιλέγεται ίδιος με τον τύπο του composition. Τότε όμως ισχύει ταυτόχρονα και η Υπόθεση Β, αφού το composition είναι ουσιαστικά κάποιος τύπος AND, και όλοι οι τύποι AND είναι αύξουσες συναρτήσεις. Βλέπουμε δηλ. ότι *για να ισχύει το απλοποιημένο μοντέλο, πρέπει το implication να είναι τύπου Ι.*

4.2. Περίπτωση κανόνων με περισσότερες υποθέσεις.

Στη συνέχεια παρουσιάζεται η περίπτωση ενός κανόνα με δύο υποθέσεις, συνδεδεμένες με την πράξη AND. Ανάλογη είναι και η περίπτωση της πράξης OR, ενώ η μέθοδος γενικεύεται με τον ίδιο τρόπο και για περισσότερες υποθέσεις συνδεδεμένες με οποιοδήποτε συνδυασμό AND και OR. Στην περίπτωση αυτή ο GMP γίνεται

Κανόνας : IF X IS A THEN Y IS B then Z IS C
Υπόθεση : X IS A', Y IS B'

Συμπέρασμα : Z IS C'

Και η (3.8) γίνεται :

$$c'_i = \max_{j,k} \{(a'_j \; T \; b'_k) \circ [(a_j \; T \; b_k) \rightarrow c_i]\} \qquad \forall i \qquad (3.18)$$

όπου $a_j = \mu_A(x_j)$, $a'_j = \mu_{A'}(x_j)$, $j=1,\ldots,M$
$b_k = \mu_B(y_k)$, $b'_k = \mu_{B'}(y_k)$, $k=1,\ldots,K$
$c_i = \mu_C(z_i)$, $c'_i = \mu_{C'}(z_i)$, $i=1,\ldots,N$

και M,K,N είναι το πλήθος στοιχείων των υπερσυνόλων αναφοράς X,Y,Z αντίστοιχα. (**ΜΟΝΤΕΛΟ I**) Βλέπουμε δηλ. ότι το πρόβλημα γίνεται τρισδιάστατο, και στη γενική περίπτωση πολυδιάστατο. Οπως και προηγουμένως, εφαρμόζοντας τις υποθέσεις A και B, η (3.18) γίνεται

$$c'_i = \max_{j,k} \{(a'_j \; T \; b'_k) \circ (a_j \; T \; b_k)\} \rightarrow c_i \qquad \forall i \qquad (3.19)$$

(**ΜΟΝΤΕΛΟ II**). Ετσι οι διαστάσεις του προβλήματος μειώνονται από 3 σε 2.

Προχωρώντας, έστω ότι ισχύει η **ΥΠΟΘΕΣΗ C**, ότι ο τύπος του composition "$\circ$" επιλέγεται ίδιος με τον τύπο του AND. Τότε λόγω της προσεταιριστικότητας του AND " T " (που δεν ισχύει όμως για το gamma product), έχουμε

$$c'_i = \max_{j,k} \{(a'_j \circ a_j) \; T \; (b'_k \circ b_k)\} \rightarrow c_i \qquad \forall i \qquad (3.20)$$

Αλλά το AND είναι αύξουσα συνάρτηση για όλους τους τύπους AND. Ετσι :

$$c'_i = [(\max_j\{a'_j \circ a_j\}) \; T \; (\max_k\{b'_k \circ b_k\})] \rightarrow c_i \qquad \forall i \qquad (3.21)$$

(**ΜΟΝΤΕΛΟ III**) και το πρόβλημα γίνεται πλέον μονοδιάστατο. Η (3.21) έχει την έννοια ότι υπολογίζουμε το βαθμό εκπλήρωσης της κάθε υπόθεσης χωριστά, συγκρίνουμε τους δύο βαθμούς με AND και εφαρμόζουμε το implication όπως στα συστήματα ελέγχου. Το Μοντέλο III επιβεβαιώθηκε όπως και το II χρησιμοποιώντας

το πρόγραμμα **comp2.m** του MATLAB.

Το Μοντέλο III ισχύει όταν οι τύποι AND, implication, και composition είναι ίδιοι, δηλ. "T" = "∘" = "→". Και πάλι δηλ. *το implication πρέπει να είναι τύπου I.*

Μια εναλλακτική τεχνική που χρησιμοποιείται για την απλοποίηση της γενικής μεθόδου [3], [8, κεφ.8], [27] είναι η *αποσύνθεση ή "σπάσιμο" (decomposition)* του κανόνα στις επιμέρους υποθέσεις, ο υπολογισμός των επιμέρους συμπερασμάτων σύμφωνα με τη σχέση (3.8) (που ισχύει για μία μόνο υπόθεση), και στη συνέχεια η *επανασύνθεση, ή επανένωση (recomposition)* των επιμέρους συμπερασμάτων σύμφωνα με τις λογικές πράξεις (AND ή OR) που συνδέουν τις υποθέσεις. Μια απλή προϋπόθεση για να είναι η μέθοδος αυτή σύμφωνη με τη γενική μέθοδο (3.18) (Μοντέλο I), είναι να ισχύει η **ΥΠΟΘΕΣΗ D** :

$$(a \; T \; b) \rightarrow c = (a \rightarrow c) \; T \; (b \rightarrow c) \qquad \forall \, a,b,c \in I \tag{3.22}$$

δηλ. το " → " να είναι επιμεριστικό ως προς το " T ". Σημειώνουμε όμως ότι η Υπόθεση αυτή *δεν* είναι πάντα σύμφωνη με την κλασσική λογική (όπως ήταν οι Υποθέσεις A, B και C), παρόλα αυτά όμως μπορεί να χρησιμοποιηθεί για την απλοποίηση των πράξεων. Αν ισχύει η (3.22), τότε η (3.18) γίνεται

$$c'_i = \max_{j,k} \{(a'_j \; T \; b'_k) \circ [(a_j \rightarrow c_i) \; T \; (b_k \rightarrow c_i)]\} \qquad \forall i \tag{3.23}$$

Αν επιπλέον ισχύει η Υπόθεση C, τότε αφού η " T " είναι προσεταιριστική και αύξουσα συνάρτηση, καταλήγουμε στην

$$c'_i = \max_{j} \{a'_j \circ (a_j \rightarrow c_i)\} \; T \; \max_{k} \{b'_k \circ (b_k \rightarrow c_i)\} \qquad \forall i \tag{3.24}$$

(**ΜΟΝΤΕΛΟ IV**) δηλ. γίνεται το "σπάσιμο" του κανόνα. Η Υπόθεση D ισχύει *μόνο* στην περίπτωση που επιλέξουμε AND : minimum και ισχύει η Υπόθεση B (δηλ. *το implication είναι τύπου I*). Επίσης στην περίπτωση που οι υποθέσεις συνδέονται με OR αντί για AND, το "σπάσιμο" των κανόνων ισχύει μόνο στην περίπτωση (i) που επιλέξουμε OR : maximum, (ii) ισχύει η υπόθεση B και (iii) *οι είσοδοι του συστήματος είναι singletons.* Παρά τους περιορισμούς αυτούς όμως, η μέθοδος χρησιμοποιείται και σε άλλες περιπτώσεις λόγω της απλότητάς της. Στο πρόγραμμα "Fuzzy Expert", μάλιστα, είναι η μόνη μέθοδος που χρησιμοποιείται.

Αν όμως οι κατάλληλες συνθήκες (δηλ. οι Υποθέσεις C και D) δεν πληρούνται μπορούν να προκύψουν σημαντικά σφάλματα, ιδίως στα συστήματα ελέγχου, με την εφαρμογή αυτής της μεθόδου. Αν για παράδειγμα επιλέξουμε "T" = " → " = product, και ένας κανόνας έχει 2 υποθέσεις που ισχύουν κατά 0.8 και ένα επακόλουθο B που παριστάνεται από το ασαφές σύνολο B του Σχήματος 3.2 τότε η μέθοδος της παρ. 2.2 δίνει το σωστό αποτέλεσμα B_1 που φαίνεται στο Σχήμα 3.2 (αφού υπολογίζεται πρώτα ο συνολικός βαθμός εκπλήρωσης του κανόνα σαν 0.8·0.8 = 0.64 και στη συνέχεια το συμπέρασμα B_1 = 0.64B). Αντίθετα η μέθοδος του

"σπασίματος" υπολογίζει πρώτα το επιμέρους συμπέρασμα από την κάθε υπόθεση, δηλ. το σύνολο $B_2 = 0.8 \cdot B$ και για τις 2 περιπτώσεις, και στη συνέχεια παίρνει το γινόμενο $B_2 \cdot B_2$ και δίνει το λανθασμένο αποτέλεσμα B_3 που φαίνεται στο Σχήμα 3.2.

Γι'αυτό το λόγο *αλλάξαμε* κάποια τμήματα του source code του προγράμματος "Fuzzy Expert" (που είναι γραμμένο σε γλώσσα C) ώστε στην ειδική περίπτωση που οι είσοδοι είναι singletons (δηλ. στα συστήματα ελέγχου) να εφαρμόζεται η γενική μέθοδος της παρ. 2.2 και όχι η μέθοδος του "σπασίματος" (αφού μάλιστα στην περίπτωση αυτή η γενική μέθοδος είναι απλούστερη).

Ολες οι τροποποιήσεις έγιναν στο αρχείο **fuzzy.c** του "Fuzzy Expert". Συγκεκριμένα, η συνάρτηση *partdecide* μετονομάστηκε σε *partdecide_fuz* και τροποποιήθηκε ως εξής :

```
void partdecide_fuz (tree, conclusion, finalresult)
RULETREE tree;
FUZZYSET conclusion;
VALUES_ARRAY finalresult;
{
VALUES_ARRAY result1,result2,result3;
double mx,rval,lval,value;
int i,pos;
if(tree->left == NULL)
 {
  if((result1 = (VALUES_ARRAY)calloc(universe,sizeof(double))) == NULL)
       error(0,0);
  if((result3 = (VALUES_ARRAY)calloc(universe,sizeof(double))) == NULL)
       error(0,0);
  evaluate(conclusion,result1,NULL,NULL);
  evaluate(tree->nodetype.anti.f_const,result3,&lval,&rval);
  pos = tree->nodetype.anti.index;
  if((assignments[cur_case][pos].type == ONE) && (single_mode == mode1))
    {
     mx=find_mx(assignments[cur_case][pos].val1,result3,lval,rval);
     for(i=0;i<universe;i++) f_implie(mx,result1[i],finalresult[i]);
    }
   else
   {
    if((result2 = (VALUES_ARRAY)calloc(universe,sizeof(double))) == NULL)
      error(0,0);
    if((assignments[cur_case][pos].type == ONE) && (single_mode == mode2))
    {
     value = assignments[cur_case][pos].val1;
     single_to_fuzzy(value,lval,rval,result2);
    }
    else
    {
     evaluate(assignments[cur_case][pos].fconst,result2,NULL,NULL);
    }
    compose(result3,result1,result2,finalresult);
    free((void *)result2);
   }
  free((void *)result3);
  free((void *)result1);
 }
 else
 {
  if((result1 = (VALUES_ARRAY)calloc(universe,sizeof(double))) == NULL)
       error(0,0);
  if((result2 = (VALUES_ARRAY)calloc(universe,sizeof(double))) == NULL)
       error(0,0);
  partdecide_fuz(tree->left,conclusion,result1);
  partdecide_fuz(tree->right,conclusion,result2);
```

```
  switch(tree->nodetype.op_type)
   {
    case _AND :set_and(result1,result2,finalresult);
              break;
    case _OR  :set_or(result1,result2,finalresult);
              break;
   }
  free((void *)result1);
  free((void *)result2);
 }
}
```

Επίσης προστέθηκαν 2 νέες συναρτήσεις, οι *partdecide* και *partdecide_det*:

```
void partdecide (tree, conclusion, finalresult)
RULETREE tree;
FUZZYSET conclusion;
VALUES_ARRAY finalresult;
{
 VALUES_ARRAY result1;
 int i,fuzzy_input=0;
 double num_result;
 for(i=0;i<input_var_num;i++)
  if (assignments[cur_case][i].type == FUZZ) fuzzy_input=1;
 if ((fuzzy_input == 0) && (single_mode == mode1))
 {
  if((result1 = (VALUES_ARRAY)calloc(universe,sizeof(double))) == NULL)
       error(0,0);
  evaluate(conclusion,result1,NULL,NULL);
  num_result=partdecide_det(tree);
  for(i=0;i<universe;i++) f_implie(num_result, result1[i], finalresult[i]);
  free((void *)result1);
 }
 else partdecide_fuz(tree, conclusion, finalresult);
}


double partdecide_det (tree)
RULETREE tree;
{
 VALUES_ARRAY result1;
 double lval,rval,num_result,num_result1,num_result2;
 int pos;
 if(tree->left == NULL)
 {
  if((result1 = (VALUES_ARRAY)calloc(universe,sizeof(double))) == NULL)
      error(0,0);
  evaluate(tree->nodetype.anti.f_const,result1,&lval,&rval);
  pos = tree->nodetype.anti.index;
  num_result=find_mx(assignments[cur_case][pos].val1,result1,lval,rval);
  free((void *)result1);
  return(num_result);
 }
 else
 {
  num_result1=partdecide_det(tree->left);
  num_result2=partdecide_det(tree->right);
  switch(tree->nodetype.op_type)
   {
    case _AND :f_and(num_result1,num_result2,num_result,and_type);
              break;
    case _OR  :f_or(num_result1,num_result2,num_result,or_type);
              break;
   }
  return(num_result);
 }
}
```

Ανακεφαλαίωση

Στο μέρος Α της διπλωματικής αυτής εργασίας έγινε μία εκτεταμένη θεωρητική μελέτη για το πώς πρέπει να συνδυάζονται μεταξύ τους οι διάφοροι τρόποι υλοποίησης των πράξεων που περιλαμβάνονται στη θεωρία ασαφούς λογικής. Επίσης αναζητήθηκαν απλούστερες μέθοδοι υπολογισμού και διερευνήθηκαν οι προϋποθέσεις κάτω από τις οποίες οι μέθοδοι αυτές μπορούν να εφαρμοστούν. Ανακεφαλαιώνοντας λοιπόν, τονίζουμε τα παρακάτω συμπεράσματα:

1) Οι πράξεις AND και OR πρέπει να συνδυάζονται μεταξύ τους σε ζεύγη ωστε να ισχύει ο νόμος De Morgan.

2) Το AND με το implication πρέπει να συνδυάζονται σύμφωνα με τον πίνακα 2.3.

3) Οι πράξεις implication τύπου I πρέπει να συνδυάζονται με aggregation τυπου OR ή SAM και με defuzzication τύπου center of area (COA). Αντίθετα, τα implication τύπου II πρέπει να συνδυάζονται με aggregation τύπου AND και με defuzzification τύπου MOM.

4) Τονίζουμε τελος ότι οι Υποθέσεις A,B,C και D είναι μόνο *ικανές* συνθήκες για να ισχύουν τα απλοποιημένα μοντέλα που παρουσιάσαμε στο κεφάλαιο 3, και όχι απαραίτητα *αναγκαίες*. Ενδέχεται επομένως, να υπάρχουν και άλλοι συνδυασμοί πράξεων που να καταλήγουν σε απλοποιημένους υπολογισμούς.

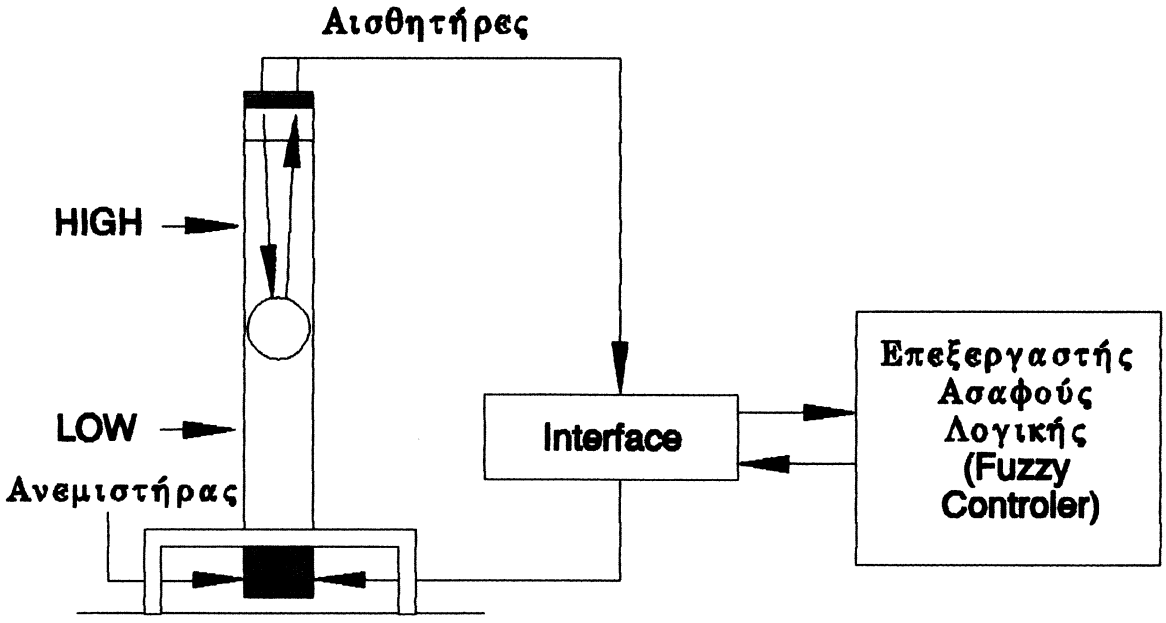
ΜΕΡΟΣ Β

ΕΠΕΞΕΡΓΑΣΤΗΣ ΑΣΑΦΟΥΣ ΛΟΓΙΚΗΣ ΚΑΙ ΕΦΑΡΜΟΓΗ ΣΕ ΕΝΑ ΠΡΑΚΤΙΚΟ ΣΥΣΤΗΜΑ ΕΛΕΓΧΟΥ

Εισαγωγή

Περιγραφή του Πειράματος

Οπως αναφέρθηκε και στην εισαγωγή, σκοπός του Μέρους Β της διπλωματικής αυτής εργασίας ήταν η ανάλυση σχεδίαση και πλήρης πρακτική κατασκευή ενός συστήματος ελέγχου που λειτουργεί με ασαφή λογική. Συγκεκριμένα, πρόκειται για έναν ασαφή, βασισμένο-σε-κανόνες ρυθμιστή, που μετακινεί μια σφαίρα (μπάλα του ping-pong) στο εσωτερικό ενός κατακόρυφου, πλαστικού και διάφανου σωλήνα ύψους 1.80m και διαμέτρου λίγο μεγαλύτερης από τη διάμετρο της σφαίρας. Η μετακίνηση γίνεται μεταξύ δύο προκαθορισμένων σημείων μεταβάλλοντας την τάση που εφαρμόζεται σε έναν ανεμιστήρα που βρίσκεται στερεωμένος στο κάτω μέρος του σωλήνα. Σκοπός του πειράματος είναι η μετακίνηση της μπάλας μεταξύ των δύο σημείων να γίνεται στο μικρότερο δυνατό χρόνο και με τις λιγότερες δυνατές ταλαντώσεις. Η διάταξη φαίνεται στο παρακάτω σχήμα όπου HIGH και LOW είναι τα δύο προκαθορισμένα σημεία.



Η πειραματική διάταξη

Οπως βλέπουμε, στο πάνω άκρο του σωλήνα είναι στερεωμένοι δύο αισθητήρες υπερήχων. Με τη βοήθεια των αισθητήρων υπολογίζεται από τα κυκλώματα του interface η θέση της σφαίρας και μέσω αυτής το σφάλμα ως προς την επιθυμητή θέση. Επίσης, μέσω παραγώγισης υπολογίζεται και η ταχύτητα της σφαίρας και στη συνέχεια οι δύο αυτές μεταβλητές τροφοδοτούνται σαν είσοδοι στον *επεξεργαστή ασαφούς λογικής (fuzzy processor)* που υλοποιήθηκε σε hardware. Ο επεξεργαστής υπολογίζει την κατάλληλη έξοδο με βάση ένα *σύνολο κανόνων (rulebase)* που μπορούμε εύκολα να μεταβάλλουμε, και η έξοδος αυτή τροφοδοτείται μέσω του interface στον ανεμιστήρα.

Η υλοποίηση του κυκλώματος ελέγχου έγινε με διακριτά στοιχεία του εμπορίου ενώ όλοι οι υπολογισμοί γίνονται εξ'ολοκλήρου με αναλογικές μεθόδους χωρίς να μεσολαβεί μετατροπή A/D ή D/A σε κανένα στάδιο της επεξεργασίας. Η μέθοδος που ακολουθήθηκε είναι παρόμοια με εκείνη που ακολουθείται στη σχεδίαση του ασαφούς μικροεπεξεργαστή που γίνεται στα πλαίσια μιας διδακτορικής διατριβής στο Εργαστήριο Ηλεκτρονικής Ε.Μ.Π.

Το πείραμα αυτό, όπως είναι προφανές, δεν επιλέχθηκε με κριτήριο τις πιθανές εφαρμογές που θα μπορούσε να έχει. Αντίθετα, είχε καθαρά χαρακτήρα επίδειξης και η σκοπιμότητά του ήταν διπλή: από τη μία να γίνει φανερή η δυνατότητα υλοποίησης μεθόδων της ασαφούς λογικής με πλήρως αναλογικά κυκλώματα, και από την άλλη να κατασκευαστεί ένα απλό σχετικά σύστημα ελέγχου έτσι ώστε όταν τελικά υλοποιηθεί ο μικροεπεξεργαστής να μπορεί να δοκιμαστεί αμέσως στην πράξη και να πάρουμε συγκριτικά αποτελέσματα.

Στη συνέχεια, παρουσιάζεται στο κεφάλαιο 4 η υλοποίηση του κυκλώματος του επεξεργαστή, στο κεφάλαιο 5 η σχεδίαση του interface και στο κεφάλαιο 6 η σχεδίαση του ασαφούς συστήματος (επιλογή των κανόνων και ρύθμιση των συναρτήσεων συμμετοχής) και τα πειραματικά αποτελέσματα.

Κεφάλαιο 4

Ο ΕΠΕΞΕΡΓΑΣΤΗΣ ΑΣΑΦΟΥΣ ΛΟΓΙΚΗΣ

4.1 Αναλογική και Ψηφιακή Υλοποίηση

Μία πρώτη προσέγγιση για την υλοποίηση κυκλωμάτων της ασαφούς λογικής σε hardware είναι τα ψηφιακά κυκλώματα, δηλ. είτε η χρησιμοποίηση ψηφιακών υπολογιστών με κατάλληλο software είτε η ανάπτυξη ειδικών ψηφιακών μικροεπεξεργαστών σε hardware. Και στις δύο περιπτώσεις μια συνάρτηση συμμετοχής προσεγγίζεται από ένα πεπερασμένο αριθμό κβαντοποιημένων σημείων, υπάρχει δηλ. *διακριτοποίηση και στους δύο άξονες* (x και μ(x)) (Σχήμα 4.1). Οι διάφορες πράξεις μεταξύ των διαφόρων συναρτήσεων συμμετοχής εκτελούνται στη συνέχεια από συνδυαστικά ψηφιακά κυκλώματα. Τέτοιες υλοποιήσεις υπάρχουν στα [27], [32], [34], [54], [56], [59].

Ενας πολύ πιο ευέλικτος τρόπος υλοποίησης είναι οι τιμές των μ(x) να παριστάνονται από *αναλογικές* τάσεις (οπότε έχουμε κυκλώματα V-mode) ή αναλογικά ρεύματα (οπότε έχουμε κυκλώματα I-mode) πάνω σε ένα πεπερασμένο αριθμό από pins που αντιστοιχούν σε ένα κβαντοποιημένο πεδίο ορισμού. 'Ετσι υπάρχει διακριτοποίηση μόνο στον άξονα των x (Σχήμα 4.2). Με τον τρόπο αυτό δεν υπάρχουν σφάλματα κβαντοποίησης στον άξονα των μ(x) και επιπλέον η υλοποίηση των πράξεων (π.χ. minimum, maximum) μπορεί να γίνει με αναλογικά κυκλώματα, που όπως αναφέρθηκε και στην εισαγωγή, είναι αποτελεσματικότερα όσον αφορά την ταχύτητα των υπολογισμών, το μέγεθος και την πολυπλοκότητα των κυκλωμάτων και το κόστος σχεδίασης και κατασκευής. Τέτοια συστήματα έχουν υλοποιηθεί στα [57], [58], [64], [66]. Αναφέρουμε επίσης ότι η σχεδίαση ολοκληρωμένων κυκλωμάτων σε I-mode είναι πολύ ευκολότερη απ'ότι σε V-mode, γιατί στην πρώτη περίπτωση τα κυκλώματα είναι πιο αναίσθητα στις μεταβολές της τροφοδοσίας, η υλοποίηση των πράξεων AND και OR (αλλά και άλλων πράξεων) είναι απλούστερη, και η άθροιση υλοποιείται με απλή σύνδεση (wiring) δύο ή περισσότερων κλάδων.

Ο τελευταίος τρόπος υλοποίησης κυκλωμάτων ασαφούς λογικής που παρουσιάζουμε είναι με κυκλώματα που έχουν *αναλογική είσοδο* και *αναλογική έξοδο*. Μ'αυτό τον τρόπο δεν υπάρχει καμία διακριτοποίηση (Σχήμα 4.3) και κανένα σφάλμα κβαντοποίησης. Επιπλέον δε χρειάζεται μεγάλος αριθμός από pins όπως στην προηγούμενη περίπτωση με αποτέλεσμα απλούστερα και μικρότερα

κυκλώματα. Τέτοιου είδους κυκλώματα έχουν προταθεί στα [60], [61], [62], [63] αλλά *δεν έχει υλοποιηθεί ολοκληρωμένο σύστημα fuzzy controller με βάση την τεχνική αυτή, όπως αυτό που κατασκευάστηκε στα πλαίσια αυτής της διπλωματικής εργασίας*. Βέβαια το κύκλωμα που κατασκευάσαμε ήταν πειραματικό και υλοποιήθηκε με διακριτά στοιχεία (σε V-mode), αλλά η υλοποίηση ενός παρόμοιου κυκλώματος σε I-mode και σε ολοκληρωμένη μορφή είναι εξίσου δυνατή. Αυτό εξάλλου είναι μέσα στο πρόγραμμα του Εργαστηρίου Ηλεκτρονικής Ε.Μ.Π.

4.2 Αρχιτεκτονική του κυκλώματος.

Η αρχιτεκτονική του ασαφούς επεξεργαστή φαίνεται στο Σχήμα 4.4. Πρόκειται για ένα σύστημα με δύο εισόδους, μία έξοδο και 16 κανόνες, που έχουν από δύο υποθέσεις συνδεδεμένες με AND. Το AND υλοποιείται σαν *minimum*, το implication σαν γινόμενο (*product*), το aggregation σαν άθροισμα (*sum*) και το defuzzification με τη μέθοδο *weighted average*. Οι συναρτήσεις συμμετοχής είναι *τριγωνικές, τραπεζοειδείς* ή *τύπου-Π* και είναι πλήρως προγραμματιζόμενες ως προς τη μέση τιμή, την κλίση και το εύρος της μικρής βάσης του τραπεζίου. Το σύστημα αποτελείται από 5 βασικά τμήματα (**block**).

Το πρώτο απ'αυτά (**block 1**) περιλαμβάνει τα *κυκλώματα ελέγχου και χρονισμού*. Επειδή το κύκλωμα υλοποιήθηκε με διακριτά στοιχεία, το μέγεθός του είναι αρκετά μεγάλο και έτσι δεν είναι δυνατή η παράλληλη επεξεργασία των κανόνων. Ετσι υλοποιήθηκε ένα βασικό κύκλωμα (block 2) που χρησιμοποιείται απ'όλους τους κανόνες *σειριακά* με το κατάλληλο άνοιγμα και κλείσιμο διακοπτών. Για το σκοπό αυτό το block 1 παράγει τα σήματα $C_1,\ldots,C_{16}$ το καθένα απ'τα οποία ενεργοποιεί ένα κανόνα. Επίσης το block 1 παράγει τα σήματα SAM, RES και CK2 από τα οποία το πρώτο χρησιμεύει για τη δειγματοληψία των εισόδων και της εξόδου σε κατάλληλες χρονικές στιγμές και τα άλλα χρησιμοποιούνται για το defuzzification (block 3).

Το **block 2** είναι εκείνο που υλοποιεί τις συναρτήσεις συμμετοχής. Το κύκλωμα έχει δύο εισόδους (το σφάλμα θέσης και την ταχύτητα της σφαίρας) οπότε χρειάζονται δύο ίδια κυκλώματα που να λειτουργούν παράλληλα και να παράγουν δύο διαφορετικές συναρτήσεις συμμετοχής. Τα κυκλώματα αυτά είναι τα block 2a, 2b. Επειδή είναι ίδια όμως στο εξής θα περιγράψουμε μόνο το ένα, που θα ονομάσουμε block 2. Ομοια από τα block 3a και 3b, που περιέχουν τους κανόνες, θα περιγράψουμε μόνο το ένα που θα ονομάσουμε block 3. Το block 2 λοιπόν, περιέχει τα *κυκλώματα εισόδου* στα οποία γίνεται και η δειγματοληψία της εισόδου **IN** και παράγεται το σήμα **IN1**, το *κύκλωμα υλοποίησης τριγωνικών και τραπεζοειδών συναρτήσεων* που έχει είσοδο το IN1 και έξοδο το **OUT1**, και τέλος το κύκλωμα υλοποίησης *συναρτήσεων τύπου Π* που έχει είσοδο το OUT1 και έξοδο το **OUT2**. Δηλ. οι συναρτήσεις τύπου Π δημιουργούνται από τις τραπεζοειδείς.

Το **block 3** περιέχει τους κανόνες, δηλ. τις πληροφορίες σχετικά με τη *μέση τιμή*, την *κλίση* και το εύρος της μικρής βάσης του τραπεζίου (που στο εξής θα ονομάζουμε απλά *εύρος*) των συναρτήσεων συμμετοχής. Τα 3 αυτά στοιχεία είναι ουσιαστικά αποθηκευμένα σε 3 μεταβλητές αντιστάσεις (τρίμμερ) και μεταβιβάζονται στο block 2 μέσω των κόμβων $\mathbf{R} = (R_1, R_2, R_3)$. Το block 3 έχει μια τριάδα από τρίμμερ για κάθε κανόνα, δηλ. 16 x 3 = 48 τρίμμερ συνολικά. Το ποιά τριάδα "ενεργοποιείται" κάθε χρονική στιγμή για να δώσει τα κατάλληλα σήματα R_1, R_2, R_3 εξαρτάται από το ποιό από τα σήματα $C_1, \ldots, C_{16}$ είναι 1, (όλα τα άλλα είναι μηδέν). Η αλλαγή από κανόνα σε κανόνα γίνεται με αναλογικούς διακόπτες που κάθε στιγμή "συνδέουν" μία μόνο τριάδα αντιστάσεων και "αποσυνδέουν" όλες τις υπόλοιπες.

Το **block 4** υλοποιεί το *defuzzification*. Εχει εισόδους τα OUT2a και OUT2b από τα block 2a και 2b αντίστοιχα, και έξοδο την τελική έξοδο του συστήματος, OUT. Για το defuzzification χρησιμοποιείται η σχέση (2.34) :

$$OUT = \frac{\mu_1 c_1 + \cdots + \mu_N c_N}{\mu_1 + \cdots + \mu_N} \qquad (4.1)$$

όπου N είναι το πλήθος των κανόνων (μέγιστο 16), C_i $(1 \le i \le N)$ η προτεινόμενη έξοδος και μ_i ο βαθμός εκπλήρωσης (DOF) του κανόνα i. Το σύστημα έχει κανόνες της μορφής

R_i : IF INa IS Ma_i AND INb IS Mb_i
THEN OUT = C

και επομένως

$$\mu_i = \mu_{Mai}(INa) \wedge \mu_{Mbi}(INb)$$

όπου **INa, INb** οι δύο είσοδοι του συστήματος (σαφείς αριθμοί), και Ma_i και Mb_i οι αντίστοιχες συναρτήσεις συμμετοχής του κανόνα i. Ομως οι συναρτήσεις συμμετοχής υλοποιούνται στο block 2, δηλ.

$$OUT2a = \mu_{Mai}(INa), \quad OUT2b = \mu_{Mbi}(INb)$$

οπότε

$$\mu_i = OUT2a \wedge OUT2b \qquad (4.2)$$

Ουσιαστικά λοιπόν με βάση τη σχέση (4.1) υπολογίζονται στο block 4 όχι μόνο το defuzzification, αλλά και το *AND*, το *implication* και το *aggregation*.

Τέλος το **block 5** περιέχει πληροφορίες για τους κανόνες όπως και το block 3. Συγκεκριμένα στο block 5 υπάρχουν για κάθε κανόνα 2 τρίμμερ (συνολικά 2 x 16 = 32 τρίμμερ). Στο πρώτο είναι αποθηκευμένο το *βάρος* (*weight*) του κανόνα,

w_i, και στο δεύτερο η *προτεινόμενη έξοδος*, c_i. Το βάρος έχει την έννοια ότι μπορούμε να κάνουμε ορισμένους κανόνες να ισχύουν λιγότερο από άλλους. Κάθε βαθμός εκπλήρωσης μ_i δηλαδή, πολλαπλασιάζεται με το βάρος του κανόνα w_i, και η (4.1) γίνεται :

$$OUT = \frac{\mu_1 w_1 c_1 + \cdots + \mu_N w_N c_N}{\mu_1 w_1 + \cdots + \mu_N w_N} \qquad (4.3)$$

δηλ. παίρνει μια μορφή που μοιάζει περισσότερο με την (2.35). Τα δεδομένα w_i και c_i μεταβιβάζονται στο block 4 μέσω των κόμβων $X = (X_1, X_2, X_3, X_4)$, όπου τα X_1, X_2, αντιστοιχούν στο w_i και τα X_3, X_4 στο c_i.

Στη συνέχεια ακολουθεί αναλυτική περιγραφή του κάθε block χωριστά.

4.3 Κυκλώματα ελέγχου και χρονισμού.

Το block 1 του σχήματος 4.4 αποτελείται από ένα αναλογικό και ένα ψηφιακό μέρος. Το πρώτο αναλύεται στην παράγραφο 4.3.1 και το δεύτερο στην 4.3.2.

4.3.1 Αναλογικό Μέρος

Το αναλογικό μέρος φαίνεται στο Σχ. 4.5 και σκοπός του είναι η παραγωγή ενός πριονωτού σήματος (sawtooth) που ονομάζουμε **SAW** και που η σκοπιμότητά του εξηγείται παρακάτω. Αυτό επιτυγχάνεται με ένα ολοκληρωτή που έχει σταθερή είσοδο −1V και παράγει μια τάση γραμμικά μεταβαλλόμενη ως προς το χρόνο. Η τάση αυτή οδηγείται σε ένα συγκριτή έτσι ώστε μόλις φτάσει το +1V η έξοδος του συγκριτή να γίνει −5V (στο σημείο αυτό τονίζουμε ότι οι τροφοδοσίες όλων των τελεστικών ενισχυτών, αλλά και των ψηφιακών κυκλωμάτων είναι ±5V, οπότε το λογικό 1 αντιστοιχεί στα +5V και το λογικό 0 στα −5V), και επομένως η έξοδος του Schmitt trigger (που η λειτουργία του θα εξηγηθεί παρακάτω) να γίνει +5V (λογικό 1). Τη στιγμή εκείνη ο διακόπτης S κλείνει και ο ολοκληρωτής μηδενίζεται.

Ετσι παίρνουμε μια πριονωτή τάση που μεταβάλλεται μεταξύ 0 και 1V, ενώ η περίοδός της ρυθμίζεται με το τρίμμερ των 100K στα 50ms. Η πριονωτή αυτή τάση φαίνεται στο Σχ. 4.6. Το σχήμα αυτό προέρχεται από μέτρηση του κυκλώματος με τον παλμογράφο που χρησιμοποιήσαμε σε όλη τη διάρκεια των πειραμάτων (τύπου HP 54510A) και εκτυπώθηκε με τον εκτυπωτή HP Thinkjet που συνεργάζεται με τον παλμογράφο. Ολα τα σχήματα που προέρχονται από πειραματικές μετρήσεις των κυκλωμάτων και παρουσιάζονται στη συνέχεια της εργασίας είναι αυτής της μορφής και γι'αυτό αξίζει στο σημείο αυτό να κάνουμε μια παρατήρηση.

Ο παλμογράφος διαθέτει τέσσερα κανάλια, τα **1**, **2**, **3** και **4**, και έξι μνήμες, τις m1, m2, m3, m4, p1 και p2. Γι'αυτό στη λεζάντα του κάθε σχήματος θα αναφερόμαστε σε κάθε κυματομορφή χρησιμοποιώντας τα παραπάνω σύμβολα. Τα σύμβολα αυτά αναγράφονται επίσης στο δεξιό μέρος κάθε σχήματος, δίπλα ακριβώς στην κυματομορφή που αναπαριστούν. Ετσι γίνεται εύκολα η αναγνώριση της κάθε κυματομορφής. Επίσης οι μετρήσεις που αναγράφονται στο κάτω μέρος κάθε σχήματος αναφέρουν το αντίστοιχο σύμβολο ώστε να φαίνεται σε ποιά κυματομορφή αντιστοιχούν.

Επιστρέφοντας τώρα στο κύκλωμά μας, εξηγούμε τη λειτουργία του Schmitt trigger. Το Schmitt trigger (που είναι το ολοκληρωμένο κύκλωμα MC14106 της Motorola) λειτουργεί ουσιαστικά σαν αναστροφέας, με τη διαφορά ότι η χαρακτηριστική μεταφοράς DC παρουσιάζει μια *υστέρηση* που φαίνεται στο Σχ. 4.7. Αν δηλαδή η τάση εισόδου V_{in} αυξάνεται, η έξοδος V_{out} αλλάζει κατάσταση τη στιγμή που $V_{in} = V_{T+}$. Αν αντίθετα η V_{in} μειώνεται, η V_{out} αλλάζει κατάσταση τη στιγμή που $V_{in} = V_{T-}$, με $V_{T-} < V_{T+}$. Στο κύκλωμά μας, με τροφοδοσία ±5V, είχαμε $V_{T+} \approx 900mv$ και $V_{T-} \approx -750mV$.

Αναφερόμενοι λοιπόν και πάλι στο Σχ. 4.5, τη στιγμή που η έξοδος του συγκριτή πέφτει κάτω από V_{T-}, το Schmitt trigger δίνει λογικό 1 και ο διακόπτης (που είναι αναλογικός διακόπτης MOS και περιέχεται στο ολοκληρωμένο CD4066 της National Semiconductors) κλείνει. Ο ολοκληρωτής μηδενίζεται σχεδόν ακαριαία και η έξοδος του συγκριτή επανέρχεται στα +5V. Ομως λόγω του βαθυπερατού φίλτρου RC, η είσοδος του συγκριτή *καθυστερεί* να ξεπεράσει το V_{T+} και στην έξοδο του Schmitt trigger σχηματίζεται ένας θετικός παλμός διάρκειας 50μs. Ετσι δίνεται στον ολοκληρωτή αρκετός χρόνος ώστε να μηδενιστεί εντελώς. Χωρίς την καθυστέρηση που εισάγει το Schmitt trigger, η λειτουργία του κυκλώματος δε θα ήταν δυνατή. Ολα τα παραπάνω φαινόμενα απεικονίζονται στο Σχ. 4.8.

Τέλος αναφέρουμε ότι σε όλα τα κυκλώματα χρησιμοποιήθηκαν τελεστικοί ενισχυτές TL084 λόγω των πολύ καλών χαρακτηριστικών τους, (διαφορικό κέρδος 200 V/mV, αντίσταση εισόδου 10^{12} Ω, εύρος ζώνης 3 MHz, slew rate 13 V/μs, τάση απόκλισης εισόδου 3mV (max), ρεύμα πόλωσης εισόδου 30 pA, περιοχή κοινού σήματος εισόδου ±15V).

4.3.2 Ψηφιακό μέρος

Το ψηφιακό μέρος του block 1 φαίνεται στο Σχ. 4.9, ενώ το Σχ. 4.10 φαίνονται δύο πρόσθετα βοηθητικά κυκλώματα. Για την επεξήγησή του αναφέρουμε αρχικά ότι ο fuzzy controller που κατασκευάσαμε λειτουργεί με δύο τρόπους. Ο πρώτος (***mode 1***) χρησιμεύει για τη ρύθμιση των συναρτήσεων συμμετοχής και ο δεύτερος (***mode 2***) αντιστοιχεί στη φυσιολογική λειτουργία του κυκλώματος σαν

controller. Στο mode 1 η αλλαγή των κανόνων γίνεται χειροκίνητα, με την πίεση ενός push-button, και σαν είσοδος τροφοδοτείται στο κύκλωμα η πριονωτή κυματομορφή SAW. Ετσι η κυματομορφή εξόδου του block 2 έχει ακριβώς τη μορφή των συναρτήσεων συμμετοχής και με απλή παρατήρησή της στον παλμογράφο μπορούμε με κατάλληλες ρυθμίσεις των trimmer να πάρουμε τις συναρτήσεις που θέλουμε.

Αντίθετα, στο mode 2 η αλλαγή των κανόνων γίνεται σύμφωνα με το εσωτερικό ρολόϊ του συστήματος, **CK1** (με περίοδο 64μs). Κάθε N παλμούς του ρολογιού (όπου **N** είναι το πλήθος των κανόνων) δημιουργείται ένας παλμός **R**, διάρκειας 300ns, που σηματοδοτεί την έναρξη ενός νέου κύκλου υπολογισμών. Ταυτόχρονα ξεκινά ένας άλλος παλμός, διάρκειας 12μs, που ονομάζουμε **SAM**, κατά τη διάρκεια του οποίου γίνεται δειγματοληψία των εισόδων (και της εξόδου). Σαν είσοδοι έρχονται στο σύστημα το σφάλμα θέσης και η ταχύτητα της σφαίρας, μέσω του interface. Ενας δυαδικός απαριθμητής (ολοκληρωμένο MC14040 της Motorola) ξεκινάει την ίδια στιγμή να μετρά από την τιμή 0, και αυξάνει κατά 1 σε κάθε παλμό CK1. Η έξοδος του απαριθμητή συμβολίζεται $\mathbf{Q = (Q_4\ Q_3\ Q_2\ Q_1)_2}$, όπου το Q_1 είναι το λιγότερο σημαντικό ψηφίο (LSB). Ο απαριθμητής είναι 12-μπιτος αλλά εμείς χρησιμοποιούμε μόνο τα 4 λιγότερο σημαντικά bits.

Στη συνέχεια ο αριθμός Q οδηγείται στην είσοδο ενός τετράμπιτου δυαδικού συγκριτή (ολοκληρωμένο MC14585 της Motorola) όπου συγκρίνεται με το δυαδικό αριθμό $\mathbf{S = (S_4\ S_3\ S_2\ S_1)_2}$ που παριστάνει τον αριθμό N-1. Αν π.χ. θέλουμε ο αριθμός των κανόνων να είναι N=8, τότε πρέπει $S = (0\ 1\ 1\ 1)_2$. Οι τιμές S_1, S_2, S_3, S_4 καθορίζονται με τέσσερις χειροκίνητους διακόπτες (DIP Switches). Ετσι δίνεται δυνατότητα αυξομείωσης του αριθμού των κανόνων ανάλογα με την εφαρμογή, χωρίς καμία αλλαγή στο hardware.

Οταν ο συγκριτής δείξει ότι Q > S (δηλ. ο αύξων αριθμός του κανόνα έχει ξεπεράσει την τιμή N-1 και ισούται με N), παράγεται ο παλμός R που οδηγείται στην είσοδο RESET του μετρητή. Ετσι μόλις ο μετρητής φτάσει την τιμή N, αμέσως μηδενίζεται. Ο συγκριτής τότε επανέρχεται στην κατάσταση Q < S, ο παλμός R σταματά και το όλο φαινόμενο διαρκεί 300ns (τόση είναι και η διάρκεια του R). Στο σημείο αυτό ο κύκλος υπολογισμών έχει τελειώσει και η διαδικασία επαναλαμβάνεται. Οσα περιγράφτηκαν παραπάνω φαίνονται στο σχήμα 4.11α όπου υποθέσαμε ότι N=3. Το σήμα SAM δημιουργείται από το R με τη χρήση του γνωστού κυκλώματος χρονισμού LM555 σε συνδεσμολογία μονοσταθούς πολυδονητή. Ετσι από έναν παλμό διάρκειας 300ns δημιουργείται ένας άλλος διάρκειας 12μs. Ο χρόνος αυτός είναι απαραίτητος για να γίνει δειγματοληψία.

Στο Σχήμα 4.11β βλέπουμε δύο νέα σήματα, το ρολόϊ **CK2** και το σήμα μηδενισμού **RES**. Οταν CK2=0, δηλ. αμέσως μετά την αλλαγή κανόνα, τα κυκλώματα υπολογισμού των συναρτήσεων συμμετοχής (block 2) αφήνονται ελεύθερα για 30μs και επιτρέπεται να μεταβάλλονται αυθαίρετα μέχρι να σταθεροποιηθούν σε κάποια τιμή OUT1 (ή OUT2). Οταν όμως περάσουν 30μs από την αλλαγή του κανόνα, έχουμε CK2=1 για 34μs και η σταθερή πλέον τιμή OUT1 (ή OUT2) χρησιμοποιείται από τα κυκλώματα του defuzzification (block 4). Αναφερόμενοι στο Σχ. 4.9, παρατηρούμε

ότι στην πράξη πρώτα παράγεται το CK2 από ένα χρονιστή 555 σε συνδεσμολογία ασταθούς πολυδονητή, και απ'αυτό δημιουργείται το CK1 με τη βοήθεια ενός υψιπερατού φίλτρου RC και δύο αναστροφέων Schmitt trigger. Επίσης ο δεύτερος αυτός χρονιστής 555 συνδυάζεται με τον προηγούμενο σε ένα μόνο διπλό χρονιστή LM 556.

Στο Σχ. 4.10(α) φαίνεται πώς δημιουργείται το σήμα RES από το SAM και πάλι με τη βοήθεια ενός Schmitt trigger σαν μονοσταθή πολυδονητή. Το σήμα RES ξεκινάει τη στιγμή που το SAM σταματά, και έχει διάρκεια 10μs. Το σήμα RES είναι απαραίτητο για το μηδενισμό δύο ολοκληρωτών που περιλαμβάνονται στο block 4. Συγκεκριμένα, μόλις τελειώσει ένας κύκλος υπολογισμών γίνεται δειγματοληψία στην έξοδο σύμφωνα με το σήμα SAM και αμέσως μετά τα εσωτερικά κυκλώματα μηδενίζονται για να ξαναρχίσουν τους υπολογισμούς. Είναι σημαντικό ότι ο παλμός RES τελειώνει *πριν* έλθει ο επόμενος παλμός CK2, γιατί από εκείνη τη στιγμή και μετά αρχίζουν ουσιαστικά οι υπολογισμοί.

Στο Σχ. 4.10(β) φαίνεται το push-button και το *κύκλωμα καθαρισμού* του, στο οποίο για μια ακόμα φορά χρησιμοποιείται ένα Schmitt trigger. Αυτό είναι απαραίτητο για να γίνει ο παλμός του διακόπτη πιο "τετράγωνος", (Σχ. 4.12) χωρίς όμως τη δημιουργία σπινθηρισμών, ώστε να είναι κατάλληλος για να λειτουργήσει σαν "ρολόϊ" για τον απαριθμητή. Το ποιός παλμός θα χρησιμοποιηθεί σαν ρολόϊ, το σήμα **BUT** του pushbutton ή το CK1, εξαρτάται από το σήμα ελέγχου S5. Το S5 καθορίζεται από ένα χειροκίνητο διακόπτη (DIP Switch) και ελέγχει ένα διπλό (μεταγωγικό) αναλογικό διακόπτη MOS (ολοκληρωμένο CD4053 της National). Ετσι, όταν S5 = 1, το σύστημα λειτουργεί σε mode 2 και το CK1 επιλέγεται σαν ρολόϊ του συστήματος. Αντίθετα, για S5 = 0, έχουμε mode 1 και το σήμα BUT λειτουργεί πλέον σαν ρολόϊ.

Τέλος, τα σήματα Q_1, Q_2, Q_3, Q_4 τροφοδοτούνται σε ένα αποκωδικοποιητή 4-σε-16 (ολοκληρωμένο MC14514 της Motorola) οπότε παράγονται τα σήματα ελέγχου $\mathbf{C_1}$, $\mathbf{C_2}$, ..., $\mathbf{C_{16}}$. Ετσι, μόλις αρχίζει ένας κύκλος υπολογισμών έχουμε $C_1 = 1$ (και όλα τα άλλα μηδέν). Με τον επόμενο παλμό ρολογιού (BUT ή CK1) έχουμε $C_2 = 1$ κ.ο.κ. Το Σχ. 4.13 δείχνει αυτήν την εναλλαγή για N = 3.

4.4 Υπολογισμός συναρτήσεων συμμετοχής

Στο κεφάλαιο αυτό θα παρουσιάσουμε τα κυκλώματα που περιλαμβάνονται στα Block 2 και Block 3 του συστήματος.

4.4.1 Κύκλωμα εισόδου

Το κύκλωμα εισόδου του Block 2 φαίνεται στο Σχ. 4.14. Καταρχήν βλέπουμε ότι με το σήμα ελέγχου **S6** (που ρυθμίζεται επίσης με ένα DIP switch) ελέγχουμε αν στα κυκλώματα υπολογισμού των συναρτήσεων συμμετοχής θα περάσει η είσοδος IN, που προέρχεται από το interface, ή η πριονωτή κυματομορφή SAW. Αν δηλαδή S6 = 0, το κύκλωμα λειτουργεί σε mode 1 και επιλέγεται η είσοδος SAW, ενώ αν S6= 1, έχουμε mode 2 και επιλέγεται η είσοδος IN.

Στη συνέχεια το σήμα εισόδου οδηγείται σε ένα κύκλωμα *δειγματοληψίας και συγκράτησης* (*sample and hold* S/H) (ολοκληρωμένο LF398 της National) που ελέγχεται απο τον παλμό SAM, και έπειτα περνάει από ενα βαθυπερατό φίλτρο RC που εξαλείφει τα "spikes" που δημιουργούνται κατά τη στιγμή της δειγματοληψίας. Τέλος με το σήμα ελέγχου **S5** (DIP Switch) ελέγχουμε αν το σήμα εισόδου θα περάσει από δειγματοληψία ή όχι. Αν δηλ. S5=0, έχουμε mode 1, το ρολόι του συστήματος προέρχεται απ'το pushbutton και επιλέγεται το σήμα *πρίν* από το S/H. Αντίθετα, αν S5=1, έχουμε mode 2, το ρολόι του συστήματος είναι το CK1 και επιλέγεται το σήμα *μετά* το S/H.

Η δειγματοληψία στο mode 2 είναι απαραίτητη γιατί οι υπολογισμοί γίνονται *σειριακά* οπότε οι είσοδοι του συστήματος πρέπει να είναι σταθερές καθ'όλη την διάρκεια ενός κύκλου υπολογισμών. Αντίθετα στο mode 1 (που η είσοδος είναι το SAW) πρέπει η μεταβολή της εισόδου να είναι συνεχής ώστε να παρατηρούμε στον παλμογράφο συνεχείς καμπύλες (τις συναρτήσεις συμμετοχής). Επομένως στην περίπτωση αυτή δε γίνεται δειγματοληψία.

4.4.2 Πύλες AND, OR και NOT.

Το κύκλωμα παραγωγής των συναρτήσεων συμμετοχής που θα παρουσιάσουμε στην επόμενη παράγραφο λειτουργεί σε V-mode και περιλαμβάνει "ασαφείς" πύλες AND και OR, δηλ. κυκλώματα που υπολογίζουν το *minimum* και το *maximum* δύο τάσεων αντίστοιχα. Οι πύλες αυτές φαίνονται στο Σχ. 4.15 μαζί με τα σύμβολά τους, που θα χρησιμοποιήσουμε παρακάτω. Η λειτουργία τους είναι απλή και μοιάζει με τη λειτουργία των ψηφιακών πυλών DTL. Η διαφορά είναι ότι οι δίοδοι βρίσκονται στο βρόχο ανάδρασης των OP AMP οπότε η τάση σημείου αγωγής V_γ (~0.6V), αλλά και η ορθή αντίσταση R_f των διόδων διαιρούνται με το κέρδος ανοιχτού βρόχου A_v των ενισχυτών. Ετσι η V_γ και η R_f ουσιαστικά εξαλείφονται και οι δίοδοι παρουσιάζουν σχεδόν ιδανική συμπεριφορά.

Συγκεκριμένα, όταν η είσοδος V_1 (στο Σχ. 4.15 (a)) γίνει αρνητικότερη από τη V_2 κατά τουλάχιστον V_g/A_v, η τάση στα άκρα της D1 υπερβαίνει τη V_γ. Έτσι η D1 άγει και ο TE1 λειτουργεί ως ακόλουθος τάσης, οπότε $V_{out} = V_1$. Αντίθετα η D2 αποκόπτεται και ο TE2 βρίσκεται στον κόρο. Τα αντίθετα συμβαίνουν όταν $V_1 > V_2$. Η λειτουργία της πύλης OR είναι ανάλογη.

Στο Σχ. 4.16 βλέπουμε τις χαρακτηριστικές μεταφοράς DC των πυλών AND και OR. Είναι φανερό ότι πράγματι πλησιάζουν την ιδανική συμπεριφορά. Αντίθετα, όπως φαίνεται στο Σχ. 4.17 οι πύλες αυτές μειονεκτούν ως προς την ταχύτητα. Η απόκρισή τους σε ένα γρήγορο παλμό είναι σχετικά αργή, αφού χρειάζονται 6.5μs περίπου για να σταθεροποιηθούν. Η καθυστέρηση αυτή οφείλεται στο γεγονός ότι κατά τη στιγμή της αλλαγής οι OP AMP εισέρχονται και εξέρχονται από τον κόρο. Η καθυστέρηση αυτή όμως δε δημιουργεί κανένα πρόβλημα στο κύκλωμά μας, όπως θα φανεί παρακάτω.

Τέλος η πύλη NOT υλοποιείται με ένα μόνο OP AMP και φαίνεται στο Σχ. 4.18. Πρόκειται για ένα αναστρέφοντα αθροιστή που δίνει έξοδο $V_{out} = 1V - V_{in}$. Αν οι "τιμές αλήθειας" απεικονίζονται στο διάστημα [0,1V] δηλ. $0 \leq V_{in} \leq 1V$, τότε πράγματι το κύκλωμα αυτό υλοποιεί την πράξη NOT.

4.4.3 Παραγωγή τριγωνικών και τραπεζοειδών συναρτήσεων

Το κύκλωμα που υλοποιεί τις συναρτήσεις αυτές λειτουργεί σε V-mode, δηλ. δέχεται είσοδο μία αναλογική τάση μεταξύ 0 και 1V (που είναι το κανονικοποιημένο υπερσύνολο αναφοράς δηλ. X = [0,1V]) και δίνει έξοδο επίσης μία αναλογική τάση μεταξύ 0 και 1V (που είναι το διάστημα των "τιμών αλήθειας"). Το κύκλωμα βασίζεται κυρίως σε τελεστικούς ενισχυτές και φαίνεται στο Σχ. 4.19, ενώ η λειτουργία του φαίνεται στο Σχ. 4.20.

Στην τάση εισόδου IN1 προστίθεται αρχικά η V_m, δηλαδή *αφαιρείται* μία τάση μεταξύ 0 και 1V. Η τάση που προκύπτει οδηγείται σε ένα μη αναστρέφοντα ενισχυτή με κέρδος ρυθμιζόμενο μεταξύ 82 και 0.82 περίπου. Έτσι παίρνουμε την τάση (1) του Σχ. 4.20 που αντιστοιχεί στον κόμβο (1) του Σχ. 4.19. Γίνεται φανερό ότι η τάση V_m καθορίζει *το σημείο μηδενισμού* της καμπύλης (1), ενώ η αντίσταση R_s την *κλίση* της. Στη συνέχεια με ένα αναστροφέα προσήμου δημιουργείται η αντίθετη τάση (2), μια πύλη maximum (OR) μας δίνει τη μέγιστη των δύο τάσεων και επομένως την *απόλυτη τιμή* της (1), και τέλος με ένα αναστρέφοντα αθροιστή παίρνουμε την τριγωνική συνάρτηση (3), της οποίας το ύψος ρυθμίζεται από την R_w μεταξύ +1V και +5V (για να γίνει καλή ρύθμιση χρησιμοποιείται και το τρίμμερ των 10 K). Τέλος με μία πύλη maximum (OR) και μία minimum (AND), η (3) "αποκόπτεται" έτσι ώστε $0 \leq OUT1 \leq 1V$.

Με την αποκοπή αυτή δημιουργείται η επιθυμητή *τραπεζοειδής* συνάρτηση. Βλέπουμε τελικά ότι η αντίσταση R_m ρυθμίζει *τη μέση τιμή* της συνάρτησης, η R_s την *κλίση* της και η R_w το εύρος, το οποίο μεταβάλλεται από μηδέν (οπότε έχουμε

την ειδική περίπτωση της *τριγωνικής* συνάρτησης) μέχρι κάποια μέγιστη τιμή που εξαρτάται από την κλίση. Αυτή η μέθοδος ρύθμισης είναι πολύ ευέλικτη, όπως αποδείχτηκε και στην πράξη. Στα σχήματα 4.21, 4.22 και 4.23 φαίνεται η τελική συνάρτηση με μεταβαλλόμενη μέση τιμή, κλίση και εύρος αντίστοιχα. Στα σχήματα φαίνονται και οι "καμπύλες" συναρτήσεις τύπου Π, που θα εξηγηθούν παρακάτω. Η είσοδος του κυκλώματος είναι η προηγούμενη κυματομορφή SAW (που δε φαίνεται στα σχήματα), η οποία μεταβάλλεται γραμμικά ως προς το χρόνο μεταξύ 0 και 1V (δηλ. καλύπτει ολόκληρο το υπερσύνολο αναφοράς) και με περίοδο 50ms (δηλ. τα παραπάνω σχήματα αντιστοιχούν σε μία μόνο περίοδο). Βλέπουμε ότι με τη βοήθεια της κυματομορφής αυτής η απεικόνιση των διαφόρων συναρτήσεων στο παλμογράφο γίνεται πολύ εύκολα.

4.4.4 Σύνολο Κανόνων (Rulebase)

Για τη δημιουργία πολλών διαφορετικών συναρτήσεων συμμετοχής χρησιμοποιείται *ένα* μόνο κύκλωμα σαν αυτό του Σχ. 4.19. Αυτό που αλλάζει κάθε φορά είναι η "τριάδα " των αντιστάσεων (R_m, R_s, R_w). Η αλλαγή των αντιστάσεων γίνεται από αναλογικούς διακόπτες (CD4066) που ελέγχονται από τα σήματα χρονισμού $C_1,\ldots,C_N$. Η συνδεσμολογία αυτή φαίνεται στο Σχ. 4.24, που δείχνει ουσιαστικά τα περιεχόμενα του block 3 (rulebase). Εκεί υπάρχουν "αποθηκευμένες" όλες οι απαραίτητες "πληροφορίες" για τη δημιουργία μέχρι 16 ανεξαρτήτων συναρτήσεων συμμετοχής. Επίσης υπάρχει και η δυνατότητα (δεν φαίνεται στο Σχ. 4.24) να μεταβάλλεται η κλίση, ή και το εύρος, όλων των συναρτήσεων *ταυτόχρονα*, δηλ. τα στοιχεία αυτά να ελέγχονται από δύο μόνο μεταβλητές ανιστάσεις που είναι *κοινές* για όλους τους κανόνες. Έτσι γίνεται πολύ ευκολότερη η διαδικασία ρύθμισης αλλά υπάρχει ο περιορισμός ότι όλες οι συναρτήσεις έχουν την ίδια κλίση και το ίδιο εύρος (και μόνο η μέση τιμή διαφέρει).

Παρατηρούμε στο σημείο αυτό ένα μειονέκτημα της μεθόδου που χρησιμοποιήσαμε. Σε κάθε κανόνα αντιστοιχούν δύο συναρτήσεις συμμετοχής, κάθε μία απ'τις οποίες χρειάζεται 3 τρίμμερ. Αν προσθέσουμε τα 2 τρίμμερ που είναι απαραίτητα για το defuzzification (που θα αναλυθεί παρακάτω) απαιτούνται 8 τρίμμερ συνολικά για καθε κανόνα και 128 (16 x 8) για όλο το σύστημα. Έτσι το κύκλωμα μεγαλώνει υπερβολικά σε μέγεθος.

Εναλλακτικά, όλα τα δεδομένα θα μπορούσαν να αποθηκευτούν σε μία ψηφιακή μνήμη RAM και να μεταβιβάζονται στο υπόλοιπο κύκλωμα μέσω κυκλωμάτων D/A. Τότε όμως δεν θα υπήρχε ευελιξία στη ρύθμιση των κανόνων, και επειδή ο χαρακτήρας αυτής της εργασίας ήταν καθαρά πειραματικός, ακολουθήθηκε η πρώτη μέθοδος. Σίγουρα πάντως για ένα fuzzy controller σε ολοκληρωμένη μορφή, θα πρέπει να χρησιμοποιηθεί η μέθοδος της μνήμης RAM.

4.4.5 Συναρτήσεις τύπου Π

Από τον ορισμό της συνάρτησης τύπου Π (σχέσεις (2.17) ως (2.20)) προκύπτει ότι αν A είναι μία τραπεζοειδής συνάρτηση συμμετοχής τότε η INT(A) είναι τύπου Π, όπου η πράξη INT ορίζεται από την (1.18). Εφόσον λοιπόν η τραπεζοειδής συνάρτηση έχει ήδη υλοποιηθεί, η δημιουργία συναρτήσεων τύπου Π ανάγεται στην υλοποίηση της μη-γραμμικής συνάρτησης (2.20) που επαναλαμβάνεται εδώ :

$$f(x) = \begin{cases} 2x^2, & 0 \le x \le 1/2 \\ 1-2(1-x)^2, & 1/2 < x \le 1 \end{cases}$$

Το κύκλωμα που κατασκευάσαμε για την υλοποίηση της συνάρτησης f φαίνεται στο Σχ. 4.25, όπου σαν συγκριτής χρησιμοποιήθηκε το ολοκληρωμένο LM393 της National, ενώ η υλοποίηση της μη γραμμικής συνάρτησης $2x^2$ αναλύεται παρακάτω. Οι δύο απομονωτές χρησιμοποιούνται για να αντισταθμίσουν την υψηλή αντίσταση εξόδου που παρουσιάζει η πύλη AND. Η λειτουργία του είναι προφανής. Οταν δηλαδή OUT1 < 0.5V, έχουμε V(1) = OUT1 και V(2) = -5V (λογικό 0), (η τάση στον αναστρέφοντα ακροδέκτη του LM393 είναι 0.5V). Επίσης έχουμε $V(3) = 2\cdot(OUT1)^2$ και V(4) = 1 - V(3). Από το μεταγωγικό διακόπτη επιλέγεται η V(3) και έτσι $V(5) = 2\cdot(OUT1)^2$. Αντίθετα, όταν OUT1 > 0.5, έχουμε V(1) = 1 - OUT1, V(2) = +5V (λογικό 1), $V(3) = 2\cdot(1-OUT1)^2$, V(4) = 1-V(3). Από το διακόπτη επιλέγεται η V(4) κι έτσι $V(5) = 1-2\cdot(1-OUT1)^2$. Τέλος, με το διακόπτη S7 (DIP Switch) επιλέγουμε αν στο κύκλωμα θα χρησιμοποιηθεί η τραπεζοειδής ή η καμπύλη συνάρτηση τύπου Π. Το ποτενσιόμετρο των 50K είναι απαραίτητο για την ακριβή ρύθμιση της τάσης του συγκριτή στα 0.5V, ώστε την στιγμή που ο διακόπτης αλλάζει κατάσταση να μην παρατηρούνται σπινθηρισμοί.

Τα αποτελέσματα ήταν πολύ ικανοποιητικά και έχουν ήδη παρουσιαστεί στα Σχ. 4.21, 4.22 και 4.23. Με κατάλληλες ρυθμίσεις, η τελική συνάρτηση ήταν εντελώς "λεία" και οι αλλαγές κατάστασης του διακόπτη δε γίνονταν αντιληπτές, ακόμα και στην πιό ευαίσθητη κλίμακα του παλμογράφου. Ανάλογες προσπάθειες χρησιμοποίησης μη γραμμικών αναλογικών κυκλωμάτων για την υλοποίηση τέτοιων συναρτήσεων έχουν παρουσιαστεί στα [62], [63] (με τη χρησιμοποίηση γραμμικών διαγωγών) ενώ στα [60], [63] προτείνεται επίσης η προσέγγιση τέτοιων συναρτήσεων με τμηματικά γραμμικές.

Τέλος το κύκλωμα που υλοποιεί τη συνάρτηση $2x^2$ βασίζεται στον αναλογικό πολλαπλασιαστή MC1595 της Motorola και φαίνεται στο Σχ. 4.26. Το MC1595 περιέχει διαγωγούς σε τεχνολογία διπολικών transistor και η πράξη του πολλαπλασιασμού πραγματοποιείται με τη μέθοδο της *μεταβλητής διαγωγιμότητας*. Ουσιαστικά η έξοδος του πολλαπλασιαστή είναι το *διαφορικό ρεύμα* μεταξύ των ακροδεκτών 2 και 14. Ετσι ο OP AMP είναι απαραίτητος για τη μετατροπή του ρεύματος αυτού σε τάση. Η σχεδίαση του κυκλώματος (δηλ. η επιλογή των αντιστάσεων) έγινε κυρίως με βάση τις οδηγίες του κατασκευαστή και ήταν αρκετά δύσκολη, γιατί απαιτείται κατάλληλη πόλωση των εσωτερικών transistor με τις

εξωτερικές αντιστάσεις, καθώς και πολύ ακριβής ρύθμιση των τάσεων απόκλισης (offset) εισόδου και εξόδου, καθώς και του κέρδους του πολλαπλασιαστή.

4.5 Υλοποίηση του defuzzification

Για την υλοποίηση του defuzzification χρησιμοποιείται η σχέση (4.3):

$$OUT = \frac{\mu_1 w_1 c_1 + \cdots + \mu_N w_N c_N}{\mu_1 w_1 + \cdots + \mu_N w_N}$$

Στη σχέση αυτή εμπλέκονται διαφόρων ειδών πράξεις, όπως πρόσθεση, πολλαπλασιασμός και διαίρεση. Σε κάθε χρονική στιγμή, ο βαθμός εκπλήρωσης του κανόνα μ_i, πρέπει να ισούται με OUT2a $\wedge$ OUT2b = min (OUT2a, OUT2b). Επομένως οι έξοδοι των block 2a και 2b οδηγούνται κατ'αρχήν σε μία πύλη AND. Στη συνέχεια ο πολ/σμός με το *βάρος* w_i γίνεται με ένα αναστρέφοντα ενισχυτή του οποίου η αντίσταση ανάδρασης, δηλ. το κέρδος, είναι ανάλογη του w_i. Έτσι απαιτείται μία αντίσταση για τη ρύθμιση του βάρους w_i του κάθε κανόνα. Ομοια γίνεται και ο πολ/σμός του $\mu_i w_i$ με το c_i, οπότε απαιτείται ακόμα μία αντίσταση για τη ρύθμιση της *προτεινόμενης εξόδου* c_i του κάθε κανόνα. Βλέπουμε λοιπόν ότι αφού τα w_i, c_i είναι σταθερά, ο πολ/σμός εκτελείται σχετικά απλά.

Στη συνέχεια η πρόσθεση πραγματοποιείται με δύο *ολοκληρωτές* (ένα για τον αριθμητή και ένα για τον παρονομαστή). Η πρόσθεση βασίζεται στο γεγονός ότι στο μεταβατικό στάδιο της αλλαγής του κανόνα, που το μ_i μεταβάλλεται, δύο διακόπτες "απομονώνουν" τους ολοκληρωτές ώστε να μην επηρεάζονται από τις μεταβολές. Οταν το μ_i έχει ήδη σταθεροποιηθεί, ο παλμός CK2 κλείνει τους διακόπτες και "επανασυνδέει" τους ολοκληρωτές για συγκεκριμένο χρονικό διάστημα (34μs). Ετσι ολοκληρώνεται μία *σταθερή τάση* και στο τέλος του χρονικού αυτού διαστήματος η *αύξηση* της τάσης του κάθε ολοκληρωτή (σε σχέση με την τιμή που είχε στην προηγούμενη επανάληψη) είναι ανάλογη της σταθερής αυτής τάσης. Ολα αυτά φαίνονται παραστατικότερα στα Σχήματα 4.27 και 4.28. Στο Σχ. 4.27, στο οποίο έχουμε N=3 (οπότε μεταξύ δύο παλμών SAM μεσολαβούν τρείς παλμοί CK1 και τρείς αλλαγές κανόνων) βλέπουμε ότι η μεταβολή της μ_i κατά την αλλαγή του κανόνα δεν είναι απότομη, αλλά παρουσιάζει ταλαντώσεις. Δεν έγινε καμία προσπάθεια για τη βελτίωση της χρονικής απόκρισης των κυκλωμάτων, αφού όπως θα δούμε αυτή δεν επηρεάζει τους υπολογισμούς.

Στο Σχ. 4.28 φαίνεται ότι οι τάσεις που οδηγούνται στους ολοκληρωτές συνδέονται με το μ_i όταν CK2 = 1 (όταν το μ_i είναι σταθερό) και με τη γη όταν CK2 = 0 (όταν το μ_i μεταβάλλεται). Στο Σχήμα αυτό υποθέσαμε για απλότητα w_i=1, ενώ $c_i < 1$, $i = 1,2,3$.

Στο Σχ. 4.29 βλέπουμε τις αντίστοιχες εξόδους των ολοκληρωτών. Οσο CK2=0, οι έξοδοι (τις οποίες ονομάζουμε **NUM** για τον αριθμητή και **DEN** για τον

παρονομαστή) παραμένουν σταθερές και ανεπηρέαστες απ'τις ταλαντώσεις του μ_i, ενώ όσο CK2=1 τα NUM και DEN αυξάνονται γραμμικά με το χρόνο, το πρώτο με ρυθμό ανάλογο του $\mu_i w_i c_i$ και το άλλο με ρυθμό ανάλογο του $\mu_i c_i$. Ετσι σχηματίζεται σταδιακά το σήμα

$$NUM = \mu_1 w_1 c_1 + \ldots + \mu_N w_N c_N$$

και το

$$DEN = \mu_1 w_1 + \ldots + \mu_N w_N$$

Ολα τα παραπάνω υλοποιούνται με το κύκλωμα του Σχ. 4.30.

Στη συνέχεια τα σήματα NUM και DEN οδηγούνται στην είσοδο ενός διαιρέτη που φαίνεται στα Σχ. 4.31 και 4.32 και βασίζεται και πάλι στον αναλογικό πολλαπλασιαστή MC1595, με τη διαφορά ότι ο πολλαπλασιαστής βρίσκεται τώρα *μέσα* στο βρόχο ανάδρασης του OP AMP. Η έξοδος του διαιρέτη, που φαίνεται στο Σχ. 4.33, υφίσταται δειγματοληψία στο τέλος του κύκλου (με το σήμα SAM) και αμέσως μετά οι ολοκληρωτές μηδενίζονται (με το σήμα RES) για να αρχίσει ένας νέος κύκλος υπολογισμών. Επειδή το κύκλωμα της διαίρεσης λειτουργεί σωστά μόνον όταν 500mV < DEN <10V, υπάρχουν ειδικά κυκλώματα που εμποδίζουν τη δειγματοληψία στην περίπτωση που η συνθήκη αυτή δεν ισχύει. Συγκεκριμένα, όταν DEN < 500mV ο παρονομαστής είναι πολύ μικρός, πράγμα που σημαίνει ότι πολύ λίγοι κανόνες έδωσαν συμπέρασμα (ή κανένας) και το αποτέλεσμα είναι αναξιόπιστο. Ετσι δεν γίνεται δειγματοληψία και το σύστημα διατηρεί την έξοδο που είχε στον προηγούμενο κύκλο. Κάτι ανάλογο συμβαίνει όταν DEN > 10V.

Κεφάλαιο 5

ΥΛΟΠΟΙΗΣΗ ΤΟΥ INTERFACE

Ο επεξεργαστής ασαφούς λογικής (fyzzy controller) που περιγράψαμε στο κεφάλαιο 4 δέχεται σαν εισόδους δύο αναλογικές τάσεις μεταξύ 0 και 1V, ενώ η έξοδος του είναι επίσης μία αναλογική τάση μεταξύ 0 και 1V. Για να λειτουργήσει το ολοκληρωμένο σύστημα ελέγχου είναι απαραίτητο η θέση και η ταχύτητα της σφαίρας να τροφοδοτηθούν κατάλληλα στον controller, ενώ η έξοδος του controller να τροφοδοτηθεί κατάλληλα στον ανεμιστήρα. Το ρόλο αυτό παίζει το *interface* μεταξύ της πειραματικής διάταξης και του controller.

5.1 Κύκλωμα Εισόδου

Το σημαντικότερο τμήμα του κυκλώματος εισόδου είναι ο *μετρητής απόστασης*. Το κύκλωμα αυτό βασίζεται σε δύο *αισθητήρες υπερήχων* (τύπου MA4052R της MuRata) από τους οποίους ο ένας λειτουργεί ως *πομπός* και ο άλλος ως *δέκτης*. Το συνολικό κύκλωμα φαίνεται στο Σχ. 5.1. Συνοπτικά η λειτουργία του έχει ως εξής. Μία παλμογεννήτρια (555) παράγει ένα τετραγωνικό παλμό Α, διάρκειας 320μs που επαναλαμβάνεται περιοδικά κάθε 12ms. Στη συνέχεια με μία πύλη NAND Schmitt trigger (ολοκληρωμένο MC14093 της Motorola) δημιουργείται μια παλμοσειρά Β συχνότητας 40 KHz. Τα σήματα Α και Β φαίνονται στο Σχ. 5.2. Η παλμοσειρά Β τροφοδοτείται μέσω δυο ζευγών αναστροφέων CMOS (ολοκληρωμένο MC14069 της Motorola) στον πομπό. Μεταξύ του πομπού και των αναστροφέων μεσολαβούν δύο πυκνωτές 100nF που κόβουν τη DC συνιστώσα και αυξάνουν το πλάτος του εκπεμπόμενου σήματος σε 20Vp-p. Η συχνότητα 40 KHz επιλέγεται γιατί είναι η συχνότητα συντονισμού των αισθητήρων, οπότε στη συχνότητα αυτή έχουμε τη μεγαλύτερη απόδοση.

Η παλμοσειρά εκπέμπεται προς το εσωτερικό του σωλήνα, ανακλάται από τη σφαίρα και επιστρέφει προς την κορυφή του σωλήνα όπου βρίσκεται ο δέκτης. Στη συνέχεια ενισχύεται από ένα ενισχυτή με κέρδος DC 100 (ή 40db) και οδηγείται στον αναστρέφοντα ακροδέκτη ενός συγκριτή, ενώ στον άλλο ακροδέκτη υπάρχει μία μεταβαλλόμενη τάση, όπως φαίνεται στο Σχ. 5.3. Αυτό γίνεται γιατί εκτός από

το *ανακλώμενο* σήμα έρχεται στο δέκτη και ένα σήμα *απ'ευθείας* από τον πομπό. Για να μην επηρεαστεί η μέτρηση από το σήμα αυτό, που φαίνεται στο Σχ. 5.3, η στάθμη του συγκριτή μεταβάλλεται με το χρόνο. Συγκεκριμένα είναι υψηλή μόνο τη στιγμή που εκπέμπεται η παλμοσειρά και μειώνεται σταδιακά ώστε όταν το ανακλώμενο σήμα επιστρέψει, η στάθμη να έχει ήδη φτάσει στο κατώτερό της σημείο (περίπου 100mV).

Αυτό φαίνεται και στο Σχ. 5.4 όπου βλέπουμε το απ'ευθείας (ισχυρότερο) και το ανακλώμενο (ασθενέστερο) σήμα, να απέχουν μεταξύ τους 6.6ms. Βλέπουμε ότι ο συγκριτής αλλάζει κατάσταση μόνο με την λήψη του ανακλώμενου σήματος, γιατί τότε μόνο η στάθμη σύγκρισης έχει κατέβει αρκετά χαμηλά.

Η έξοδος του συγκριτή οδηγείται σε ενα μανδαλωτή RS που εξαλείφει τους σπινθηρισμούς και δίνει ένα και μόνο παλμό τη στιγμή που λαμβάνεται το ανακλώμενο σήμα. Με τη βοήθεια ενος ολοκληρωτή παράγεται ένα σήμα που είναι ανάλογο προς τη διάρκεια του παλμού αυτού, η οποία με τη σειρά της είναι ανάλογη με την απόσταση της σφαίρας από τους αισθητήρες. Ετσι παίρνουμε στην έξοδο ένα αναλογικό σήμα μεταξύ 0 και 1V, ανάλογο της μετρούμενης απόστασης.

Σημειώνουμε ότι η ταχύτητα του ήχου στον αέρα είναι περίπου 340m/s οπότε στο συγκεκριμένο παράδειγμα η απόσταση της σφαίρας είναι 340m/s · 6.6ms / 2 = 1,12m (6.6ms είναι ο χρόνος που χρειάζεται το ηχητικό σήμα για να φτάσει τη σφαίρα και να επιστρέψει, γι'αυτό διαιρούμε δια 2).

Το υπόλοιπο κύκλωμα εισόδου φαίνεται στα Σχ. 5.5 και 5.6 και σκοπός του είναι ο υπολογισμός του *σφάλματος θέσης* και της *ταχύτητας* της σφαίρας. Το σφάλμα θέσης, e, υπολογίζεται από το σήμα του μετρητή απόστασης, x, απλά αφαιρώντας μία τάση που είναι μεταξύ 0 και 1V και που ρυθμίζεται από το ποτενσιόμετρο των 10 K. Η τάση αυτή παριστάνει την "*επιθυμητή*" θέση της σφαίρας.

Το σφάλμα ταχύτητας, ė, υπολογίζεται με παραγώγιση. Για να αποφευχθούν τα προβλήματα θορύβου που παρουσιάζει το κύκλωμα παραγώγισης προσθέσαμε την αντίσταση των 10K και τον πυκνωτή των 330mF που εισάγουν δύο πόλους στο σύστημα και μειώνουν το θόρυβο υψηλών συχνοτήτων. Επίσης υπενθυμίζουμε ότι *οι μεταβλητές e και ė είναι κανονικοποιημένες στο διάστημα [0,1V] και επομένως η τιμή "ισορροπίας" τους δεν είναι μηδέν αλλά 0.5.*

5.2 Κύκλωμα εξόδου

Το κύκλωμα εξόδου φαίνεται στο Σχ. 5.6. Αν y είναι η έξοδος του controller, τότε μετά το μη αναστρέφοντα ενισχυτή η τάση αυτή γίνεται y' = 8y + 2.8V. Αφού λοιπόν η y παίρνει τιμές στο διάστημα [0,1V], η y' παίρνει τιμές στο διάστημα [3V,11V] περίπου. Οι τάσεις αυτές είναι κατάλληλες για την

τροφοδότηση του μοτέρ του ανεμιστήρα, αφού 3V είναι η ελάχιστη και 11V η μέγιστη τάση έτσι ώστε να έχουμε μέγιστη ταχύτητα καθόδου και ανόδου της σφαίρας, αντίστοιχα. Στη συνέχεια ο δεύτερος OP AMP σε συνδυασμό με ένα darlington ισχύος (TIP120 της Texas Instruments) λειτουργεί σαν ακόλουθος τάσης με τη διαφορά ότι μπορεί να τροφοδοτήσει το μοτέρ με ρεύμα μέχρι και 1A, που είναι απαραίτητο για τη φυσιολογική λειτουργία του.

Κεφάλαιο 6

ΣΧΕΔΙΑΣΗ ΤΟΥ ΑΣΑΦΟΥΣ ΣΥΣΤΗΜΑΤΟΣ ΚΑΙ ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Το ασαφές σύστημα ελέγχου που σχεδιάσαμε έχει κανόνες της μορφής

R_i : IF e IS A_i AND ė IS B_i THEN y = c_i

όπου e το σφάλμα θέσης, ė το σφάλμα ταχύτητας, y η εξοδος και οι "ετικέττες" A_i και B_i παίρνουν τις τιμές :

MP : Medium Positive
SP : Small Positive
ZE : Zero
SN : Small Negative
MN : Medium Negative

δηλαδή το υπερσύνολο αναφοράς X των εισόδων διαμερίζεται σε πέντε ασαφείς περιοχές, σύμφωνα με τις συναρτήσεις συμμετοχής που φαίνονται στο Σχ. 6.1. (χρησιμοποιήθηκαν τραπεζοειδείς συναρτήσεις και τύπου Π). Οι μεταβλητές εισόδου e και ė, όπως προκύπτουν από τα κυκλώματα του interface, παίρνουν τιμές στο διάστημα [-1V, +1V] και το σημείο "ισορροπίας" του συστήματος είναι e = ė = 0. Στη συνέχεια όμως το διάστημα [-1V, +1V] κανονικοποιείται και γίνεται X = [0,1V], οπότε το σημείο ισορροπίας γίνεται πλέον e = ė = 0.5V. Το υπερσύνολο αναφοράς της εξόδου y είναι επίσης το X = [0,1V], δηλ. η έξοδος y=0 αντιστοιχεί στην ελάχιστη τροφοδοσία του ανεμιστήρα (και στην μέγιστη ταχύτητα *πτώσης* της σφαίρας) ενώ η y = 1V στη μέγιστη τροφοδοσία (και στη μέγιστη ταχύτητα *ανόδου* της σφαίρας). Στη συνέχεια το διάστημα αυτό *αποκανικοποιείται* έτσι ώστε η μέγιστη και η ελάχιστη τάση τροφοδοσίας του ανεμιστήρα να είναι 11V και 3V αντίστοιχα (το μοτέρ του ανεμιστήρα λειτουργεί κανονικά με 12V DC, αλλά μεταβάλλοντας την τάση τροφοδοσίας του μεταβάλλεται και η ταχύτητα περιστροφής του). Οι ακραίες τιμές 3V και 11V καθορίστηκαν μεταβάλλοντας χειροκίνητα την τάση που εφαρμόζεται στον ανεμιστήρα και παρατηρώντας την επίδραση στην κίνηση της σφαίρας. Με τον ίδιο τρόπο καθορίστηκαν και προτεινόμενες έξοδοι του κάθε κανόνα, όπως θα δούμε παρακάτω.

Μία αναλυτική μελέτη της δυναμικής συμπεριφοράς του συστήματος θα μπορούσε να γίνει χρησιμοποιώντας στοιχεία ρευστοδυναμικής. Ομως μία τέτοια ανάλυση θα ήταν άσκοπη αφού το νόημα της ασαφούς λογικής είναι ότι μπορεί να λειτουργήσει σε συστήματα των οποίων οι εξισώσεις δεν είναι γνωστές, ή είναι ιδιαίτερα πολύπλοκες. Ετσι, με μία "εμπειρική " ανάλυση μπορέσαμε να καθορίσουμε ένα σύνολο κανόνων (rulebase) που έδωσε τελικά πολύ ικανοποιητικά αποτελέσματα.

Οι κανόνες αναπτύχθηκαν ξεκινώντας από την υπόθεση ότι η επιθυμητή απόκριση ήταν να κινήσουμε τη σφαίρα από το σημείο HIGH στο LOW (και αντίστροφα) με τη μέγιστη δυνατή ταχύτητα, με την ελάχιστη δυνατή υπερύψωση (overshoot), και τις ελάχιστες δυνατές ταλαντώσεις. Ετσι καθορίστηκε ένα σύνολο κανόνων που να μπορεί να δώσει μέγιστη επιτάχυνση όσο η σφαίρα βρίσκεται "μακριά" από το στόχο και μέγιστη επιβράδυνση όταν η σφαίρα πλησιάζει "κοντά" στο στόχο. Σημειώνουμε ότι η χρησιμοποίηση της ταχύτητας ė σαν μεταβλητή εισόδου είναι απαραίτητη ώστε αν η σφαίρα *πλησιάζει* προς το στόχο να αρχίζει η διαδικασία της επιβράδυνσης *πρίν* η σφαίρα φτάσει στο τελικό σημείο. Αντίθετα, αν η σφαίρα *απομακρύνεται* από το στόχο πρέπει να έχουμε μέγιστη επιτάχυνση προς την κατεύθυνσή του.

Οι κανόνες που αναπτύχθηκαν με αυτό τον τρόπο φαίνονται στον Πίνακα 6.1.

		e				
		MN	SN	ZE	SP	MP
ė	MN					
	SN		0.9		0.6	
	ZE	1.0	0.8	0.6	0.4	0.2
	SP		0.6		0.3	
	MP					

Πίνακας 6.1: *Οι κανόνες του ασαφούς συστήματος*

Εφόσον η μεταβλητές e και ė μπορουν να παίρνουν 5 διαφορετικές "ασαφείς" τιμές (MP, SP, ZE, SN, MN), οι δυνατοί συνδυασμοί τους είναι 25. Τόσοι επομένως πρέπει να είναι και οι κανόνες. Στην πράξη όμως οι 9 μόνο κανόνες που φαίνονται στον πίνακα 6.1 (στα κενά δεν αντιστοιχεί κανένας κανόνας) ήταν αρκετά για τη σωστή κίνηση της σφαίρας. Η αναπαράσταση των κανόνων με τη μορφή του πίνακα 6.1 ονομάζεται *Fuzzy Associative Memory (FAM)* [8], και είναι πρακτική γιατί επιτρέπει τη "συμπυκνωμένη" απεικόνιση τους. Κάθε στοιχείο του πίνακα αντιστοιχεί σε ένα κανόνα. Για παράδειγμα, ο κανόνας

IF e IS SP AND ė IS ZE THEN y = 0.4

παριστάνεται στον πίνακα από το στοιχείο 0.4 (*κανονικοποιημένη* έξοδος) που βρίσκεται στη στήλη SP και τη σειρά ZE. Η "λογική" που εκφράζει αυτό το σύνολο κανόνων γίνεται αμέσως προφανής με απλή παρατήρηση του παραπάνω πίνακα. Από την 3η σειρά του πίνακα βλέπουμε όταν το σφάλμα θέσης είναι αρνητικό (π.χ. MN) η έξοδος είναι μεγάλη ώστε η σφαίρα να ανέβει προς τη θέση ισορροπίας, ενώ όσο το σφάλμα μεγαλώνει τόσο η έξοδος μικραίνει. Στη θέση ισορροπίας (ZE) η έξοδος είναι 0.6, που αντιστοιχεί σε τέτοια ρύθμιση του ανεμιστήρα ώστε η σφαίρα να είναι περίπου σταθερή. Επίσης, παρατηρώντας τη 2η στήλη, που αντιστοιχεί σε μία θέση της σφαίρας λίγο πιο χαμηλά από το σημείο ισορροπίας, βλέπουμε ότι η έξοδος επηρεάζεται σημαντικά από την ταχύτητα ὲ. Αν δηλαδή η σφαίρα πλησιάζει προς το στόχο (ὲ IS SP) η έξοδος είναι μικρή (0.6) ενώ αν απομακρύνεται (ὲ IS SN) η έξοδος είναι μεγάλη (0.9).

Παρατηρούμε ότι το σύστημα είναι ουσιαστικά κατά το ήμισυ μόνο ασαφές, αφού η έξοδος του κάθε κανόνα είναι ένας σαφής αριθμός. Σύμφωνα όμως με την ανάλυση της παραγράφου 2.3.7 η μέθοδος αυτή είναι ισοδύναμη με τη γενική, αφού επιλέξαμε implication: product, aggregation: sum and defuzzification: weighted average. Οι αριθμητικές τιμές του πίνακα 6.1 είναι ενδεικτικές (εξάλλου σε ένα τέτοιο σύστημα η αριθμητική ακρίβεια δεν έχει ιδιαίτερη σημασία) και προέκυψαν μετά από μετρήσεις και πειραματισμούς ώστε η συμπεριφορά του συστήματος να είναι η καλύτερη δυνατή, δηλαδή να ελαχιστοποιηθεί ο χρόνος μετάβασης μεταξύ των δύο setpoints, καθώς και η υπερύψωση (overshoot). Η διαδικασία ρύθμισης πάντως ήταν ιδιαίτερα εύκολη και έγινε σε σύντομο χρονικό διάστημα. Ετσι έγινε φανερή η ανωτερότητα της ασαφούς λογικης που δίνει τη δυνατότητα να μετατρέπει τις "σκέψεις" σε μαθηματικές εξισώσεις ώστε να μπορούμε μέσα σε λίγη ώρα να έχουμε ένα σύστημα ελέγχου "κομμένο και ραμμένο" για τη συγκεκριμένη εφαρμογή.

Τα αποτελέσματα του πειράματος φαίνονται στο Σχήμα 6.2, όπου βλέπουμε τη χρονική απόκριση του συστήματος, όταν η επιθυμητή θέση ισορροπίας αλλάζει απότομα από μια τιμή σε μια άλλη. Ο χρόνος που χρειάζεται η σφαίρα για να σταθεροποιηθεί στη νέα αυτή θέση (στο συγκεκριμένο παράδειγμα 3.5s) καθώς και το ποσοστό υπερύψωσης (περίπου 5%) μπορούν εύκολα να ρυθμιστούν μεταβάλλοντας τις τιμές c_i του κάθε κανόνα. Επίσης μπορεί εύκολα να διαπιστωθεί ότι κάθε κανόνας επηρεάζει διαφορετικό "τμήμα" της συμπεριφοράς του συστήματος (π.χ. το ανέβασμα ή το κατέβασμα της σφαίρας κλπ.)

Τέλος από τις καμπύλες του σχήματος 6.2 μπορούμε να δούμε ότι η απόκριση του συστήματος είναι πιο γρήγορη με τη χρησιμοποίηση καμπυλών τύπου Π (α) παρά τραπεζοειδών συναρτήσεων (β). Περαιτέρω μελέτη μπορεί να γίνει ως προς την επίδραση του παράγοντα της *υπερκάλυψης (overlap)* των πέντε συναρτήσεων συμμετοχής του σχήματος 6.1. Πάντως μετά από πειραματισμούς βρέθηκε ότι για 50% overlap περίπου είχαμε την καλύτερη απόκριση.

Συμπεράσματα

Οπως αναφέρθηκε και στον πρόλογο, το πείραμα αυτό είχε καθαρά χαρακτήρα επίδειξης και κύριος σκοπός του ήταν να γίνει φανερή η δυνατότητα υλοποίησης των μεθόδων της ασαφούς λογικής με πλήρως αναλογικά κυκλώματα. Μάλιστα η μέθοδος που ακολουθήθηκε στη σχεδίαση του fuzzy controller θα μπορούσε εύκολα μεταφερθεί στη σχεδίαση ενός microcontroller σε ολοκληρωμένη μορφή.

Γι'αυτό το λόγο πρέπει καταρχήν να τονίσουμε ότι δεν έγινε προσπάθεια *βελτιστοποίησης* του ασαφούς συστήματος (κάτι που θα μπορούμε να γίνει με κάποιο αυτοματοποιημένο αλγόριθμο εκμάθησης), αφου η ικανοποιητική λειτουργία του μας ήταν αρκετή.

Ενα δεύτερο σημείο που θα πρέπει να τονίσουμε είναι ότι στην πειραματική διάταξη που επιλέξαμε θα μπορούσε άνετα να χρησιμοποιηθεί και ένα *κλασσικό σύστημα ελέγχου*, αφού για να γίνει η επίδειξη έπρεπε το σύστημα να είναι αρκετά απλό. Αντίθετα, ο στόχος είναι η εφαρμογή της ασαφούς λογικής σε πολυπλοκότερα συστήματα (πχ. έλεγχος χημικών διεργασιών ή αυτόματη οδήγηση αυτοκινήτου) τα οποία απαιτούν "γνώση" και "εμπειρία" και στα οποία οι κλασσικές μέθοδοι δε μπορούν να εφαρμοστούν. Πάντως ένα πείραμα παρόμοιο με το δικό μας που υλοποιήθηκε με ψηφιακές μεθόδους και παρουσιάζεται στο [55] έδειξε ότι ακόμα και στην περίπτωση ενός τέτοιου απλού συστήματος η ασαφής λογική ξεπέρασε σε επιδόσεις την κλασσική μέθοδο (PID controller).

Ως προς την υλοποίηση ενός ασαφούς μικροεπεξεργαστή (fuzzy microprocessor) θα πρέπει να σημειώσουμε ότι είναι απαραίτητο να βρεθεί η "χρυσή τομή" μεταξύ του *βαθμού ελευθερίας* που δίνεται στο χρήστη ως προς τη ρύθμιση των κανόνων και της *ευελιξίας* της ρύθμισης αυτής. Για παράδειγμα στο σύστημά μας δόθηκε η δυνατότητα ρύθμισης των συναρτήσεων συμμετοχής σε κάθε κανόνα χωριστά, δηλαδή σε κάθε κανόνα οι καμπύλες μπορούν να έχουν διαφορετική *κλίση* και *εύρος*. Η ελευθερία αυτή επιτρέπει να έχουμε μεγαλύτερη "πυκνότητα" κανόνων γύρω από το επιθυμητό σημείο ισορροπίας και επομένως καλύτερο έλεγχο γύρω από το σημείο αυτό . Η ρύθμιση όμως τόσων πολλών στοιχείων αποδείχτηκε τόσο δύσκολη και χρονοβόρα στην πράξη ώστε προτιμήθηκε τελικά όλες οι καμπύλες να έχουν την ίδια μορφή και να διαφέρουν μόνο στη μέση τιμή τους. Αυτή προφανώς δεν ήταν η βέλτιστη λύση αλλά με τον τρόπο αυτό σχεδιάστηκε μέσα σε λιγότερο από μια μέρα ένα σύστημα που λειτουργεί ικανοποιητικά. Τέτοιες αποφάσεις θα πρέπει να ληφθούν ώστε να μην αυξάνεται άσκοπα η πολυπλοκότητα των κυκλωμάτων.

Συμπερασματικά μπορούμε να αναφέρουμε ότι η ασαφής λογική έναι ένας κλάδος ταχύτατα αναπτυσσόμενος και πρόσφορος για περαιτέρω έρευνα. Η υλοποίηση μεθόδων της ασαφούς λογικής με αναλογικά κυκλώματα VLSI μπορεί να προσφέρει πολλά στον τομέα αυτό και μέχρι στιγμής έχουν γίνει λίγες τέτοιες προσπάθειες. Στο μέλλον τέτοια κυκλώματα θα μπορούσαν να αποτελέσουν τη βάση για την κατασκευή αναλογικών "ασαφών" υπολογιστών έκτης γενιάς [58].

Βιβλιογραφικές αναφορές

[1] Ε. Λεμπέσης, Ρ.Ε. Κίνγκ, *Στοιχεία ασαφούς λογικης και εφαρμογή στον έμπειρο έλεγχο διεργασιών*, Πανεπιστήμιο Πατρών 1991.

[2] Σ. Τζαφέστας, *Εισαγωγή στην τεχνητή νοημοσύνη και τα έμπειρα συστήματα*, Εθνικό Μετσόβιο Πολυτεχνείο, Αθήνα 1988.

[3] Ι. Βασιλείου, *Συνοπτική παρουσίαση της ασαφούς λογικής και των εφαρμογών της - Πρόγραμμα προσομοίωσης μεθόδων της ασαφούς λογικής*, (Διπλωματική εργασία), Ε.Μ.Π. Αθήνα 1992.

[4] L.A. Zadeh, "Fuzzy Sets", *Information and control*, Vol. 8, 1965.

[5] L.A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes", *IEEE Trans. Systems, Man and Cybernetics*, Vol. 3, 1973.

[6] L.A. Zadeh, "Fuzzy Logic", *IEEE Computer*, April 1988.

[7] L.A. Zadeh, "Making computers think like people", *IEEE Spectrum*, Aug. 1984.

[8] B. Kosko, *Neural networks and fuzzy systems - a dynamical systems approach to machine intelligence*, Englewood Cliffs, NJ, Prentice Hall, 1992.

[9] H.J. Zimmermann, *Fuzzy set Theory - and its applications*, Norwell, Mass., Kluwer Academic Press, 1987.

[10] E.Cox, "Solving problems with fuzzy logic", *AI Expert*, March 1992.

[11] E.H. Mamdani & D. Stipaničev, "Fuzzy sets theory and process control - past, present and future", *Intelligent Control Gazette, IEEE Control Systems Society Technical Committee on Intelligent Control*, Summer 1988.

[12] E.H. Mamdani & S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller", *International Journal of Man-Machine Studies*, Vol. 7, Jan. 1975.

[13] Chuen Chien Lee, "Fuzzy logic in control systems: fuzzy logic controller - Part I", *IEEE Trans. Systems, Man and Cybernetics*, Vol. 20, No.2, March/April 1990.

[14] E. Cox, "Adaptive fuzzy systems", *IEEE Spectrum*, Feb. 1993.

[15] E. Cox, "Fuzzy fundamentals", *IEEE Spectrum*, Oct. 1992.

[16] D.G. Schwartz & G.J.Klir, "Fuzzy logic flowers in Japan", *IEEE Spectrum*, July 1992.

[17] D.I. Brubaker, "Fuzzy logic basics: intuitive rules replace complex math", *EDN*, June 18, 1992.

[18] R. Wiggins, "Docking a Truck: a genetic fuzzy approach", *AI Expert*, May 1992.
[19] J.H. Holland, "Genetic Algorithms", *Scientific American*, July 1992.
[20] P.Davis, "Is nature mathematical ?", *New Scientist*, March 21, 1992.
[21] L.A. Zadeh, "The calculus of fuzzy if/then rules", *AI Expert*, March 1992.
[22] A. Sangalli, "Fuzzy logic goes to market", *New Scientist*, Feb. 1992.
[23] E. Cox, "The great myths of fuzzy logic", *AI Expert*, Jan. 1992.
[24] D.J. Dubois & H. Prade, *Fuzzy sets and systems: Theory and applications*, New York, Academic Press, 1980.
[25] T. Williams, "Fuzzy logic simplifies complex control problems", *Computer Design*, March 1, 1991.
[26] T.J. Schwartz, "Fuzzy systems in the real world", *AI Expert*, Aug. 1990.
[27] M.H. Lim & Y. Takefuji, "Implementing fuzzy rule-based systems on silicon chips", *IEEE Expert*, Feb. 1990.
[28] K.J. Schmuker, *Fuzzy sets, natural language computations and risk analysis*, Computer Science Press, Rockville, Md., 1984.
[29] N. Rescher, *Many-valued logic*, McGraw-Hill, New York, 1969.
[30] R. Herman, "Computing with a human face", *New Scientist*, May 6, 1982.
[31] S. Tzafestas, "Fuzzy expert control: recent results with applications to robotic systems", 1989.
[32] C. von Altrock, B. Krause & H.J. Zimmerman, "Advanced fuzzy logic control of a model car in extreme situations", *Fuzzy Sets and Systems* **48**, 1992.
[33] M. Mitsumoto & H.J. Zimmermann, "Comparison of fuzzy reasoning methods", *Fuzzy sets and systems* **18**, 1982.
[34] M. Sugeno, T. Murofushi, T. Mori, T. Tatematsu & J. Tanaka, "Fuzzy algorithmic control of a model car by oral instructions", *Fuzzy Sets and Systems* **32**, 1989.
[35] J.C. Bezdek, "Editorial: fuzzy models - what are they, and why ?", *IEEE Trans. Fuzzy Systems*, Vol. 1, No. 1, Feb. 1993.
[36] M.Sugeno & T.Yasukawa, "A Fuzzy-logic-based approach to qualitative modeling", *IEEE Trans. Fuzzy Systems*, Vol. 1, No.1, Feb. 1993.
[37] R.R. Yager & D.P. Filev, "SLIDE: A simple adaptive defuzzification method", *IEEE Trans. Fuzzy Systems*, Vol. 1, No.1, Feb. 1993.
[38] J.J. Barron, "Putting fuzzy logic into focus", *BYTE*, April 1993.
[39] E. Cox, "Integrating fuzzy logic into neural nets", *AI Expert*, June 1992.
[40] M.M. Gupta, "Cognition, perception and uncertainty", *Fuzzy computing: theory, hardware and applications*, edited by M.M. Gupta & T. Yamakawa, North Holland, 1988.
[41] D. Dubois, R. Martin-Clouaire & H. Prade, "Practical computing in fuzzy logic", *Fuzzy computing: theory, hardware and applications*, edited by M.M. Gupta & T. Yamakawa, North Holland, 1988.
[42] Wu Wangming, "A multivalued logic system with respect to T-norms", *Fuzzy computing: theory, hardware and applications*, edited by M.M.

Gupta & T. Yamakawa, North Holland, 1988.

[43] N. Nakajima, "A generalized fuzzy set and its representation", *Fuzzy computing: theory, hardware and applications*, edited by M.M. Gupta & T. Yamakawa, North Holland, 1988.

[44] G. Bortolan & R. Degani, "Linguistic approximation of fuzzy certainty factors in computerized electrocardiography", *Fuzzy computing: theory, hardware and applications*, edited by M.M. Gupta & T. Yamakawa, North Holland, 1988.

[45] R.E. King & F.C. Karonis, "Multilevel expert control of a large scale industrial process", *Fuzzy computing: theory, hardware and applications*, edited by M.M. Gupta & T. Yamakawa, North Holland, 1988.

[46] K.Sugiyama, "Rule-based self-organising controller", *Fuzzy computing: theory, hardware and applications*, edited by M.M. Gupta & T. Yamakawa, North Holland, 1988.

[47] X.-T. Peng, S.-M. Liu, T. Yamakawa, P. Wang & X. Liu, "Self-regulating PID controllers and their applications to a temperature controlling process", *Fuzzy computing: theory, hardware and applications*, edited by M.M. Gupta & T. Yamakawa, North Holland, 1988.

[48] I.B. Turksen, "An approximate reasoning framework for production management", *Fuzzy computing: theory, hardware and applications*, edited by M.M. Gupta & T. Yamakawa, North Holland, 1988.

[49] M. Black, "Vagueness: An exercise in logical analysis", *Philosophy of Science*, Vol. 4, pp. 427-455, 1937.

[50] R.Zwick, E.Karlstein & D.V. Budescu, "Measures of similarity between fuzzy concepts: a comparative analysis", Carnegie Mellon Univ., Pittsburgh, Penn., 1986.

[51] J.C. Bezdek, *Pattern recognition with fuzzy objective function algorithm*, New York, Plenum Press, 1981.

[52] Y.Fukuyama & M.Sugeno, "A new method of choosing the number of clusters for fuzzy c-means method", *Proc. 5th fuzzy systems symposium*, pp. 247-250, 1989.

[53] E. Cox & T.J. Schwartz, "Around the world with fuzzy products", *AI Expert*, March 1993.

[54] D. Brubaker, "An accelerated kernel for fuzzy systems", *AI Expert*, March 1993.

[55] *The fuzzy source*, Vol. 1, Issue 2, Togai Infralogic Inc., Fall/ Winter 1991.

[56] H. Watabane, W.D. Dettloff & K.E. Yount, "A VLSI fuzzy logic controller with reconfigurable, cascadable architecture", *IEEE Journal of solid-state circuits*, Vol. 25, No.2, April 1990.

[57] T.Yamakawa, "Fuzzy logic circuits in current mode', *Analysis of fuzzy information*, Vol. I, chap. 17, editor J.C. Bezdek, CRC Press, 1987

[58] T. Yamakawa & T. Miki, "The current mode fuzzy logic integrated circuits fabricated by the standard CMOS process", *IEEE Trans. on Computers*, Vol. c-35, No.2, Feb. 1986.

[59] J.M. Sibigtroth, "Implementing fuzzy expert rules in hardware", *AI*

Expert, April 1982.

[60] T. Kettner, C. Heite & K. Schumacher, "Analog CMOS realization of fuzzy logic membership functions".

[61] J.-J. Chen, C.-C. Chen & H.-W. Tsao, "Tunable membership function circuit for fuzzy control systems using CMOS technology", *Electronics Letters*, Vol. 28, No. 22, October 1992.

[62] E. Seevinck & R.F. Wassenaar, "A versatile CMOS linear transconductor/ square-law function circuit", *IEEE Journal of solid-state circuits*, Vol. sc-22, No. 3, June 1987.

[63] E. Sánchez-Sinencio, J. Ramírez-Angulo, B. Linares-Barranco & A. Rodriguez-Vázquez, "Operational transconductance amplifier-based nonlinear function syntheses", *IEEE Journal of solid-state circuits*, Vol. 24, No. 6, December 1989.

[64] T. Yamakawa, "Intrinsic fuzzy electronic circuits for 6th generation computer", *Fuzzy computing: theory, hardware and applications*, edited by M.M. Gupta & T. Yamakawa, North Holland, 1988.

[65] K. Hirota, K. Ozawa, "Fuzzy flip-flop as a basis of fuzzy memory modules", *Fuzzy computing: theory, hardware and applications*, edited by M.M. Gupta & T. Yamakawa, North Holland, 1988.

[66] T. Yamakawa, "A simple fuzzy computer hardware system employing MIN & MAX operations", *Preprints of 2nd IFSA Congress*, Tokyo, July 20-25, 1987, pp. 827-830.

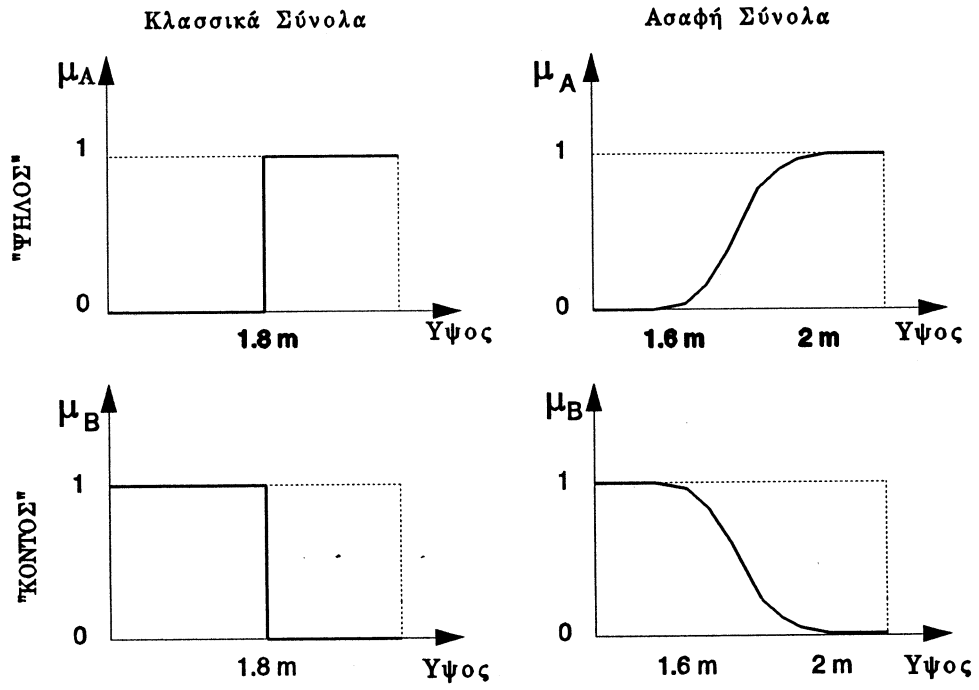
[67] G. Boole, *An investigation of the laws of thought*, New York, Dover Pub., 1854.

[68] C.L. Karr & E.J. Gentry, "Fuzzy control of pH using genetic algorithms", *IEEE Trans. Fuzzy Systems*, Vol. 1, No. 1, Feb. 1993.

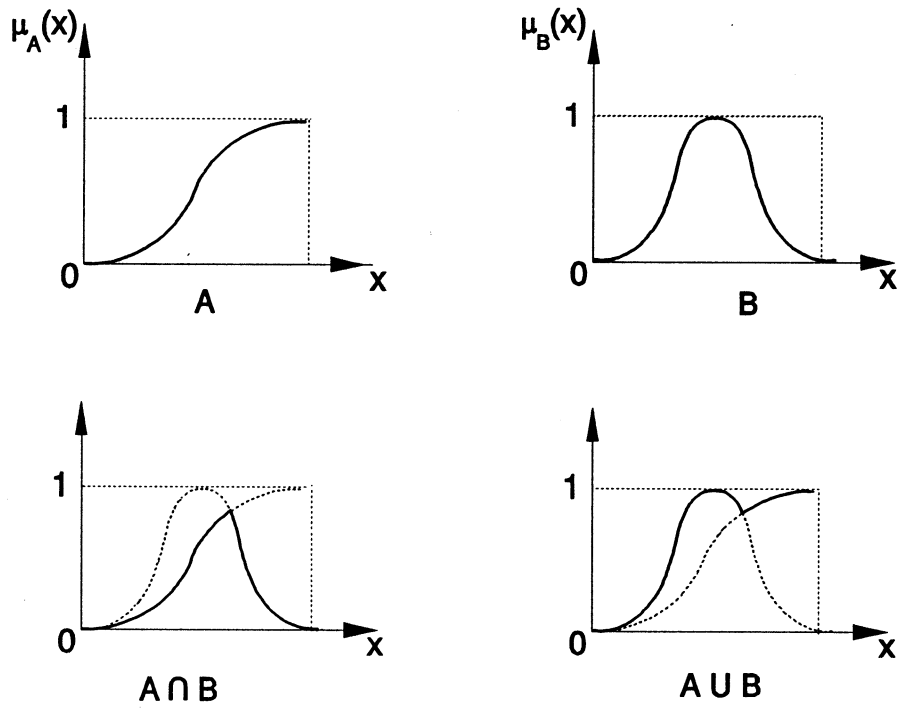
[69] R.R. Yager, "Inference in a multivalued system", *Inter. J. Man-Machine Studies* **23**, pp. 27-44, 1985.

[70] D. Dubois & H. Prade, "Fuzzy logic and the generalized modus ponens revisited", *Cybernetics and Systems*, Vol. 15, pp. 293-331, 1984.

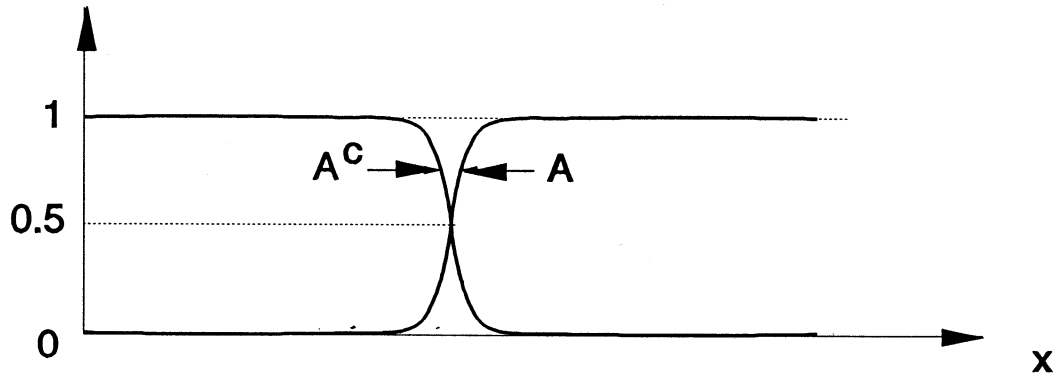
ΣΧΗΜΑΤΑ



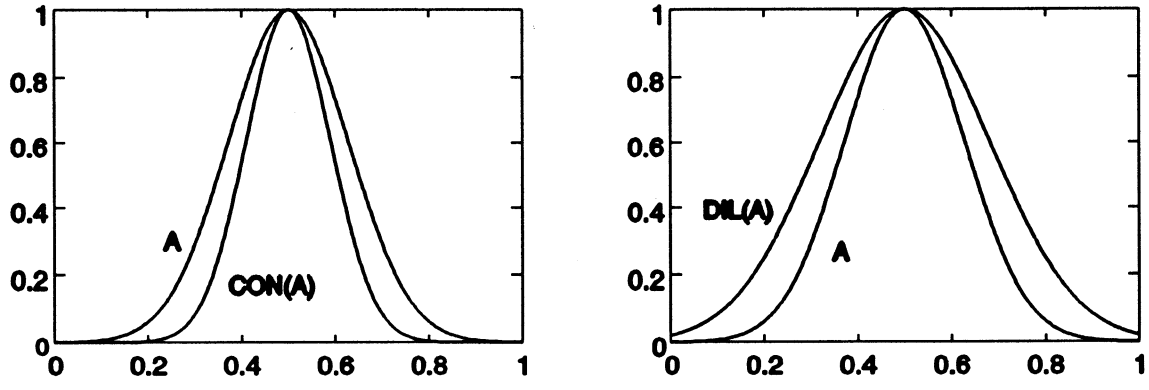
Σχήμα 1.1: Κλασσικά και ασαφή σύνολα για τις έννοιες "ΚΟΝΤΟΣ" και "ΨΗΛΟΣ".



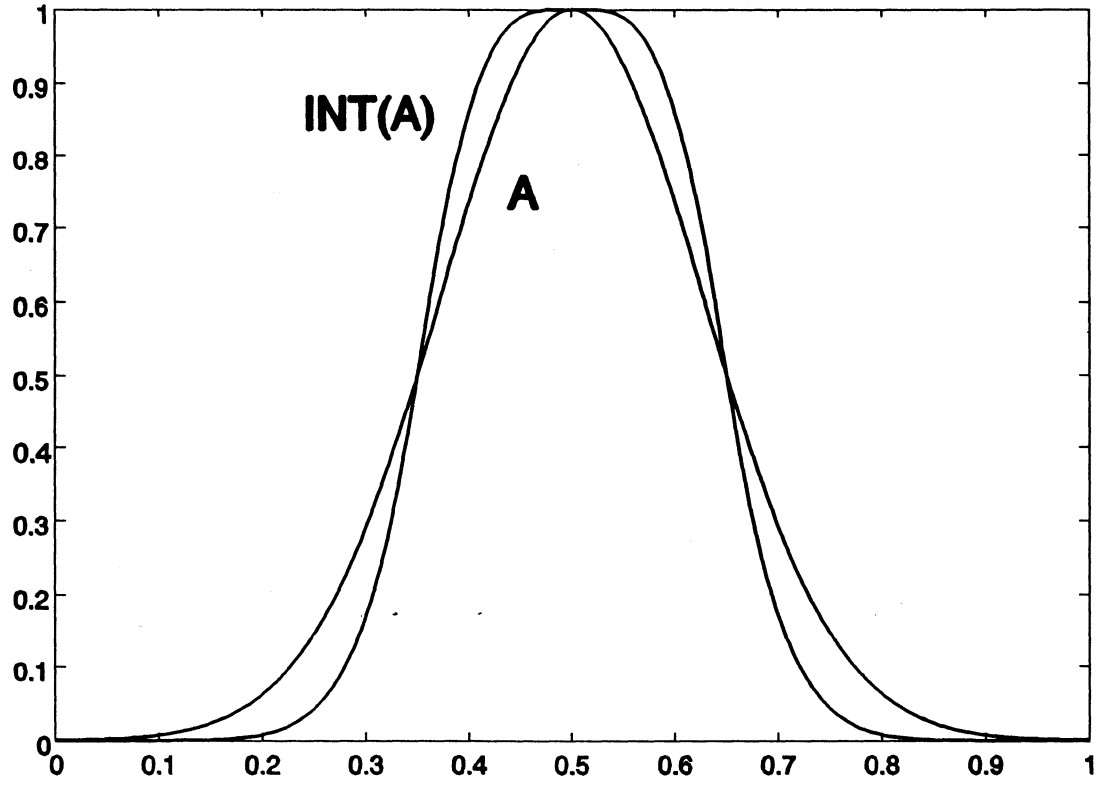
Σχήμα 1.2: Ενωση και τομή ασαφών συνόλων.



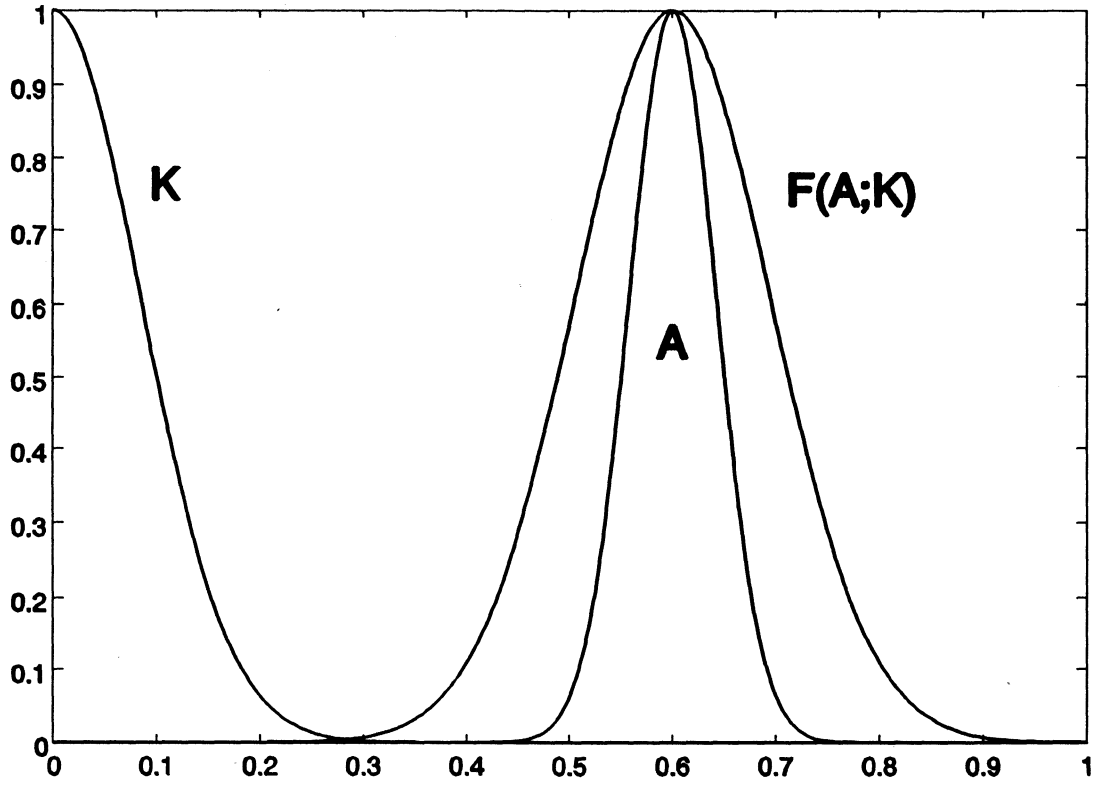
Σχήμα 1.3: Ασαφές συμπλήρωμα.



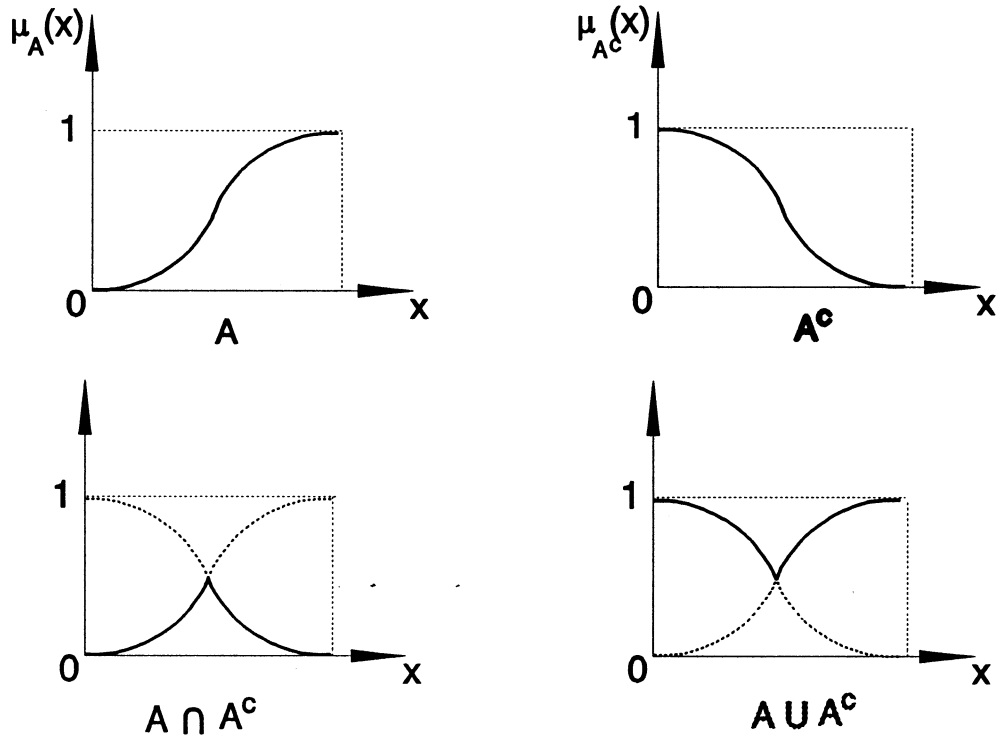
Σχήμα 1.4: Συμπύκνωση και διαστολή ασαφών συνόλων.



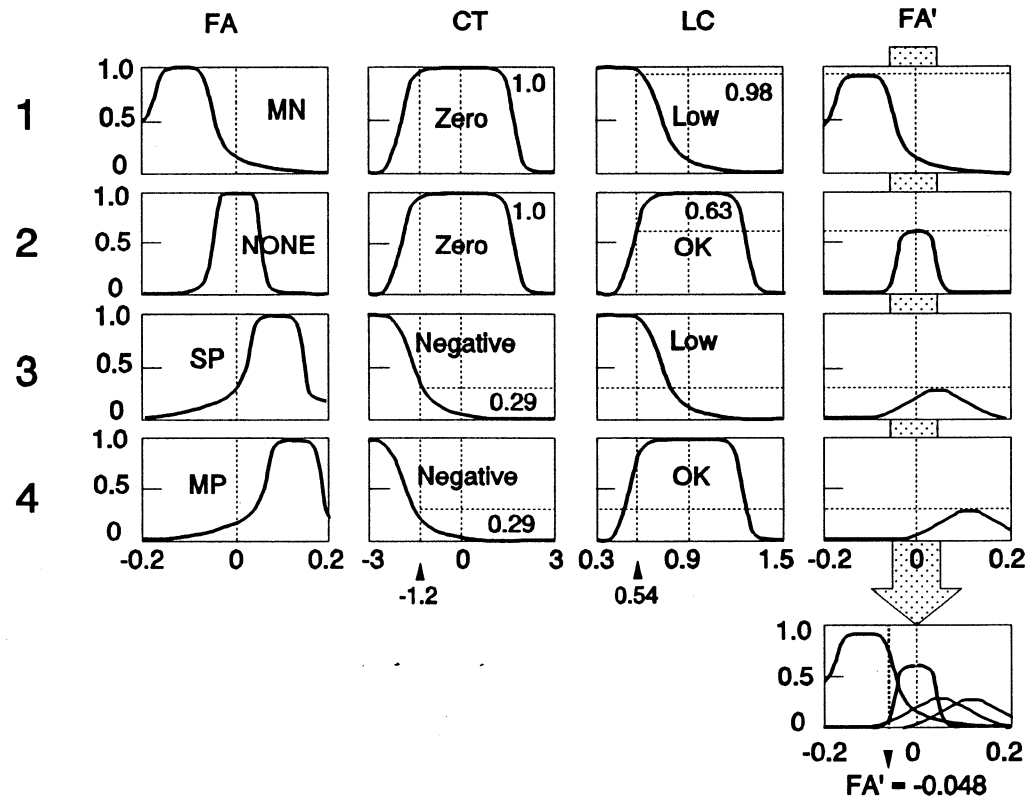
Σχήμα 1.5: Αύξηση αντίθεσης ασαφών συνόλων.



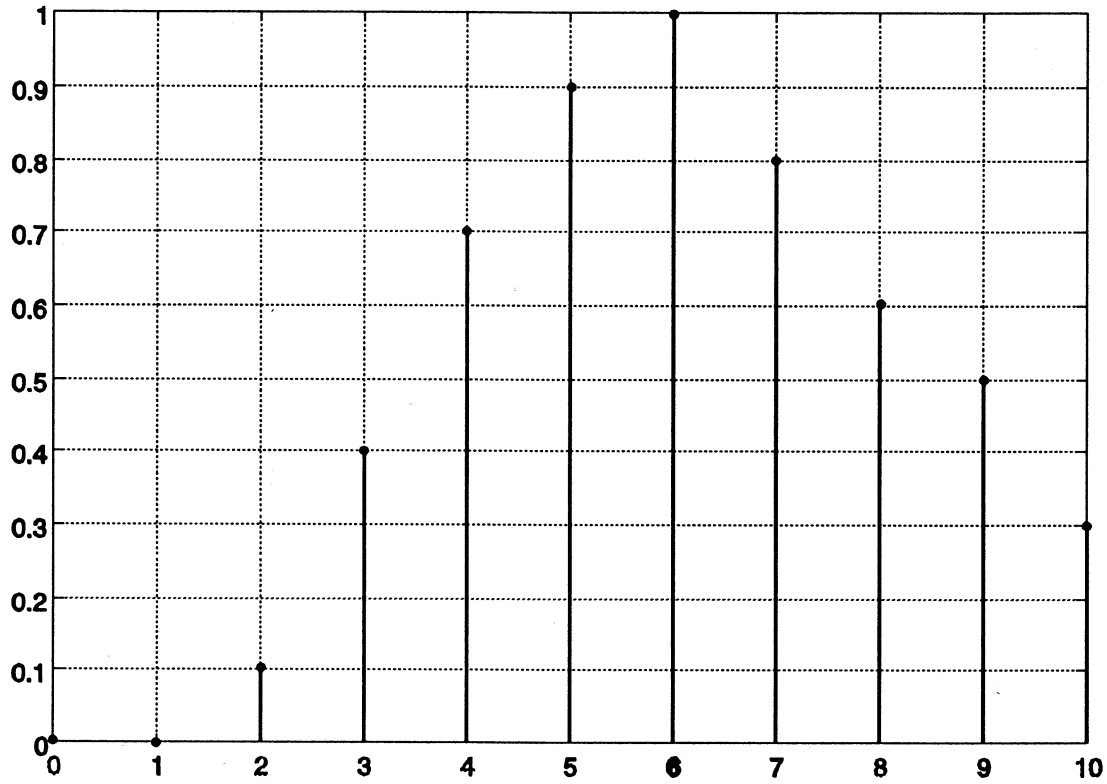
Σχήμα 1.6: Ασαφοποίηση.



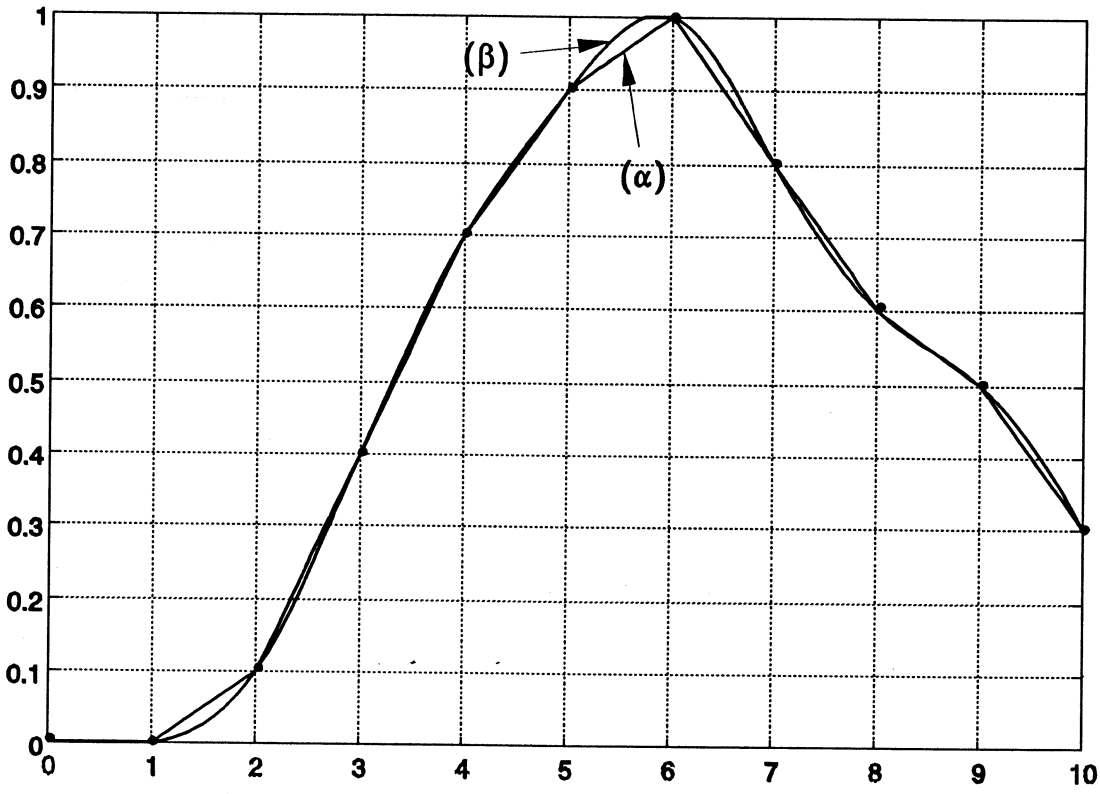
Σχήμα 1.7: Καταπάτηση των δύο νόμων της κλασσικής λογικής.



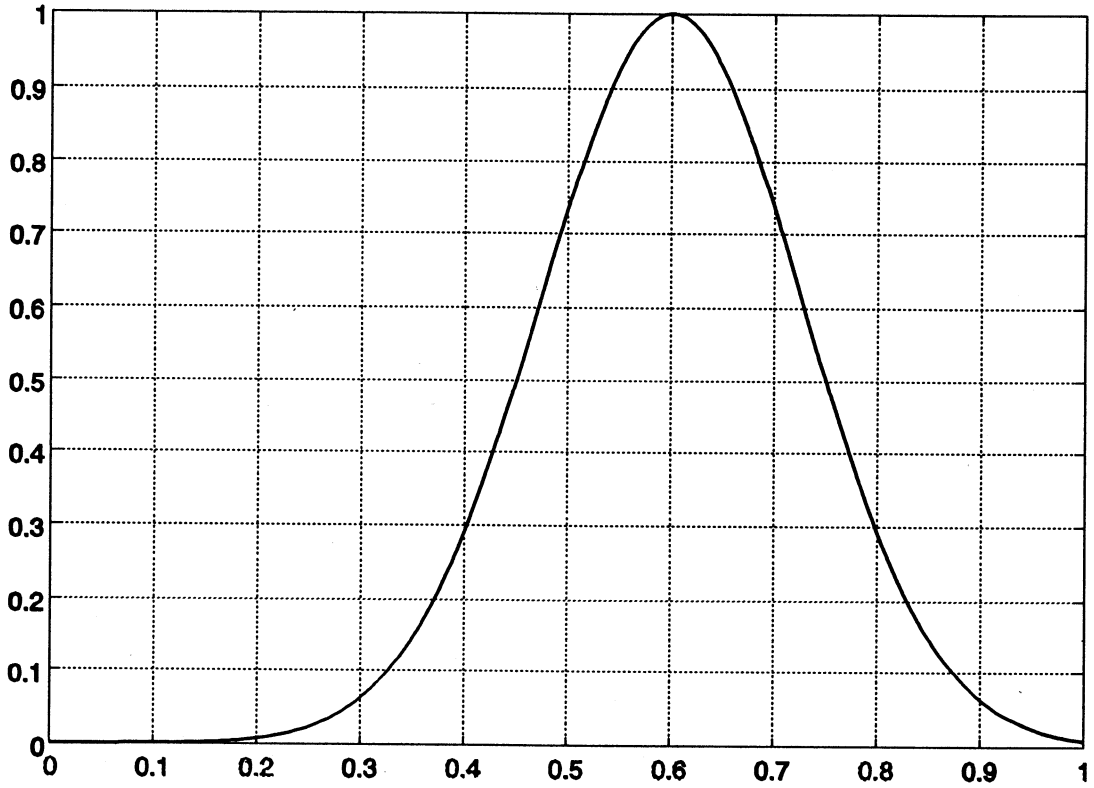
Σχήμα 2.1: Ο τρόπος λειτουργίας ενός ασαφούς συστήματος ελέγχου.



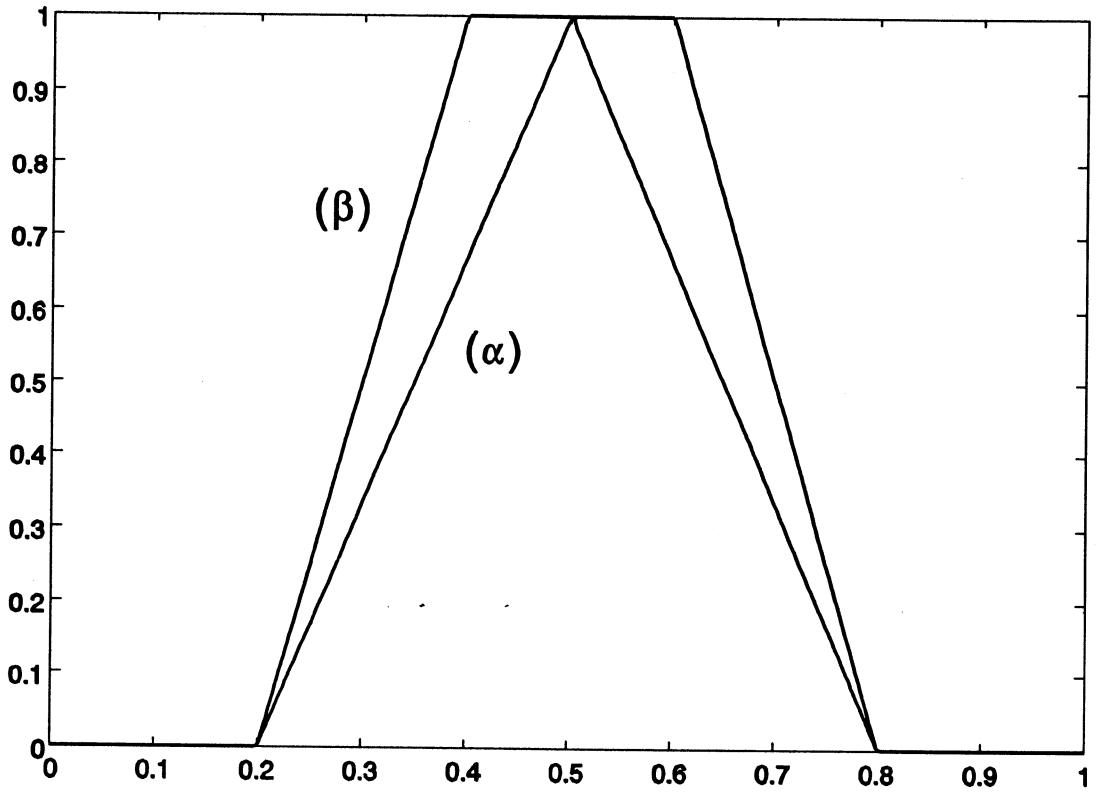
Σχήμα 2.2: Διακριτή συνάρτηση συμμετοχής.



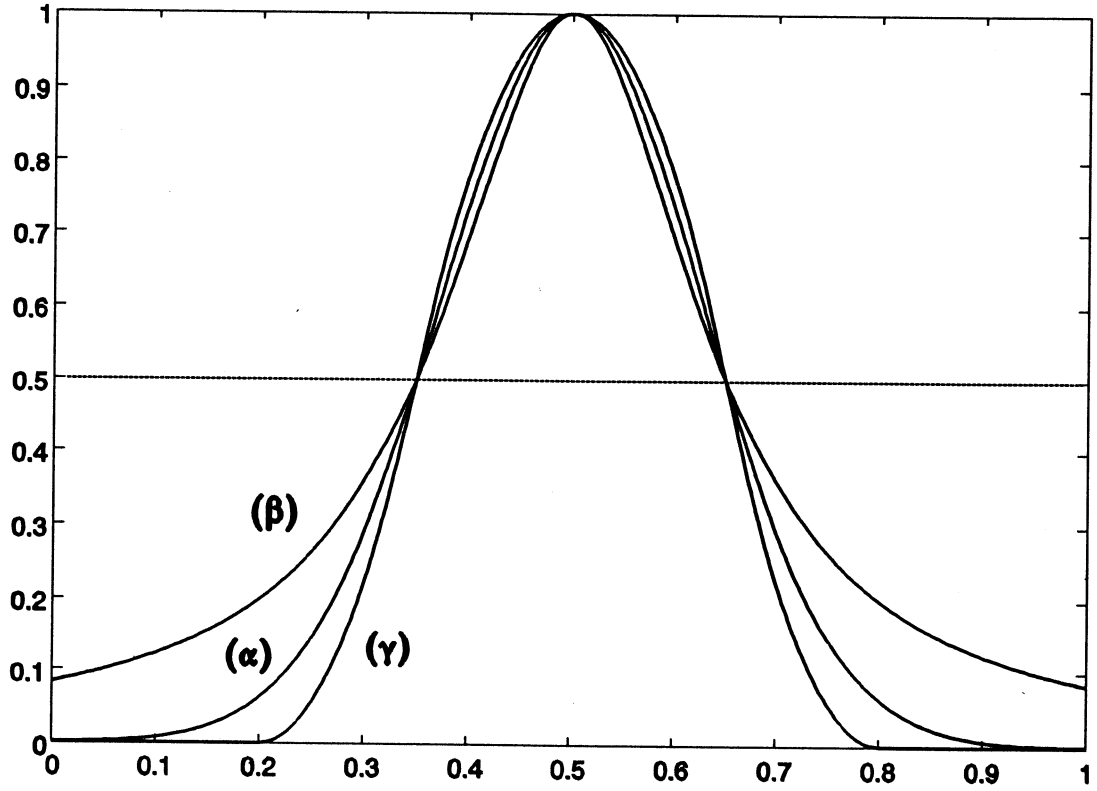
Σχήμα 2.3: Συνεχής συνάρτηση ορισμένη με παρεμβολή. (α) Linear, (β) Biharmonic interpolation.



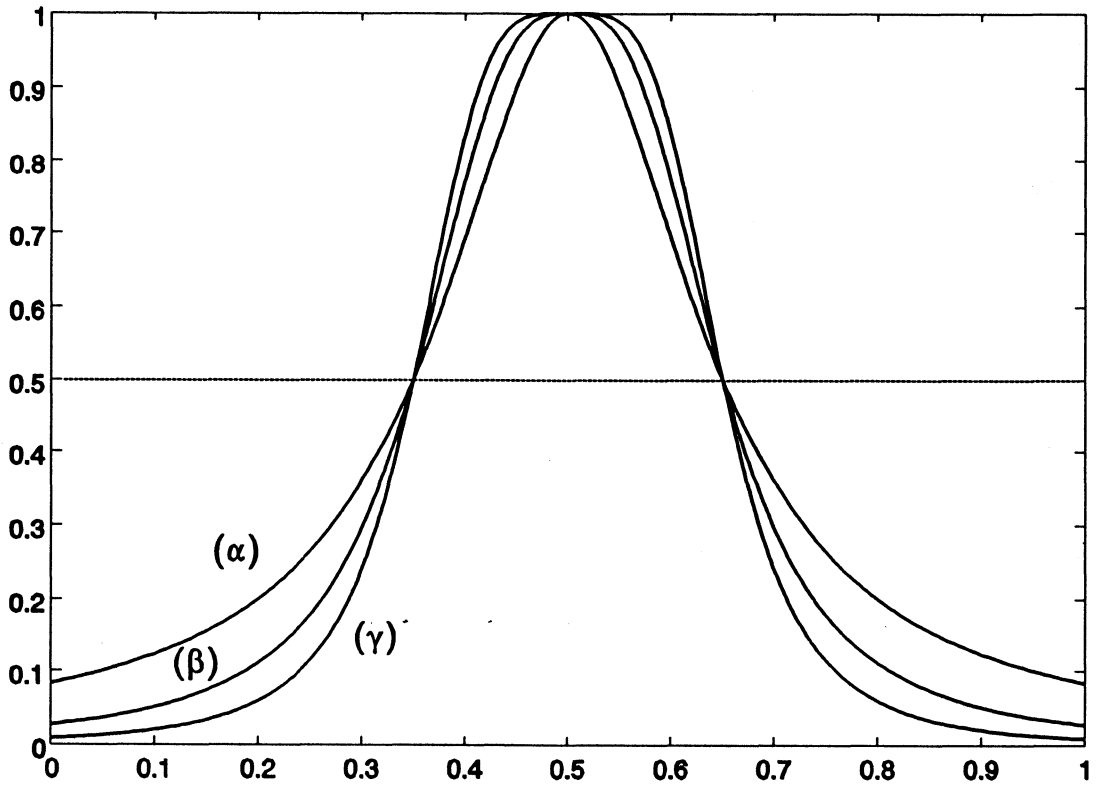
Σχήμα 2.4: Συνεχής συνάρτηση συμμετοχής ορισμένη παραμετρικά.



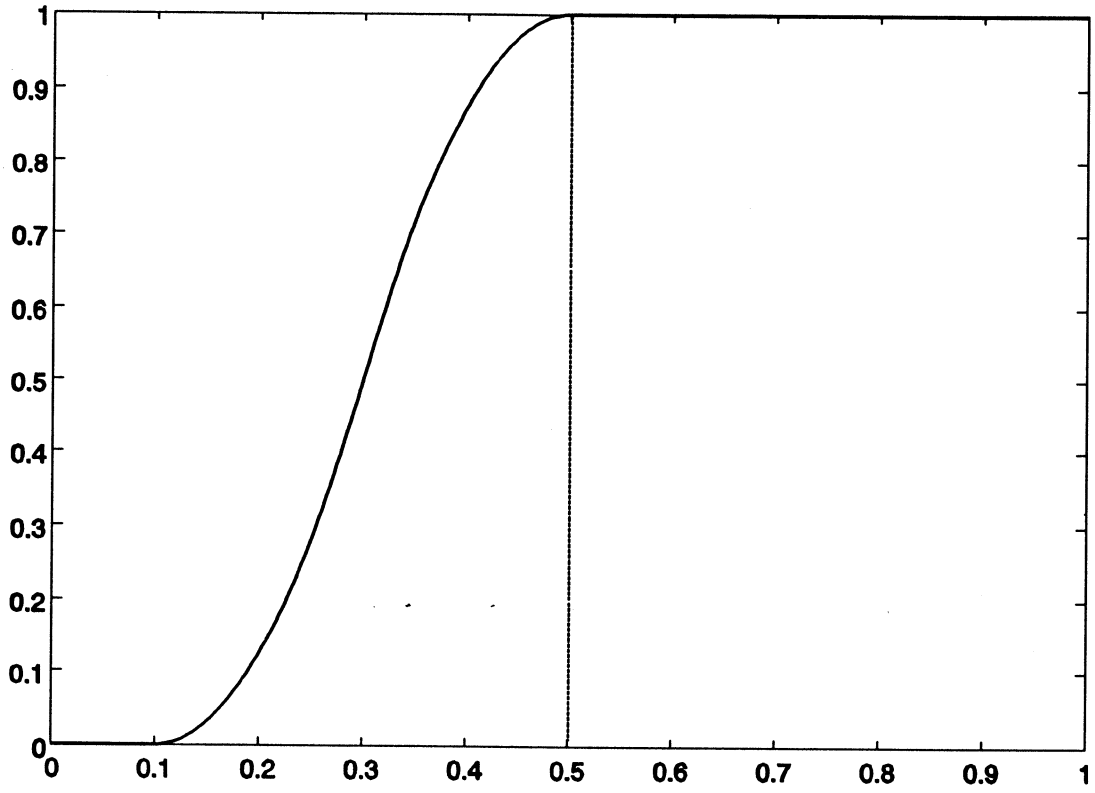
Σχήμα 2.5: Συναρτήσεις triang (α) & trapez (β) για a=0.5, s=0.3, p=0.2



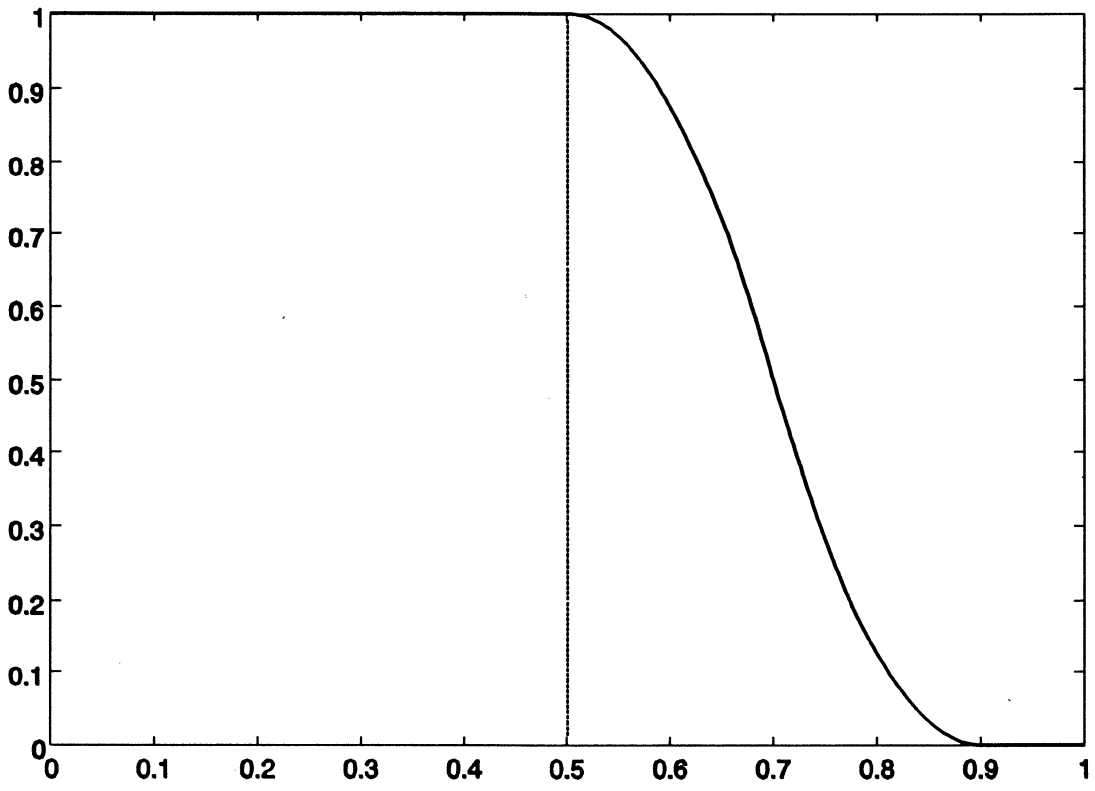
Σχήμα 2.6: Συναρτήσεις gauss (α), lorentz (β) & Π (γ) για a=0.5, s=0.3



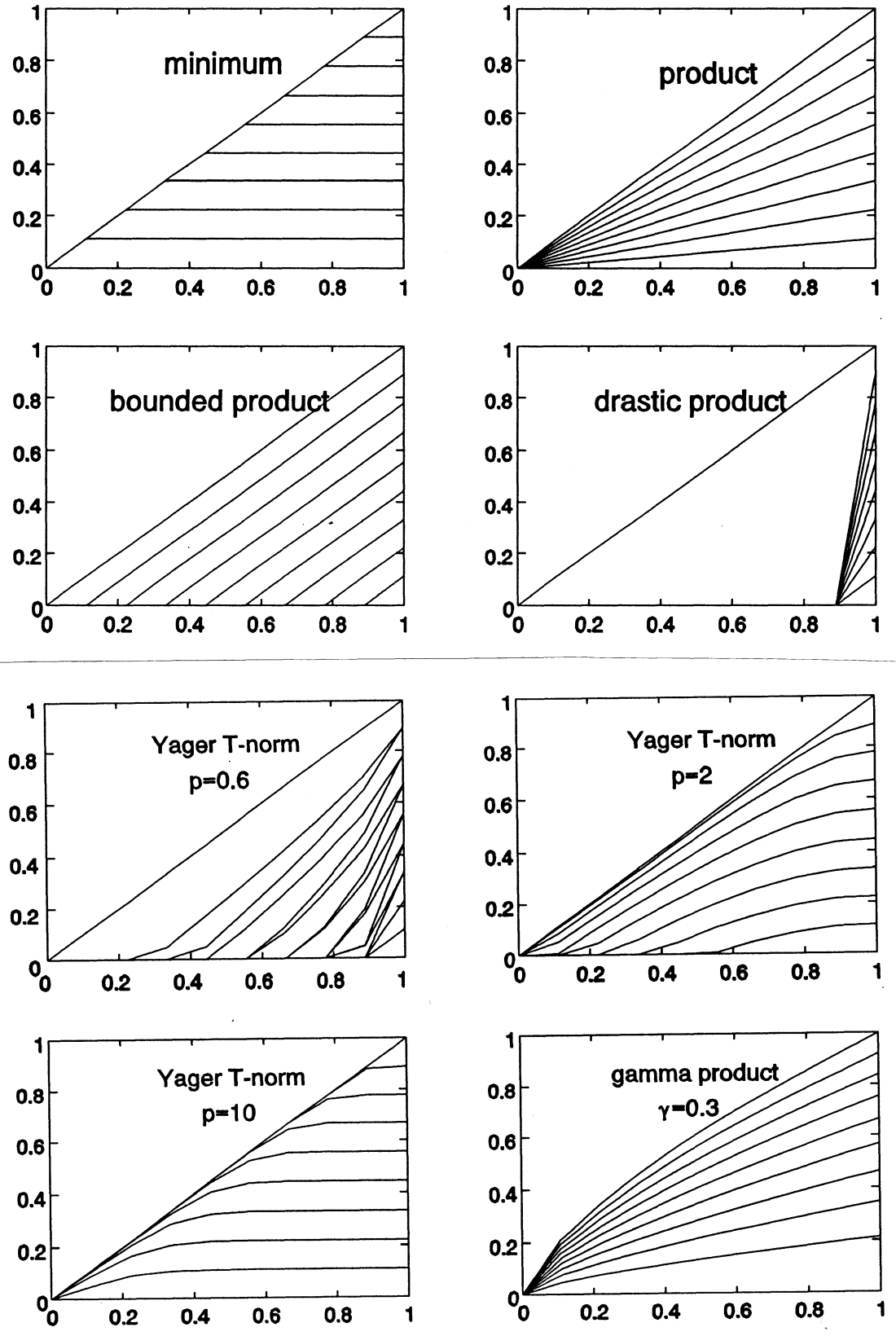
Σχήμα 2.7: Γενικευμένη συνάρτηση lorentz για a=0.5, s=0.3, p=1.6 (α), p=2 (β) και p=4 (γ).



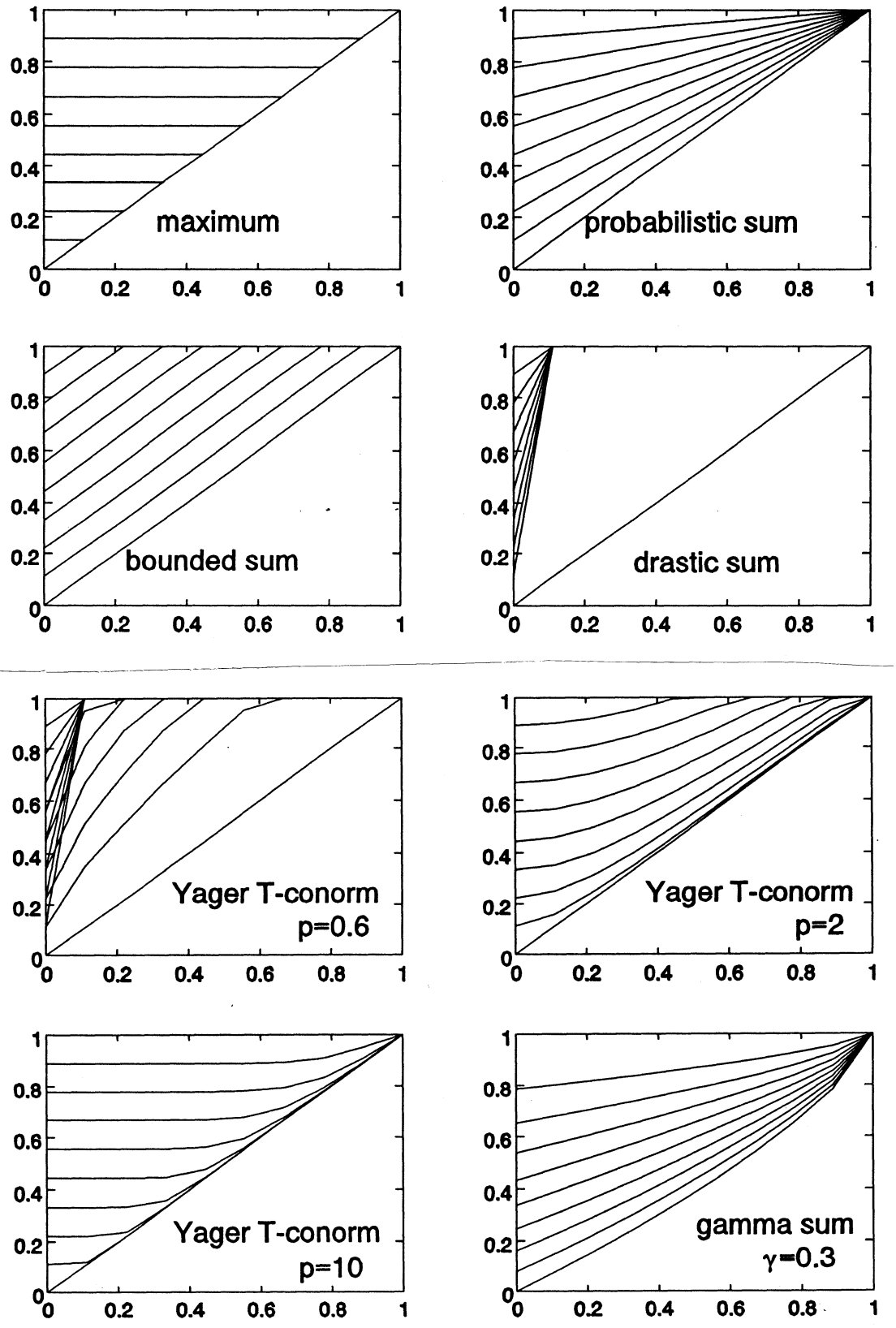
Σχήμα 2.8: Η συνάρτηση S για a=0.5, s=0.4



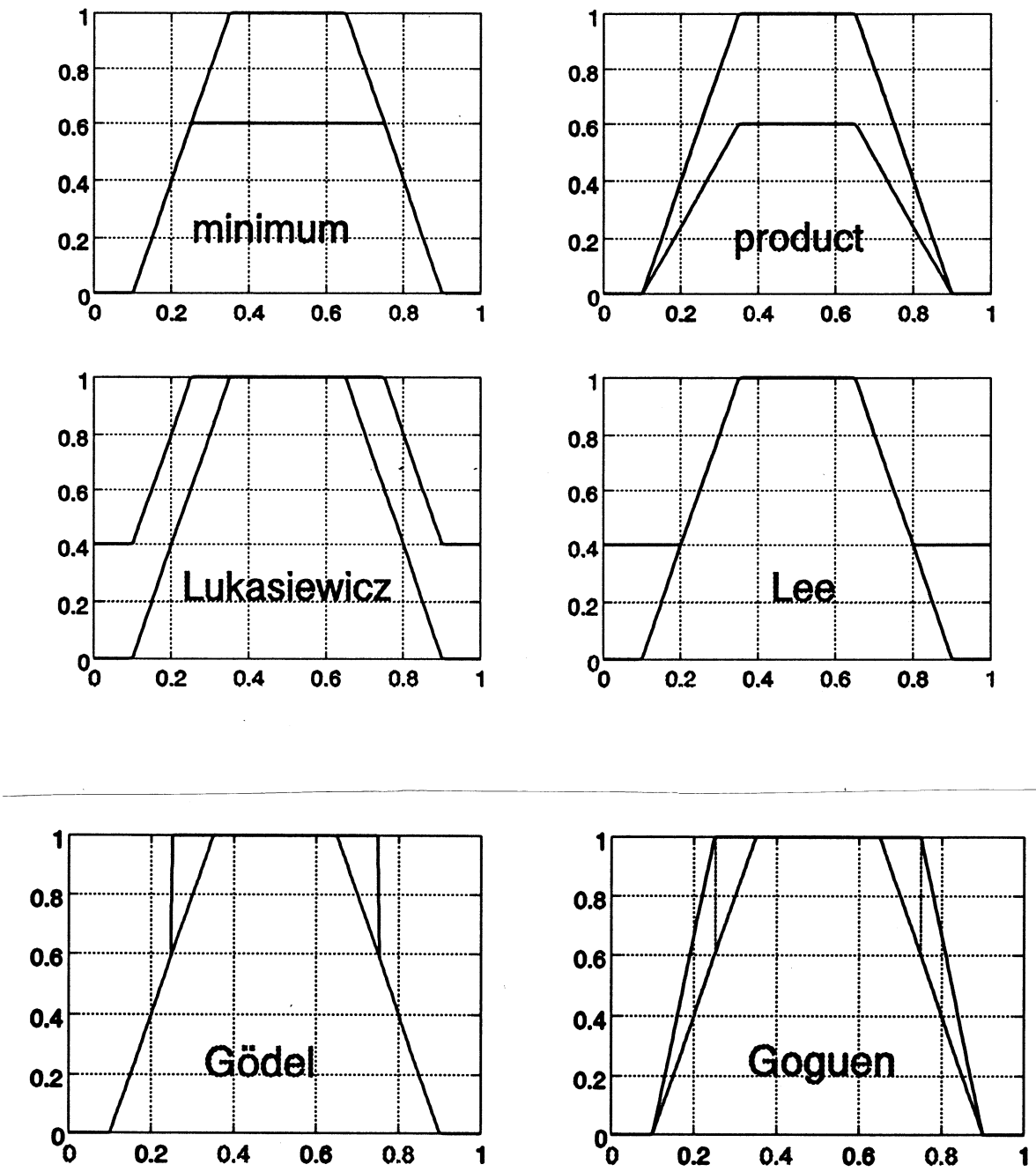
Σχήμα 2.9: Η συνάρτηση Z για a=0.5, s=0.4



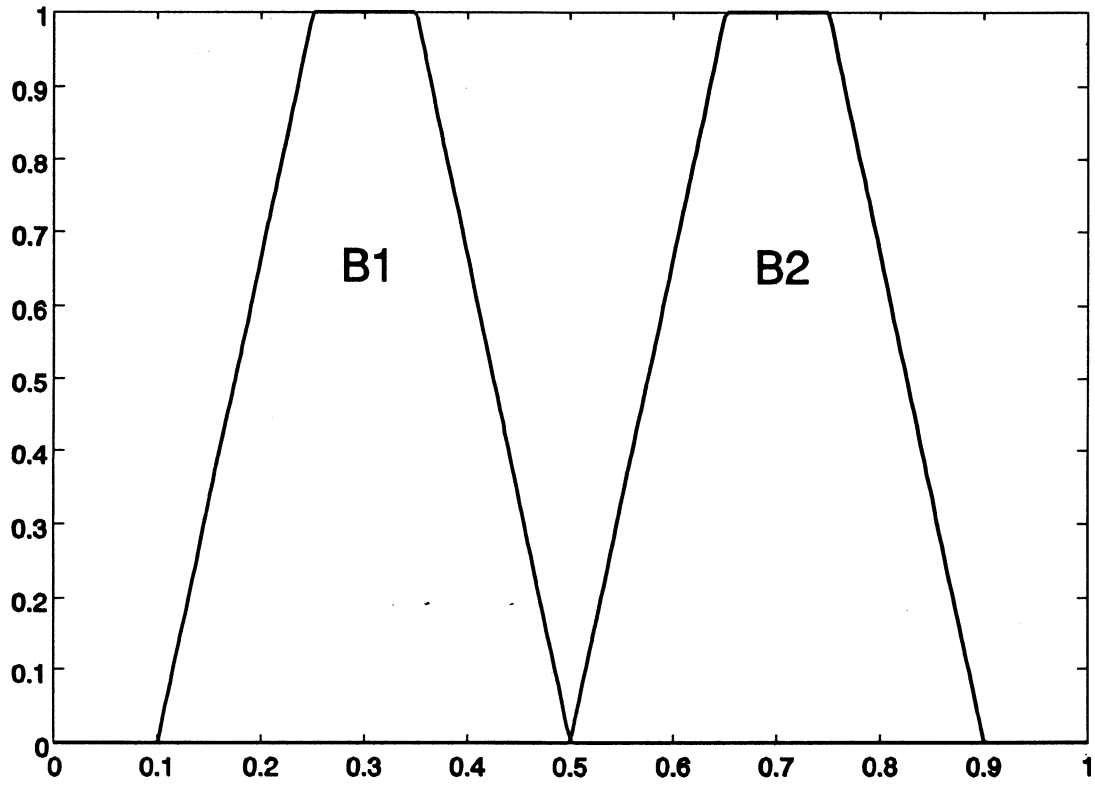
Σχήμα 2.10: Τρόποι υλοποίησης της πράξης AND.



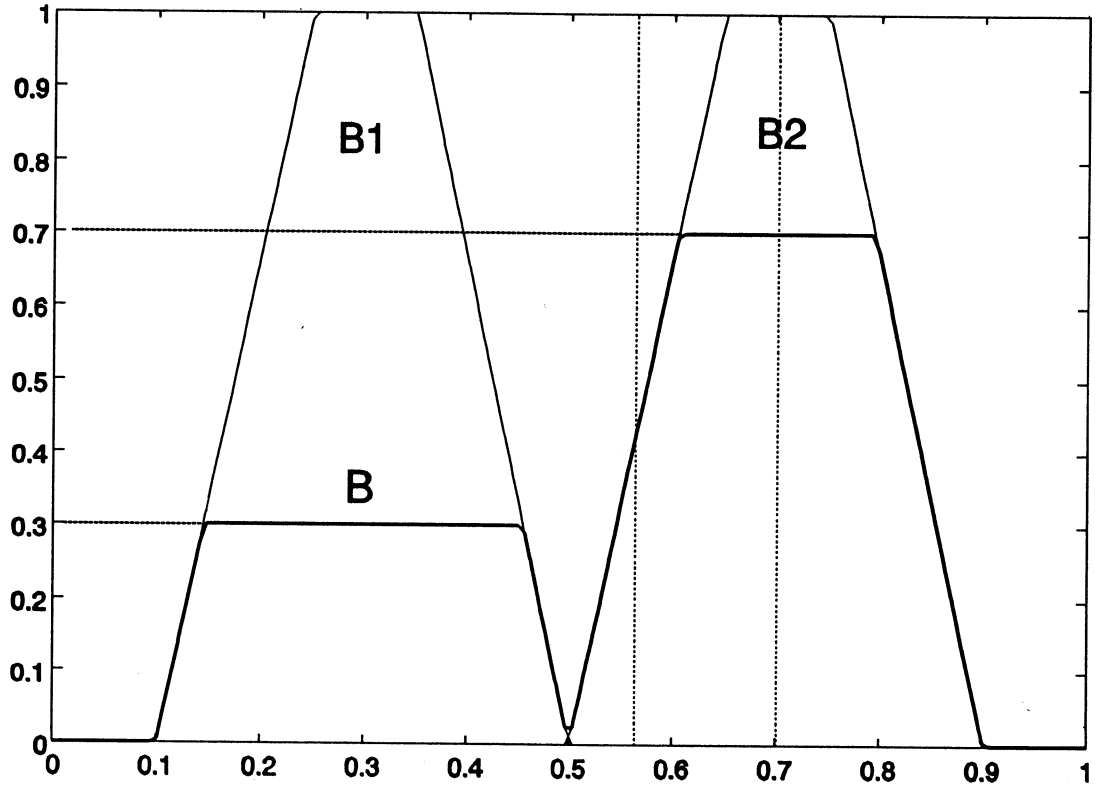
Σχήμα 2.11: Τρόποι υλοποίησης της πράξης OR.



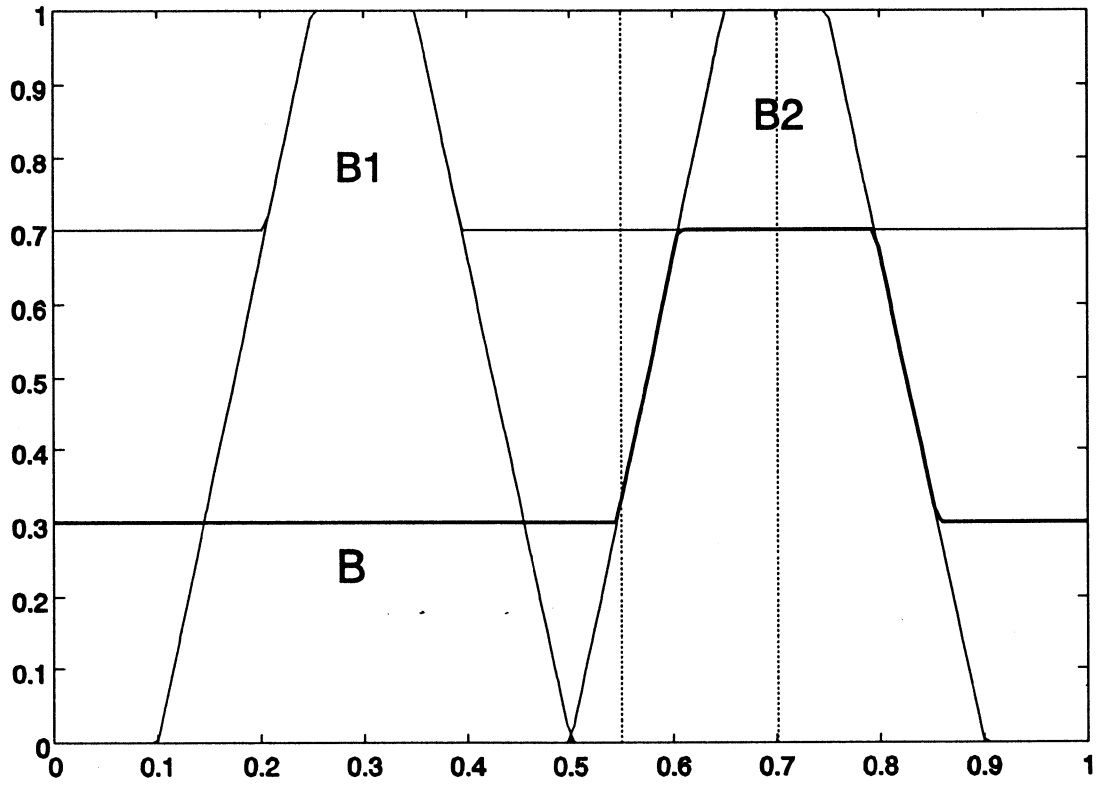
Σχήμα 2.12: Σύγκριση των διάφορων τύπων implication.



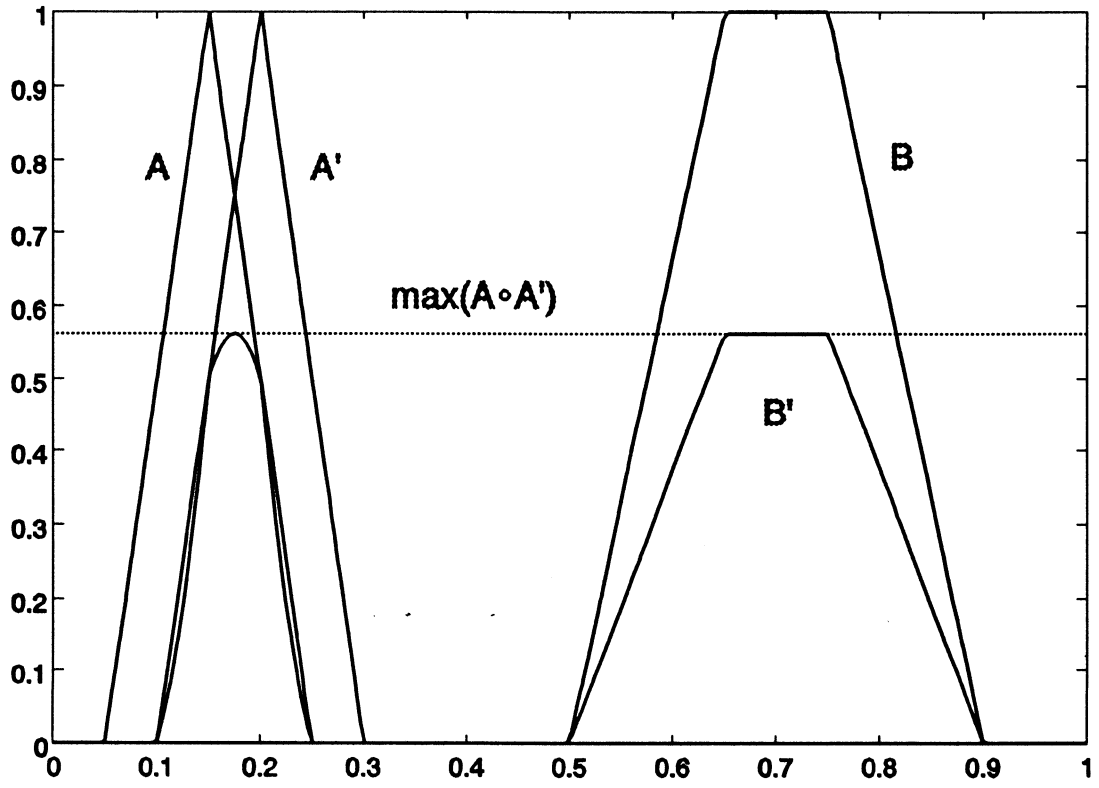
Σχήμα 2.13: Τα δύο ασαφή σύνολα B1 και B2.



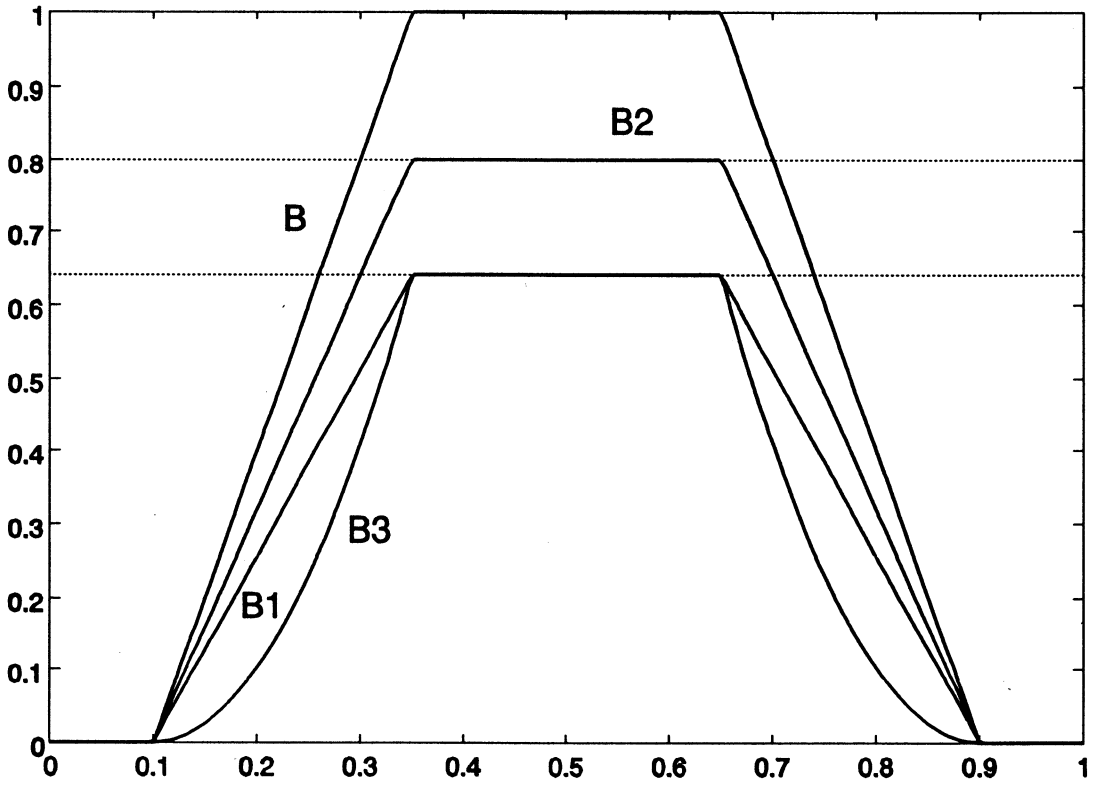
Σχήμα 2.14: Implication: minimum, aggregation: OR (maximum).



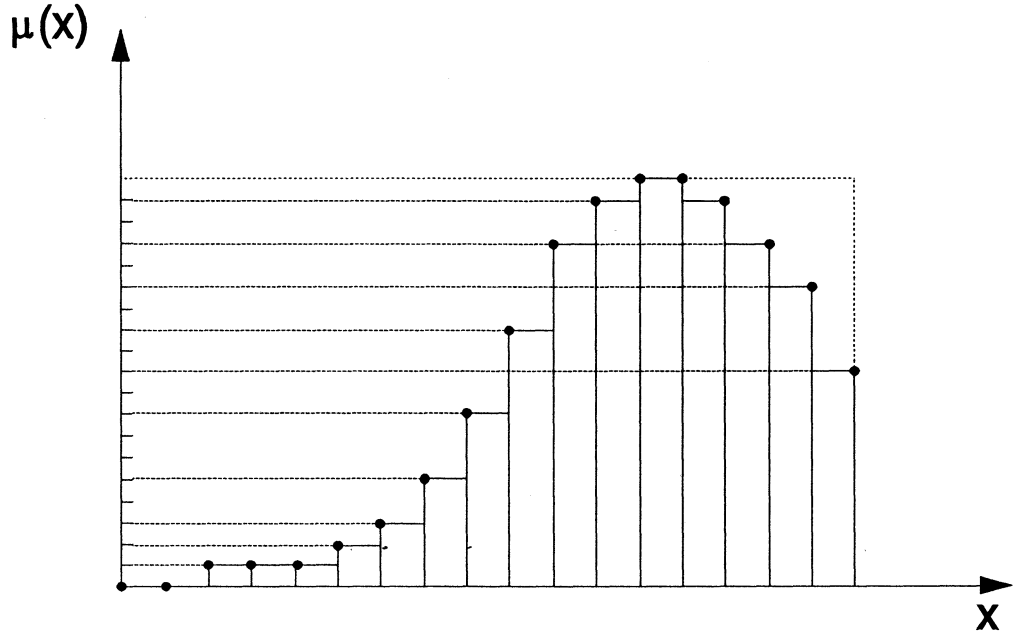
Σχήμα 2.15: Implication: Lee, aggregation: AND (minimum).



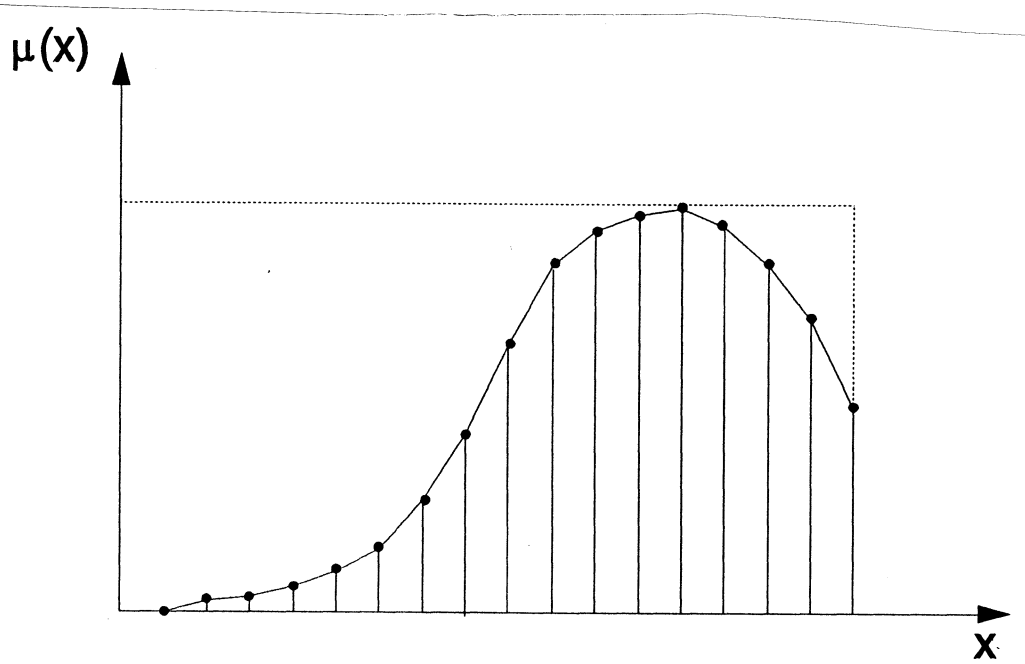
Σχήμα 3.1: Απεικόνιση της μεθόδου που χρησιμοποιείται στο μοντέλο II.



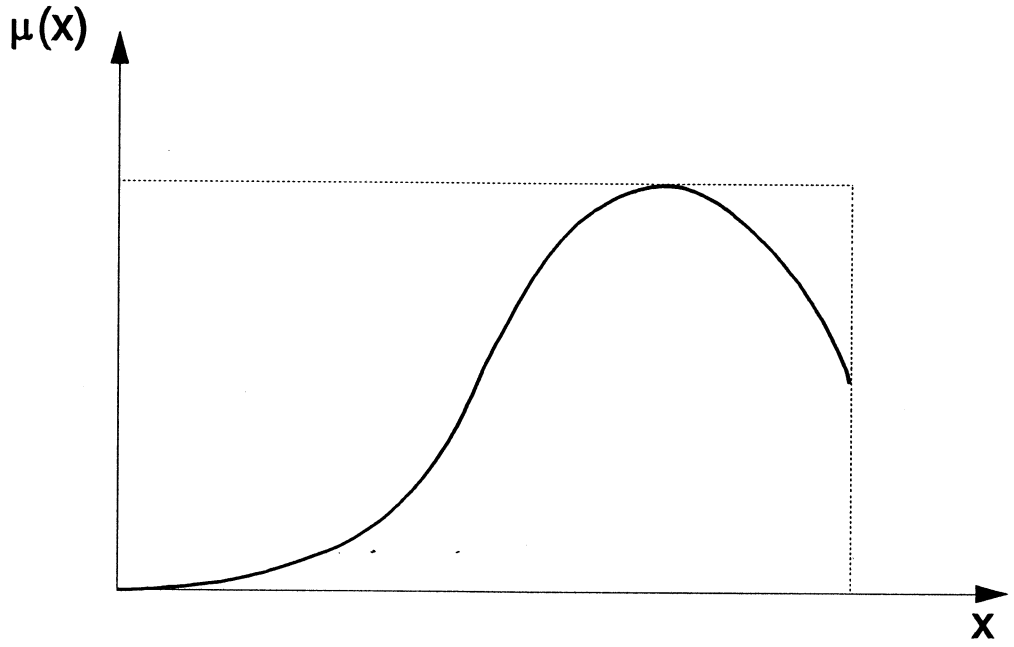
Σχήμα 3.2: Σφάλμα από την εφαρμογή της μεθόδου του "σπασίματος".



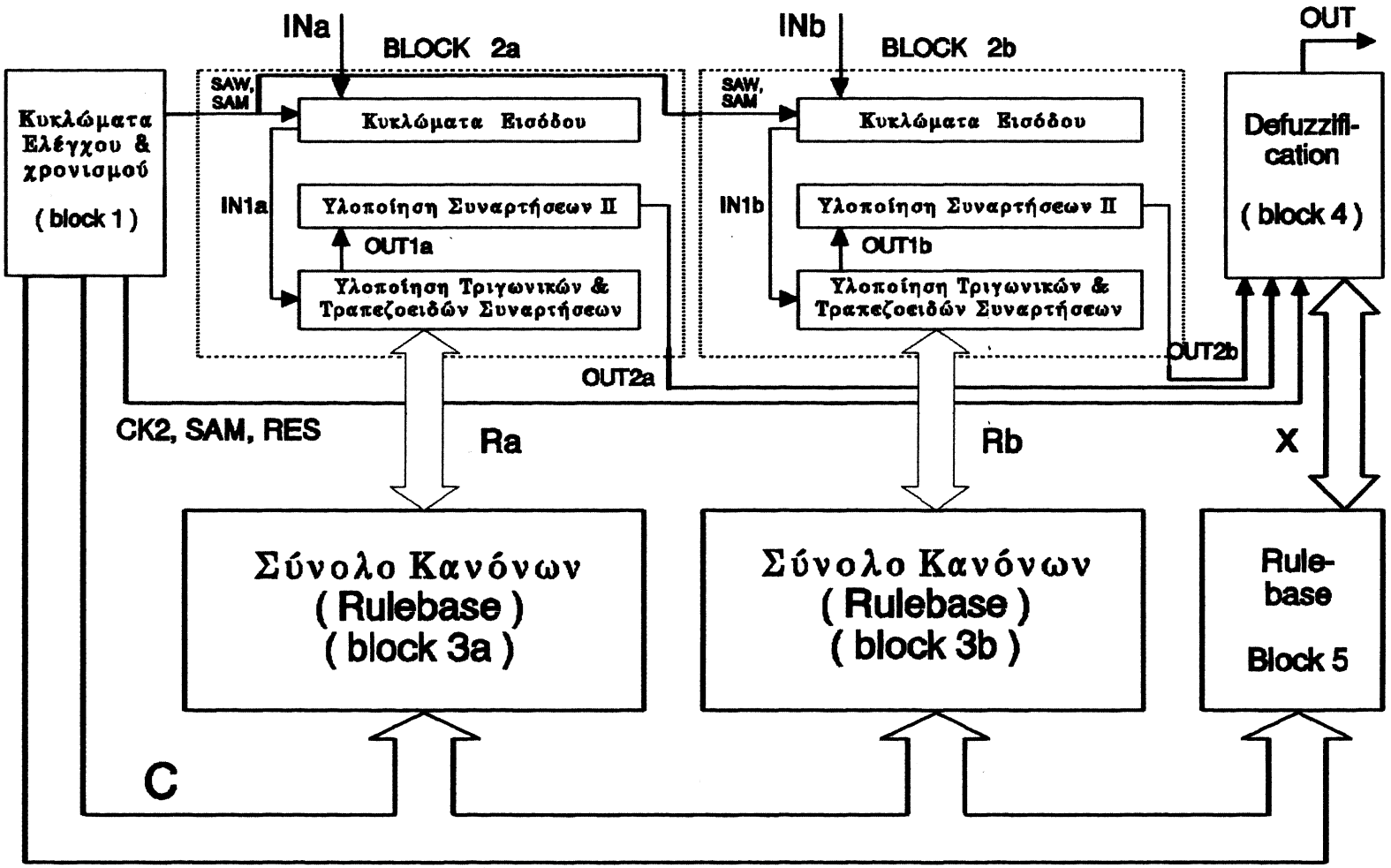
Σχήμα 4.1: Διακριτοποίηση και στους δύο άξονες.



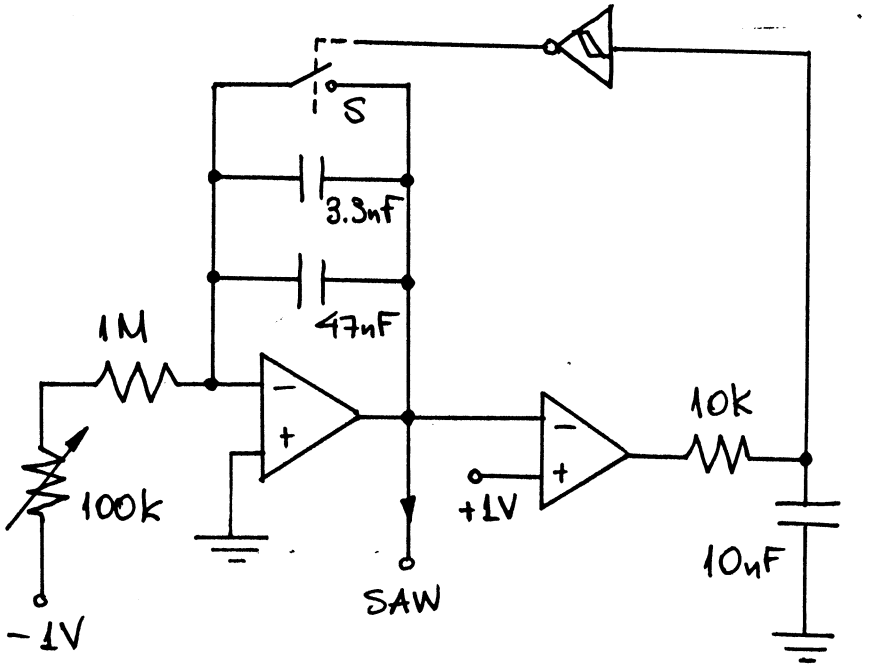
Σχήμα 4.2: Διακριτοποίηση στον άξονα των x μόνο.



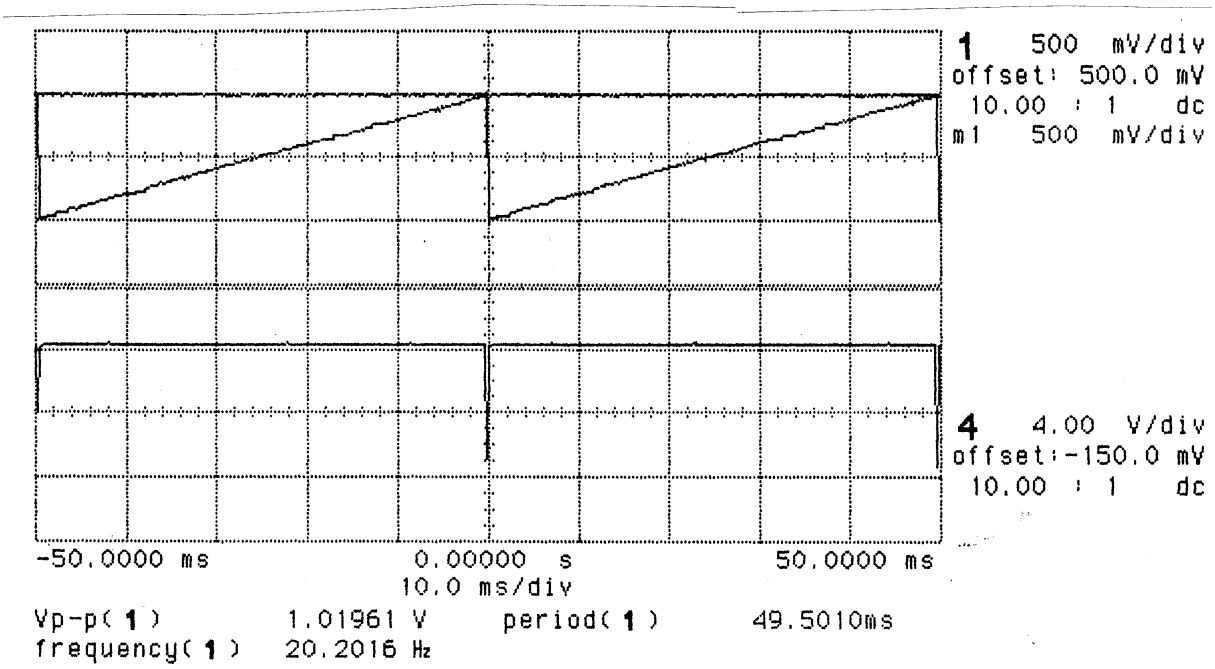
Σχήμα 4.3: Πλήρης αναλογική υλοποίηση.



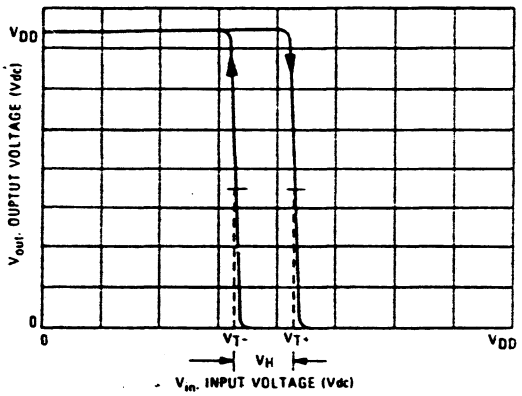
Σχήμα 4.4 : Η αρχιτεκτονική του επεξεργαστή



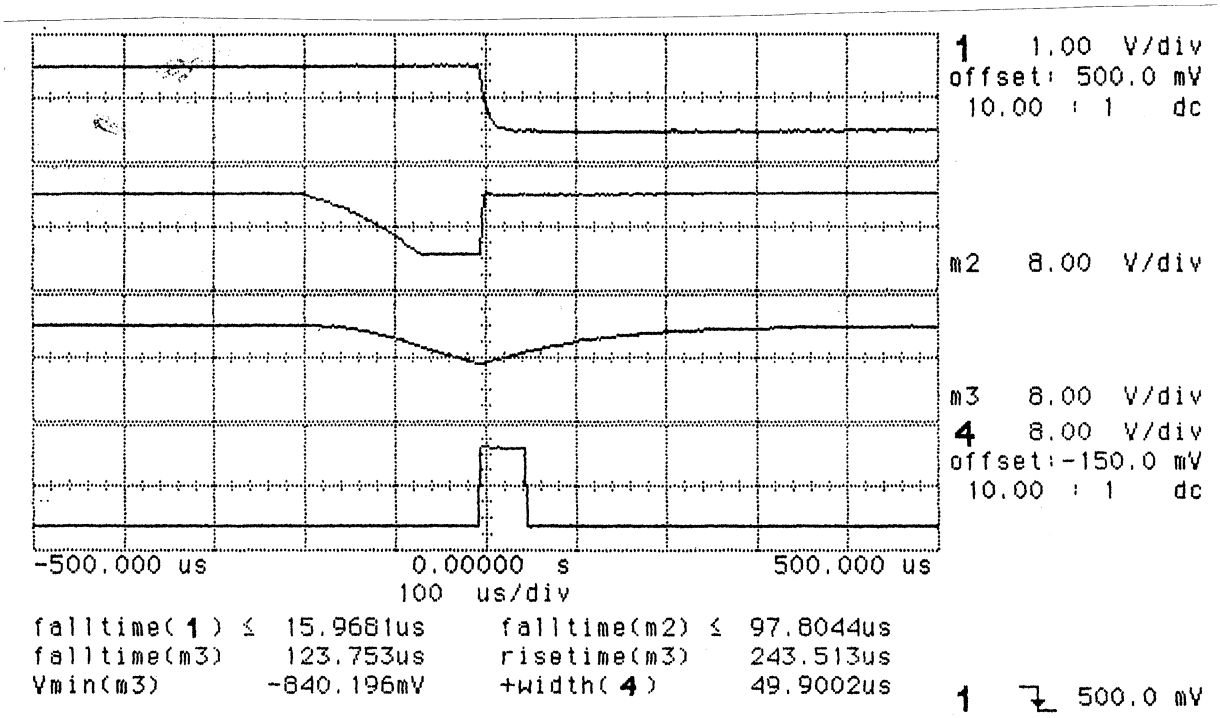
Σχήμα 4.5: Το αναλογικό μέρος του block 1.



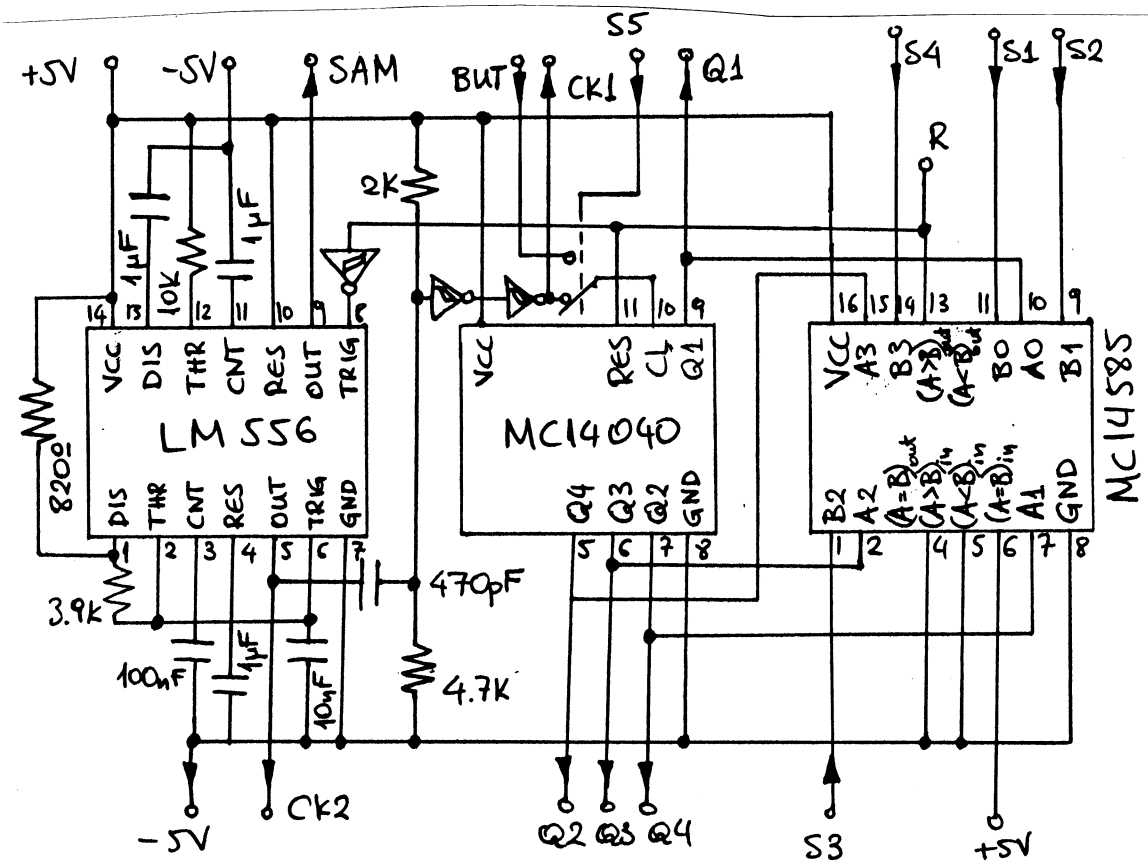
Σχήμα 4.6: (1) Η πριονωτή κυματομορφή SAW, (4) Η έξοδος του συγκριτή.



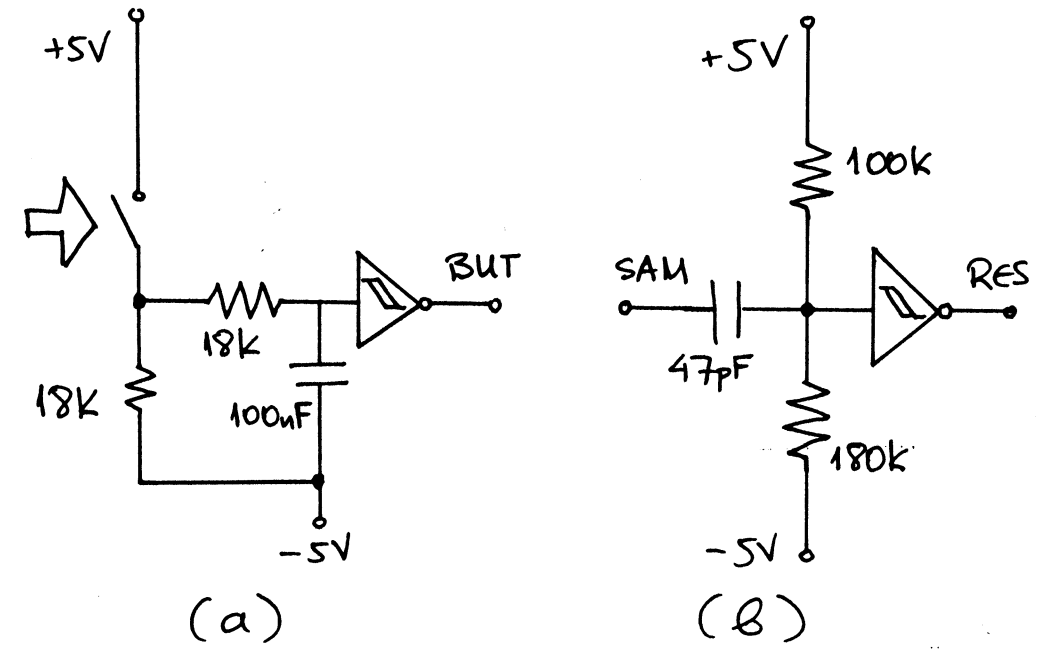
Σχήμα 4.7: Χαρακτηριστική μεταφοράς του Schmitt trigger.



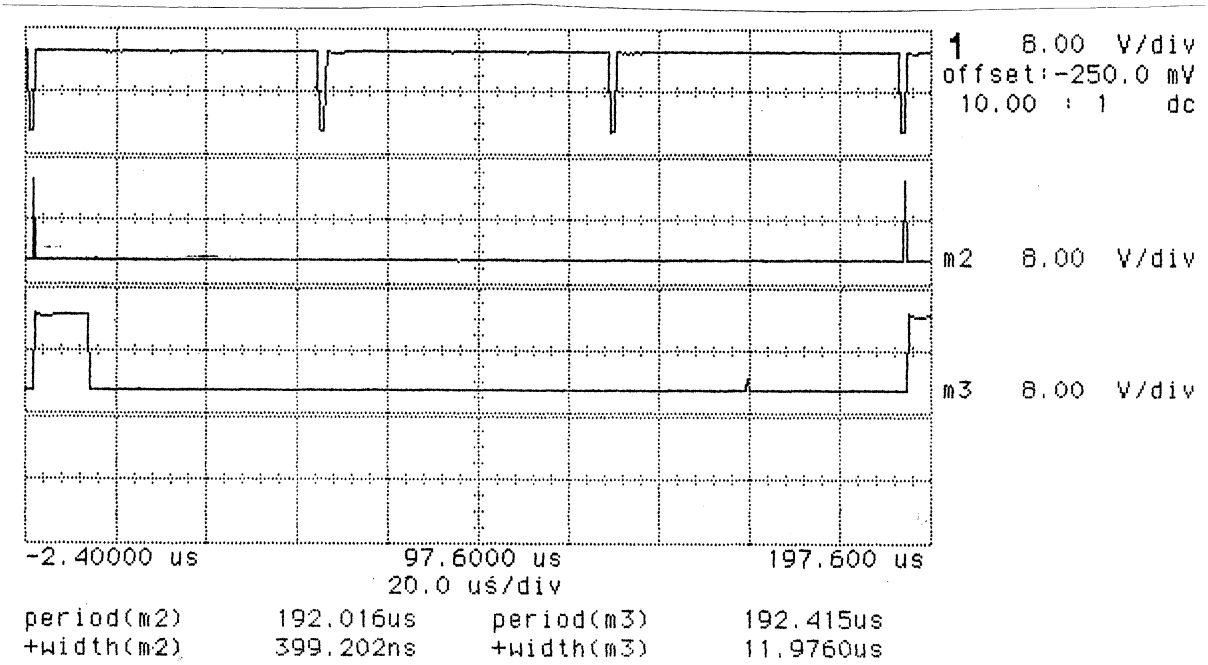
Σχήμα 4.8: (1) Κυματομορφή SAW, (m2) Εξοδος συγκριτή, (m3-4) Είσοδος-έξοδος Schmitt trigger



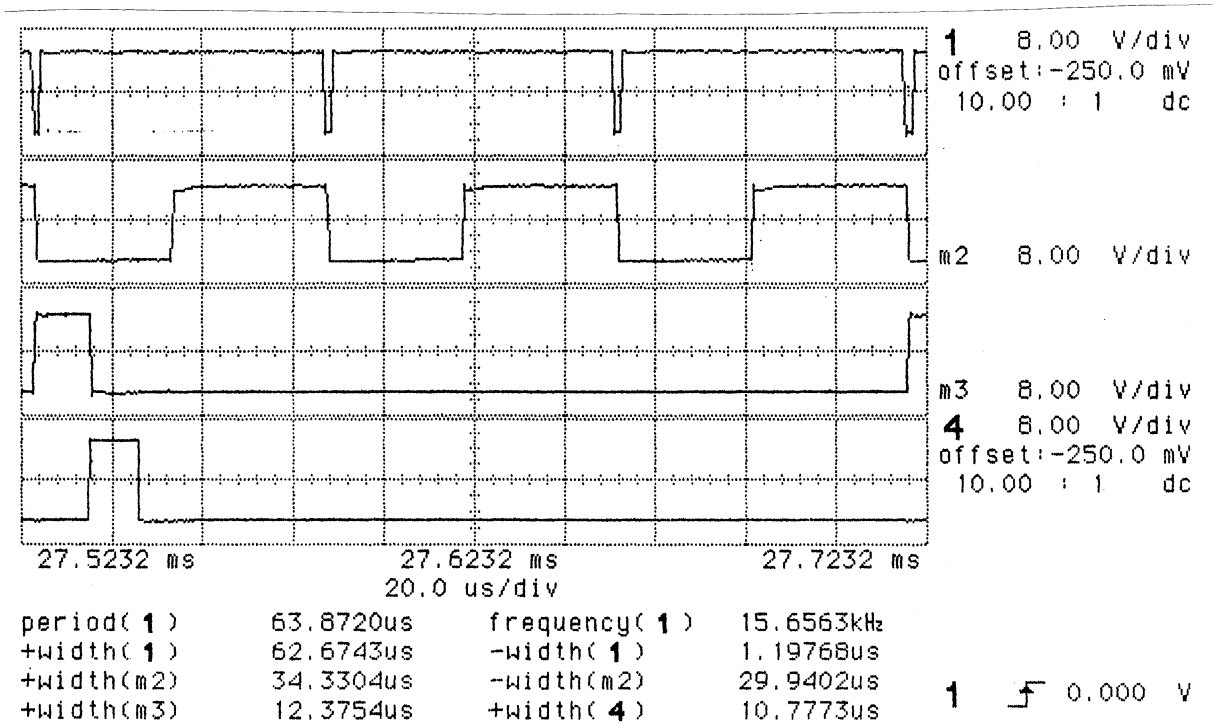
Σχήμα 4.9: Το ψηφιακό τμήμα του block 1.



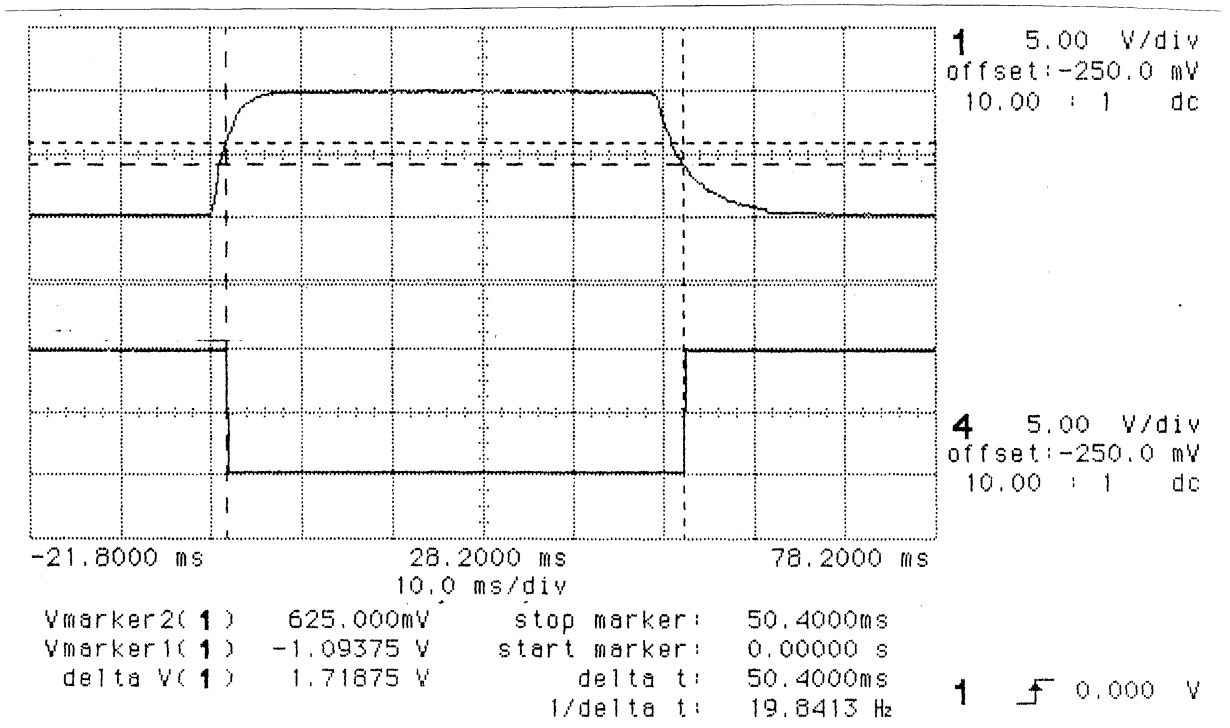
Σχήμα 4.10: (α) Κύκλωμα καθαρισμού push-button, (β) Σήματα Sample & Reset.



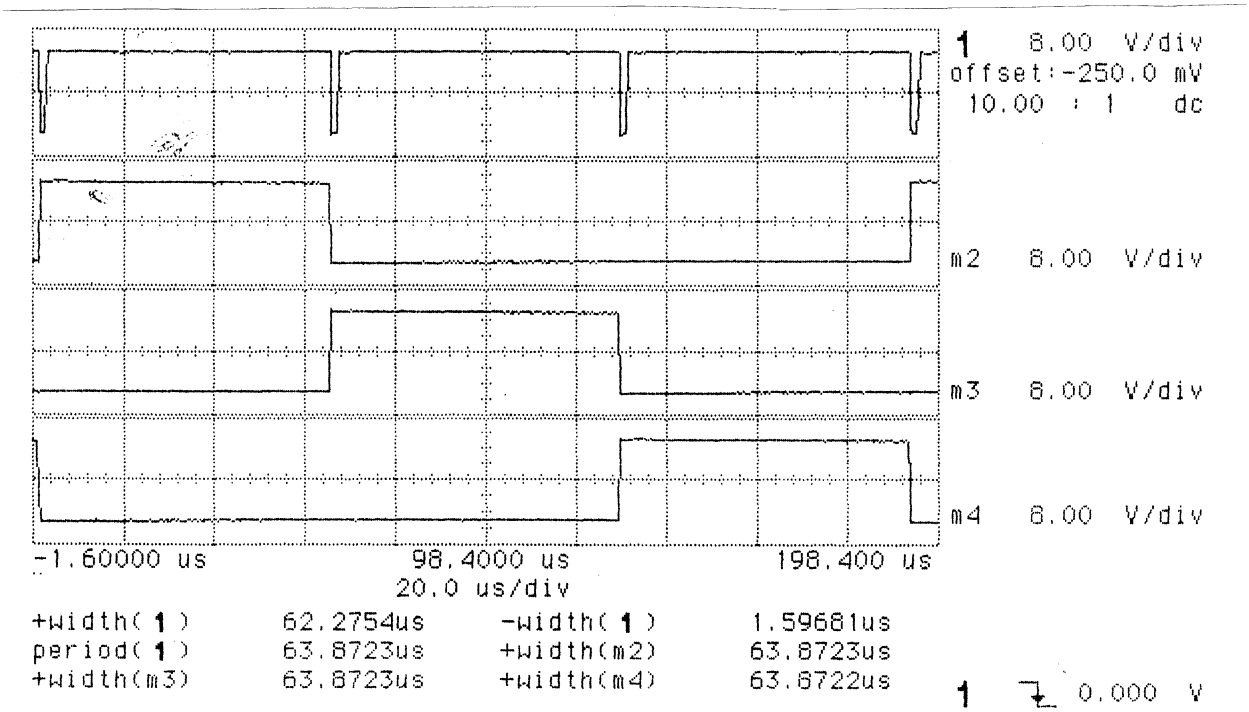
Σχήμα 4.11α: (1) Ρολόι CK1, (m2) Παλμός μηδενισμού R, (m3) Σήμα δειγματοληψίας SAM



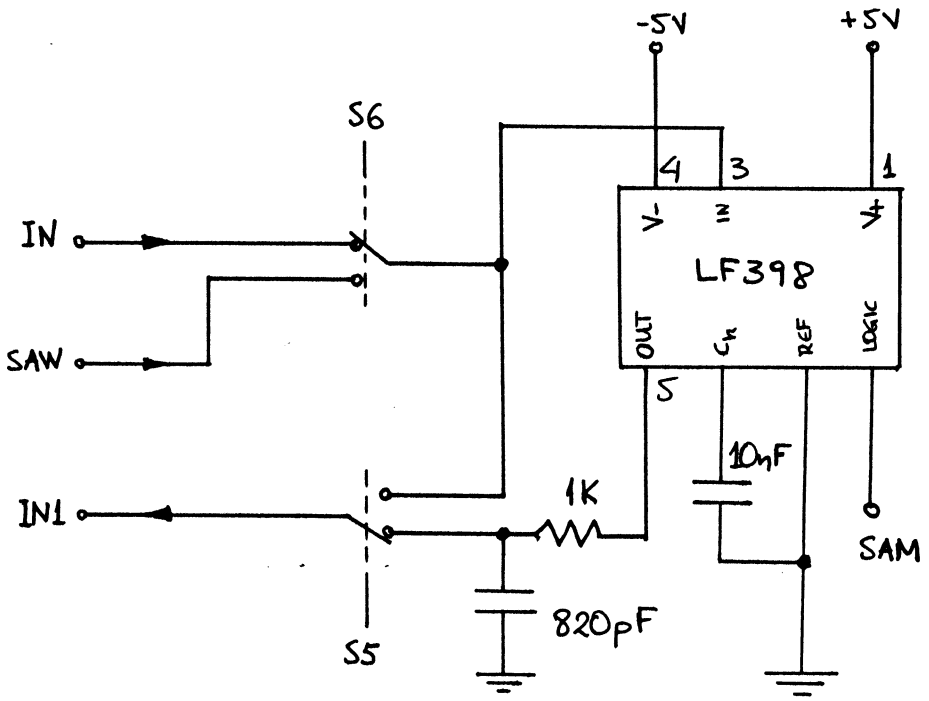
Σχήμα 4.11β: (1) CK1, (m2) CK2, (m3) SAM, (4) Σήμα μηδενισμού RES



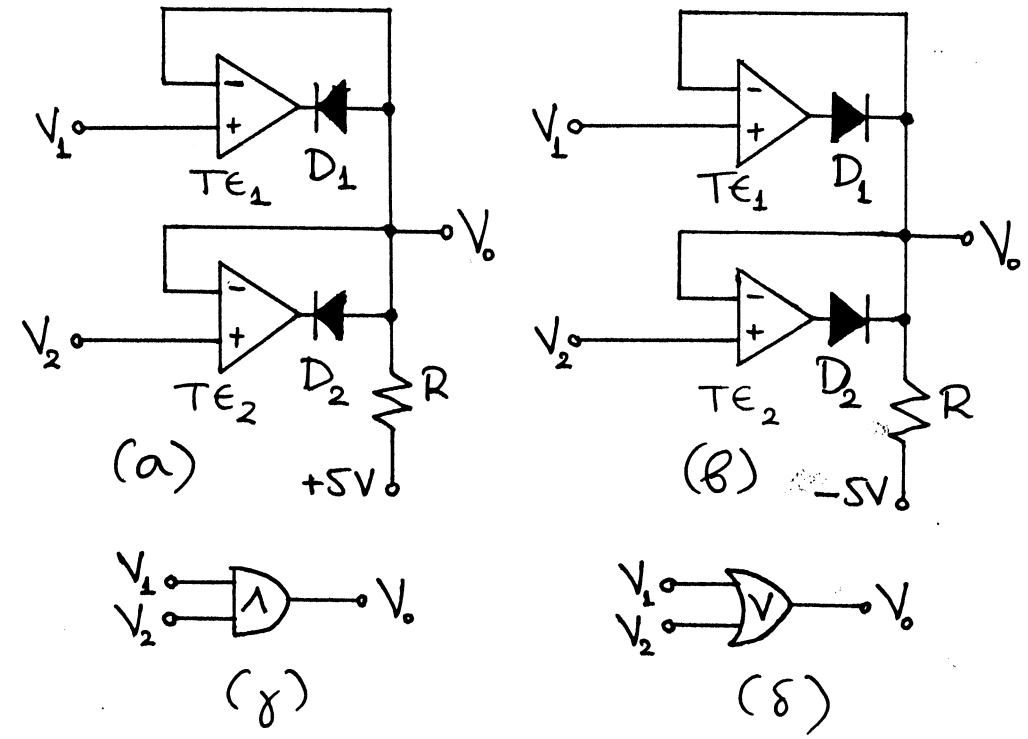
Σχήμα 4.12: Καθαρισμός του push-button. (1) Είσοδος, και (4) Εξοδος του Schmitt trigger.



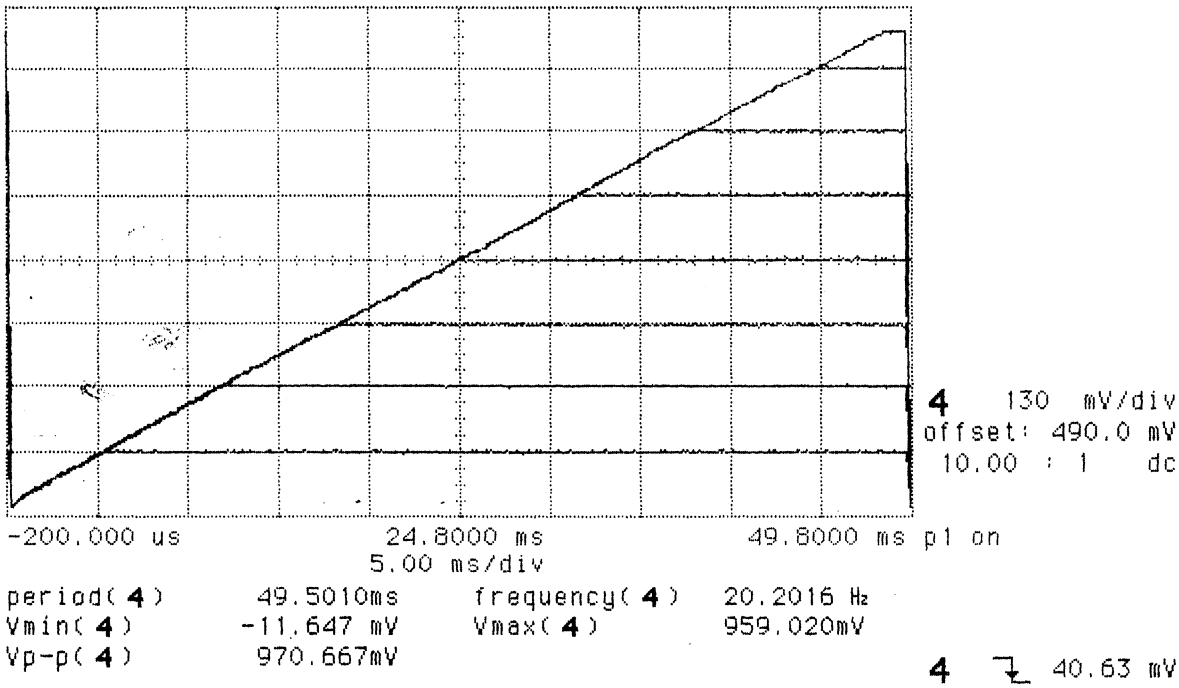
Σχήμα 4.13: (1) Ρολόι CK1, (m2,m3,m4) Σήματα ελέγχου C_1, C_2, C_3.



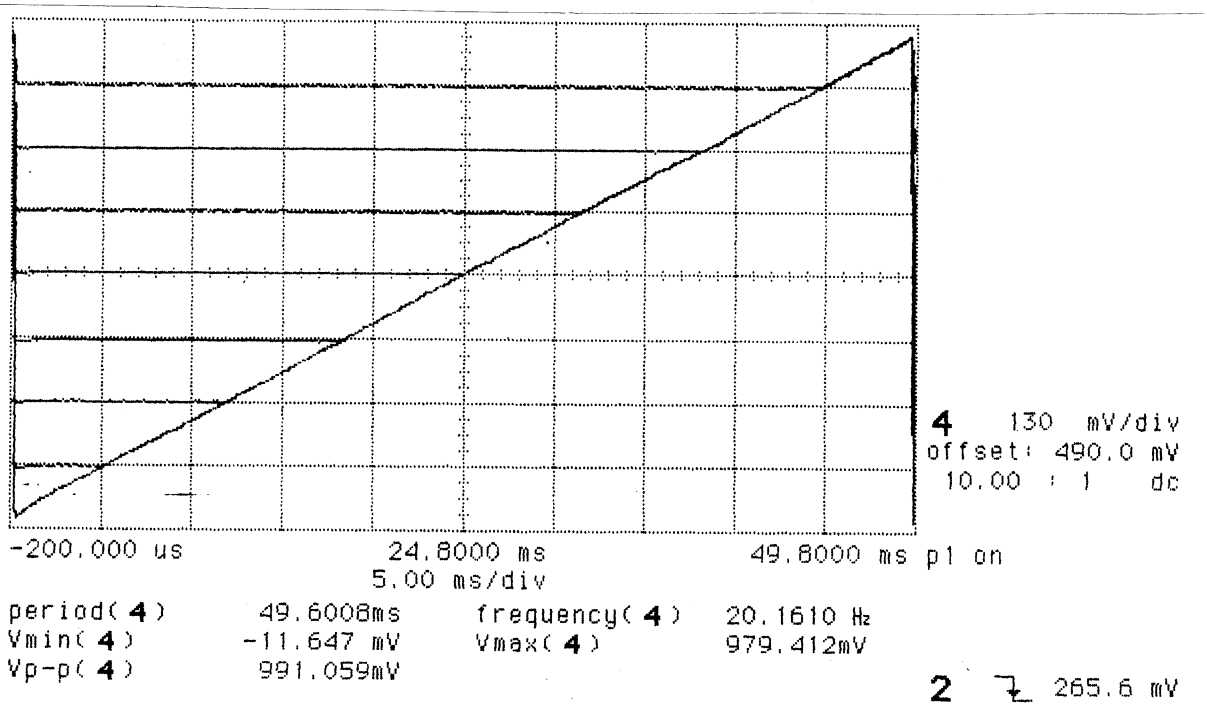
Σχήμα 4.14: Το κύκλωμα εισόδου.



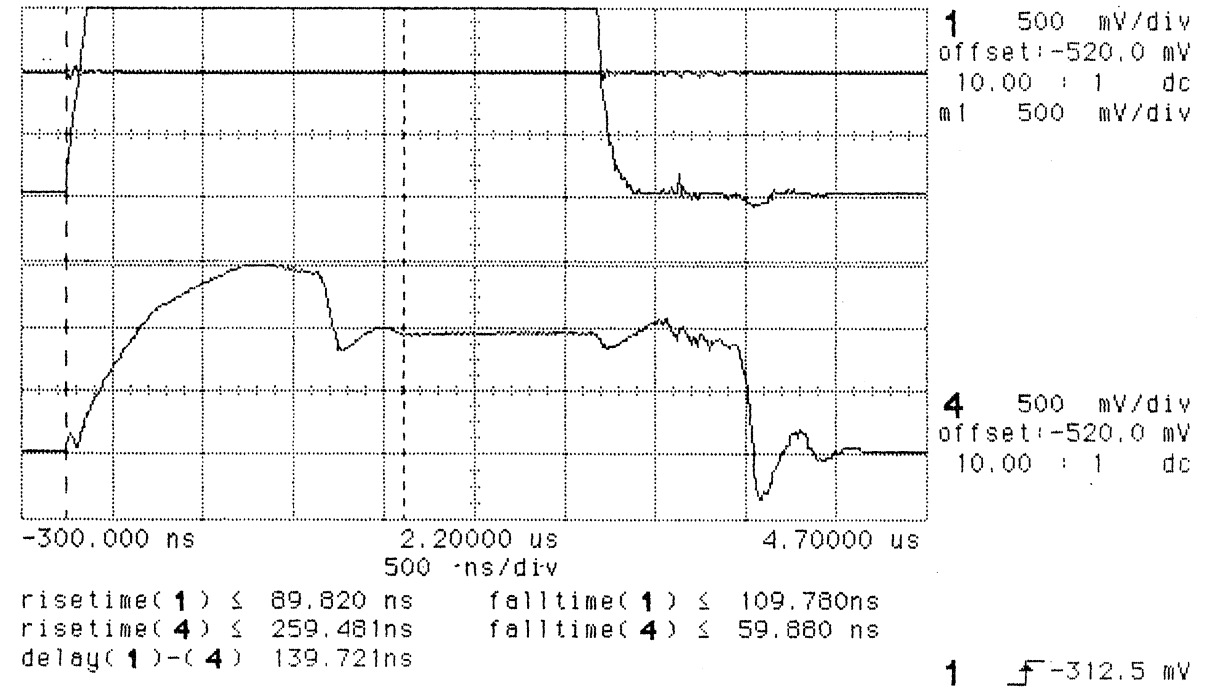
Σχήμα 4.15: (α,β) Πύλες AND και OR, (γ,δ) Τα αντίστοιχα σύμβολα.



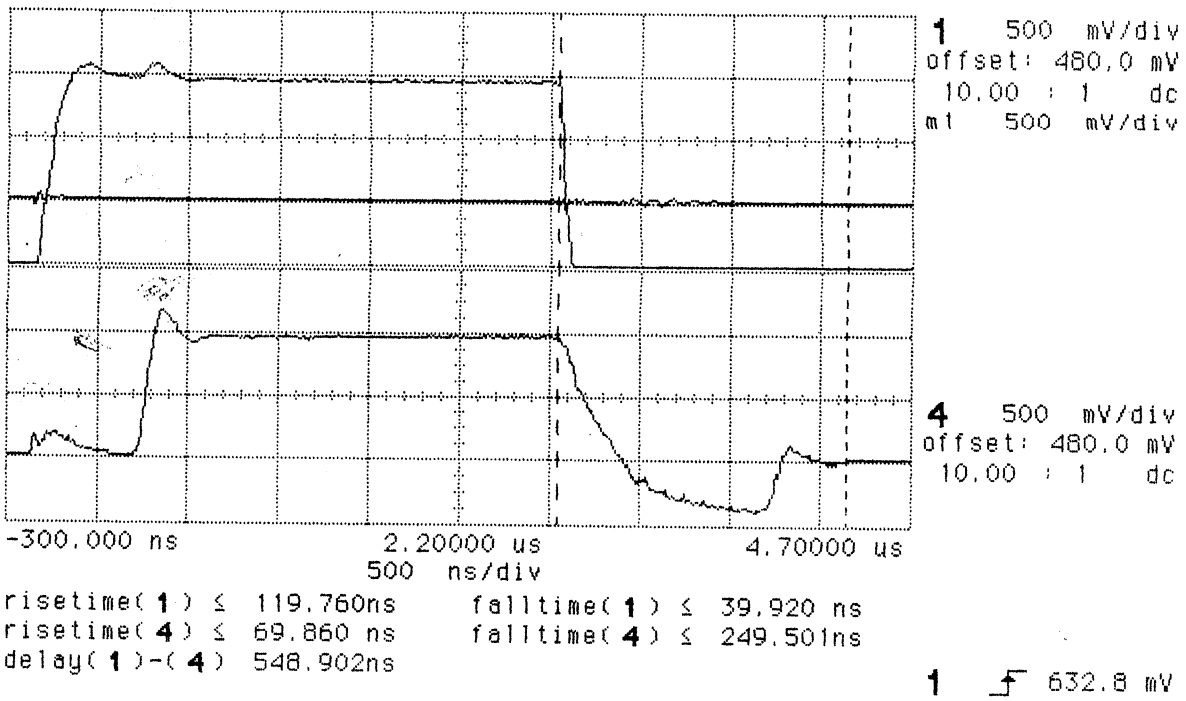
Σχήμα 4.16α: Χαρακτηριστική μεταφοράς της πύλης AND.



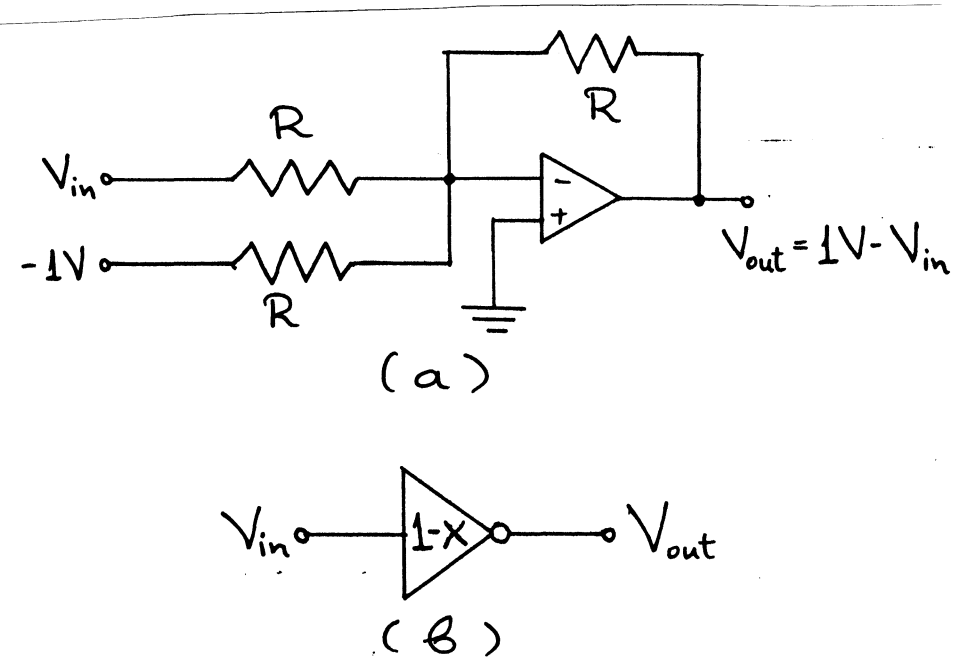
Σχήμα 4.16β: Χαρακτηριστική μεταφοράς της πύλης OR.



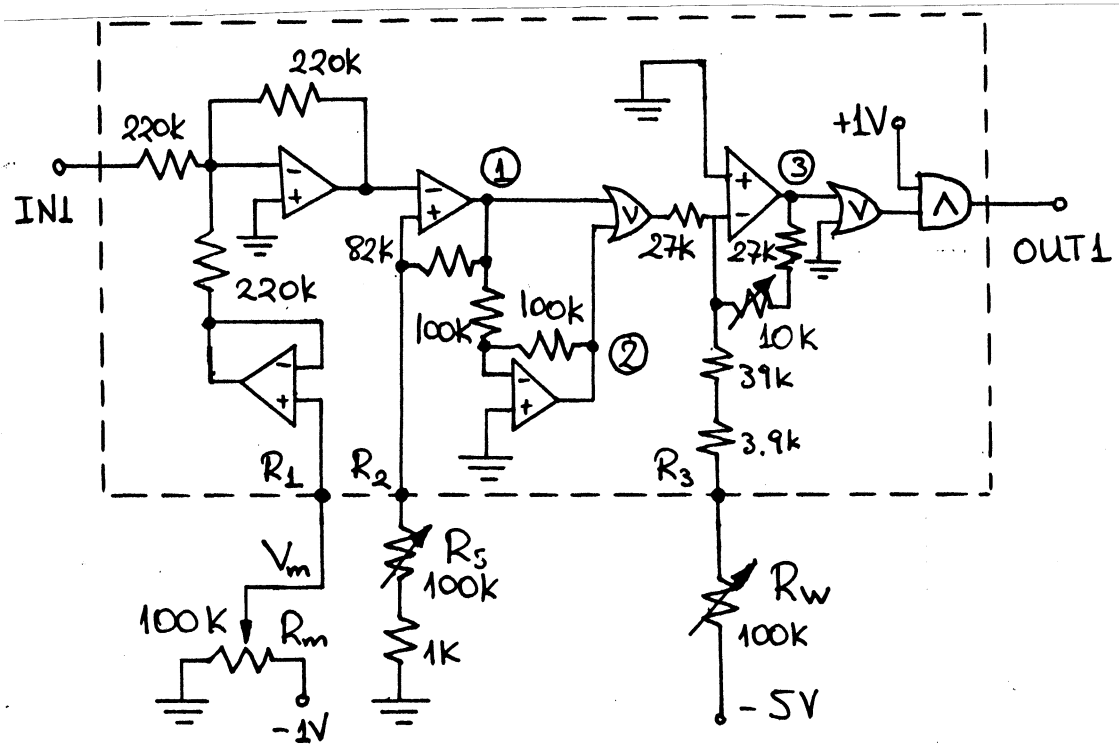
Σχήμα 4.17α: Απόκριση της πύλης AND σε ένα υψίσυχνο τετραγωνικό παλμό.



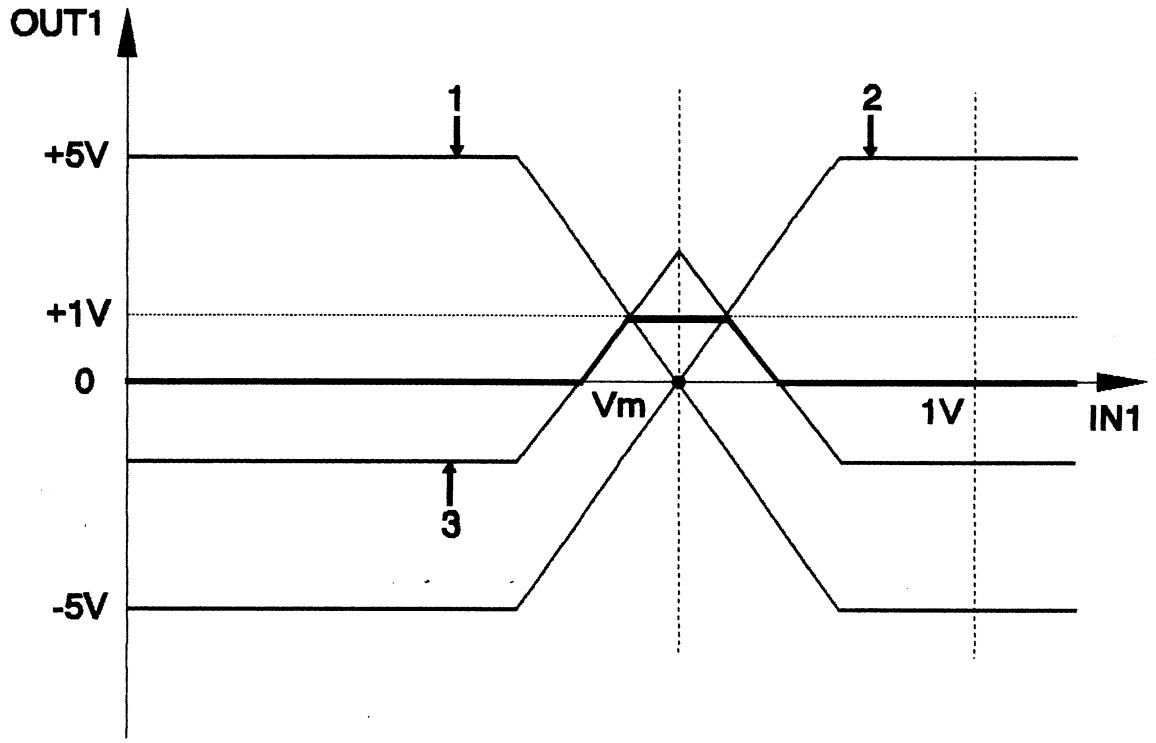
Σχήμα 4.17β: Απόκριση της πύλης OR σε ένα υψίσυχνο τετραγωνικό παλμό.



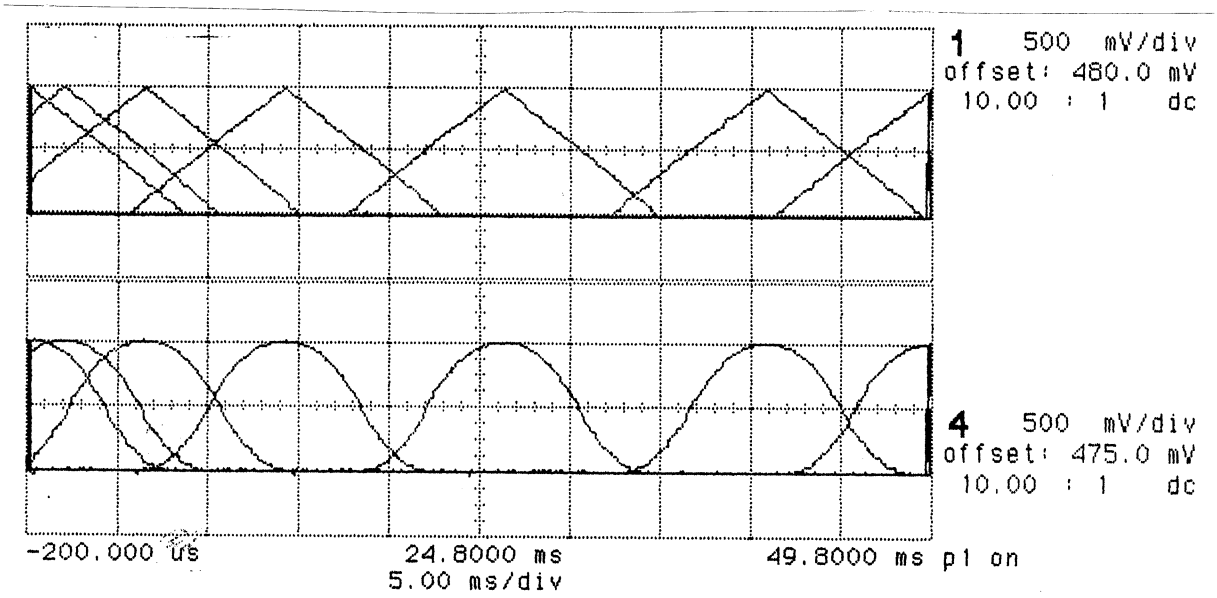
Σχήμα 4.18: (α) Η πύλη NOT, (β) Το σύμβολό της.



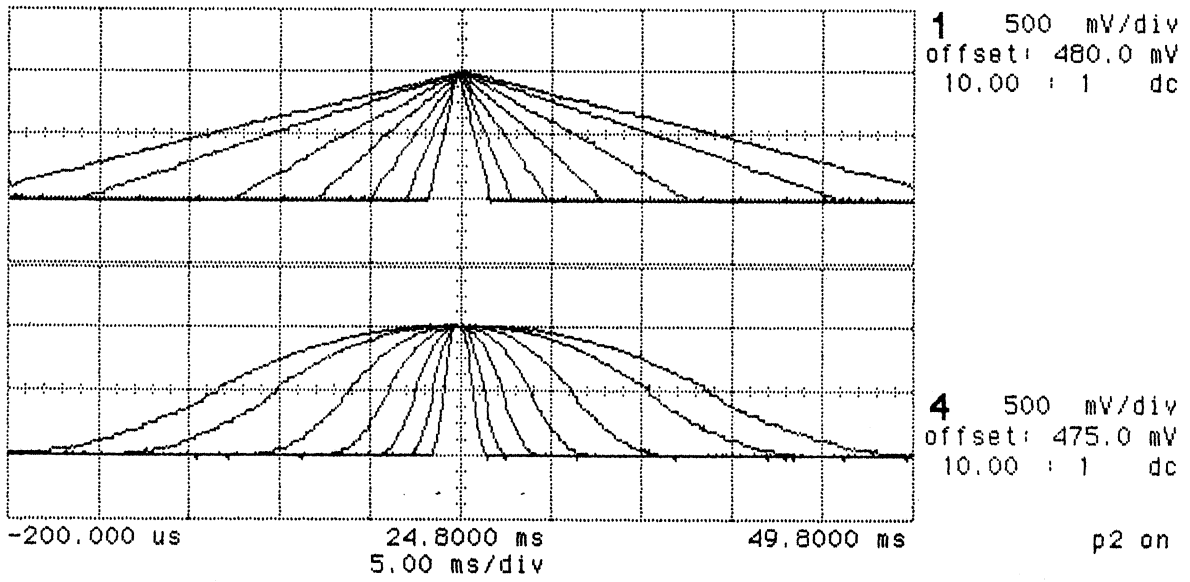
Σχήμα 4.19: Παραγωγή τριγωνικών και τραπεζοειδών συναρτήσεων.



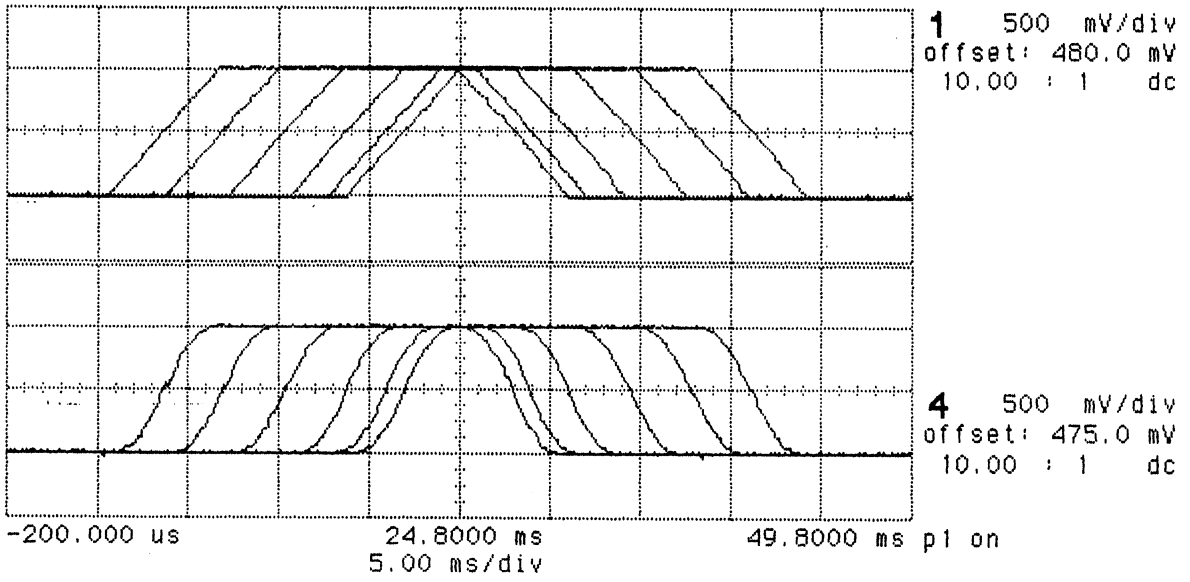
Σχήμα 4.20: Η λειτουργία του προηγούμενου κυκλώματος.



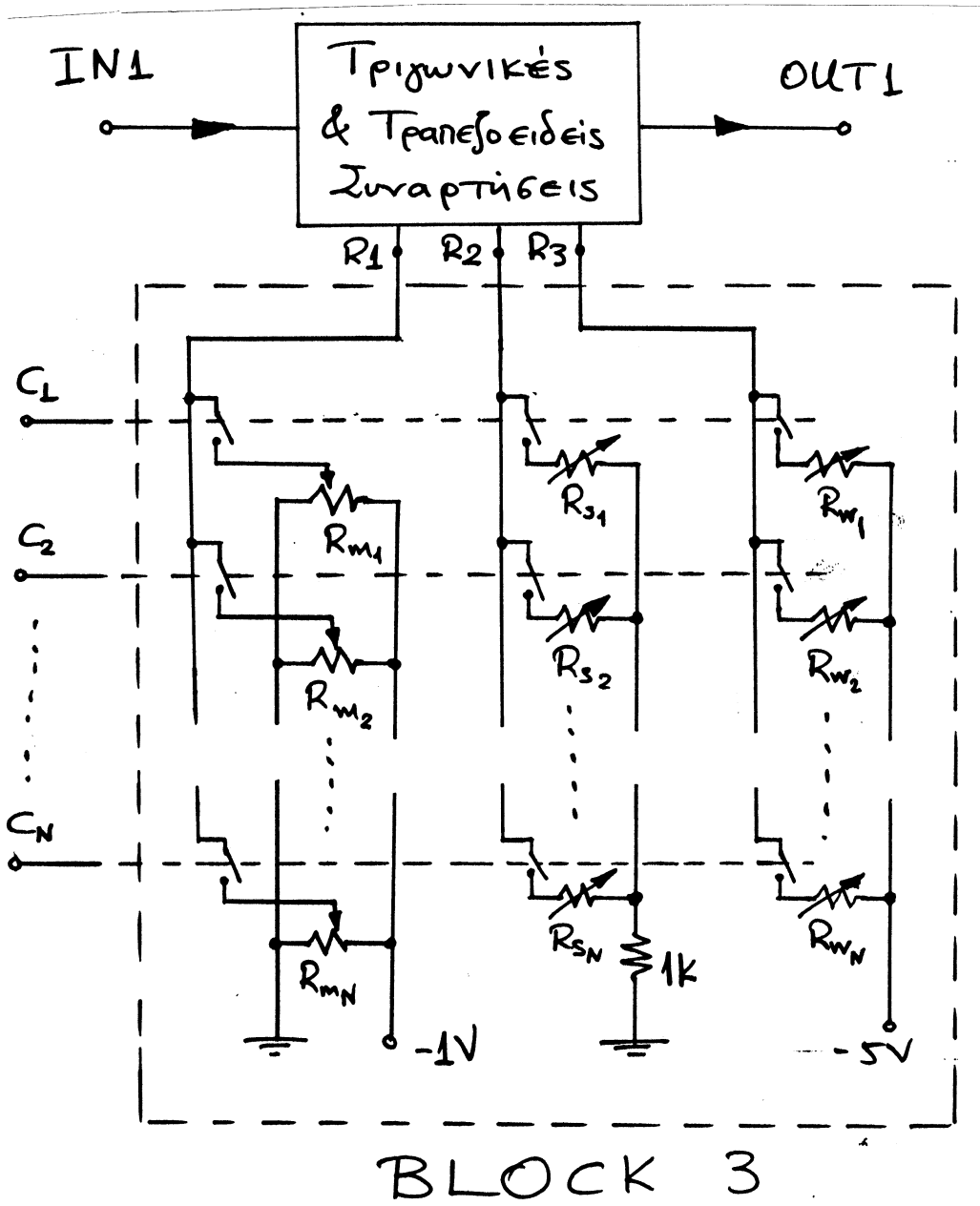
Σχήμα 4.21: Μεταβαλλόμενη μέση τιμή (1) Τριγωνικές, (4) Τύπου Π



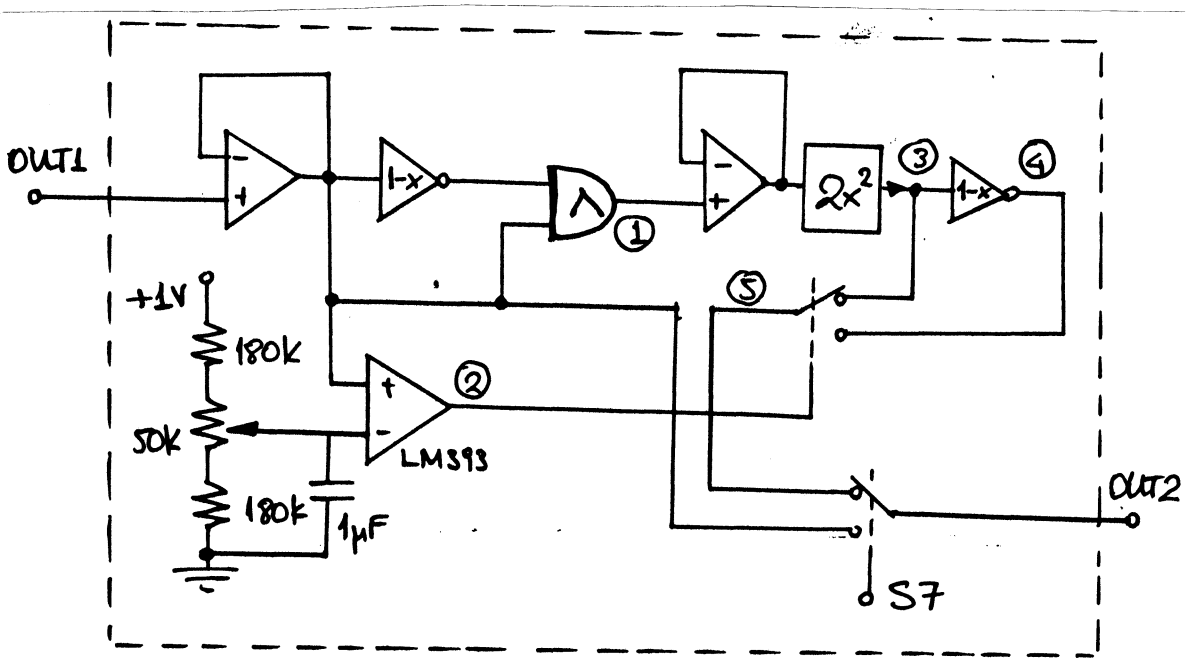
Σχήμα 4.22: Συναρτήσεις συμμετοχής με μεταβαλλόμενη κλίση.



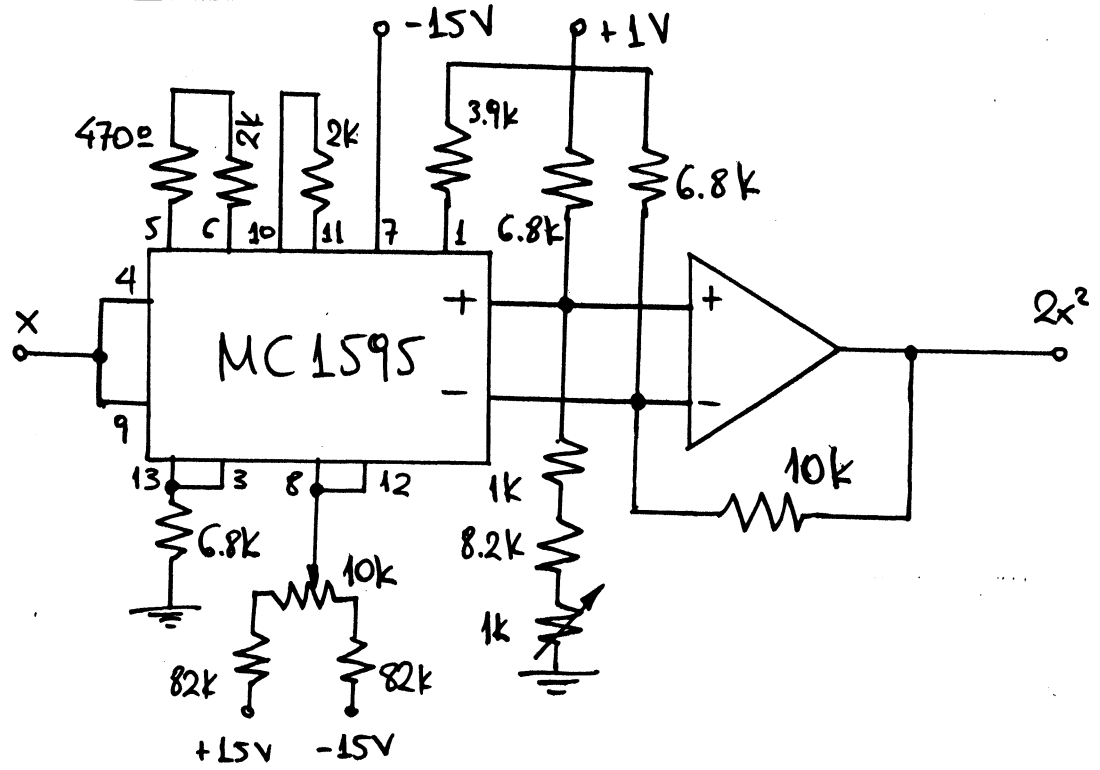
Σχήμα 4.23: Συναρτήσεις συμμετοχής με μεταβαλλόμενο εύρος.



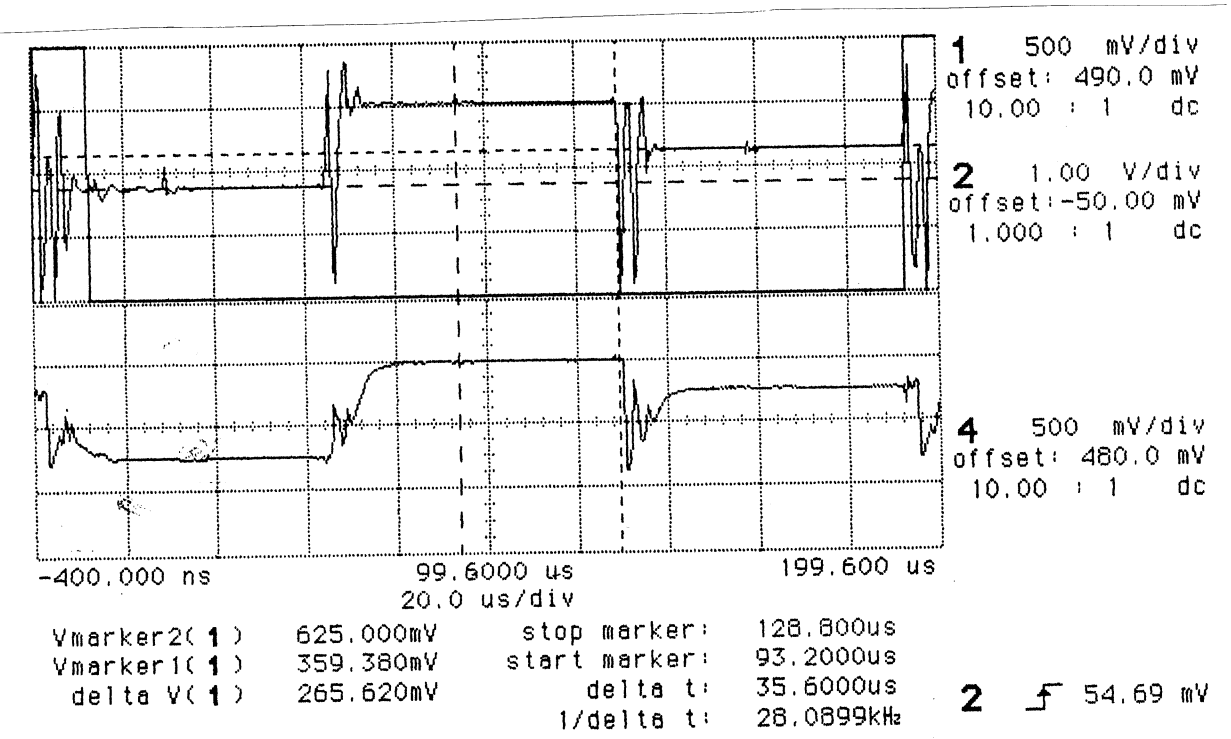
Σχήμα 4.24: Το σύνολο των συναρτήσεων συμμετοχής (block 3).



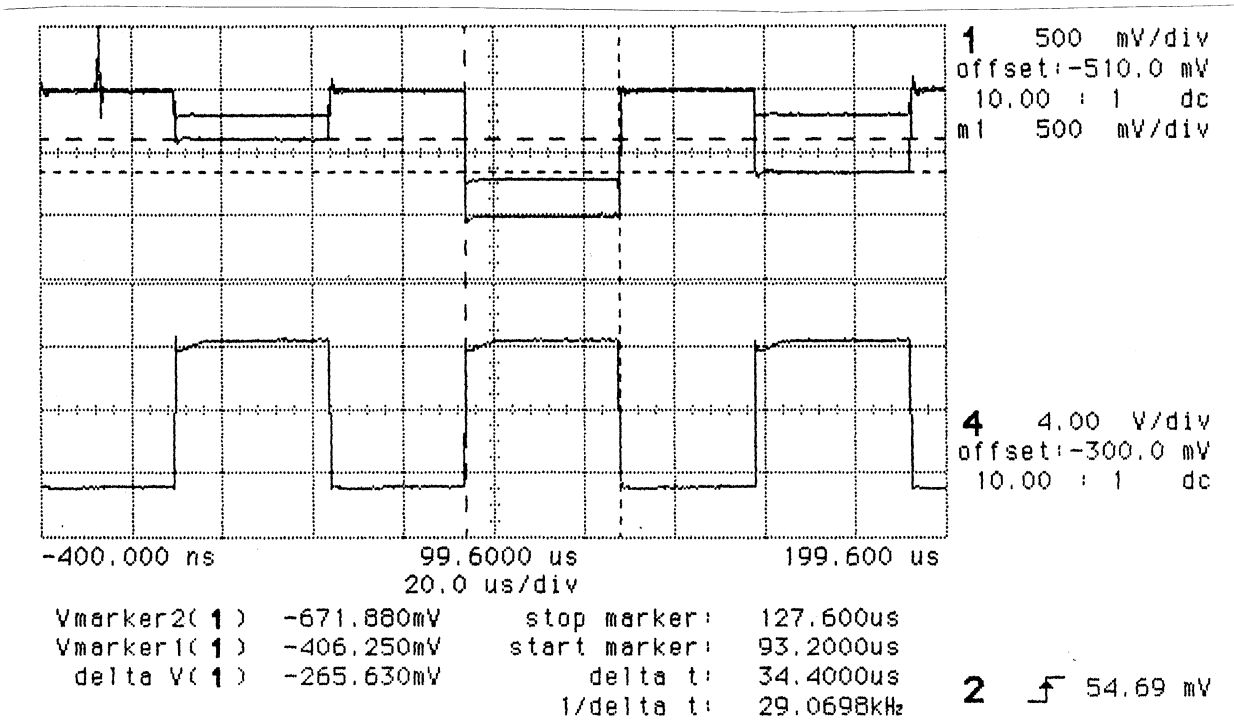
Σχήμα 4.25: Υλοποίηση της πράξης INT.



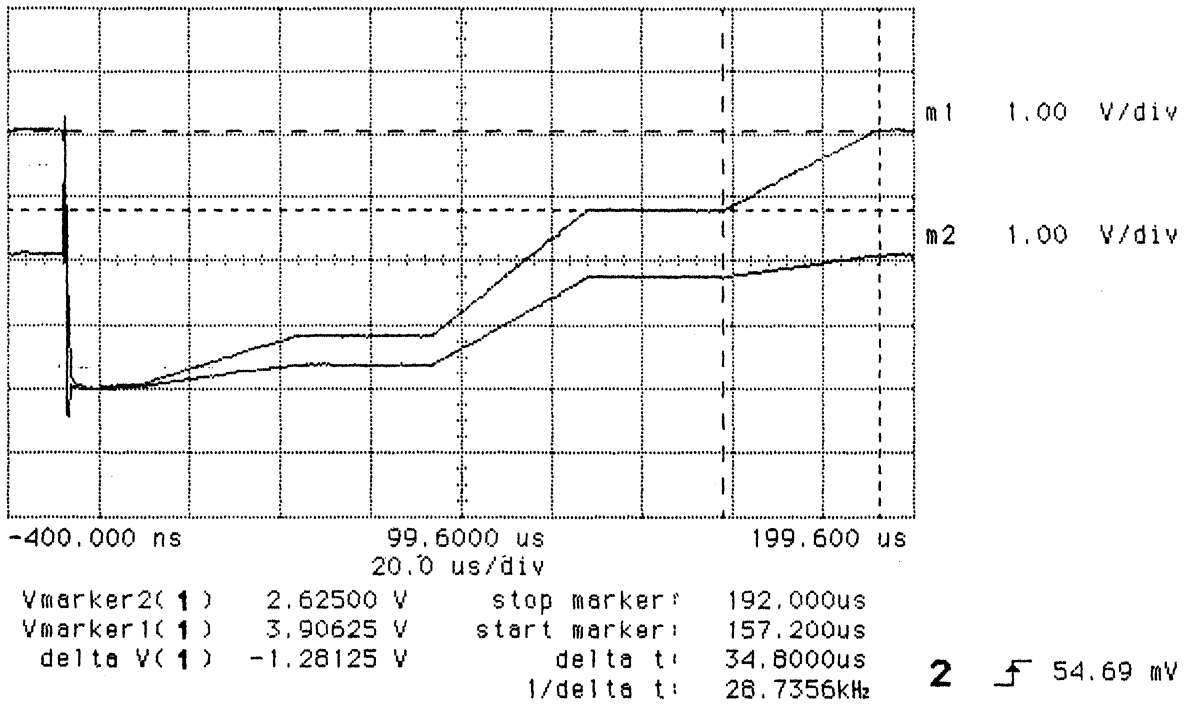
Σχήμα 4.26: Υλοποίηση της συνάρτησης $2x^2$.



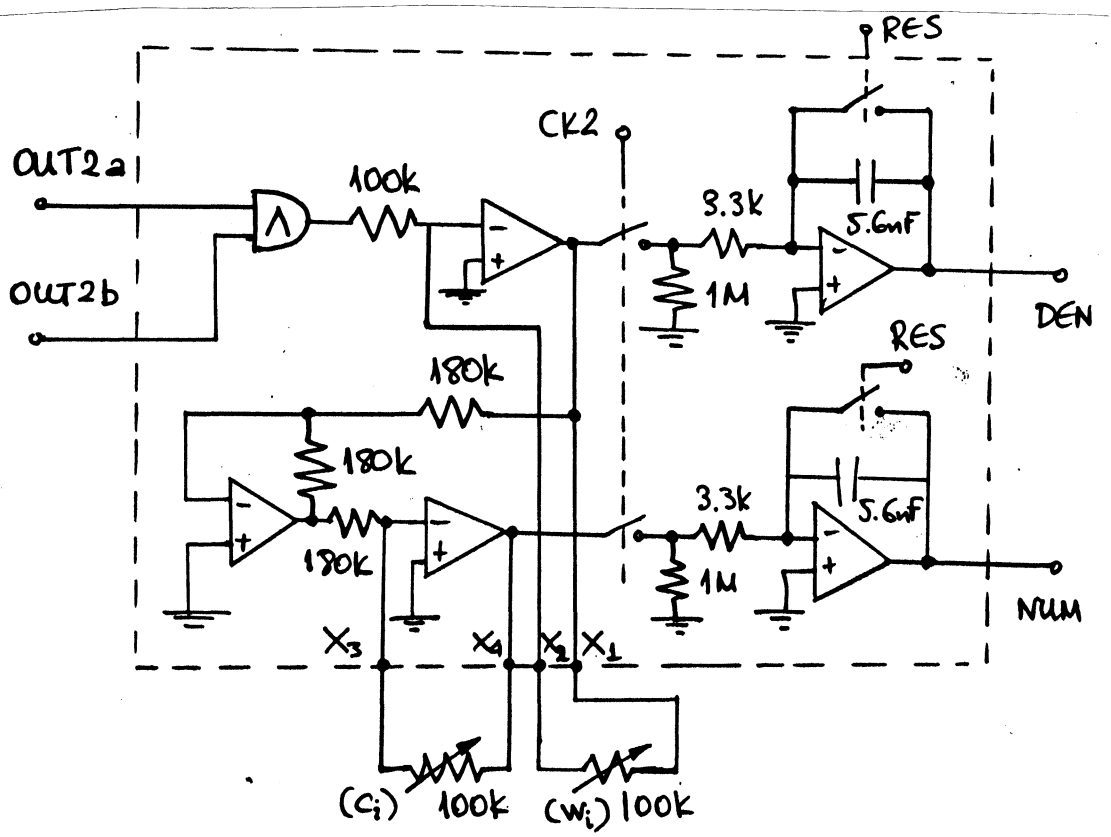
Σχήμα 4.27: Η τάση μ_i για τραπεζοειδείς συναρτήσεις (1) και τύπου Π (4)



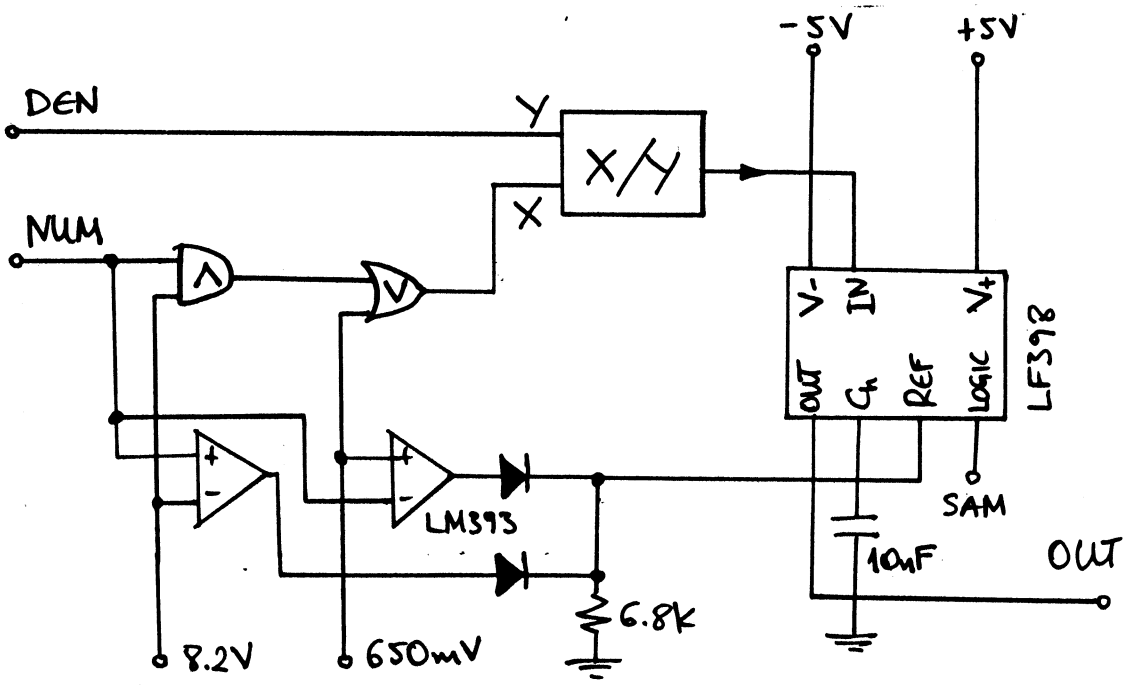
Σχήμα 4.28: Είσοδος ολοκληρωτών αριθμητή (1) και παρονομαστή (m1).



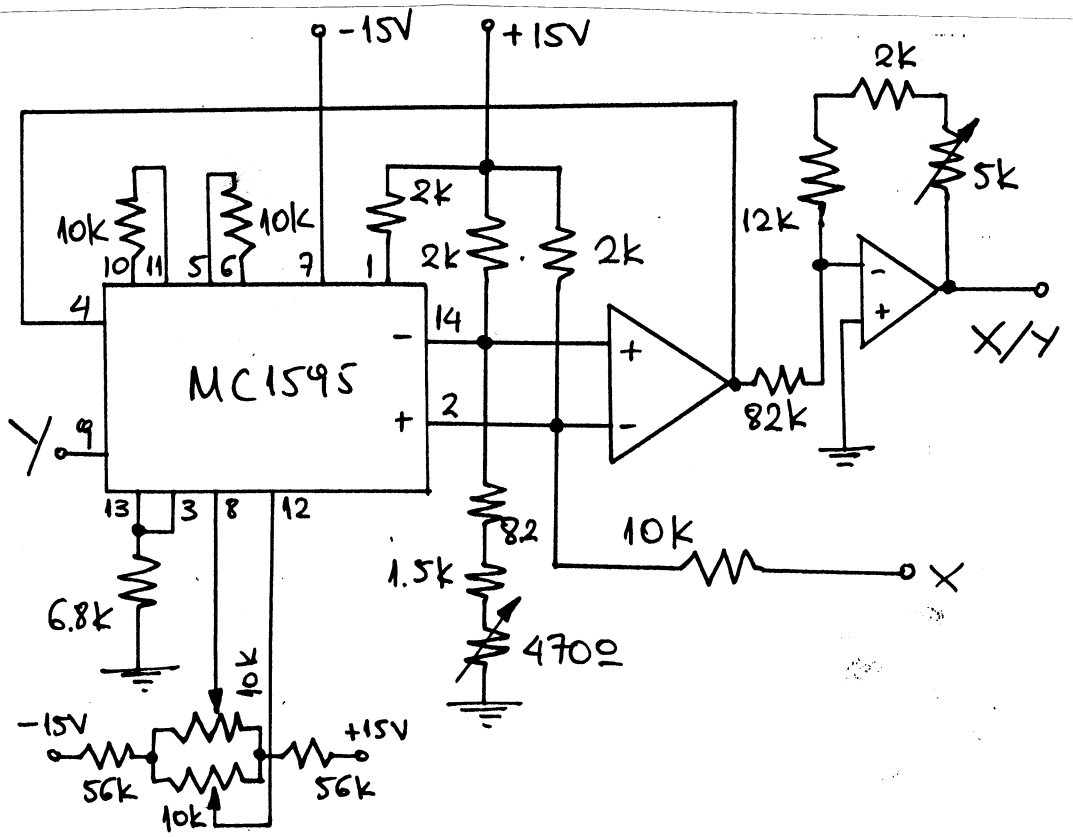
Σχήμα 4.29: Εξοδος ολοκληρωτών αριθμητή (m2) και παρονομαστή (m1).



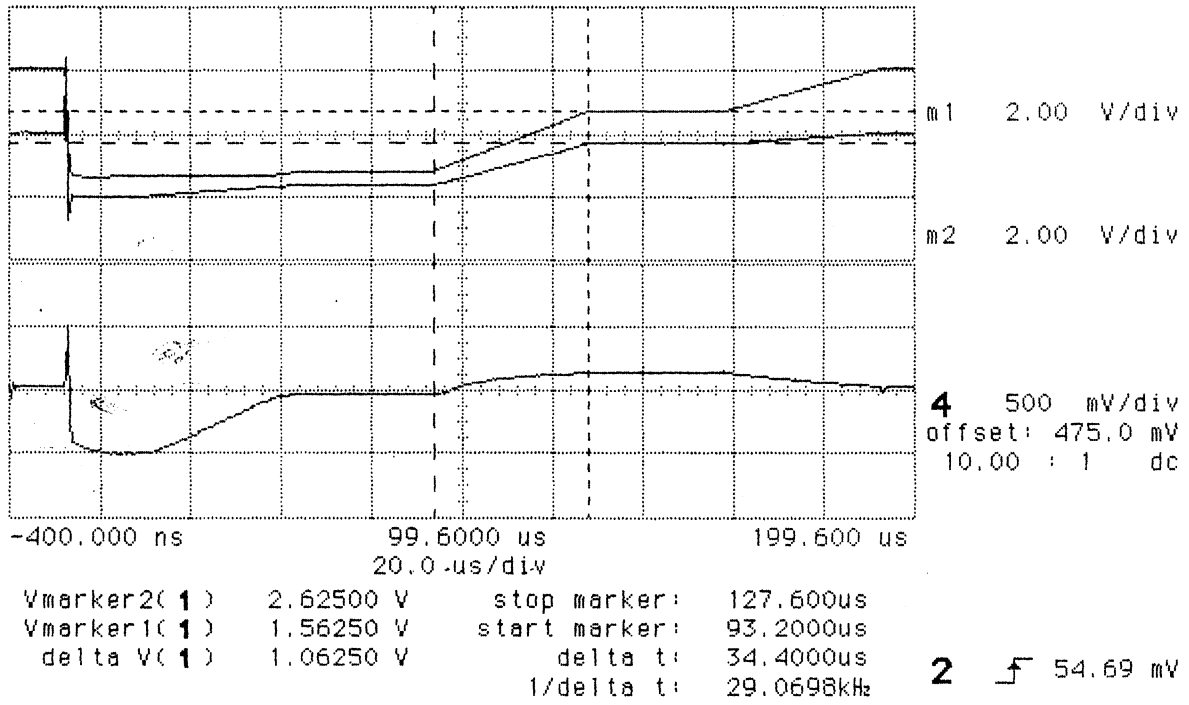
Σχήμα 4.30: Κύκλωμα υπολογισμού αριθμητή και παρονομαστή.



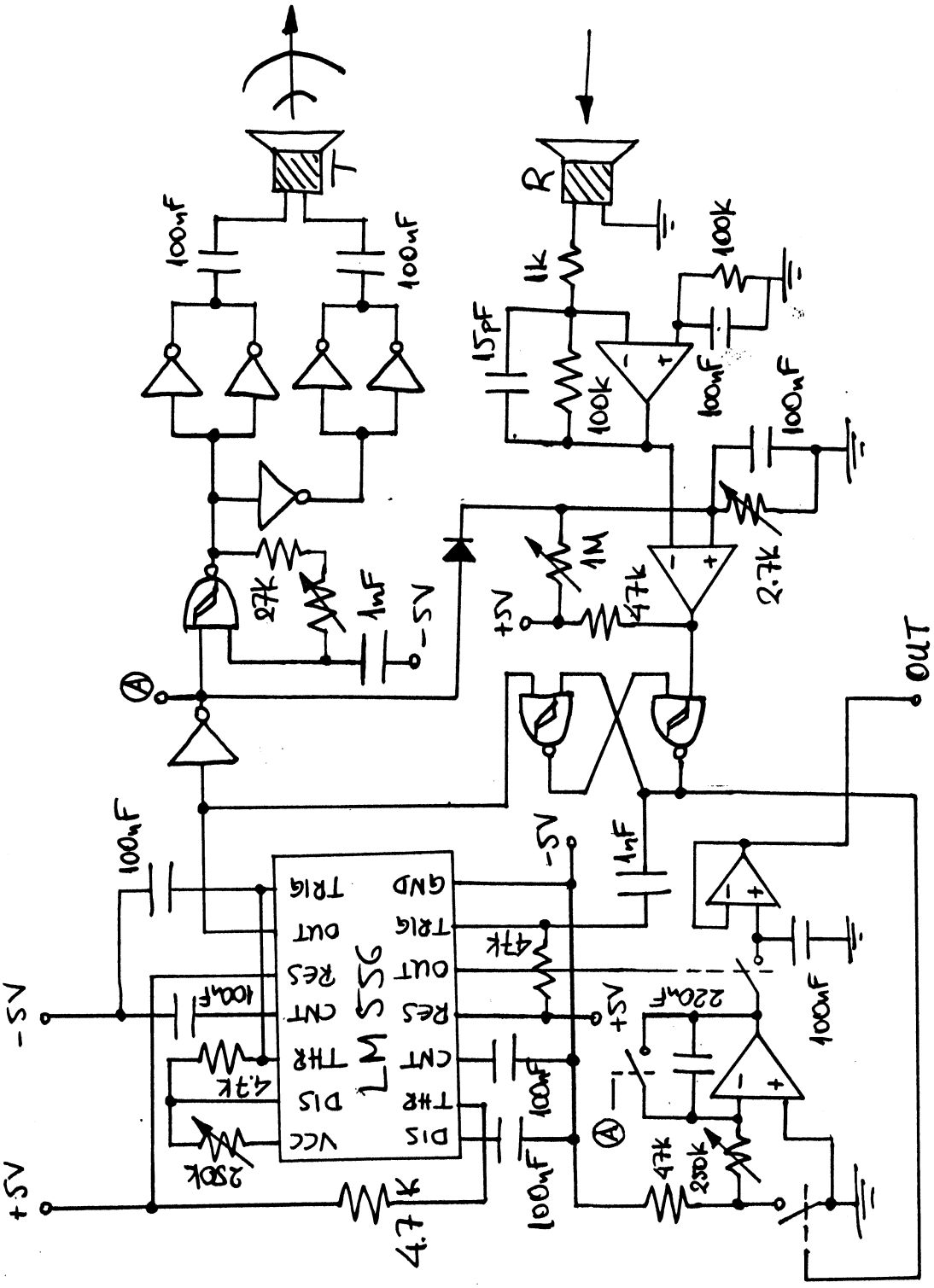
Σχήμα 4.31: Τελική φάση του defuzzification και κύκλωμα εξόδου.



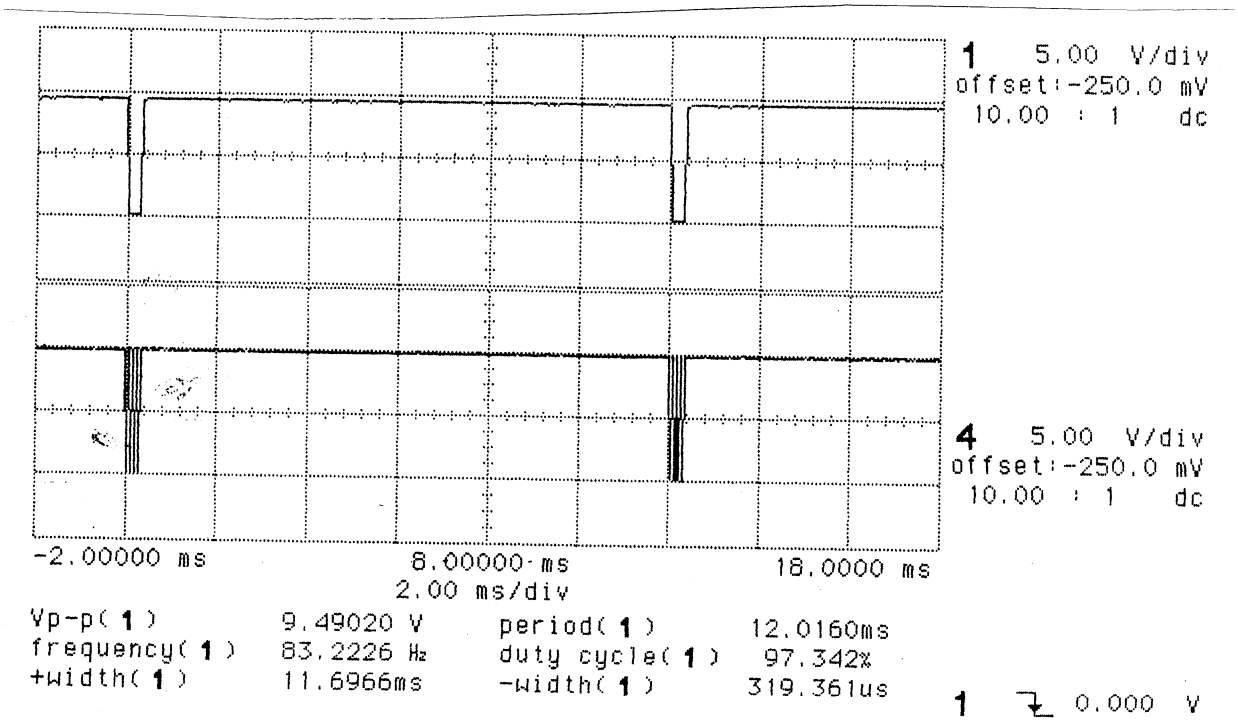
Σχήμα 4.32: Κύκλωμα διαίρεσης.



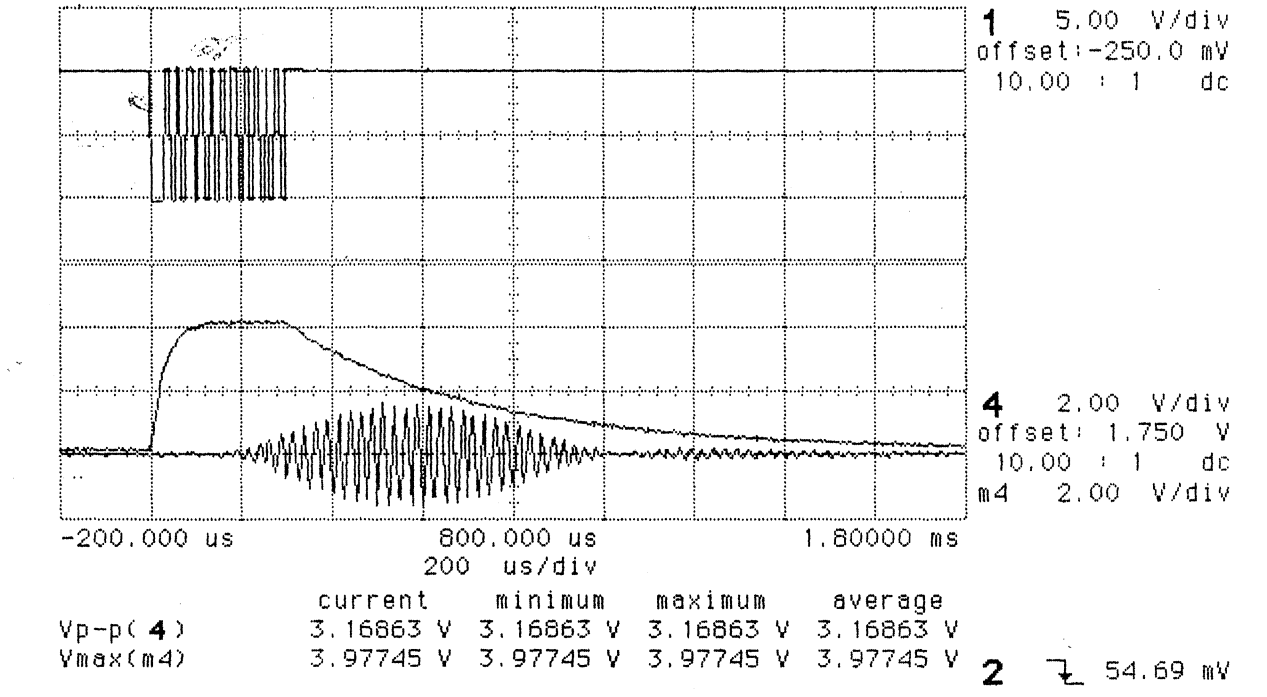
Σχήμα 4.33: (m2) Αριθμητής, (m1) παρονομαστής, (4) αποτέλεσμα διαίρεσης.



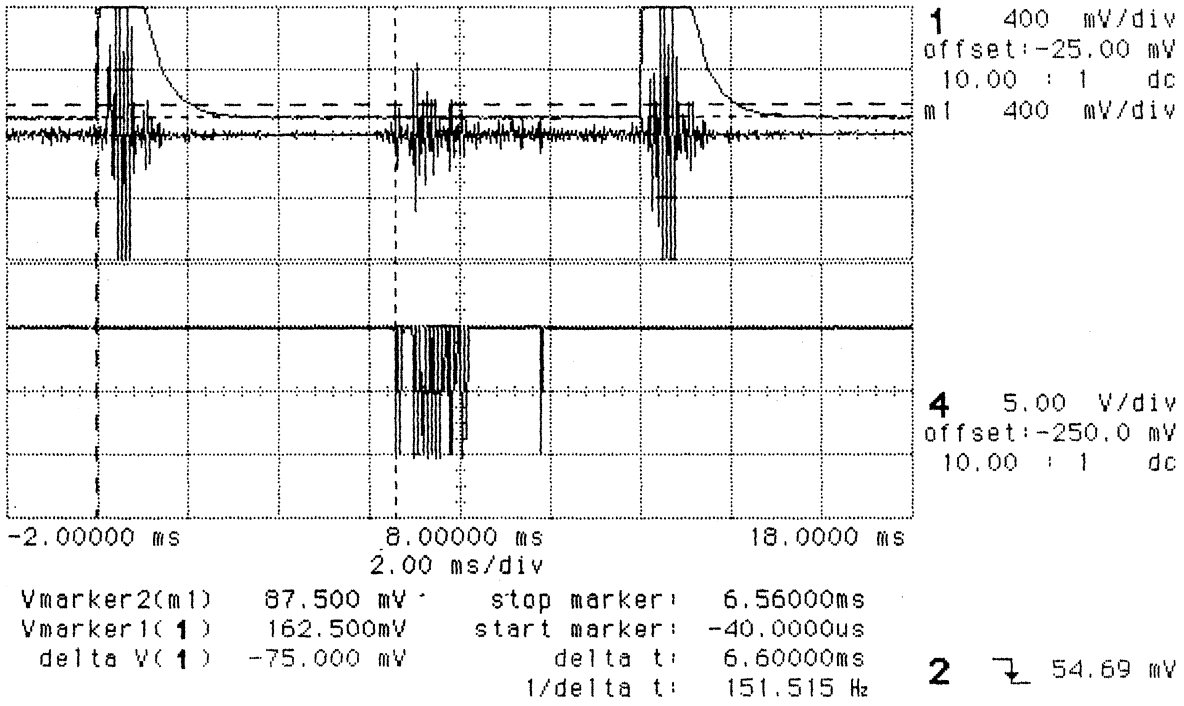
Σχήμα 5.1: Μετρητής απόστασης με υπερήχους.



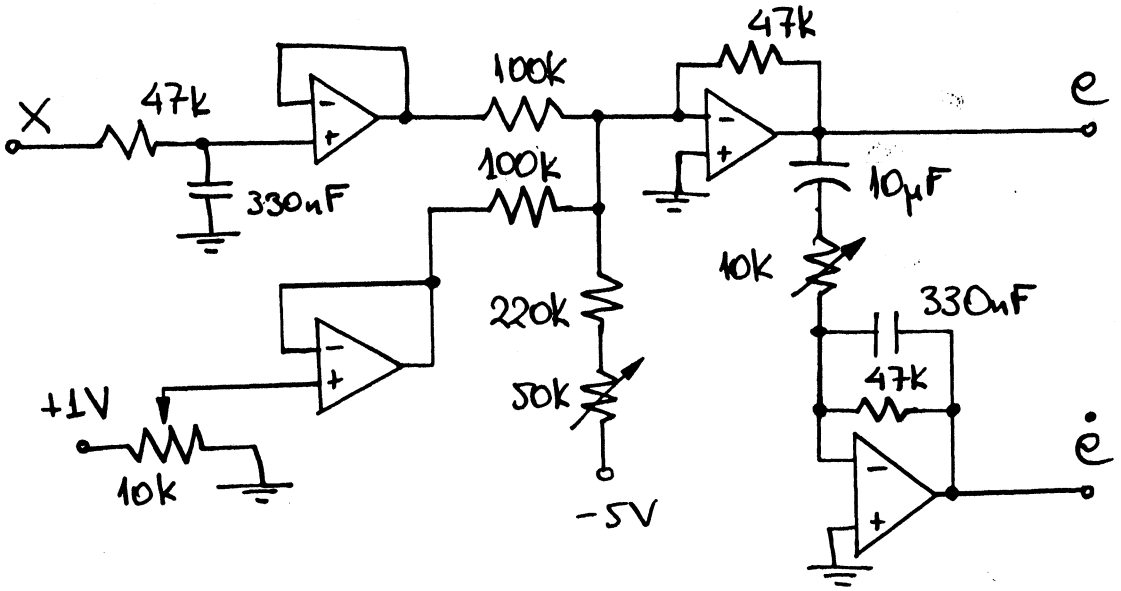
Σχήμα 5.2: (1) Ο παλμός Α, (4) η παλμοσειρά Β.



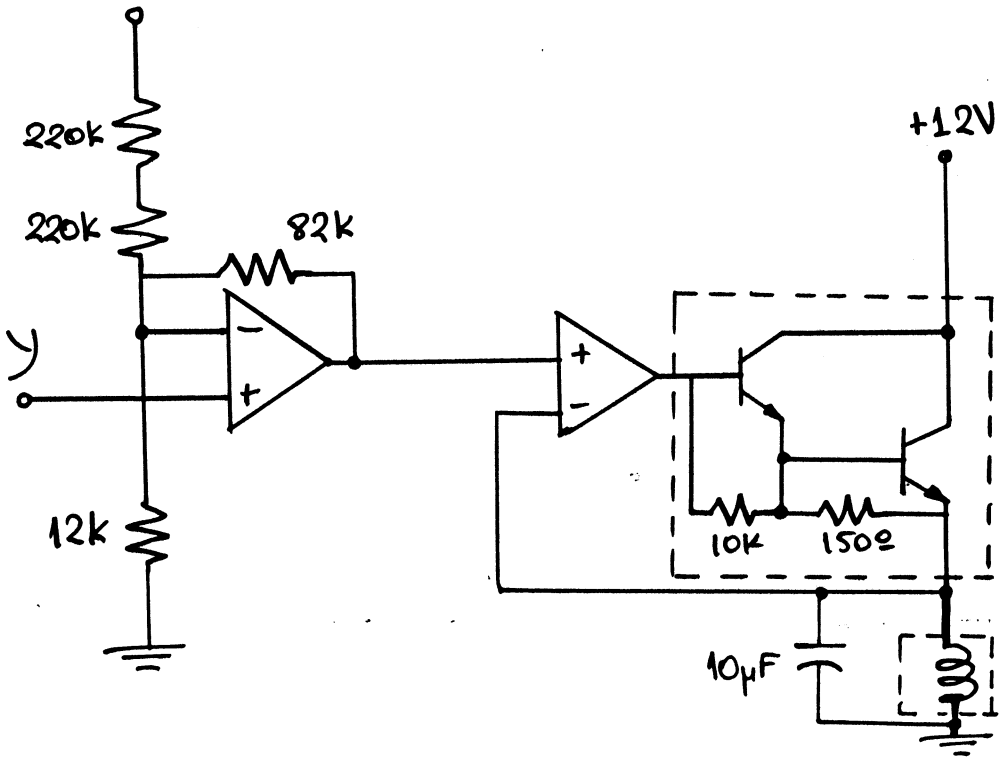
Σχήμα 5.3: (1) Εκπεμπόμενο και (4) λαμβανόμενο σήμα. (m4) Στάθμη σύγκρισης.



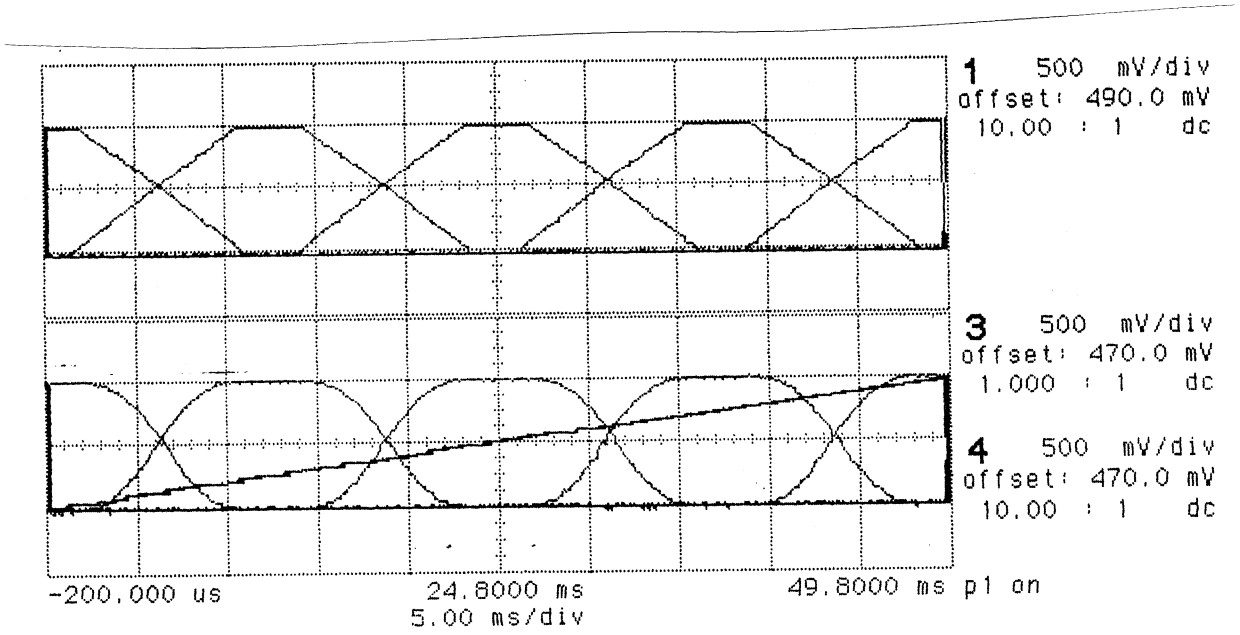
Σχήμα 5.4: (1) Λαμβανόμενο σήμα (m1) Στάθμη σύγκρισης (4) Εξοδος συγκριτή.



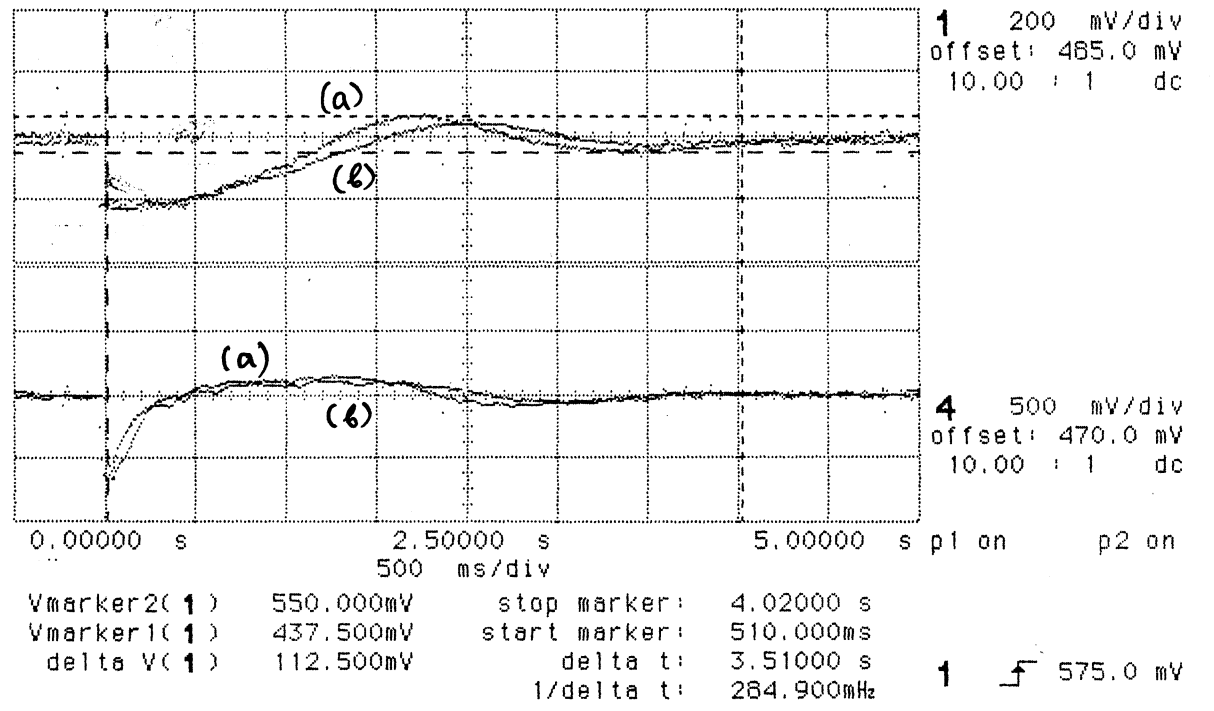
Σχήμα 5.5: Υπολογισμός σφάλματος θέσης και ταχύτητας.



Σχήμα 5.6: Στάδιο εξόδου του interface.



Σχήμα 6.1: Διαμέριση του X σε πέντε ασαφείς περιοχές.



Σχήμα 6.2: Χρονική απόκριση του συστήματος. (1) θέση, (4) ταχύτητα.

ΠΑΡΑΡΤΗΜΑ

ΑΡΧΕΙΑ ΕΙΣΟΔΟΥ ΤΟΥ MATLAB

```
function x=and(a,b,c)

% Fuzzy AND operator. Syntax:
%     1. and
%     2. and(type)
%     3. and(a, b)
%     4. and(a, b, type)
%     5. and([a1; a2; ... ; aN])
%     6. and([a1; a2; ... ; aN], type)
%
% - Syntax 1 creates a menu for selection of the AND operator type.
% - Syntax 2 sets the AND type to a specific value without any promts.
%     Valid values for type are: 'min', 'yag', 'prd', 'gpr', 'bpr' and 'dpr'.
% - Syntax 3 calculates a AND b, using the AND type stored in the global
%     variable 'and_type'. a and b represent fuzzy sets and must be vectors
%     of size (1,univ), where univ is also a global variable. See also INIT.
% - Syntax 4 is the same with syntax 3, but uses type instead of the
%     global variable 'and_type'.
% - Syntax 5 and 6 are the same with syntax 3 and 4, but calculate
%     a1 AND a2 AND ... AND aN.
%
% See also INIT, REPORT, SHOW, OR, NOT, FUZ, IMPL, ENTROPY, SUB, SUP.

% Yannis Avrithis, 11-5-93.

if nargin==0,
  k=0;
  while(k<1 | k>6),
     k=menu('Select type of fuzzy AND:', ..
            'min (Minimum)', ..
            'yag (Yager t-norm)', ..
            'prd (Product)', ..
            'gpr (Gamma Product)', ..
            'bpr (Bounded Product)', ..
            'dpr (Drastic Product)' );
  end
  types=['min';'yag';'prd';'gpr';'bpr';'dpr'];
  and_type=types(k,:);
  report

elseif nargin==1,
   if isstr(a),
     if (any(a~='min') & any(a~='yag') & any(a~='prd') & any(a~='gpr') & ..
         any(a~='bpr') & any(a~='dpr'))
       error('Bad argument type')
     else and_type=a;
     end
   else x=and(a,setstr(and_type));
   end

elseif nargin==2,
   if isstr(b),
     [M N]=size(a);
     if M<2, error('Bad argument type'), end
     if b=='min', x=min(a);
     elseif b=='yag', p=yag_par;
                      x=max(0,1-sum((1-a).^p).^(1/p));
     elseif b=='prd', x=prod(a);
     elseif b=='gpr', x=prod(a).^(1-gamma_par);
     elseif b=='bpr', x=max(sum(a)-M+1,0);
     elseif b=='dpr', x=zeros(1,N);
                      s=sum(a==1);
```

```
                    f1=find(s==M);
                    [M1 N1]=size(f1);
                    x(f1)=ones(1,N1);
                    f2=find(s==M-1);
                    t=min(a);
                    x(f2)=t(f2);
     end
   else x=and(a,b,setstr(and_type));
   end

elseif nargin==3,
   [Ma Na]=size(a);
   [Mb Nb]=size(b);
   if [Ma Na]==[1 1], a=a*ones(Mb,Nb);
   elseif [Mb Nb]==[1 1], b=b*ones(Ma,Na);
   elseif Na~=Nb, error('Bad argument type')
   end
   x=and([a;b],c);

end
```

```
function y=clip(x)

% CLIP.  y=clip(x) forces all elements of x to be between 0 and 1
% by clipping the elements that are outside this region. x may be
% a scalar, vector or matrix.
%
% See also INIT, SEE, POINTS, RESIZE, NORMAL, NRM, CONVEX.

% Yannis Avrithis, 12-5-93.

y=max(min(x,1),0);
```

```
function x=comp(a,b,c,typ)

% Fuzzy COMPOSITION operator. Syntax:
%     1. comp
%     2. comp(type)
%     3. b'=comp(a', a, b)
%     4. b'=comp(a', a, b, type)
%
% - Syntax 1 creates a menu for selection of the COMP operator type.
% - Syntax 2 sets the COMP type to a specific value without any promts.
%     Valid values for type are: 'min', 'yag', 'prd', 'gpr', 'bpr' and 'dpr'.
% - Syntax 3 calculates max(a' COMP (a IMPL b)), using the COMP type stored
%     in the global variable 'comp_type' and the IMPL type stored in the
%     global variable 'impl_type'. a', a and b represent fuzzy sets and
%     must be vectors of size (1,univ), where univ is also a global variable.
%     (See also INIT). It is equivalent to the generalized modus ponens:
%     { IF a THEN b,  a' }  ->  b'.
% - Syntax 4 is the same with syntax 3, but uses type instead of the
%     global variable 'comp_type'.
%
% See also INIT, REPORT, COMP2, IMPL, FELSE.

% Yannis Avrithis, 11-5-93.

if nargin==0,
  k=0;
  while(k<1 | k>6),
```

```
k=menu('Select type of fuzzy COMPOSITION:','min (AND: Min -> COMP: max-min)',..
'yag (AND: Yager)','prd (AND: Product -> COMP: max-*)','gpr (AND: Gamma Prod.)',..
'bpr (AND: Bounded Prod. -> COMP: max-theta)','dpr (AND: Drastic Prod. -> COMP: max-lambda)');

  end
  types=['min';'yag';'prd';'gpr';'bpr';'dpr'];
  comp_type=types(k,:);
  report

elseif nargin==1,
   if isstr(a),
     if (any(a~='min') & any(a~='yag') & any(a~='prd') & any(a~='gpr') & ..
         any(a~='bpr') & any(a~='dpr'))
       error('Bad argument type')
     else comp_type=a;
     end
   else error('Bad argument type');
   end

elseif nargin==3,
   x=comp(a,b,c,setstr(comp_type));

elseif nargin==4,
   [M1 N1]=size(a);
   [M2 N2]=size(b);
   if (N1~=N2 | M1~=1 | M2~=1), error('Bad argument type'), end
   x=-inf;
   for k=1:N1
     x=max(x,and( a(k), impl(b(k),c,setstr(impl_type)), typ ));
   end

else error('Wrong argument count')
end
```

```
function x=comp2(a1,a,b1,b,c,typ)

% Fuzzy COMPOSITION operator for 2 antecedents. Syntax:
%     1. b'=comp(a', a, b', b, c)
%     2. b'=comp(a', a, b', b, c, type)
%
% - Syntax 1 calculates
%
%      max { max { (a' AND b') COMP ( (a AND b) IMPL c ) } },
%       a     b
%
%     using the COMP type stored in the global variable 'comp_type' and
%     the IMPL type stored in the global variable 'impl_type'.
%     a', a, b', b and c represent fuzzy sets and must be vectors of size
%     (1,univ), where univ is also a global variable (See also INIT).
%     It is equivalent to the generalized modus ponens:
%     { IF a AND b THEN c,  { a', b' } }  ->  c'.
%
% - Syntax 2 is the same with syntax 3, but uses type instead of the
%     global variable 'comp_type'.
%
% See also INIT, REPORT, COMP, IMPL, FELSE.

% Yannis Avrithis, 31-5-93.

if nargin==5,
   x=comp2(a1,a,b1,b,c,setstr(comp_type));

elseif nargin==6,
   [Ma1 Na1]=size(a1);
```

```
   [Ma Na]=size(a);
   [Mb1 Nb1]=size(b1);
   [Mb Nb]=size(b);
   if (Na1~=Na | Ma1~=1 | Ma~=1), error('Bad argument size'), end
   if (Nb1~=Nb | Mb1~=1 | Mb~=1), error('Bad argument size'), end
   x=-inf;
   for ka=1:Na
   for kb=1:Nb
     x=max(x, and( and(a1(ka),b1(kb)) , impl( and(a(ka),b(kb)) , ..
       c, setstr(impl_type) ) , typ ));
   end
   end

else error('Wrong argument count')
end
```

function Y=conc(X,n)

```
% CONCENTRATION of a fuzzy set. Y=conc(X,n) (n>1) raises the membership values
% of the elements of set X to power n and produces the concetrated (if n>1)
% fuzzy set Y. If n is omitted, it is assumed 2. Note that X is dilated
% if 0<n<1.
%
% See also DIL, CONC2, DIL2, INTEN, INTEN2, FUZZIF, AND, OR, NOT.

% Yannis Avrithis, 15-5-93.

if nargin==1, n=2; end
Y=X.^n;
```

function Y=conc2(X,a)

```
% CONCENTRATION of a fuzzy set. Y=conc2(X,a) (a>0) computes x/(1+a*(1-x))
% for every membership value x of the fuzzy set X, and produces the
% concetrated (if a>0) fuzzy set Y. If a is omitted, it is assumed 2.
% Note that X is dilated if -1<a<0.
%
% See also CONC, DIL, DIL2, INTEN, INTEN2, FUZZIF, AND, OR, NOT.

% Yannis Avrithis, 19-5-93.

if nargin==1, a=2; end
Y=X./(1+a*(1-X));
```

function Y=convex(X)

```
% CONVEX closure of a fuzzy set (vector of size (1, univ)).
% Y=convex(X), where X is any fuzzy set, produces the respective
% convex fuzzy set Y.
%
% See also CLIP, NRM, NORMAL, DENORMAL, RANGE.

% Yannis Avrithis, 24-5-93.

Y=X(:).' ;                      % Y must be a row vector
D=diff(sign(diff([0 Y 0])));  % Approximate 2nd derivative
```

```
S=find(D<0);                    % Local minima
M=max(size(S));
k=1; b=S(1);
while k<M
  a=b; P=Y(S(k+1:M));
  k = k + min( find( P >= min(Y(S(k)),max(P)) ) );
  b=S(k); Y(a:b)=max( Y(a:b), min(Y(a),Y(b)) );
end
```

```
function x=defuz(a,b)

% Defuzzification. Syntax:
%     1. defuz
%     2. defuz(type)
%     3. defuz(a)
%     4. defuz(a, type)
%     5. defuz([a1; a2; ... ; aN])
%     6. defuz([a1; a2; ... ; aN], type)
%
% - Syntax 1 creates a menu for selection of the defuzzification type.
% - Syntax 2 sets the DEFUZ type to a specific value without any promts.
%     Valid values for type are: 'cnt' and 'mom'.
% - Syntax 3 calculates the defuzzified value of a, using the DEFUZ type
%     stored in the global variable 'defuz_type'. a represents a fuzzy
%     (consequent) set and must be a vector of size (1,univ), where univ
%     is also a global variable. See also INIT.
% - Syntax 4 is the same with syntax 3, but uses type instead of the
%     global variable 'defuz_type'.
% - Syntax 5 and 6 are the same with syntax 3 and 4, but first aggregate
%     the matrix [a1; a2; ... aN] to a single vector using FELSE.
%
% See also INIT, REPORT, FELSE, RULE, SEE_DEF, DENORMAL, FUZ.

% Yannis Avrithis, 11-5-93.

if nargin==0,
  k=0;
  while(k<1 | k>2),
     k=menu('Select type of DEFUZZIFICATION:', ..
            'cnt (Centroid)', ..
         'mom (Mean of Max)' );
  end
  types=['cnt';'mom'];
  defuz_type=types(k,:);
  report

elseif nargin==1,
   if isstr(a),
     if (any(a~='cnt') & any(a~='mom'))
       error('Bad argument type')
     else defuz_type=a;
     end
   else x=defuz(a,setstr(defuz_type));
   end

elseif nargin==2,
   [M N]=size(a);
   if M>1, a=felse(a); end
   k=(0:univ-1)/(univ-1);
   if b=='cnt', x=sum(k.*a)/sum(a);
   elseif b=='mom', f=find(a<max(a));
                    [M N]=size(f);
            a(f)=zeros(M,N);
            x=sum(k.*a)/sum(a);
```

```
   end

end
```

```
function y=denormal(x,S)

% Denormalization of outputs from the region [0,1] to S=[a,b]
% Syntax: y=denormal(x,S). x may be a scalar, vector or matrix,
% but its elements must belong to the region [0,1].
%
% See also DEFUZ, NORMAL, NRM.

% Yannis Avrithis, 11-5-93.

a=S(1);
b=S(2);
% if (x<0 | x>1), error('Bad argument value'), end
y=x .* (b-a) + a;
```

```
function Y=dil(X,n)

% DILATION of a fuzzy set. Y=dil(X,n) (0<n<1) raises the membership values
% of the elements of set X to power n and produces the dilated (f n<1) fuzzy
% set Y. If n is omitted, it is assumed 0.5. Note that X is concentrated
% if n>1.
%
% See also CONC, CONC2, DIL2, INTEN, INTEN2, FUZZIF, SEE, AND, OR, NOT.

% Yannis Avrithis, 15-5-93.

if nargin==1, n=0.5; end
Y=X.^n;
```

```
function Y=dil2(X,a)

% DILATION of a fuzzy set. Y=dil2(X,a) (-1<a<0) computes x/(1+a*(1-x))
% for every membership value x of the fuzzy set X, and produces the
% dilated (if -1<a<0) fuzzy set Y. If a is omitted, it is assumed -0.6.
% Note that X is concentrated if a>0.
%
% See also CONC, DIL, CONC2, INTEN, INTEN2, FUZZIF, AND, OR, NOT.

% Yannis Avrithis, 19-5-93.

if nargin==1, a=-0.6; end
Y=X./(1+a*(1-X));
```

```
function E=entropy(A);

% Fuzzy ENTROPY. E=entropy(A), where A represents a fuzzy set (and is
% usually a vector of size (1,univ)), measures the fuzzy entropy of A,
% i.e. how fuzzy A is. Due to the fuzzy entropy theorem,
```

```
%
%          M(A and A')
%    E(A)=-------------
%          M(A or A')
%
% where A'= NOT A, and_type='min', or_type='max' and M(.) is the
% cardinality measure or sigma-count of a fuzzy set (here norm(.,1)
% is used). The entropy of a crisp or nonfuzzy set (Boolean corner
% of the hypercube) equals 0, while the entropy of the midpoint
% (A=1/2 everywhere) is 1. Note that due to the entropy-subsethood
% theorem, E(A)=S(A and A', A or A'), where S means subsethood (see SUB).
%
% Source: Bart Kosko, "Neural Networks & Fuzzy Systems", 1992.
% See also INIT, REPORT, AND, OR, NOT, SUB, SUP.

% Yannis Avrithis, 12-5-93.

A1=not(A);
E=norm(and(A,A1),1)/norm(or(A,A1),1);
```

function F=eq(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12)

```
% Function F=eq(A1,A2, ... , AN) returns 1 if all of the arguments
% are exactly the same and 0 otherwise. Very useful for comparing strings.
% For strings, leading and trailing spaces are ignored.

% Yannis Avrithis, 24-5-93.

F=1;
B=A1;
for k=2:nargin
  A=B;
  B=eval(['A',num2str(k)]);
  if isstr(A),
    A=A(find(A~=' '));
  end
  if isstr(B),
    B=B(find(B~=' '));
  end
  if any(size(A)~=size(B)), F=0; break
  elseif any(any(A~=B)), F=0; break, end
end
```

function x=felse(a,b)

```
% Fuzzy ELSE (Aggregation) between the consequents of the rules.
% Syntax:
%     1. felse
%     2. felse(type)
%     3. felse([a1; a2; ... ; aN])
%     4. felse([a1; a2; ... ; aN], type)
%
% - Syntax 1 creates a menu for selection of the ELSE operator type.
% - Syntax 2 sets the ELSE type to a specific value without any promts.
%     Valid values for type are: 'and', 'or_' and 'sum'.
% - Syntax 3 calculates ELSE(a1, a2, ... aN), using the ELSE type stored
%     in the global variable 'else_type'. a1, a2, ... aN represent fuzzy
%     (consequent) sets and must be vectors of size (1,univ), where univ
%     is also a global variable (See also INIT). [a1; a2; ... aN] is
%     usually created by the RULE command.
```

```
% - Syntax 4 is the same with syntax 3, but uses type instead of the
%     global variable 'else_type'.
%
% See also INIT, REPORT, COMP, IMPL, RULE, DEFUZ.

% Yannis Avrithis, 11-5-93.

if nargin==0,
  k=0;
  while(k<1 | k>3),
     k=menu('Select type of fuzzy ELSE (Aggregation):', ..
            'and (Same as Fuzzy AND)', ..
            'or_ (Same as Fuzzy OR)', ..
         'sum (Sum of Consequents)');
  end
  types=['and';'or_';'sum'];
  else_type=types(k,:);
  report

elseif nargin==1,
   if isstr(a),
     if (any(a~='and') & any(a~='or_') & any(a~='sum'))
       error('Bad argument type')
     else else_type=a;
     end
   else x=felse(a,setstr(else_type));
   end

elseif nargin==2,
   [M N]=size(a);
   if M<2, x=a; else
     if b=='and', x=and(a);
     elseif b=='or_', x=or(a);
     elseif b=='sum', x=normal(sum(a));
     end
   end
end
```

function y=fuz(x,A);

```
% Fuzzifier. y=fuz(x,A) is the membership value of element x
% when the membership function is A. It's like evaluating the
% expression "x is A". x must be between 0 and 1. y is calculated
% by linear interpolation.
%
% See also NORMAL, POINTS, SEE, SEE_MEMB.

% Yannis Avrithis, 11-5-93

x=max(x,eps);
x=min(x,1-eps);
y1=(0:univ-1)/(univ-1);
y=(table1([y1' A'],x'))';
```

function Y=fuzzif(X,K)

```
% FUZZIFICATION. B=fuzzif(A,K), where A is a fuzzy set (vector of size
% (1,univ)), produces a fuzzified version of A, using the kernel K
% (B is something like the correlation of A and K). K should be any
% fuzzy set (gaussian, lorentzian, triangular, trapezoidal, zadeh's pi)
```

```
% centered at 0 (having its maximum value at 0). If K is omitted,
% K=gauss(0,.2) is assumed. Usually FUZZIF is used to fuzzify a singleton,
% producing a fuzzy set similar to K, but centered somewhere else.
%
% See also CONC, DIL, INTEN, SGLTON, SEE, AND, OR, NOT.

% Yannis Avrithis, 15-5-93.

if nargin==1, K=gauss(0,.2); end

X = X(:).';             % Make sure X is a row vector
K = K(:).';             % Make sure K is a row vector
n = max(size(X));
m = max(size(K));

if n<m
  K=K(1:n);
elseif n>m
  C=zeros(1,n);
  C(1:m)=K;
  K=C;
end

Y=zeros(1,n);           % Allocate Y

for i=1:n
  Y = max(Y, X(i) * [K(i:-1:2) K(1:n-i+1)] );
end
```

```
function y=gauss(a,s)

% Gaussian Function. gauss(a,s) produces a vector of size (1, univ)
% (see INIT) representing a gaussian membership function centered
% at a and with width s. a and s must be between 0 and 1.
%
% See also TRIANG, TRAPEZ, LORENTZ, SIGMA, ZETA, ZAD_PI, SGLTON, POINTS,
% SEE, SHFT, TEAM, MAT_TEAM, INIT.

% Yannis Avrithis, 11-5-93.

x=(0:univ-1) / (univ-1) ;

y=exp(-(x-a).^2 / ((s/2)^2) * log(2) );

y=max(y,y==max(y));
```

```
function Y=hedge(c,X);

if ~isstr(c),
  hedges  =  [  'very         '; ..
        'not          '; ..
        'more_or_less '; ..
        'somewhat     '; ..
        'sort_of      '; ..
        'kind_of      '; ..
        'pretty       '; ..
        'rather       '; ..
        'quite        '; ..
        'slightly     '; ..
        'strictly     '; ..
```

```
        'exceptionally'; ..
        'extremely    '; ..
        'hardly       ' ];
  c=hedges(c,:);
end

if     eq(c,'very'),          Y=conc(X);
elseif eq(c,'not'),           Y=not(X);
elseif eq(c,'more_or_less'),  Y=dil(X);
elseif eq(c,'somewhat'),      Y=dil2(X,3);
elseif eq(c,'sort_of'),       Y=nrm(and(inten(dil(X)),inten(dil(not(X)))));
elseif eq(c,'kind_of'),       Y=nrm(and(inten(dil(X)),inten(dil(not(X)))));
elseif eq(c,'pretty'),        Y=nrm(and(inten(X),not(inten(conc(X)))));
elseif eq(c,'rather'),        Y=nrm(and(inten(conc(X)),not(conc(X))));
elseif eq(c,'quite'),         Y=nrm(and(inten(conc(X)),not(conc(X))));
elseif eq(c,'slightly'),      Y=nrm(and(X,not(conc(X))));
elseif eq(c,'strictly'),      Y=inten(X);
elseif eq(c,'exceptionally'), Y=conc2(X);
elseif eq(c,'extremely'),     Y=conc2(X,4);
elseif eq(c,'hardly'),        Y=conc(not(X));

end
```

```
function x=impl(a,b,c)

% Fuzzy IMPLICATION operator. Syntax:
%     1. impl
%     2. impl(type)
%     3. impl(a, b)
%     4. impl(a, b, type)
%
% - Syntax 1 creates a menu for selection of the IMPL operator type.
% - Syntax 2 sets the IMPL type to a specific value without any promts.
%     Valid values for type are: 'min', 'yag', 'prd', 'gpr', 'bpr', 'dpr',
%     'luk', 'lee', 'god' and 'gog'.
% - Syntax 3 calculates a IMPL b, using the IMPL type stored in the global
%     variable 'impl_type'. a and b represent fuzzy sets and must be vectors
%     of size (1,univ), where univ is also a global variable. (See also INIT).
%     If a or b are scalars, they are expanded to the appropriate vectors and
%     a vector is returned. Otherwise a matrix is returned, containing all
%     possible combinations of elements a(i) IMPL b(i).
% - Syntax 4 is the same with syntax 3, but uses type instead of the
%     global variable 'impl_type'.
%
% See also INIT, REPORT, SHOW, COMP, FELSE.

% Yannis Avrithis, 11-5-93.

if nargin==0,
  k=0;
  while(k<1 | k>10),
     k=menu('Select type of fuzzy IMPLICATION:', ..
     'min (Minimum)', ..
     'yag (Yager)', ..
     'prd (Product)', ..
     'gpr (Gamma Product)', ..
     'bpr (Bounded Product)', ..
     'dpr (Drastic Product)', ..
     'luk (Lukaciewicz)', ..
     'lee (Lee)', ..
     'god (Godel)', ..
     'gog (Goguen)' );
  end
  types=['min';'yag';'prd';'gpr';'bpr';'dpr';'luk';'lee';'god';'gog'];
```

```
  impl_type=types(k,:);
  report

elseif nargin==1,
   if ~isstr(a), error('Bad argument type'), end
   if (any(a~='min') & any(a~='yag') & any(a~='prd') & any(a~='gpr') & ..
       any(a~='bpr') & any(a~='dpr') & any(a~='luk') & any(a~='lee') & ..
       any(a~='god') & any(a~='gog'))
   error('Bad argument type')
   else impl_type=a;
   end

elseif nargin>=2,
   if nargin==2, c=impl_type; end
   [Ma Na]=size(a);
   [Mb Nb]=size(b);
   if (Na==1 | Nb==1),
     if Na==1, a=a*ones(Mb,Nb); N=Nb;
     elseif Nb==1, b=b*ones(Ma,Na); N=Na; end
     if c=='min', x=min(a,b);
     elseif c=='yag', p=yag_par;
                      x=max(0 , 1 - ((1-a).^p + (1-b).^p) .^ (1/p));
     elseif c=='prd', x=a .* b;
     elseif c=='gpr', x=(a .* b) .^ (1-gamma_par);
     elseif c=='bpr', x=max(a+b-1,0);
     elseif c=='dpr', x=zeros(1,N); n=find(max(a,b)==1);
                        x(n)=min(a(n),b(n));
     elseif c=='luk', x=min(1,1-a+b);
     elseif c=='lee', x=max(1-a,b);
     elseif c=='god', x=ones(1,N); n=find(a>b); x(n)=b(n);
     elseif c=='gog', x=ones(1,N); n=find(a>b);
                        x(n)=b(n) ./ a(n);
     end
   else
     x=zeros(Na,Nb);
     for k=1:Na
       x(k,:)=impl(a(k),b,c);
     end
   end
end
```

```
% Initialization of Fuzzy Expert. INIT is a script file which creates and
% initializes the following global variables to their default values:

% Name        Default            Possible Values
% ------------------------------------------------------------------
% univ          300     Universe of discourse. Any integer, preferably
%                       between 20 and 600.
% gamma_par     0.3     gamma<1. Preferably -1 < gamma < 0.9 . Used
%                       only in 'gpr' AND or 'gsm' OR.
% yag_par         2     Any integer > 0. Used in 'yag' AND or OR.
% not_par         0     not_par > -1. Usually not_par=0. Used by NOT.
% and_type      'min'   'min', 'yag', 'prd', 'gpr', 'bpr' or 'dpr'.
% or_type       'max'   'max', 'yag', 'sum', 'gsm', 'bsm' or 'dsm'.
% impl_type     'min'   'min','prd','bpr','dpr','luk','lee','god','gog'.
% else_type     'or_'   'and', 'or_' or 'sum'.
% comp_type     'min'   Same as and_type.
% defuz_type    'cnt'   'cnt' or 'mom'.
% Variables univ, not_par and gamma_par can afterwards be changed very simply
% (e.g. 'univ=200'). The types of the various fuzzy operators can also
% be changed by executing the respective command (e.g. 'and') and then
% choosing from a menu.
%
% See also REPORT, SHOW, SEE, AND, OR, NOT, IMPL, FELSE, COMP, FUZ, DEFUZ.
```

```
% Yannis Avrithis, 11-5-93.

global and_type or_type impl_type else_type comp_type defuz_type ..
       univ gamma_par yag_par not_par

univ=300;
gamma_par=0.3;
yag_par=2;
not_par=0;
and_type='min';
or_type='max';
impl_type='min';
else_type='or_';
comp_type='min';
defuz_type='cnt';

report
```

function Y=inten(X,n)

```
% CONTRAST INTENSIFICATION of a fuzzy set. Y=inten(X,n) (n>1) is a
% function similar to ZADEH_F but more general. Y=inten(X) assumes n=2
% and is the same as Y=zadeh_f(X).
%
% See also ZADEF_F, CONC, DIL, CONC2, DIL2, INTEN2, FUZZIF, AND, OR, NOT.

% Yannis Avrithis, 15-5-93.

if nargin==1, n=2; end
[M N]=size(X); Y=zeros(M,N);

f=find(0<=X & X<0.5);
Y(f)=(.5)^(1-n) * X(f) .^ n;

f=find(0.5<=X & X<=1);
Y(f)=1 - (.5)^(1-n) * (1-X(f)) .^ n;
```

function Y=inten2(X,a)

```
% CONTRAST INTENSIFICATION of a fuzzy set. Y=inten2(X,a) (a>0) is a
% concentration function based on CONC2 (and DIL2). The function is
% defined as follows:
%
%          x                                    1-x
% y = -----------  (0<=x<0.5), and y = 1 -  -----------  (0.5<=x<=1)
%     1+a*(0.5-x)                          1+a*(x-0.5)
%
% where x is membership value of fuzzy set X, and y is the respective
% value of Y. Y=inten2(X) assumes a=2. Do not use a<0.
%
% See also CONC, DIL, INTEN, CONC2, DIL2, FUZZIF, AND, OR, NOT.

% Yannis Avrithis, 15-5-93.

if nargin==1, a=2; end
[M N]=size(X); Y=zeros(M,N);

f=find(0<=X & X<0.5);
Y(f)=X(f) ./ (1+a*(0.5-X(f)));
```

```
f=find(0.5<=X & X<=1);
Y(f)=1 - (1-X(f)) ./ (1+a*(X(f)-0.5));
```

function y=lorentz(a,s,p)

```
% Lorentzian Function. lorentz(a,s,p) produces a vector of size (1, univ)
% (see INIT) representing a lorentzian membership function centered
% at a and with width s and contrast (fuzziness) p. a and s must be between
% 0 and 1, while p must be >0. p is assumed 2 if it is omitted.
%
% See also TRIANG, TRAPEZ, GAUSS, SIGMA, ZETA, ZAD_PI, SGLTON, POINTS, SEE,
% SHFT, TEAM, MAT_TEAM, INIT.

% Yannis Avrithis, 12-5-93.

x=(0:univ-1) / (univ-1) ;

if nargin<3, p=2; end
y=1 ./ ( 1 + ( abs(x-a) / (s/2) ) .^p );

y=max(y,y==max(y));
```

function y=mat_team(n,c,s,p)

```
% Definition of membership functions as a team, but in the form
% of a matrix. y=mat_team(n,c,s,p) creates a matrix with n rows,
% representing n membership functions. See TEAM for details on
% parameters c,s,p. At least 2 parameters, n and c, must be declared.
%
% See also TEAM, SEE, TRIANG, TRAPEZ, GAUSS, LORENTZ, SIGMA, ZETA, ZAD_PI,
% SGLTON.

% Yannis Avrithis, 11-5-93.


if nargin<3, s=1/(n-1); end
if nargin>=3, if isstr(s),
   [m1 n1]=size(s);
   if s(n1)=='%', eval(['s=0.5*(1+',s(1:n1-1),'/100)/(n-1);']);
   else s=1/(n-1); end
end, end
if nargin<4, p=(2/3)/(n-1); end
for k=1:n
  a=(k-1)/(n-1);
  com=['y=[y; ',c];
  if c(1:4)=='sglt',                    % Singleton
    com=[com,'([0 1],a)];'];
  elseif c(1:4)=='trap',                % Trapezoidal
    com=[com,'(a,s,p)];'];
  else
    com=[com,'(a,s)];'];
  end
  eval(com)
end
```

```
function y=normal(x,S)

% Normalization of inputs from the region S=[a,b] to [0,1]
% Syntax: y=normal(x,S) or y=normal(x). x may be a scalar,
% vector or matrix, but its elements must belong to the region [a,b].
% If S is omitted, S=[min(x), max(x)] is assumed.
%
% See also CLIP, NRM, DENORMAL, CONVEX, FUZ.

% Yannis Avrithis, 11-5-93.

if nargin==1,
  a=min(min(x));
  b=max(max(x));
elseif nargin==2,
  a=S(1);
  b=S(2);
end

% if (x<a | x>b), error('Bad argument value'), end
y=(x-a) ./ (b-a);
```

```
function y=not(x,a)

% Fuzzy NOT (Generalized negation) operator. y=not(x,a) calculates
% NOT x, that is, (1-x)/(1+a*x), where x may be a scalar, vector or
% matrix, and a is a parameter that must be greater than -1. If a is
% omitted, a=not_par is assumed, where not_par is a global variable
% (See INIT). If a=0, then (NOT x) = 1-x.
%
% See also INIT, REPORT, AND, OR, CONC, DIL, INTEN, FUZZIF, ENTROPY, SUB, SUP.

% Yannis Avrithis, 12-5-93.

if nargin==1, a=not_par; end
y=(1-x)./(1+a*x);
```

```
function Y=nrm(X)

% Normalization of fuzzy sets. Y=nrm(X), where X may be a scalar,
% vector or matrix, devides X by its maximum element so that the maximum
% element of Y is always 1.
%
% See also CLIP, NORMAL, DENORMAL, CONVEX, FUZ.

% Yannis Avrithis, 24-5-93.

Y=X ./ max(X);
```

```
function x=or(a,b,c)

% Fuzzy OR operator. Syntax:
%     1. or
%     2. or(type)
%     3. or(a, b)
%     4. or(a, b, type)
```

```
%     5. or([a1; a2; ... ; aN])
%     6. or([a1; a2; ... ; aN], type)
%
% - Syntax 1 creates a menu for selection of the OR operator type.
% - Syntax 2 sets the OR type to a specific value without any promts.
%     Valid values for type are: 'max', 'yag', 'sum', 'gsm', 'bsm' and 'dsm'.
% - Syntax 3 calculates a OR b, using the OR type stored in the global
%     variable 'or_type'. a and b represent fuzzy sets and must be vectors
%     of size (1,univ), where univ is also a global variable. See also INIT.
% - Syntax 4 is the same with syntax 3, but uses type instead of the
%     global variable 'or_type'.
% - Syntax 5 and 6 are the same with syntax 3 and 4, but calculate
%     a1 OR a2 OR ... OR aN.
%
% See also INIT, REPORT, SHOW, AND, NOT, FUZ, IMPL, ENTROPY, SUB, SUP, RANGE.

% Yannis Avrithis, 11-5-93.

if nargin==0,
  k=0;
  while(k<1 | k>6),
   k=menu('Select type of fuzzy OR:', ..
          'max (Maximum)', ..
          'yag (Yager t-norm)', ..
          'sum (Probabilistic Sum)', ..
          'gsm (Gamma Sum)', ..
          'bsm (Bounded Sum)', ..
          'dsm (Drastic Sum)' );
  end
  types=['max';'yag';'sum';'gsm';'bsm';'dsm'];
  or_type=types(k,:);
  report

elseif nargin==1,
   if isstr(a),
     if (any(a~='max') & any(a~='yag') & any(a~='sum') & any(a~='gsm') & ..
         any(a~='bsm') & any(a~='dsm'))
       error('Bad argument type')
     else or_type=a;
     end
   else x=or(a,setstr(or_type));
   end

elseif nargin==2,
   if isstr(b),
     [M N]=size(a);
     if M<2, error('Bad argument type'), end
     if b=='max', x=max(a);
     elseif b=='yag', p=yag_par;
                      x=min(1,sum(a.^p).^(1/p));
     elseif b=='sum', x=1-prod(1-a);
     elseif b=='gsm', x=1-prod(1-a).^(1-gamma_par);
     elseif b=='bsm', x=min(1,sum(a));
     elseif b=='dsm', x=ones(1,N);
                      s=sum(a==0);
              f1=find(s==M);
              [M1 N1]=size(f1);
              x(f1)=zeros(1,N1);
              f2=find(s==M-1);
              t=max(a);
                      x(f2)=t(f2);
     end
   else x=or(a,b,setstr(or_type));
   end

elseif nargin==3,
   [Ma Na]=size(a);
   [Mb Nb]=size(b);
```

```
   if [Ma Na]==[1 1], a=a*ones(Mb,Nb);
   elseif [Mb Nb]==[1 1], b=b*ones(Ma,Na);
   elseif Na~=Nb, error('Bad argument type')
   end
   x=or([a;b],c);

end



function DX=pend(THETA,DTHETA)

% FVAR THETA, DTHETA;
% CONSEQUENTS DX;

THETA=normal(THETA,[-90,90]);
DTHETA=normal(DTHETA,[-40,40]);


[NM, NS, ZE, PS, PM]=team('trapez');
[NMD,NSD,ZED,PSD,PMD]=team('trapez');
[NMX,NSX,ZEX,PSX,PMX]=team('trapez');

% TEAM NM, NS, ZE, PS, PM [-90,90] theta;
% TEAM NMD,NSD,ZED,PSD,PMD [-40,40] dtheta;
% TEAM NMX,NSX,ZEX,PSX,PMX [-0.2,0.2] dx;

R=[];

R=rule(R,impl(and(fuz(THETA,ZE),fuz(DTHETA,ZED)),ZEX));

% IF (THETA IS ZE) AND (DTHETA IS ZED) THEN DX IS ZEX;


R=rule(R,impl(and(fuz(THETA,PS),fuz(DTHETA,ZED)),NSX));
R=rule(R,impl(and(fuz(THETA,PM),fuz(DTHETA,ZED)),NMX));
R=rule(R,impl(and(fuz(THETA,NS),fuz(DTHETA,ZED)),PSX));
R=rule(R,impl(and(fuz(THETA,NM),fuz(DTHETA,ZED)),PMX));

% IF (THETA IS PS) AND (DTHETA IS ZED) THEN DX IS NSX;
% IF (THETA IS PM) AND (DTHETA IS ZED) THEN DX IS NMX;
% IF (THETA IS NS) AND (DTHETA IS ZED) THEN DX IS PSX;
% IF (THETA IS NM) AND (DTHETA IS ZED) THEN DX IS PMX;


R=rule(R,impl(and(fuz(THETA,ZE),fuz(DTHETA,NSD)),PSX));
R=rule(R,impl(and(fuz(THETA,ZE),fuz(DTHETA,NMD)),PMX));
R=rule(R,impl(and(fuz(THETA,ZE),fuz(DTHETA,PSD)),NSX));
R=rule(R,impl(and(fuz(THETA,ZE),fuz(DTHETA,PMD)),NMX));

% IF (THETA IS ZE) AND (DTHETA IS NSD) THEN DX IS PSX;
% IF (THETA IS ZE) AND (DTHETA IS NMD) THEN DX IS PMX;
% IF (THETA IS ZE) AND (DTHETA IS PSD) THEN DX IS NSX;
% IF (THETA IS ZE) AND (DTHETA IS PMD) THEN DX IS NMX;


R=rule(R,impl(and(fuz(THETA,PS),fuz(DTHETA,NSD)),ZEX));
R=rule(R,impl(and(fuz(THETA,NS),fuz(DTHETA,PSD)),ZEX));

% IF (THETA IS PS) AND (DTHETA IS NSD) THEN DX IS ZEX;
% IF (THETA IS NS) AND (DTHETA IS PSD) THEN DX IS ZEX;


DX=see_def(R);
%DX=defuz(R);
DX=denormal(DX,[-0.2,0.2]);
```

function y=points(S,X,int_type)

```
% Construction of a membership function by giving values directly,
% using linear or biharmonic interpolation. Syntax:
%    1. y=points(S,X)
%    2. y=points(X)
%    3. y=points(S,X,int_type)
%    4. y=points(X,int_type)
% where S is the domain of the respective variable (assumed [0,1] if
% omitted), and X is a matrix containing the appropriate values:
%
% X=[  x1    x2   ...  xN;
%     m(x1) m(x2) ... m(xN) ]
%
% Linear interpolation is used if int_type='l' or if int_type is omitted,
% and biharmonic interpolation if int_type='h'.
%
% See also FUZ, SEE, SEE_MEMB, TRIANG, TRAPEZ, GAUSS, LORENTZ, SIGMA, ZETA,
% ZAD_PI, SGLTON, TEAM, RESIZE.

% Yannis Avrithis, 11-5-93.

if nargin==1,
  X=S; S=[0 1]; int_type='l';
end

if nargin==2,
  if isstr(X), int_type=X; X=S; S=[0 1];
  else int_type='l'; end
end

[M N]=size(X);
n=(0:univ-1)/(univ-1);
X(1,:)=normal(X(1,:),S);
if X(1,N)~=1, X=[X [1; 0]]; end
if X(1,1)~=0, X=[[0; 0] X]; end
if int_type=='h',

  % Biharmonic interpolation
  y=clip( interp2( X(1,:)', X(2,:)', n' )' );

else

  % Linear interpolation
  y=clip( table1(X',n')' );

end
```

function C=range(A,B)

```
% C=range(A,B), where A, B are two (usually normal) fuzzy sets,
% produces the convex fuzzy set CONVEX(A OR B), where OR is usually
% set to maximum. If, for example, A is approximately 0.3 and B is
% approximately 0.6, then C is true for all the values between
% approximately 0.3 and 0.6.
%
% See also OR, CONVEX, HEDGE.

% Yannis Avrithis, 24-5-93.

C=convex(or(A,B));
```

```
% COMPUTING WITH FUZZY SETS. The following M-files are available:
%
% - INIT, REPORT
% - SEE, SEE_DEF, SEE_MEMB, SHOW
% - POINTS, GAUSS, LORENTZ, SGLTON, TRIANG, TRAPEZ, SIGMA, ZETA,
%   ZAD_PI, ZADEH_F
% - TEAM, MAT_TEAM
% - CLIP, CONVEX, NRM, NORMAL, DENORMAL, SHFT, RESIZE
% - CONC, DIL, INTEN, CONC2, DIL2, INTEN2, FUZZIF, HEDGE, RANGE
% - ENTROPY, SUB, SUP
% - FUZ, AND, OR, NOT, IMPL, COMP, COMP2, RULE, FELSE, DEFUZ
%
% Applications: TEST1, TEST2, TEST3, TEST4, PEND
%
% Auxiliary: EQ
%
% Help: README
```

```
% Status Report. REPORT is a script file that displays the current
% values of the global variables that are used to control the various
% parameters and operator types of Fuzzy Expert. See INIT for the names,
% defaults and possible values of these variables.
%
% See also README, INIT, AND, OR, NOT, IMPL, COMP, FELSE, FUZ, DEFUZ.

% Yannis Avrithis, 11-5-93.

disp([' '])
disp('Selections:')
disp('-----------')
disp(['univ:       ' num2str(univ)])
disp(['gamma_par:  ' num2str(gamma_par)])
disp(['yag_par:    ' num2str(yag_par)])
disp(['not_par:    ' num2str(not_par)])
disp(['and_type:   ' and_type])
disp(['or_type:    ' or_type])
disp(['impl_type:  ' impl_type])
disp(['else_type:  ' else_type])
disp(['comp_type:  ' comp_type])
disp(['defuz_type: ' defuz_type])
disp([' '])
```

```
function yi=resize(y,int_type)

% Previously created membership functions of size different than the
% the current value of univ cannot be used and must be resized.
% Y=resize(X,int_type) resizes vector X so that it has size (1,univ).
% Linear interpolation is used if int_type='l', and biharmonic
% interpolation if int_type='h'. Omitting int_type assumes 'l'.
%
% See also INIT, CLIP, NORMAL, SEE, MEM_FUNC.

% Yannis Avrithis, 12-5-93.

y=y(:);
m=length(y);
```

```
x=((0:m-1)/(m-1))';
xi=((0:univ-1)/(univ-1))';
if nargin<2, int_type='l'; end
if int_type=='h',

  % Biharmonic Interpolation
  yi=clip(interp2(x,y,xi)');

else

  % Linear Interpolation
  yi=clip(table1([x y], xi)');

end
```

function R=rule(R,C)

```
% R=rule(R,C) adds the consequent C of a rule to the rule consequent
% matrix R. C represents a fuzzy set and must be a vector of size
% (1, univ) where univ is a global variable.
%
% See also FELSE, DEFUZ.

% Yannis Avrithis, 11-5-93.

if exist('R')~=1, R=[]; end
R=[R; C];
```

function see(x1,x2,x3,x4,x5,x6,x7,x8,x9)

```
% Shows membership functions. Syntax:
%   see(S, x1, x2, ...) or
%   see(S, [x1; x2; ...])
% x1, x2, ... represent fuzzy sets and must be vectors of size (1, univ)
% (see INIT for univ). They are plotted versus the region S=[a b].
% If S is omitted, it is assumed [0 1].
%
% See also SEE_MEMB, SEE_DEF, POINTS, TRIANG, TRAPEZ, GAUSS, LORENTZ,
% SIGMA, ZETA, ZAD_PI, SGLTON, SHFT, CONC, DIL, INTEN, FUZZIF, SHOW.

% Yannis Avrithis, 11-5-93.

if nargin==0, shg
else
  X=[];
  [m n]=size(x1);
  if n==2, l=2; S=x1; else l=1; S=[0 1]; end
  for k=1:nargin
    com=['X=[X; x', num2str(k), '];'];
    eval(com)
  end
  N=denormal((0:univ-1)/(univ-1),S);
  axis([S 0 1])
  plot(N, X)
  axis;
end
```

```
function x=see_def(A)

% Shows Defuzzification procedure. Same syntax as DEFUZ.
%
% See also DEFUZ, SEE, SEE_MEMB.

% Yannis Avrithis, 11-5-93.

a=felse(A);
see([A;a])
hold on
x=defuz(a);
plot([x x],[0 1],'w:');
hold off
```

```
function see_memb(A,x1,x2,x3,x4,x5,x6,x7,x8,x9)

% SEE_MEMB shows membership function A and membership of values
% of xi (0<=xi<=1). Syntax: see_memb(A, x1, x2, ...).
%
% See also SEE, SEE_DEF, POINTS, FUZ.

% Yannis Avrithis, 11-5-93.

see(A)
hold on
for k=1:nargin-1
  com=['x=x',num2str(k),';'];
  eval(com)
  y=fuz(x,A);
  X=[x x 0];
  Y=[0 y y];
  plot(X,Y,'w:')
end
hold off
```

```
function X=sglton(S,x)

% Definition of a membership function as a singleton. Syntax:
%    1. A=sglton(S,x)
%    2. A=sglton(x)
% where S is the domain of the respective variable (assumed [0,1] if
% omitted), and x is the value (belonging in S) at which you want A
% to be 1. The fuzzy set A looks like:
%
%   A = [  0  0  0  ....  1  ....  0  0  ]
%          a     ....     x    ....   b
%
% where S=[a b]. x can also be a horizontal vector containing multiple
% values at which you want A to be 1. The resulting A is then a vector
% of 1's and 0's.
%
% See also SEE, TRIANG, TRAPEZ, GAUSS, LORENTZ, SIGMA, ZETA,
% ZAD_PI, POINTS, SHFT, TEAM, FUZZIF, FUZ, COMP.

% Yannis Avrithis, 16-5-93.

if nargin==1,
  x=S; S=[0 1];
end
```

```
[m,n]=size(x);
x=normal(x,S);
x=denormal(x,[1 univ]);
X=zeros(1,univ);
X(x)=ones(m,n);
```

function y=shft(x,t,S)

```
% Y=shft(X,t,S) is the shifted version of a 1*N vector X.
% t is the lag, which is included in the region S. t is normalized
% to the region [0 1], and all new elements equal zero.
%
% See also POINTS, CONC, DIL, INTEN, FUZZIF, SEE.

% Yannis Avrithis, 22-4-93.

[M,N]=size(x);
if M>1, error('Bad argument type'), end
if nargin==2, S=[0 1]; end
t=normal(t,S);
t=denormal(t,[0 N]);
y=zeros(1,3*N);
n=N+1:2*N;
y(n+t)=x;
y=y(n);
```

function R=show(c,n)

```
% R=show(c,n) demonstrates in various ways the behaviour of operators
% AND, OR and IMPL. c can be 'and', 'or' or 'impl' and n can be 1, 2 or 3.
% The current types of operators are used.
%
% - If n=1, a matrix is returned, containing the result of the operator
%   for all the combinations of Boolean inputs (0 or 1).
%
% - If n=2, the result of the operator for all the combinations of
%   inputs (between 0 and 1) is displayed in the form of multiple 2-D
%   curves.
%
% - If n=3, the result of the operator for all the combinations of
%   inputs (between 0 and 1) is displayed in the form of a 3-D diagram.
%
% In the 2 last cases, univ should be rather small (about 20).
%
% See also INIT, REPORT, AND, OR, IMPL, SEE.

% Yannis Avrithis, 11-5-93.

if nargin==1, n=2; end

if n==1,
  for k=0:1
    R=[R [k k; 0 1; feval(c,k,[0 1])]];
  end
  R=R';

else
  R=zeros(univ,univ);
  x=(0:univ-1)/(univ-1);
  for k=(0:univ-1)/(univ-1)
```

```
    R(:,k*(univ-1)+1)=feval(c,k,x)';
  end

  if n==2, plot(x,R)

  elseif n==3, mesh(R,[-20 60])
  end
end
```

```
function y=sigma(a,s)

% Sigma Function. sigma(a,s) produces a vector of size (1, univ)
% (see INIT) representing a 'sigma' membership function centered
% at a and with width s. a and s must be between 0 and 1.
%
% See also TRIANG, TRAPEZ, GAUSS, LORENTZ, ZETA, ZAD_PI, SGLTON,
% POINTS, SEE, SHFT, TEAM, MAT_TEAM, INIT.

% Yannis Avrithis, 11-5-93.

x=(0:univ-1) / (univ-1) ;
y=zeros(1,univ);

f=find(a-s<x & x<=a);
y(f)=zadeh_f(1+(x(f)-a)/s);

y=max(y,a<x);

y=max(y,y==max(y));
```

```
function S=sub(A,B);

% Fuzzy SUBSETHOOD. S=sub(A,B), where A, B represent fuzzy sets (and are
% usually vectors of size (1,univ)), measures the fuzzy subsethood of A
% in B, i.e. the degree to which A is a subset of B. Due to the subsethood
% theorem,
%
%            M(A and B)
%    S(A,B)=------------
%               M(A)
%
% where and_type='min', or_type='max' and M(.) is the cardinality measure
% or sigma-count of a fuzzy set (here norm(.,1) is used). Note that for
% every A, S(A,0)=0, S(0,A)=1, S(A,X)=1, where 0 denotes the empty set
% and X the domain. Note also that SUPERSETHOOD(A,B)=1-S(A,B).
%
% Source: Bart Kosko, "Neural Networks & Fuzzy Systems", 1992.
% See also INIT, REPORT, AND, OR, NOT, ENTROPY, SUP.

% Yannis Avrithis, 12-5-93.

S=norm(and(A,B),1)/norm(A,1);
```

```
function S=sup(A,B);

% Fuzzy SUPERSETHOOD. S=sup(A,B), where A, B represent fuzzy sets (and are
```

```
% usually vectors of size (1,univ)), measures the fuzzy supersethood of B
% in A, i.e. the degree to which A is a superset of B. Due to the subsethood
% theorem,
%
%                                          M(A and B)
%    SUPERSETHOOD(A,B) = 1-S(A,B) = 1 -  ------------
%                                             M(A)
%
% where and_type='min', or_type='max', M(.) is the cardinality measure
% or sigma-count of a fuzzy set (here norm(.,1) is used), and S(A,B)
% is the subsethood of A in B, i.e. the degree to which A is a subset of B.
%
% Source: Bart Kosko, "Neural Networks & Fuzzy Systems", 1992.
% See also INIT, REPORT, AND, OR, NOT, ENTROPY, SUB.

% Yannis Avrithis, 12-5-93.

S=1-norm(and(A,B),1)/norm(A,1);
```

```
function [y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11,y12]=team(c,s,p)

% Definition of membership functions as a team. Syntax:
%     1. [y1, y2, ... yN] = team (c)
%     2. [y1, y2, ... yN] = team (c, s)
%     3. [y1, y2, ... yN] = team (c, s, p)
%     4. [y1, y2, ... yN] = team (c, 'string', p)
% - Syntax 1 creates N membership functions, which devide the region [0, 1]
%     into N fuzzy regions of equal width. c is a string declaring the
%     type of membership functions to be used, and can be 'triang',
%     'trapez', 'gauss', 'sigma', 'zeta' or 'zad_pi'.
% - Syntax 2 is the same, but s defines the width of the membership
%     functions. s can be omitted, as in syntax 1, in which case it is
%     assumed that s=1/(N-1). It can also be a string of the form 's%',
%     meaning that there is s% overlap between the functions.
% - Syntax 3 and 4 are used only in the case of trapezoidal functions,
%     in which case p denotes the top width. p can be omitted, as in
%     syntax 1, 2 or 3, in which case it is assumed that p=(2/3)/(N-1).
%     It can also be a string of the form 'p%' (see TRAPEZ for more
%     details). In case you want to define p but not s, you can use
%     syntax 5, where 'string' is an arbitrary string, but not of the
%     form 's%'. In that case, 's%' is interpreted as in syntax 2.
% See also MAT_TEAM, SEE, TRIANG, TRAPEZ, GAUSS, LORENTZ, SIGMA, ZETA, ZAD_PI,
%          SGLTON.

% Yannis Avrithis, 11-5-93.

N=nargout;
if nargin<2, s=1/(N-1); end
if nargin>=2, if isstr(s),
   [m1 n1]=size(s);
   if s(n1)=='%', eval(['s=0.5*(1+',s(1:n1-1),'/100)/(N-1);']);
   else s=1/(N-1); end
end, end
if nargin<3, p=(2/3)/(N-1); end
for k=1:N
  a=(k-1)/(N-1);
  com=['y',num2str(k),'=',c];
  if c(1:4)=='sglt',                    % Singleton
    com=[com,'([0 1],a);'];
  elseif c(1:4)=='trap',                % Trapezoidal
    com=[com,'(a,s,p);'];
  else
    com=[com,'(a,s);'];
  end
```

```
  eval(com)
end



function E=test1(b)

% TEST1. E=test1(b) takes as input a scalar value b and produces two
% matrices representing the following equivalent propositions:
%    R1:  a1 -> (a2 -> b)
%    R2:  (a1 AND a2) -> b
% for all values of a1, a2 between 0 and 1. The 2 matrices (which have
% size (univ,univ)) are plotted and the maximum error E between them
% is returned. It can be confirmed that the following types of AND
% should be chosen with the respective types of IMPLICATION ( -> ),
% if we want the results of R1, R2 to be the same:
%
%     IMPLICATION     ==>      AND
% ------------------------------------------
%         min          |         min
%       product        |       product
%   bounded product    |    bounded procuct
%   drastic product    |    drastic product
%     Lukeciewicz      |      Lukeciewicz
%         Lee          |         min
%        Godel         |       product
%        Goguen        |         min
%
% See also IMPL, AND.

% Yannis Avrithis, 13-5-93.

  R1=zeros(univ,univ);
  R2=R1;
  a1=(0:univ-1)/(univ-1);
  a2=a1;
  R1=impl(a1,impl(a2,b));
  for a2=a2
    R2(:,a2*(univ-1)+1)=impl(and(a1,a2),b)';
  end
  clg
  subplot(221)
  E=max(max(abs(R1-R2)));
  title(num2str(E));
  mesh(R1,[-20 60])
  mesh(R2,[-20 60])
  plot(R1)
  plot(R2)




% TEST2. Composition test for 1 antecedent.
% Try any combination of implication and composition to see
% if comp(A1,A,C) = max(A1 o A) -> C     (1),
% where comp(A1,A,C) = max( A1 o (A -> C) ),
% and 'o' means composition.
% (1) is true if the implication type is min, prd, bpr, dpr or yag
% and the composition type is the same as the implication type.
%
% See also IMPL, COMP, TEST3.

% Yannis Avrithis, 31-5-93.

univ=10;
```

```
A1=trapez(.2,.2,.2);
A=trapez(.4,.2,.2);
M=and(A1,A,setstr(comp_type));
see(A1,A,M)
hold on
univ=50;
C=lorentz(.5,.2);
C1=comp(A1,A,C);
C2=impl(max(M),C);
N=(0:univ-1)/(univ-1);
plot(N,[C;C1;C2],'--')
title([num2str(max(M)),'  ', num2str(max(C1)),'  ', num2str(max(C2))])
hold off
```

```
% TEST3. Composition test for 2 antecedents.
% Try any combination of AND, implication and composition to see
% if comp(A1,A,B1,B,C)=(max(A1 o A) AND max(B1 o B)) -> C    (1),
% where comp(A1,A,B1,B,C)=max( (A1 AND B1) o ((A AND B) -> C) ),
% and 'o' means composition.
% (1) is true if the implication type is min, prd, bpr, dpr or yag
% and the AND and composition types are the same as the implication type.
%
% See also AND, IMPL, COMP, COMP2, TEST2, TEST4, TEST5.

% Yannis Avrithis, 31-5-93.

univ=10;
A1=trapez(.2,.2,.2);
A=trapez(.4,.2,.2);
B1=A1;
B=A;
M=and(A1,A,setstr(comp_type));
see(A1,A,M)
hold on
univ=50;
C=lorentz(.5,.2);
C1=comp2(A1,A,B1,B,C);
C2=impl(and(max(M),max(M)),C);
N=(0:univ-1)/(univ-1);
plot(N,[C;C1;C2],'--')
title([num2str(max(M)),'  ', num2str(max(C1)),'  ', num2str(max(C2))])
hold off
```

```
% TEST4. Rule Decomposition test for 2 antecedents.
% Try any combination of AND, implication and composition to see
% if comp(A1,A,B1,B,C)=comp(A1,A,C) AND comp(B1,B,C)    (1),
% where comp(A1,A,B1,B,C)=max( (A1 AND B1) o ((A AND B) -> C) ),
% comp(A1,A,C)=max(A1 o (A -> C)),  and 'o' means composition.
%
% See also AND, IMPL, COMP, COMP2, TEST2, TEST3.

% Yannis Avrithis, 31-5-93.

univ=10;
A1=trapez(.2,.2,.2);
A=trapez(.4,.2,.2);
B1=A1;
B=A;
M=and(A1,A);
see(A1,A,M)
```

```
hold on
univ=50;
C=lorentz(.5,.2);
C1=comp2(A1,A,B1,B,C);
C2=and(comp(A1,A,C),comp(B1,B,C));
N=(0:univ-1)/(univ-1);
plot(N,[C;C1;C2],'--')
title([num2str(max(M)),'  ', num2str(max(C1)),'  ', num2str(max(C2))])
hold off
function y=test10(a,b,c,d)



% TEST 5. Tests to see if (a OR b) o (c OR d) = (a o c) OR (b o d)
% for all a,b,c,d, where 'o' means COMPOSITION (same as AND). This
% would be true if all the elements of each row of the resulting matrix
% were equal. Unfortunately, this doesn't happen for any OR or AND type.
%
% See also AND, OR, COMP, TEST1, TEST3.

% Yannis Avrithis, 5-6-93.

y=[and(or(a,b),or(c,d)), or(and(a,c),and(b,d)), or(and(a,d),and(b,c)); ..
   and(or(a,c),or(b,d)), or(and(a,b),and(c,d)), or(and(a,d),and(c,b)); ..
   and(or(a,d),or(b,c)), or(and(a,b),and(d,c)), or(and(a,c),and(d,b)) ];



function y=trapez(a,s,p)

% Trapezoidal Function. trapez(a,s,p) produces a vector of size (1, univ)
% (see INIT) representing a trapezoidal membership function centered
% at a and with base width s and top width p. a and s must be between
% 0 and 1. p can also be a string of the form 'P%'. For example,
% p='60%' means that the top width is 60% of the base width.
%
% See also TRIANG, GAUSS, LORENTZ, SIGMA, ZETA, ZAD_PI, SGLTON, POINTS, SEE,
% SHFT, TEAM, MAT_TEAM, INIT.

% Yannis Avrithis, 11-5-93.

x=(0:univ-1) / (univ-1) ;
if isstr(p),
  [m1 n1]=size(p);
  eval(['p=(',p(1:n1-1),'/100)*(2*s);'])
end
p=min(p,2*s);
y=zeros(1,univ);

f=find(a-s<x & x<=min(a-p/2,a));
y(f)=(x(f)-a+s)/(s-p/2+eps);

y=max(y,(a-p/2<x & x<=a+p/2));

f=find(max(a+p/2,a)<x & x<=a+s);
y(f)=(a+s-x(f))/(s-p/2+eps);

y=max(y,y==max(y));
```

```
function y=triang(a,s)

% Triangural Function. triang(a,s) produces a vector of size (1, univ)
% (see INIT) representing a triangular membership function centered
% at a and with width s. a and s must be between 0 and 1.
%
% See also TRAPEZ, GAUSS, LORENTZ, SIGMA, ZETA, ZAD_PI, SGLTON, POINTS, SEE,
% SHFT, TEAM, MAT_TEAM, INIT.

% Yannis Avrithis, 11-5-93.

x=(0:univ-1) / (univ-1) ;
y=zeros(1,univ);

f=find(a-s<x & x<=a);
y(f)=1+(x(f)-a)/s;

f=find(a<x & x<=a+s);
y(f)=(a+s-x(f))/s;

y=max(y,y==max(y));
```

```
function y=zad_pi(a,s)

% Zadeh's Pi Function. zad_pi(a,s) produces a vector of size (1, univ)
% (see INIT) representing a Zadeh's 'PI' membership function centered
% at a and with width s. a and s must be between 0 and 1.
%
% See also TRIANG, TRAPEZ, GAUSS, LORENTZ, SIGMA, ZETA, ZADEH_F, SGLTON,
% POINTS, SEE, SHFT, TEAM, MAT_TEAM, INIT.

% Yannis Avrithis, 11-5-93.

x=(0:univ-1) / (univ-1) ;

f=find(x<=a);
y1=sigma(a,s);
y(f)=y1(f);

f=find(a<x);
y1=zeta(a,s);
y(f)=y1(f);

y=max(y,y==max(y));
```

```
function y=zadeh_f(x);

% Y=zadeh_f(X) is the function f, used by functions sigma, zeta and
% zad_pi (Zadeh's Pi). X must be between 0 and 1. When X is a vector,
% Y is a vector of the same size.
%
% See also SIGMA, ZETA, ZAD_PI, INTEN.

% Yannis Avrithis, 11-5-93.

if isempty(x), y=[]; return, end
[M N]=size(x); y=zeros(M,N);

f=find(0<=x & x<0.5);
y(f)=2*x(f).^2;
```

```
f=find(0.5<=x & x<=1);
y(f)=1-2*(1-x(f)).^2;
```

```
function y=zeta(a,s)

% Zeta Function. zeta(a,s) produces a vector of size (1, univ)
% (see INIT) representing a 'zeta' membership function centered
% at a and with width s. a and s must be between 0 and 1.
%
% See also TRIANG, TRAPEZ, GAUSS, LORENTZ, SIGMA, ZAD_PI, SGLTON, POINTS,
% SEE, SHFT, TEAM, MAT_TEAM, INIT.

% Yannis Avrithis, 11-5-93.

x=(0:univ-1) / (univ-1) ;
y=1-sigma(a+s,s);
```