Ατομική Διπλωματική Εργασία

ΕΠΙΧΕΙΡΗΜΑΤΟΛΟΓΙΑ ΜΕ ΕΛΛΙΠΗ ΓΝΩΣΗ

Ραφαέλα Λουκά

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2016

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΠΙΧΕΙΡΗΜΑΤΟΛΟΓΙΑ ΜΕ ΕΛΛΙΠΗ ΓΝΩΣΗ

Ραφαέλα Λουκά

Επιβλέπων Καθηγητής
Γιάννης Δημόπουλος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2016

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής μου εργασίας Δρ. Γιάννη Δημόπουλο για τη καθοδήγηση του και την πολύτιμη βοήθεια που μου πρόσφερε κατά την διάρκεια της διπλωματικής μου εργασίας.

Θα ήθελα επίσης να ευχαριστήσω την οικογένεια μου, και ιδιαίτερα τους γονείς μου Παναγιώτη και Ειρήνη, για την ηθική και οικονομική υποστήριξη τους στα τέσσερα χρόνια σπουδών μου στο Πανεπιστήμιο Κύπρου.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ στη φίλη και συμφοιτήτρια μου Ευανθία Τιγγίρη για την αμέριστη συμπαράσταση και στήριξη της.

Περίληψη

Τα τελευταία χρόνια, η επιχειρηματολογία αποτελεί κυρίαρχο μέρος στην μοντελοποίηση συστημάτων πολλαπλών πρακτόρων. Υπάρχουν διάφοροι τύποι διαλόγων, οι οποίοι έχουν ένα κοινό στόχο. Ο στόχος αυτό είναι ένας πράκτορας Χ να αλλάξει τις πεποιθήσεις ενός πράκτορα Υ χρησιμοποιώντας επιχειρήματα,, με τρόπο που θα επιτρέψει στον πράκτορα Υ να αποδεχτεί την άποψη του Χ. Πιο συγκεκριμένα, η πρόκληση που έχει ο προτείνων πράκτορας είναι να βρει ένα σύνολο ισχυρών επιχειρημάτων, τα οποία θα καταφέρουν να αλλάξουν την άποψη του αντιπάλου του, ούτως ώστε να συμφωνήσουν οι δυο τους. Ωστόσο, το προφίλ των αντιπάλων συνήθως δεν είναι καθόλου γνωστό και σε κάποιες περιπτώσεις μερικώς γνωστό.

Η παρούσα διπλωματική εργασία ασχολείται με προβλήματα επιχειρηματολογίας με ελλιπή γνώση και έχει σαν γενικότερο στόχο την εύρεση ενός συνόλου ισχυρών επιχειρημάτων, τα οποία να είναι ανεξάρτητα από το προφίλ των αντιπάλων.

Ουσιαστικά ο σκοπός της διπλωματικής είναι δεδομένου ενός συνόλου επιχειρημάτων και μερικές υποθέσεις, να βρεθεί κάποιο σύνολο ισχυρών επιχειρημάτων που αν εισαχθούν στον αρχικό σύνολο επιχειρημάτων να προκαλέσουν την αποδοχή ενός συγκεκριμένου επιχειρήματος που μέχρι στιγμής δεν ήταν αποδεκτό.

Η τεχνική που χρησιμοποιήθηκε για τον υπολογισμό του συνόλου ισχυρών επιχειρημάτων είναι ο προτείνων πράκτορας να χρησιμοποιήσει την δική του θεωρία για να επιλέξει τα καλύτερα επιχειρήματα του. Στη συνέχεια με βάση την θεωρία του αντιπάλου του θα πρέπει να επιλέξει τα κατάλληλα επιχειρήματα που υποστηρίζουν τα δικά του με σκοπό να πείσει τον αντίπαλο του να αποδεχτεί την άποψη του. Αυτή η τεχνική βασίζεται την επαναφορά επιχειρημάτων. Αν ο αντίπαλος δεν πειστεί, ο προτείνων θα πρέπει χρησιμοποιήσει την διαδικασία για την επαναφορά ενός από τα επιχειρήματα του, τα οποία ήταν αποδεκτά από την δική του θεωρία αλλά απορρίπτονταν από την θεωρία του αντιπάλου του.

Περιεχόμενα

Κεφάλαιο **1** **Εισαγωγή**……**1**

1.1 Σκοπός Διπλωματικής Εργασίας 1

1.2 Δομή Διπλωματικής Εργασίας 2

Κεφάλαιο **2** **Επιχειρηματολογία** ……**3**

2.1 Τι είναι η επιχειρηματολογία 3

2.2 Αφηρημένο σύστημα επιχειρηματολογίας αλά Dung 5

2.3 Σημασιολογία με βάση τις επεκτάσεις (extension-based) 7

2.4 Σημασιολογία με βάση τις ετικέτες (labelling-based) 14

Κεφάλαιο **3** **Προτασιακή Ικανοποιησιμότητα (SAT) και Ποσοτικοποιημένος Δυαδικός Τύπος (QBF)**……...……..……**18**

3.1 Ορισμός Προβλημάτων Ικανοποίησης Περιορισμών (CSP) 18

3.2 Πρόβλημα Ικανοποιησιμότητας (SAT) 19

3.3 Ποσοτικοποιημένος Δυαδικός Τύπος(QBF) 21

3.4 Ο προτασιακός επιλυτής Quantom 28

Κεφάλαιο **4** **Ισχυρά σύνολα και Ποσοτικοποιημένος Δυαδικός Τύπος(QBF)**.....**29**

4.1 Το πρόβλημα των γενικευμένων ισχυρών συνόλων 29

4.2 Υπολογισμός γενικευμένων ισχυρών συνόλων 32

Κεφάλαιο **5** **Σύστημα Υλοποίησης και Πειράματα**……**40**

5.1 Επεξήγηση συστήματος υλοποίησης 40

5.2 Πειραματικά Αποτελέσματα 46

Κεφάλαιο **6** **Συμπεράσματα και Μελλοντικές Επεκτάσεις**……..…..**50**

6.1 Συμπεράσματα 50

6.2 Μελλοντικές Επεκτάσεις 51

Β ι β λ ι ο γ ρ α φ ί α ……**52**

Π α ρ ά ρ τ η μ α Α……..……..……..……..**53**

Κεφάλαιο 1

Εισαγωγή

1.1 Σκοπός Διπλωματικής Εργασίας 1
1.2 Δομή Διπλωματικής Εργασίας 2

1.1 Σκοπός Διπλωματικής Εργασίας

Σκοπός της παρούσας διπλωματικής εργασίας είναι να βρεθούν τα ισχυρά επιχειρήματα σε ένα πρόβλημα επιχειρηματολογίας.
Πιο συγκεκριμένα, στον πραγματικό κόσμο, οι άνθρωποι όταν αλληλεπιδρούν για κάποιο σκοπό πάντα προσπαθούν να κάνουν υποθέσεις με βάση το προφίλ του αντιπάλου τους. Αυτές οι υποθέσεις συνήθως βασίζονται σε εικασίες καθοδηγούμενες από κοινές αποδεκτές πεποιθήσεις (πχ ένας 20χρονος πελάτης συνήθως ενδιαφέρετε περισσότερο να αγοράσει ένα τετραπύρηνο κινητό παρά ένας 70χρόνος πελάτης) ή σε εκμάθηση (πχ ένας πελάτης που απορρίπτει συνεχώς κινητά της ίδιας μάρκας τότε δεν θέλει την συγκεκριμένη μάρκα κινητού). Η χρησιμοποίηση αυτής της πληροφορίας που βασίζεται πάνω στο προφίλ του αντιπάλου επιτρέπει στον προτείνοντα να επιλέξει τα κατάλληλα επιχειρήματα ούτως ώστε όταν τα επιχειρήματα αυτά εισαχθούν στις πεποιθήσεις του αντιπάλου, να οδηγήσει τον αντίπαλο να δεχθεί τα επιχειρήματα του. Αυτό όμως σημαίνει ότι προσπαθώντας να πείσουμε διαφορετικούς ανθρώπους σχετικά με το ίδιο θέμα το πιο πιθανόν να χρειαζόμαστε διαφορετικά επιχειρήματα για κάθε άνθρωπο.

Επομένως, αυτό που θέλουμε να καταφέρουμε σε αυτή την διπλωματική είναι να βρούμε τα κατάλληλα επιχειρήματα που θα οδηγήσουν το αντίπαλο μας να δεχθεί μια άποψη την οποία δεν την δεχόταν αρχικά. Δηλαδή, δοθέντος ενός προβλήματος επιχειρηματολογίας θα πρέπει να βρεθούν τα ισχυρά επιχειρήματα για το συγκεκριμένο πρόβλημα.

1.2 Δομή Διπλωματικής Εργασίας

Αρχικά, στο κεφάλαιο 2 γίνεται εκτενής αναφορά στην θεωρία της επιχειρηματολογίας. Πιο συγκεκριμένα, παρουσιάζονται επεξηγηματικά παραδείγματα για κάποιες βασικές έννοιες της επιχειρηματολογίας, καθώς επίσης και κάποιοι ορισμοί τους οποίους θα χρειαστούμε για την καλύτερη κατανόηση της υλοποίησης.

Στ επόμενο κεφάλαιο επεξηγούνται τα προβλήματα ικανοποίησης περιορισμών, το πρόβλημα ικανοποιησιμότητας (SAT) και ο ποσοτικοποιημένος δυαδικός τύπος (QBF), τα οποία σχετίζονται άμεσα με το πρόβλημα που υλοποιήθηκε. Ακόμη, δίνεται μια σύντομη περιγραφή του επιλυτή (solver) που χρησιμοποιήθηκε για την εύρεση του συνόλου ισχυρών επιχειρημάτων.

Στο κεφάλαιο 4, αρχικά, παρουσιάζεται η έννοια των γενικευμένων ισχυρών συστημάτων (generalized potent sets) τα οποία αντιπροσωπεύουν την ελλιπή γνώση στην θεωρία του αντιπάλου. Στη συνέχεια, εξηγείται πως οι πρόσφατες εξειδικευμένες τεχνικές χρησιμοποίησης ποσοτικοποιημένων δυαδικών τύπων μπορούν να χρησιμοποιηθούν στην αντιμετώπιση του προβλήματος εύρεσης γενικευμένων ισχυρών συνόλων. Επίσης, περιγράφεται ο τρόπος μετάφρασης ενός προβλήματος επιχειρηματολογίας σε ποσοτικοποιήμενο δυαδικό τύπο (QBF).

Στο κεφάλαιο 5, γίνεται μια σύντομη περιγραφή της υλοποίησης του συστήματος και περιγράφονται κάποια πειράματα που πραγματοποιήθηκαν κατά την διάρκεια της πειραματικής μελέτης.

Τέλος, στο κεφάλαιο 6, παρουσιάζονται μερικά γενικά συμπεράσματα σχετικά με το πρόβλημα που είχαμε να αντιμετωπίσουμε καθώς επίσης και κάποιες μελλοντικές επεκτάσεις των ιδεών που παρουσιάστηκαν στην παρούσα διπλωματική εργασία.

Κεφάλαιο 2

Επιχειρηματολογία

2.1 Τι είναι η επιχειρηματολογία	3
2.2 Αφηρημένο σύστημα επιχειρηματολογίας αλά Dung	5
2.3 Σημασιολογία με βάση τις επεκτάσεις (extension-based)	7
2.4 Σημασιολογία με βάση τις ετικέτες (labelling-based)	14

2.1 Τι είναι η επιχειρηματολογία

Το κύριο χαρακτηριστικό της επιχειρηματολογίας είναι τα επιχειρήματα. Τα επιχειρήματα είναι το σύνολο των λογικών προτάσεων με τις οποίες υποστηρίζουμε μια πεποίθηση ή θέση ή ανακρούομε μια άλλη. Συνήθως χρησιμοποιούνται για να πείσουμε κάποιον για ένα θέμα ή να παρουσιάσουμε λόγους για την αποδοχή ενός συμπεράσματος. Ένα επιχείρημα αποτελείται από προϋποθέσεις που είναι μία ή περισσότερες προτάσεις, και το συμπέρασμα που είναι και αυτό μία πρόταση. Οι προϋποθέσεις και το συμπέρασμα είναι προτάσεις που μπορούν να είναι αληθείς ή ψευδείς, ενώ ολόκληρο το επιχείρημα μπορεί να είναι έγκυρο ή άκυρο.

Επομένως επιχειρηματολογία είναι μια γλώσσα με την οποία ένας άνθρωπος, ο πομπός, παρουσιάζοντας ένα αριθμό από επιχειρήματα υποστηρίζει μια θέση και προσπαθεί να μεταβάλει την άποψη ενός άλλου ανθρώπου, του δέκτη, για ένα συγκεκριμένο θέμα. Αποτελεί εργαλείο για την κριτική σκέψη και την αξιολόγηση των απόψεων άλλων ανθρώπων. Αναλυτικότερα, η επιχειρηματολογία είναι μια διεπιστημονική μελέτη που εξετάζει πως οι άνθρωποι μπορούν να εξάγουν συμπεράσματα μέσω λογικού συλλογισμού. Περιλαμβάνει τέχνες και επιστήμες της πολιτικής συζήτησης, διάλογο, συζήτηση, πειθώ και διαπραγμάτευση. Επιπρόσθετα, μελετά κανόνες εξαγωγής συμπερασμάτων, λογική και διαδικαστικούς κανόνες τόσο σε τεχνικό όσο και σε πραγματικό κόσμο.

Η επιχειρηματολογία είναι επίσης μια τέχνη, ίσως και επιστήμη, η οποία εκτός των άλλων περιλαμβάνει τον εριστικό διάλογο, κλάδος της κοινωνικής συζήτησης, όπου ως κύριο στόχο έχει τη νίκη απέναντι στον αντίπαλο. Αυτή η τέχνη λοιπόν είναι ένας τρόπος με τον οποίο οι άνθρωποι προστατεύουν τις πεποιθήσεις και τα πιστεύω τους κατά την διάρκεια ενός ορθολογικού διαλόγου .

Υπάρχουν αρκετά παραδείγματα στην καθημερινή μας ζωής που είτε εμείς είτε οι γύρω μας χρησιμοποιούν επιχειρηματολογία. Ένα από αυτά τα παραδείγματα είναι στο επάγγελμα των δικηγόρων και πιο συγκεκριμένα στις δίκες. Για παράδειγμα ο δικηγόρος θα πρέπει να προετοιμάσει τα επιχειρήματα που θα παρουσιάσει στο δικαστήριο με σκοπό να πείσει τους δικαστές και εισαγγελείς για την αθωότητα του πελάτη του. Ακόμη ένα κλασσικό παράδειγμα που συναντούμε επιχειρηματολογία είναι στο επάγγελμα του πωλητή. Ο πωλητής χρησιμοποιώντας τα κατάλληλα επιχειρήματα, ανάλογα με το προϊόν, θα πρέπει να μπορεί να πείσει τον πελάτη να αγοράσει το προϊόν του.

Ένα σύστημα επιχειρηματολογίας αποτελείται από ένα αριθμό επιχειρημάτων και αντεπιχειρημάτων που σχετίζονται με ένα θέμα. Ακόμη μπορεί να παρέχει συγκρίσεις ή αξιολογήσεις επιχειρημάτων καθώς επίσης και εξαγωγή συμπερασμάτων.

Πιο κάτω φαίνεται ένα απλό παράδειγμα επιχειρηματολογίας.
<u>Παράδειγμα Κατανόησης 2.1</u>
Άννα: "Η Μαρία θα αγοράσει αυτοκίνητο."
Βασίλης: "Η Μαρία δεν έχει λεφτά για να αγοράσει αυτοκίνητο."
Γιώργος: "Η Μαρία θα κάνει δάνειο για να πάρει λεφτά."

Στο πιο πάνω παράδειγμα έχουμε 3 άτομα, την Άννα, τον Βασίλη και το Γιώργο, οι οποίοι επιχειρηματολογούν για το αν η Μαρία θα αγοράσει αυτοκίνητο. Όπως μπορούμε να δούμε η Άννα και ο Βασίλης έχουν αντίθετες απόψεις. Άρα λέμε ότι ο Βασίλης επιτίθεται στην Άννα. Ο Γιώργος, όμως, με το επιχείρημα του υποστηρίζει την άποψη της Άννας και επιτίθεται στην άποψη του Βασίλη αφού προσφέρει λύση στο ότι η Μαρία δεν έχει λεφτά για να αγοράσει αυτοκίνητο. Η Άννα, ο Βασίλης και ο Γιώργος

ονομάζονται πράκτορες και χρησιμοποιούν επιχειρήματα για να υποστηρίξουν τις απόψεις τους.

2.2 Αφηρημένο σύστημα επιχειρηματολογίας αλά Dung

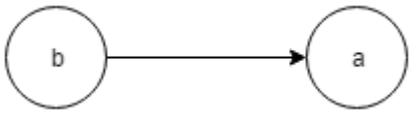
Το αφηρημένο σύστημα επιχειρηματολογίας ή αλλιώς πλαίσιο επιχειρηματολογίας (argumentation framework) εισήχθηκε από τον επιστήμονα και καθηγητή, Phan Minh Dung[1].

Σύμφωνα με τον Dung, ένα σύστημα επιχειρηματολογίας ορίζεται ως ένα ζεύγος <A,R>, όπου το A είναι ένα σύνολο από επιχειρήματα και R η δυαδική σχέση επίθεσης πάνω στο σύνολο A. Δηλαδή, ένα επιχείρημα a∈A επιτίθεται σε ένα επιχείρημα b∈A αν και μόνο αν (a,b) ∈ R.

Ένα αφηρημένο σύστημα επιχειρηματολογίας μπορεί να αναπαρασταθεί ως ένας κατευθυνόμενος γράφος όπου οι κόμβοι θα απεικονίζουν τα επιχειρήματα και οι ακμές την δυαδική σχέση επίθεσης μεταξύ των επιχειρημάτων. Η ακμή θα έχει κατεύθυνση από το επιχείρημα που κάνει επίθεση στο επιχείρημα που δέχεται επίθεση.

Ένα απλό παράδειγμα αναπαράστασης ενός συστήματος επιχειρηματολογίας φαίνεται πιο κάτω στο σχήμα 2.1.

Σύστημα επιχειρηματολογίας: AF: {a,b},{(b,a)}
Μπορούμε να δούμε ότι έχουμε δύο επιχειρήματα, το a και το b. Επίσης, βλέπουμε ότι το επιχείρημα b επιτίθεται στο επιχείρημα a. Επομένως, θα έχουμε μια ακμή που θα κατευθύνεται από τον κόμβο b στον κόμβο a.



Σχήμα 2.1

Παρόμοια πιο κάτω στο σχήμα 2.2 φαίνεται η αναπαράσταση του παραδείγματος κατανόησης 2.1 σε γράφο.

Παράδειγμα Κατανόησης 2.1

Άννα: "Η Μαρία θα αγοράσει αυτοκίνητο."

Βασίλης: "Η Μαρία δεν έχει λεφτά για να αγοράσει αυτοκίνητο."

Γιώργος: "Η Μαρία θα κάνει δάνειο για να πάρει λεφτά."

Στο παράδειγμα 2.1 έχει τρεις πράκτορες, την Άννα, τον Βασίλη και τον Γιώργο. Επομένως θα έχουμε 3 κόμβους για κάθε επιχείρημα. Επίσης, θα έχουμε και 2 ακμές αφού όπως αναφέρθηκε και πιο πάνω ο Βασίλης επιτίθεται στο επιχείρημα της Άννας και ο Γιώργος στο επιχείρημα του Βασίλη.

Σχήμα 2.2

Το αφηρημένο σύστημα επιχειρηματολογίας που αντιστοιχεί στο παράδειγμα κατανόησης 2.1 είναι το εξής:

AF = {Άννα, Βασίλης, Γιώργος}, {(Βασίλης, Άννα), (Γιώργος, Βασίλης)}

A-Σύνολο Επιχειρημάτων

R - Δυαδικές σχέσεις επίθεσης

Ένα επιχείρημα χαρακτηρίζεται ως αποδεκτό (ή δικαιολογημένο) αν με κάποιο τρόπο επιβιώνει της επίθεσης που δέχεται, αν όχι, τότε χαρακτηρίζεται ως μη αποδεκτό (ή αδικαιολόγητο). Η διαδικασία για να καθορίσουμε αν ένα επιχείρημα είναι αποδεκτό ή όχι ονομάζεται αξιολόγηση επιχειρήματος (argument evaluation). Η αξιολόγηση του επιχειρήματος είναι αναγκαία γιατί σε μια θεωρία δεν μπορούν όλα τα επιχειρήματα να είναι ευσταθή όταν υπάρχουν σχέσεις επίθεσης μεταξύ τους. Η σημασιολογία ενός αφηρημένου συστήματος επιχειρηματολογίας είναι η διαδικασία αξιολόγησης του επιχειρήματος.

Οι δύο κύριοι τρόποι για να καθορίσουμε αν ένα επιχείρημα είναι αποδεκτό ή όχι είναι:

α) η σημασιολογία με βάση τις επεκτάσεις (extension-based) και

β) η σημασιολογία με βάση τις ετικέτες (labelling-based).

Αυτούς τους δύο τρόπους θα τους μελετήσουμε αναλυτικότερα στα επόμενα δύο υποκεφάλαια.

2.3 Σημασιολογία με βάση τις επεκτάσεις (extension-based)

Όπως αναφέρθηκε και στο προηγούμενο υποκεφάλαιο, η σημασιολογία με βάση τις επεκτάσεις είναι ένας από τους δύο κύριους τρόπους αξιολόγησης των επιχειρημάτων.

Αναλυτικότερα, η σημασιολογία με βάση τις επεκτάσεις ορίζει ένα σύνολο επεκτάσεων E από ένα σύστημα επιχειρηματολογίας <A,R>, όπου A το σύνολο των επιχειρημάτων και R η δυαδική σχέση επίθεσης. Το σύνολο επεκτάσεων E είναι υποσύνολο του συνόλου A και αντιπροσωπεύει ένα σύνολο από επιχειρήματα που μπορούν να επιβιώσουν μαζί ή είναι αποδεκτά.

Στη συνέχεια θα εξηγηθούν κάποιοι βασικοί ορισμοί που σχετίζονται με την σημασιολογία με βάση τις επεκτάσεις.

1. Απουσία Συγκρούσεων (Conflict-free):

   Αν ένα επιχείρημα a επιτίθεται σε ένα άλλο επιχείρημα b, τότε τα επιχειρήματα a και b δεν μπορούν να βρίσκονται μαζί σε ένα σύνολο επεκτάσεων.

   Αυστηρός Ορισμός:

   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο $S \subseteq A$ δεν έχει συγκρούσεις αν και μόνο αν $\nexists$ a,b $\in$ S και (a,b) $\in$ R.

   Παράδειγμα Κατανόησης 2.2:

   Έστω η θεωρία AF=<{a,b,c,d},{(a,c), (b,a), (d,a)}>. Μας δίνονται τα σύνολα S={a,b,d} και G={c,b,d} και θα πρέπει να καθορίσουμε αν αυτά τα σύνολα υπακούουν στην ιδιότητα απουσίας συγκρούσεων.

   Αρχικά θα ελέγξουμε το σύνολο S={a,b,d}. Όπως μπορούμε να δούμε το σύνολο αυτό δεν έχει την ιδιότητα απουσίας συγκρούσεων αφού τα επιχειρήματα b και d επιτίθενται στο επιχείρημα a.

Αντιθέτως, το σύνολο G={c,b,d} έχει την ιδιότητα απουσίας συγκρούσεων αφού δεν υπάρχουν επιθέσεις μεταξύ των επιχειρημάτων c, b, d.

2. Αποδεκτό Επιχείρημα (acceptable):

   Ένα επιχείρημα είναι αποδεκτό αν υπάρχει ένα σύνολο από επιχειρήματα τα οποία αντεπιτίθενται σε όλα τα επιχειρήματα που κάνουν επίθεση στο επιχείρημα αυτό.

   Αυστηρός Ορισμός:

   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R> και ενός συνόλου $S \subseteq A$, ένα επιχείρημα $a \in A$ είναι αποδεκτό αν και μόνο αν $\forall b \in A$ όπου $(b,a) \in R$, τότε $\exists\ c \in S$ τέτοιο ώστε $(c,b) \in R$.

   Παράδειγμα Κατανόησης 2.3:

   Έστω η θεωρία AF=<{a,b,c,d},{(a,d), (b,a), (d,c), (c,a)}>. Θα πάρουμε ένα σύνολο S από το σύνολο το επιχειρημάτων A και θα δούμε αν στο σύνολο S υπάρχει κάποιο επιχείρημα το οποίο μπορούμε να χαρακτηρίσουμε ως αποδεκτό.

   Έστω παίρνουμε το σύνολο S={a,c}. Θα πρέπει να ελέγξουμε αν κάποιο από τα επιχειρήματα αυτά είναι αποδεκτό.

   Πρώτα θα ελέγξουμε το επιχείρημα a. Όπως μπορούμε να δούμε το επιχείρημα a δέχεται επίθεση από το επιχείρημα b και c. Για να είναι το a αποδεκτό θα πρέπει να υπάρχει επιχείρημα από το σύνολο S, το οποίο επιτίθεται στο b και c. Όπως μπορούμε να δούμε δεν υπάρχει τέτοιο επιχείρημα και επομένως το a δεν είναι αποδεκτό.

   Τώρα θα κάνουμε το ίδιο και για το επιχείρημα c. Το επιχείρημα c δέχεται επίθεση από το επιχείρημα d. Για να είναι το c αποδεκτό θα πρέπει να υπάρχει επιχείρημα από το σύνολο S, το οποίο επιτίθεται στο d. Όπως μπορούμε να δούμε το επιχείρημα a επιτίθεται στο d. Άρα το επιχείρημα c είναι αποδεκτό.

3. Παραδεκτό σύνολο (Admissible set)

   Ένα σύνολο επιχειρημάτων, έστω S, χαρακτηρίζεται ως παραδεκτό αν και μόνο αν ικανοποιεί την ιδιότητα απουσίας συγκρούσεων(conflict-free) και κάθε επιχείρημα που ανήκει στο σύνολο S είναι αποδεκτό (acceptable).

   Αυστηρός Ορισμός:

   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο $S \subseteq A$ είναι παραδεκτό αν και μόνο αν cf(S)(δηλαδή αν στο S δεν υπάρχουν συγκρούσεις) και $\forall a \in S$ είναι αποδεκτό ως προς το σύνολο S.

   Παράδειγμα Κατανόησης 2.4:

   Έστω η θεωρία AF=<{a,b,c,d,e},{(a,d), (b,a), (d,c), (c,b), (e,b)}>. Θα πάρουμε δύο σύνολα επιχειρημάτων και θα αποφασίσουμε αν τα σύνολα αυτά είναι αποδεκτά.

   Παίρνουμε τα σύνολα S={a,c} και G={b,d}.

   Το σύνολο S={a,c} είναι παραδεκτό γιατί όλα τα επιχειρήματα που ανήκουν σε αυτό είναι αποδεκτά και δεν υπάρχουν ούτε συγκρούσεις μέσα σε αυτό.

   Αντιθέτως, το σύνολο G={b,d} δεν είναι παραδεκτό γιατί παρόλο που δεν υπάρχουν συγκρούσεις μέσα σε αυτό, το επιχείρημα b δεν είναι αποδεκτό.

Στη συνέχεια, θα μελετήσουμε τις τέσσερις βασικές επεκτάσεις οι οποίες είναι οι εξής:

1. Πλήρης Επέκταση (Complete Extensions)
2. Ευσταθής Επέκταση (Stable Extensions)
3. Προτιμούμενη Επέκταση (Preferred Extensions)
4. Στηριγμένη Επέκταση (Grounded Extensions)

1. Πλήρης Επέκταση (Complete Extensions):

   Η πλήρης επέκταση συμβολίζεται με E_{CO}.

   Ένα σύνολο E χαρακτηρίζεται ως πλήρης επέκταση αν και μόνο αν είναι παραδεκτό σύνολο και κάθε αποδεκτό επιχείρημα από το σύνολο επιχειρημάτων A ανήκει στο σύνολο E.

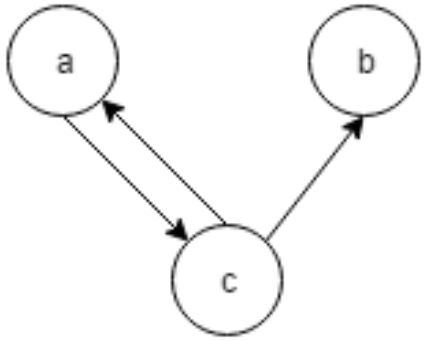
Αυστηρός Ορισμός:

Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο $E \subseteq A$ είναι πλήρης επέκταση αν και μόνο αν το E είναι παραδεκτό(admissible) σύνολο και αν $\forall a \in A$ είναι αποδεκτό(acceptable), τότε $a \in E$.

Παράδειγμα Κατανόησης 2.5:

Έστω η θεωρία AF=<{a,b,c},{(a,c), (c,a), (c,b)}>.

Στο σχήμα 2.3 φαίνεται η αναπαράσταση της θεωρίας αυτής σε γράφο.



Σχήμα 2.3

Οι πλήρης επεκτάσεις για την πιο πάνω θεωρία είναι: {a,b}, {c}.

2. Ευσταθής Επέκταση (Stable Extensions):

   Η ευσταθής επέκταση συμβολίζεται με E_{ST}.

   Ένα σύνολο E χαρακτηρίζεται ως ευσταθής επέκταση αν και μόνο αν δεν έχει συγκρούσεις μεταξύ των στοιχείων του(conflict-free) και έχει την ιδιότητα να επιτίθεται σε όλα τα επιχειρήματα που δεν περιλαμβάνονται σε αυτό.

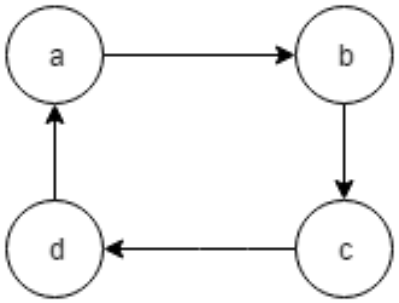
   Αυστηρός Ορισμός:

   Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο $E \subseteq A$ είναι ευσταθής επέκταση αν και μόνο αν το E έχει την ιδιότητα απουσίας συγκρούσεων (conflict-free) και $\forall a \in A, a \notin E, \exists b \in E$ τέτοιο ώστε $(b,a) \in R$.

Παράδειγμα Κατανόησης 2.5:

Έστω η θεωρία AF=<{a,b,c,d},{(a,b), (b,c), (c,d), (d,a) }}>.

Στο σχήμα 2.4 φαίνεται η αναπαράσταση της θεωρίας αυτής σε γράφο.



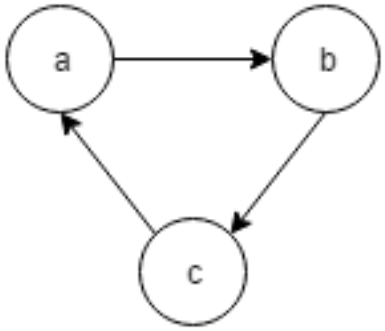
Σχήμα 2.4

Οι ευσταθής επεκτάσεις για την πιο πάνω θεωρία είναι: {a,c}, {b,d}.

Παράδειγμα Κατανόησης 2.6:

Έστω η θεωρία AF=<{a,b,c },{(a,b), (b,c), (c,a)}}>.

Στο σχήμα 2.5 φαίνεται η αναπαράσταση της θεωρίας αυτής σε γράφο.



Σχήμα 2.5

Σε αυτό το παράδειγμα δεν υπάρχουν ευσταθής επεκτάσεις, δηλαδή {∅}.

3. Προτιμούμενη Επέκταση (Preferred Extensions):

Η προτιμούμενη επέκταση συμβολίζεται με E_{PR}.

Ένα σύνολο E χαρακτηρίζεται ως προτιμούμενη επέκταση αν και μόνο αν είναι το μέγιστο παραδεκτό σύνολο. Πιο συγκεκριμένα το σύνολο που είναι όσο το

δυνατό μεγαλύτερο και μπορεί να υπερασπιστεί τον εαυτό του από τις επιθέσεις που δέχεται.

Θα πρέπει να σημειωθεί ότι σε όλα τα συστήματα επιχειρηματολογίας θα υπάρχει τουλάχιστον ένα σύνολο που θα είναι προτιμούμενη επέκταση λόγω του ότι υπάρχει πάντα το κενό σύνολο το οποίο είναι παραδεκτό.

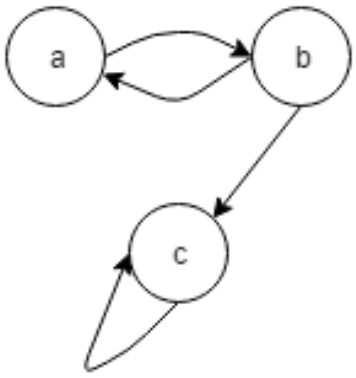
Αυστηρός Ορισμός:

Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο $E \subseteq A$ είναι προτιμούμενη επέκταση αν και μόνο αν το E είναι το μέγιστο σύνολο του AS(AF).

Παράδειγμα Κατανόησης 2.7:

Έστω η θεωρία AF=<{a,b,c},{(a,b), (b,α), (b,c), (c,c)}>.

Στο σχήμα 2.6 φαίνεται η αναπαράσταση της θεωρίας αυτής σε γράφο.



Σχήμα 2.6

Οι προτιμούμενες επεκτάσεις για την πιο πάνω θεωρία είναι: {a}, {b}.

4. Στηριγμένη Επέκταση (Grounded Extensions):

   Η στηριγμένη επέκταση συμβολίζεται με E_{GR}.

   Ένα σύνολο E χαρακτηρίζεται ως προτιμούμενη επέκταση αν είναι το μικρότερο παραδεκτό σύνολο που είναι πλήρης επέκταση.

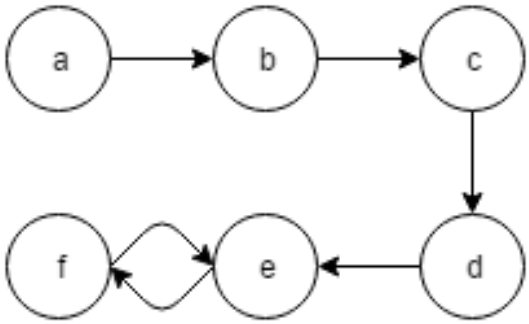
Αυστηρός Ορισμός:

Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>, ένα σύνολο $E \subseteq A$ είναι στηριγμένη επέκταση αν και μόνο αν το E είναι το ελάχιστο παραδεκτό σύνολο και $\forall a \in A$ είναι αποδεκτό επιχείρημα, τότε $a \in E$.

Παράδειγμα Κατανόησης 2.8:

Έστω η θεωρία AF=<{a,b,c},{(a,b), (b,α), (b,c), (c,c)}>.

Στο σχήμα 2.7 φαίνεται η αναπαράσταση της θεωρίας αυτής σε γράφο.



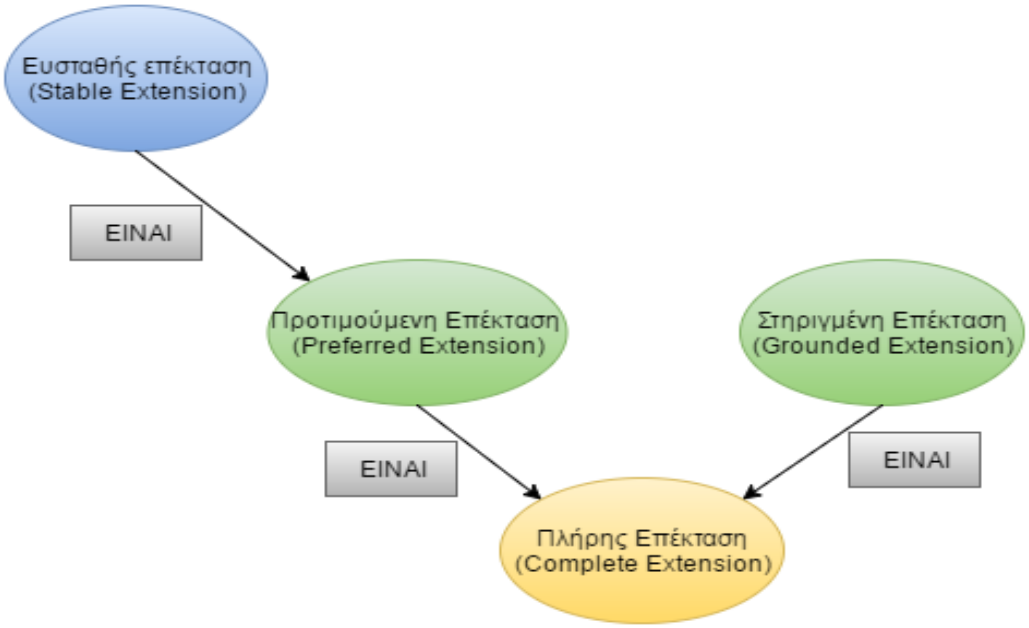
Σχήμα 2.7

Η στηριγμένη επέκταση για την πιο πάνω θεωρία είναι: {a,c}

Η σχέση μεταξύ των τεσσάρων βασικών επεκτάσεων που αναφέρθηκαν πιο πάνω είναι:

- Κάθε ευσταθής επέκταση (stable extension) είναι προτιμούμενη (preferred),
- Κάθε προτιμούμενη επέκταση (preferred extension) είναι πλήρης (complete),
- Η στηριγμένη επέκταση (grounded extension) είναι πλήρης (complete).

Για καλύτερη κατανόηση, πιο κάτω, στο σχήμα 2.8 φαίνεται η αναπαράσταση των προαναφερθέντων σχέσεων.[2]



Σχήμα 2.8

2.4 Σημασιολογία με βάση τις ετικέτες (labelling-based)

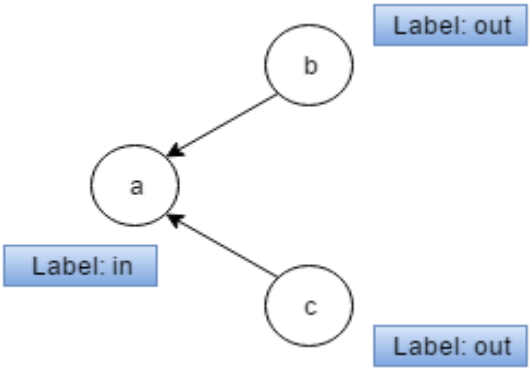
Όπως αναφέρθηκε και στο προηγούμενο υποκεφάλαιο, η σημασιολογία με βάση τις ετικέτες είναι ένας από τους δύο κύριους τρόπους αξιολόγησης των επιχειρημάτων.

Η σημασιολογία με βάση τις ετικέτες ορίζεται ως ένα ζεύγος (argument, label). Οι ετικέτες είναι ένας πιο εκφραστικός τρόπος, σε σχέση με τις επεκτάσεις, που μπορούμε να εκφράσουμε την αποδοχή ενός επιχειρήματος. Αναλυτικότερα, οι ετικέτες που μπορεί ένα επιχείρημα να πάρει είναι μέσα (in), έξω (out) και αναποφάσιστο (undec). Η ετικέτα μέσα (in) υπονοεί ότι το επιχείρημα είναι αποδεκτό, η ετικέτα έξω (out) ότι το επιχείρημα δεν είναι αποδεκτό και η ετικέτα αναποφάσιστο (undec) ότι δεν ξέρουμε αν το επιχείρημα είναι αποδεκτό ή όχι.

Η ετικέτα κάθε επιχειρήματος μπορεί να καθοριστεί σύμφωνα με τους πιο κάτω κανόνες:

- Ένα επιχείρημα έχει ετικέτα μέσα (in) αν και μόνο αν όλοι οι κόμβοι-επιχειρήματα που του επιτίθενται έχουν ετικέτα out.
  Αυστηρός ορισμός:
  Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>.
  ∀ a∈A, L(a) = in, αν και μόνο αν ∀ b∈A τέτοιο ώστε (b,a)∈R, L(b) = out.

Σχηματική Απεικόνιση:



Σχήμα 2.9

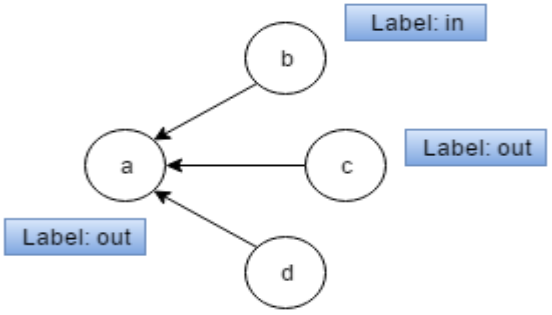
- Ένα επιχείρημα έχει ετικέτα έξω (out) αν και μόνο αν υπάρχει τουλάχιστον ένας κόμβος-επιχείρημα από αυτούς που του επιτίθενται, ο οποίος έχει ετικέτα μέσα (in).

  Αυστηρός ορισμός:

  Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>.

  ∀ a∈A, L(a) = out, αν και μόνο αν ∃ b∈A τέτοιο ώστε (b,a)∈R, L(b) = in.

  Σχηματική Απεικόνιση:



Σχήμα 2.10

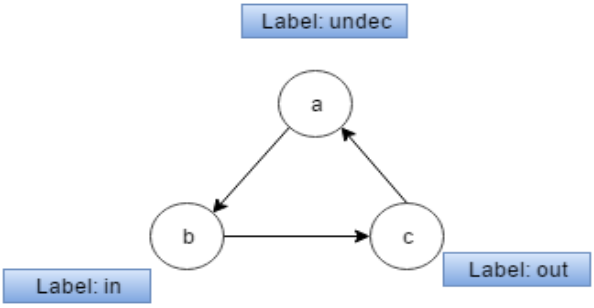
- Ένα επιχείρημα έχει ετικέτα αναποφάσιστο (undec) αν δεν μας δίνετε κάποια πληροφορία που να μπορούμε να καθορίσουμε αν είναι αποδεκτό η όχι.

  Αυστηρός ορισμός:

  Δεδομένου ενός συστήματος επιχειρηματολογίας AF=<A,R>.

  ∀ a∈A, L(a) = undec, αν και μόνο L(a) ≠ in και L(a) ≠ out.

Σχηματική Απεικόνιση:



Σχήμα 2.12

Επεξήγηση:

Το επιχείρημα θα έχει ετικέτα αναποφάσιστο γιατί δεν μπορούμε να καθορίσουμε αν πρέπει να πάρει την ετικέτα μέσα ή έξω.

Για να κατανοηθεί καλύτερα θα πάρουμε τις δύο πιθανές ετικέτες που μπορεί να πάρει το επιχείρημα a με βάση το σχήμα 2.12 και θα δείξουμε ότι δεν μπορεί να έχει αυτές τις ετικέτες.

Πρώτα, θα δούμε την περίπτωση που το επιχείρημα a έχει ετικέτα μέσα (in). Αν ισχύει αυτό τότε παραβιάζεται το επιχείρημα b, το οποίο έχει ετικέτα μέσα. Πιο συγκεκριμένα, αφού το επιχείρημα b έχει ετικέτα μέσα, τότε τα επιχειρήματα που του επιτίθενται πρέπει να έχουν ετικέτα έξω. Το επιχείρημα που του επιτίθεται είναι το a. Επομένως φτάνουμε σε αντίφαση.

Τώρα, για την περίπτωση που το επιχείρημα a έχει ετικέτα έξω (out), τότε θα πρέπει να υπάρχει τουλάχιστον ένα επιχείρημα από αυτούς που του επιτίθενται, ο οποίος έχει ετικέτα μέσα. Όπως μπορούμε να δούμε, το μόνο επιχείρημα που επιτίθεται στο a είναι το c και έχει ετικέτα έξω. Άρα, φτάνουμε και σε αυτή την περίπτωση αντίφαση.

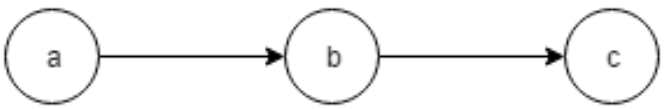
Επομένως το επιχείρημα a θα πάρει την ετικέτα αναποφάσιστο.

Στη συνέχεια θα δοθούν κάποια παραδείγματα για να κατανοηθούν καλύτερα οι προαναφερθέντες κανόνες.

Παράδειγμα Κατανόησης 2.9:

Έστω η θεωρία AF=<{a,b,c},{(a,b), (b,c)}}>.

Στο σχήμα 2.12 φαίνεται η αναπαράσταση της θεωρίας αυτής σε γράφο.



Σχήμα 2.12

Οι ετικέτες για την πιο πάνω θεωρία καθορίζονται ως εξής: L(a)=in, L(b)=out, L(c)=in.

Παράδειγμα Κατανόησης 2.10:

Έστω η θεωρία AF=<{a,b},{(a,b), (b,a)}>.

Στο σχήμα 2.13 φαίνεται η αναπαράσταση της θεωρίας αυτής σε γράφο.

Σχήμα 2.13

Σε αυτή την περίπτωση έχουμε τρεις πιθανές αναθέσεις ετικετών.

Πιο συγκεκριμένα, οι ετικέτες για την πιο πάνω θεωρία καθορίζονται ως εξής:

1. L(a) = in, L(b) = out
2. L(a) = out, L(b) = in
3. L(a) = undec, L(b) = undec

Κεφάλαιο 3

Προτασιακή Ικανοποιησιμότητα (SAT) και Ποσοτικοποιημένος Δυαδικός Τύπος(QBF)

3.1 Ορισμός Προβλημάτων Ικανοποίησης Περιορισμών (CSP) 18
3.2 Πρόβλημα Ικανοποιησιμότητας (SAT) 19
3.3 Ποσοτικοποιημένος Δυαδικός Τύπος (Quantified Boolean formulas/QBF) 21
3.4 Ο προτασιακός επιλυτής Quantom 28

3.1 Ορισμός Προβλημάτων Ικανοποίησης Περιορισμών (CSP)

Τα προβλήματα ικανοποίησης περιορισμών (constraint satisfaction problems - CSP) είναι μαθηματικά προβλήματα οριζόμενα ως ένα σύνολο από αντικείμενα των οποίων η κατάσταση τους πρέπει να ικανοποιεί ένα αριθμό από περιορισμούς. Τα προβλήματα αυτά αντιπροσωπεύουν τις οντότητες ενός προβλήματος ως μια ομοιογενές συλλογή από πεπερασμένους περιορισμούς πάνω στις μεταβλητές, τα οποία λύνονται με μεθόδους ικανοποίησης περιορισμών. Τα προβλήματα ικανοποίησης περιορισμών αποτελούν αντικείμενο έντονης έρευνας στον τομέα της τεχνητής νοημοσύνης καθώς επίσης και στην επιχειρηματική έρευνα. Επιπρόσθετα, τα προβλήματα αυτά συνήθως παρουσιάζουν υψηλή πολυπλοκότητα, απαιτώντας ένα συνδυασμό από ευρετικά και συνδυαστικές μεθόδους αναζήτησης για να μπορέσουν να λυθούν σε εύλογο χρόνο.

Ένα πρόβλημα ικανοποίησης περιορισμών ορίζεται από:

- Ένα σύνολο από μεταβλητές(variables) $X_1,X_2,...,X_n$.
- Ένα σύνολο πεδίων τιμών(domain) $D_1,D_2,...,D_n$ για κάθε μια από τις μεταβλητές.
- Ένα σύνολο από περιορισμούς(constraints) $C_1,C_2,...,C_n$. Ένας περιορισμός καθορίζει του επιτρεπτούς συνδυασμούς τιμών για ένα υποσύνολο του συνόλου των μεταβλητών.

Δηλαδή ορίζεται ως μια τριπλή <X,D,C>.

Ανάλογα με το πόσες μεταβλητές περιλαμβάνει ένας περιορισμός χαρακτηρίζεται ως[3]:

- Μοναδιαίος (unary) όταν περιλαμβάνει μία μεταβλητή,
- Δυαδικός (binary) όταν περιλαμβάνει δύο μεταβλητές,
- Ανώτερης τάξης (higher order) όταν περιλαμβάνει περισσότερες από 2 μεταβλητές.

Μια κατάσταση του προβλήματος, ορίζεται ως μια ανάθεση τιμών σε κάποιες από τις μεταβλητές. Στην περίπτωση που η ανάθεση δεν παραβιάζει κανέναν από τους περιορισμούς τότε ονομάζεται συνεπής(consistent). Όταν η ανάθεση περιλαμβάνει όλες τις μεταβλητές τότε ονομάζετε πλήρης(complete). Λύση του προβλήματος ικανοποίησης περιορισμών θεωρείται μια πλήρης και συνεπής ανάθεση. Συνεπώς, μία λύση του προβλήματος αποτελείται από την πλήρη ανάθεση τιμών στις μεταβλητές όταν ικανοποιούνται όλοι οι περιορισμοί.

3.2 Πρόβλημα Ικανοποιησιμότητας (SAT)

Το πρόβλημα ικανοποιησιμότητας στην προτασιακή λογική είναι γνωστό ως SAT και αποτελεί θέμα μελέτης εδώ και πάρα πολλά χρόνια και είναι μια ειδική κατηγορία προβλημάτων ικανοποίησης περιορισμών. Το πρόβλημα της ικανοποιησιμότητας είναι ένα από τα βασικά προβλήματα στην θεωρία υπολογισμού αλλά και στην μαθηματική λογική.

Μια έκφραση προτασιακής λογικής, που ονομάζεται και ως Boolean έκφραση, αποτελείται από μεταβλητές, τελεστές και παρενθέσεις. Οι τελεστές που μπορούν να χρησιμοποιηθούν είναι: AND (σύζευξη, και συμβολίζεται με $\wedge$) OR (διάζευξη, και συμβολίζεται με $\vee$) και NOT (άρνηση, και συμβολίζεται με $\neg$).

Το πρόβλημα ικανοποιησιμότητας(Boolean Satisfiability Problem) είναι το πρόβλημα καθορισμού αν υπάρχει μια ανάθεση τιμών που ικανοποιεί μια Boolean έκφραση. Με άλλα λόγια, αναθέτοντας τις τιμές αληθές ή ψευδές στις μεταβλητές μιας έκφρασης, προσπαθεί να βρει μια ανάθεση τιμών που οδηγεί το αποτέλεσμα της έκφρασης σε αληθής. Σε αυτή την περίπτωση, η έκφραση χαρακτηρίζεται ως ικανοποιήσιμη.

Αντιθέτως, αν δεν υπάρχει καμία ανάθεση τιμών στις μεταβλητές που να δίνει αποτέλεσμα true στην έκφραση, η έκφραση χαρακτηρίζεται ως μη ικανοποιήσιμη.

Παράδειγμα Κατανόησης 3.1:

Η έκφραση (a AND NOT b) είναι ικανοποιήσιμη γιατί υπάρχει ανάθεση που οδηγεί την έκφρασε σε true. Η ανάθεση αυτή είναι a= true και b=false, που κάνει την (a AND NOT b) = true. Αντιθέτως, η έκφραση (a AND NOT a) είναι μη ικανοποιήσιμη.

Ακολουθεί ένα πιο σύνθετο παράδειγμα.

Παράδειγμα Κατανόησης 3.2:

Δεδομένου της έκφρασης:

(a ∨ b ∨ c) ∧ (ד a ∨ ד b) ∧ (d ∨ a) ∧ (c ∨ ד e) ∧ (a ∨ e ∨ c ∨ d) ∧ (ד a ∨ ד f) ∧ (ד c) ∧ (f)

Θα καθορίσουμε αν είναι ικανοποιήσιμη ή μη ικανοποιήσιμη.

Μπορούμε να δούμε ότι υπάρχει μια ανάθεση που ικανοποιεί την πιο πάνω έκφραση. Η ανάθεση αυτή είναι a = False, b = True, c = False, d = True, e = False, f = True. Οπότε, αφού υπάρχει μια τέτοια ανάθεση, η πρόταση είναι ικανοποιήσιμη.

Το πρόβλημα ικανοποιησιμότητας είναι ένα από τα πρώτα προβλήματα που αποδείχθηκαν ότι είναι NP-Complete. Αυτό σημαίνει ότι τα SAT προβλήματα είναι πολύ δύσκολο να λυθούν και μέχρι τώρα δεν ανακαλύφθηκε κάποιος αποδοτικός αλγόριθμος. Γενικά πιστεύεται ότι δεν θα υπάρξει ένας τέτοιος αλγόριθμος αλλά αυτό δεν έχει αποδειχθεί μαθηματικά.

Χρησιμοποιώντας SAT solvers μπορούμε να λύσουμε πολλές περιπτώσεις προβλημάτων SAT. Για παράδειγμα, προβλήματα στον τομέα της τεχνητής νοημοσύνης, στο σχεδιασμό κυκλωμάτων και στην απόδειξη θεωρημάτων.

Υπάρχουν δύο κατηγορίες αλγορίθμων υψηλής απόδοσης για την επίλυση προβλημάτων SAT. Ο αλγόριθμος που χρησιμοποιείται ποιο πολύ στην πράξη είναι ο αλγόριθμος DPLL.

Αλγόριθμος DPLL (Davis-Putnam-Logemann-Loveland):

Ο αλγόριθμος DPLL αναπτύχθηκε το 1962 από τους M. Davis, H. Putnam ,G. Logemann και D. Loveland. Είναι ένας από τους πιο διαδεδομένους αλγόριθμους επίλυσης του προβλήματος της ικανοποιησιμότητας και οι περισσότεροι SAT επιλυτές(solvers) στηρίζονται στον αλγόριθμο αυτό. [4]

Ο αλγόριθμος αυτός χρησιμοποιεί την διαδικασία αναζήτησης με οπισθοδρόμηση. Στην αρχή της εκτέλεσης του αλγορίθμου οι μεταβλητές δεν έχουν κάποια τιμή. Έπειτα γίνεται ανάθεση σε μια μεταβλητή και γίνεται έλεγχος κατά πόσο επηρεάζει η ανάθεση αυτή τις υπόλοιπες μεταβλητές. Αν ο αλγόριθμος δεν εντοπίσει κάποια σύγκρουση συνεχίζει με μια νέα ανάθεση τιμών σε μια άλλη μεταβλητή στην οποία δεν έχει αναθέσει τιμή μέχρι τώρα. Στην περίπτωση όμως που εντοπίστηκε σύγκρουση, τότε γίνεται οπισθοδρόμηση (backtracking) και αφαιρεί από την μεταβλητή την τιμή που της είχε ανατέθηκε και συνεχίζει με την ανάθεση νέας τιμής στην μεταβλητή αυτή. Συνεχίζεται αυτή η διαδικασία μέχρι ότου να ανατεθούν τιμές σε όλες τις μεταβλητές ή να μην υπάρχει κάποια τιμή για να ανατεθεί σε κάποια από τις μεταβλητές. Στην δεύτερη περίπτωση το πρόβλημα θα είναι μη ικανοποιήσιμη .

3.3 Ποσοτικοποιημένος Δυαδικός Τύπος (QBF)

Οι ποσοτικοποιημένοι δυαδικοί τύποι (QBF) είναι πολύ χρήσιμοι σε προβλήματα μοντελοποίησης στην Τεχνητή Νοημοσύνη και στην Πληροφορική.

Είναι επέκταση των προτασιακών τύπων, επιτρέποντας ποσοδείκτες (quantifiers) πάνω σε προτασιακές μεταβλητές. Οι ποσοδείκτες είναι δύο σύμβολα, το "∃" (exists, υπάρχει) και το "∀" (all, για κάθε) που ονομάζονται αντίστοιχα υπαρξιακός και καθολικός ποσοδείκτης. Ο καθολικά ποσοτικοποιημένος τύπος ∀x φ(x) θα είναι αληθής αν και μόνο αν όλα τα φ(x) είναι αληθής. Παρόμοια, ένας υπαρξιακός ποσοτικοποιημένος τύπος ∃y φ(y) θα είναι αληθής αν και μόνο αν τουλάχιστον ένα από τα f(y) είναι αληθής. [5]

Συντακτικά, μια έκφραση QBF περιγράφεται με την ακόλουθη γραμματική:

$x ::= δυαδική\ μεταβλητή$

$Q ::= \exists \mid \forall$

$\phi ::= x \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid true \mid false$

$\Phi ::= \phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid Qx\ \Phi$

Σύμφωνα με τον αυστηρό ορισμό του QBF, ένα "κανονικό" QBF είναι στη μορφή Q_1X_1 $Q_2X_2 \ldots Q_nX_n\Phi$, όπου Φ ο προτασιακός τύπος, $Q_i \in \{\forall, \exists\}$, $Q_i \neq Q_i+1$, $X_1, X_2, \ldots, X_n$ ξένα σύνολα από προτασιακές μεταβλητές τέτοια ώστε $X_1 \cup X_2 \cup \ldots \cup X_n$ να συμπίπτει με το σύνολο των προτασιακών μεταβλητών της Φ. Οι πιθανές τιμές που μπορούν να πάρουν οι μεταβλητές $X_1, X_2, \ldots, X_n$ είναι αληθές και ψευδές δηλαδή, $(\{X_1, X_2, \ldots, X_n\} \rightarrow \{True,False\})$.[7]

<u>Παράδειγμα Κατανόησης 3.3:</u>

Η πιο κάτω έκφραση:

$\exists x \forall y(x \vee \neg y) \wedge (\neg x \vee y)$

Σε φυσική γλώσσα θα μεταφραζόταν ως εξής:

"Υπάρχει κάποια τιμή για το x, τέτοια ώστε για όλες τις τιμές του y η έκφραση να είναι αληθής;"

<u>Παράδειγμα Κατανόησης 3.4:</u>

Η πιο κάτω έκφραση:

$\forall y \exists x(x \vee \neg y) \wedge (\neg x \vee y)$

Σε φυσική γλώσσα θα μεταφραζόταν ως εξής:

"Για όλες τις τιμές του y, υπάρχει τιμή για το x τέτοια ώστε η έκφραση να είναι αληθής;"

Μια QBF έκφραση χαρακτηρίζεται ως "κλειστή" αν και μόνο αν κάθε μεταβλητή δεσμεύεται από ποσοδείκτη. Αν δεν ισχύει αυτό, τότε η έκφραση χαρακτηρίζεται ως "ανοικτή". Στην παρούσα διπλωματική εργασία θα ασχοληθούμε μόνο με κλειστές εκφράσεις.

Σε σχέση με την προτασιακή λογική, οι αναπαραστάσεις μέσω ποσοτικοποιημένων δυαδικών τύπων συνήθως είναι αρκετά μικρότερες, πιο εύκολες να διαβαστούν και πιο φυσικές. Αλλά, όλα αυτά οδηγούν στην αύξηση της πολυπλοκότητας της χειρότερης περίπτωσης του προβλήματος ικανοποιησιμότητας. Πιο συγκεκριμένα τα QBF προβλήματα έχουν πολυπλοκότητα τάξης μεγαλύτερη από την NP-Complete που έχει το SAT.[6] Επομένως, πρακτικά το QBF πρόβλημα είναι πιο δύσκολο να λυθεί απ' ότι το SAT.

Πολλοί μοντέρνοι QBF επιλυτές χρησιμοποιούν τεχνικές οι οποίες αναπτύχθηκαν για SAT επιλυτές,

Υπάρχει ένας απλός αναδρομικός αλγόριθμος που καθορίζει αν το QBF είναι true.
Ο αλγόριθμος αυτός δουλεύει ως εξής:
Δεδομένου ενός QBF, $Q_1X_1\ Q_2X_2 \ldots Q_nX_n\Phi(X_1, X_2, \ldots, X_n)$.
Αν η έκφραση δεν περιέχει ποσοδείκτες, μπορούμε να επιστέψουμε απλώς την έκφραση. Αλλιώς βγάζουμε πρώτα τον πρώτο ποσοδείκτη και ελέγχουμε τις δύο πιθανές τιμές για την πρώτη μεταβλητή. Δηλαδή θα έχουμε:
$A = Q_2X_2 \ldots Q_nX_n\Phi(0, X_2, \ldots, X_n)$,
$B = Q_2X_2 \ldots Q_nX_n\Phi(1, X_2, \ldots, X_n)$.
Αν ο $Q_1 = \exists$, τότε επιστρέφει $A \vee B$.
Αν, όμως, $Q_1 = \forall$, τότε επιστρέφει $A \wedge B$.
Συνεχίζει αυτή την διαδικασία μέχρι να εξαλειφθούν όλα οι ποσοδείκτες της έκφρασης.[7]

<u>Παράδειγμα Κατανόησης 3.5:</u>
Έστω η ακόλουθη QBF έκφραση:

$$\forall a \exists b, c\ (a \vee b) \wedge (\neg a \vee c) \wedge (\neg b \vee \neg c)$$

Θα εξετάσουμε αν η πιο πάνω QBF έκφραση είναι ικανοποιήσιμη.
Η πιο πάνω έκφραση σε φυσική γλώσσα:
"Για κάθε τιμή της a, υπάρχουν τιμές για τις μεταβλητές b και c τέτοια ώστε η έκφραση $(a \vee b) \wedge (\neg a \vee c) \wedge (\neg b \vee \neg c)$ να είναι αληθής;"

Οι πιθανές τιμές που έχει η μεταβλητή a είναι αληθής και ψευδής.

Πρώτα θα αναθέσουμε την μεταβλητή a με αληθές και θα βρούμε τιμές για τις μεταβλητές b και c. Άρα αφού a = True, για να ικανοποιείται η 2η πρόταση της έκφρασης (δηλαδή η (¬a ∨ c)) θα πρέπει να ανατεθεί η τιμή αληθές στην μεταβλητή c. Τώρα, αφού c = False, για να ικανοποιείται η 2η πρόταση της έκφρασης (δηλαδή η (¬b ∨ ¬c)), θα πρέπει να αναθέσουμε την μεταβλητή b με ψευδές. Επομένως, η πρώτη ανάθεση που έχουμε είναι {a = True, b = False, c = True}.
Στη συνέχεια, θα εξετάσουμε και την περίπτωση όπου αναθέτουμε την μεταβλητή a με ψευδές. Αν η a είναι ψευδές, τότε η μεταβλητή b θα πρέπει να είναι αληθές για να ικανοποιηθεί η 1η πρόταση (δηλαδή η (*a* ∨ b)). Αφού, η μεταβλητή b είναι αληθές, θα πρέπει να αναθέσουμε την μεταβλητή c με ψευδές για να ικανοποιηθεί η 3η πρόταση ((¬b ∨ ¬c)) της έκφρασης. Άρα, η δεύτερη ανάθεση τιμών είναι η {a = False, b = True, c = False}.

Συνεπώς, αφού για κάθε τιμή της a υπάρχει ανάθεση τιμών που οδηγεί στην ικανοποίηση της (a ∨ b) ∧ (¬a ∨ c) ∧ (¬b ∨ ¬c), τότε η QBF έκφραση είναι ικανοποιήσιμη.

Παράδειγμα Κατανόησης 3.6:

Έστω η ακόλουθη QBF έκφραση:

∃b∀*a*∃c (a ∨ b) ∧ (¬a ∨ ¬c) ∧ (¬b ∨ ¬c) ∧ (c)

Θα εξετάσουμε αν η πιο πάνω QBF έκφραση είναι ικανοποιήσιμη.

Θα αρχίσουμε με την ανάθεση της b με ψευδής. Αν αναθέσουμε την μεταβλητή a την τιμή ψευδής. Τότε δεν ικανοποιείται λόγω της πρότασης (a ∨ b). Επομένως απορρίπτεται απευθείας αυτό τα σενάριο αφού δεν ισχύει για όλες τις τιμές της a.
Τώρα θα εξετάσουμε την περίπτωση που αναθέτουμε αληθές την μεταβλητή b. Αν αναθέσουμε την μεταβλητή a με αληθής, τότε για να ικανοποιηθεί η πρόταση (¬a ∨ ¬c), θα πρέπει να αναθέσουμε με ψευδές την μεταβλητή c. Όμως, αυτό δεν μπορεί να γίνει αφού μετά δεν θα μπορεί να ικανοποιηθεί η πρόταση (c). Άρα, απορρίπτεται και αυτή η περίπτωση απευθείας αφού δεν μπορεί να ισχύει για όλες τις τιμές της a.
Συνεπώς, η QBF έκφραση είναι μη-ικανοποιήσιμη αφού δεν υπάρχει κάποια ανάθεση τιμών που να την ικανοποιεί.

Όπως είδαμε στο προηγούμενο παράδειγμα, υπάρχουν περιπτώσεις όπου μια QBF έκφραση δεν έχει λύση. Σε αυτές τις περιπτώσεις έχουμε τις ευέλικτες μεταβλητές, οι οποίες έχουν την δυνατότητα να αλλάζουν επίπεδα με σκοπό να βρεθεί ένα επίπεδο για κάθε ευέλικτη μεταβλητή που να ικανοποιεί την έκφραση. Πιο κάτω θα εξηγηθούν εκτενέστερα οι ευέλικτες μεταβλητές για να κατανοηθούν καλύτερα, καθώς θα χρησιμοποιηθούν στα επόμενα κεφάλαια.

QBF με ευέλικτες μεταβλητές (soft variables):

Η σύνταξη των ευέλικτων μεταβλητών στο πλαίσιο του QBF ορίζεται ως εξής:

Σε μια QBF, $Q_1X_1Q_2X_2\ldots Q_nX_n$, το επίπεδο ποσοτικοποίησης (quantification level) μιας μεταβλητής $x \in X_i$ είναι σταθερό. Οι ελαστικές μεταβλητές προκαθορίζονται με το σύμβολο $Q^{\forall,\exists}_{L}$, όπου το L είναι σύνολο από τα επιτρεπόμενα επίπεδα ποσοτικοποίησης (quantification level).

Επίπεδο Ποσοτικοποίησης: $\underbrace{\exists a,b}_{\text{Level 1}}\ \overbrace{\forall y_0}^{\text{Level 2}}\ \underbrace{\exists x_0x_1x_2s}_{\text{Level 3}}$

Στο QBF με ελαστικές μεταβλητές επιτρέπεται η αλλαγή επιπέδων των μεταβλητών

Για παράδειγμα:

Η έκφραση $F = Q^{\forall,\exists}_{\{1,2,3\}}x\forall y\exists z\Phi$, επιτρέπει η μεταβλητή x να μετακινηθεί στα επίπεδα 1,2 και 3 προκαλώντας τρεις πιθανές εκφράσεις:

1. $F_1 = \exists x\forall y\exists z\ \Phi$
2. $F_2 = \forall xy\exists z\ \Phi$
3. $F_3 = \forall y\exists xz\ \Phi$

Για κάθε επιτρεπόμενο επίπεδο ποσοτικοποίησης (quantification level) καθορίζεται από τον χρήστη ένα σχετικό κόστος και συμβολίζεται με:

$\sigma(\mu\varepsilon\tau\alpha\beta\lambda\eta\tau\acute{\eta}, \varepsilon\pi\acute{\iota}\pi\varepsilon\delta o) = \kappa\acute{o}\sigma\tau o\varsigma.$

Στο παράδειγμα μας τα κόστη καθορίζονται ως εξής: σ(x, 1) =3, σ(x, 2) =2, σ(x, 3) =1.

Η λύση για το F είναι η αληθής ανάθεση που ικανοποιεί την F_1. Αν η F_1 είναι μη-ικανοποιήσιμη, η λύση του F είναι οποιαδήποτε ικανοποιητική ανάθεση του F_2. Παρόμοια το F_3 είναι η λύση αν η F_2 είναι μη-ικανοποιήσιμη.

Στις εκφράσεις QBF με ελαστικές μεταβλητές προκύπτει το πρόβλημα της βελτιστοποίησης. Το πρόβλημα αυτό είναι η εύρεση ενός επιπέδου για κάθε ευέλικτη μεταβλητή, τέτοιο ώστε η σχετική προτασιακή έκφραση F να ικανοποιείτε, και το άθροισμα των κοστών από τις ελαστικές μεταβλητές να μεγιστοποιείται.

Στη συνέχεια θα μελετήσουμε κάποια παραδείγματα εκφράσεων QBF οι οποίες είναι μη-ικανοποιήσιμες και με την χρήση των ευέλικτων μεταβλητών θα προσπαθήσουμε να βρούμε λύση που να τις ικανοποιεί.

Παράδειγμα Κατανόησης 3.7:

Έστω η ακόλουθη QBF έκφραση:

$$\exists d \forall a, b \exists c\ (a \vee c) \wedge (d \vee \neg c) \wedge (\neg b \vee c) \wedge (d \vee \neg c)$$

Η οποία είναι μη-ικανοποιήσιμη.

Έστω ότι d είναι ψευδές, αν αναθέσουμε ψευδές και τις μεταβλητές a και b, θα πρέπει να αναθέσουμε την μεταβλητή c με αληθές για να ισχύει η πρόταση $(a \vee c)$. Όμως, μετά δεν θα ισχύουν οι προτάσεις $(d \vee \neg c)$ και $(d \vee \neg c)$. Επομένως, αφού η $(a \vee c) \wedge (d \vee \neg c) \wedge (\neg b \vee c) \wedge (d \vee \neg c)$ δεν ισχύει για κάθε τιμή στη a και b, η έκφραση QBF είναι μη-ικανοποιήσιμη.

Όπως είδαμε η έκφραση δεν είναι ικανοποιήσιμη.

Θα κάνουμε την μεταβλητή a ευέλικτη μεταβλητή αφού το πρόβλημα ήταν ότι δεν μπορεί να ισχύει για όλες τις τιμές της μεταβλητής a και b.

Επομένως, η QBF έκφραση με ευέλικτες μεταβλητές θα είναι η εξής:

$$Q^{\forall, \exists}_{\{1, 2, 3\}} a\ \exists d\ \forall b\ \exists c\ (a \vee c) \wedge (d \vee \neg c) \wedge (\neg b \vee c) \wedge (d \vee \neg c)$$

Οι πιθανές μετακινήσεις της μεταβλητής a που έχουμε είναι:

- $F_1 = \exists d\ \forall a,b\ \exists c$
- $F_2 = \exists d,a\ \forall b\ \exists c$
- $F_3 = \exists d\ \forall b\ \exists c,a.$

Η πρώτη πιθανή μετακίνηση είναι η ίδια με αυτή που είδαμε στο παράδειγμα κατανόησης 3.6.

Οπότε θα ελέγξουμε τις άλλες δυο.

Πρώτα για την $F_2 = \exists d,a\, \forall b\, \exists c$:

$$\exists d, a\, \forall b\, \exists c\, (a \vee c) \wedge (d \vee \neg c) \wedge (\neg b \vee c) \wedge (d \vee \neg c)$$

1. Έστω ότι αναθέτουμε τις μεταβλητές d και a με ψευδές.
    - Αναθέτουμε την b με αληθές. Για να ισχύει η πρόταση $(d \vee \neg c)$ πρέπει να αναθέσουμε την μεταβλητή c ίση με ψευδές. Όμως, αν γίνει αυτό δεν θα ισχύει η πρόταση $(\neg b \vee c)$.
2. Αναθέτουμε την μεταβλητή a με αληθές και την d με ψευδές.
    - Αναθέτουμε την b με αληθές. Για να ισχύει η πρόταση $(d \vee \neg c)$ πρέπει να αναθέσουμε την μεταβλητή c ίση με ψευδές. Όμως, αν γίνει αυτό δεν 0θα ισχύει η πρόταση $(\neg b \vee c)$.
3. Αναθέτουμε την μεταβλητή a με ψευδές και την d με αληθές.
    - Αναθέτουμε την b με αληθές. Για να ισχύει η πρόταση $(\neg b \vee c)$ πρέπει να αναθέσουμε την μεταβλητή c ίση με αληθές. Αν γίνει αυτό θα ισχύουν όλες οι προτάσεις
    - Αναθέτουμε την b με ψευδές. Για να ισχύει η πρόταση $(a \vee c)$ πρέπει να αναθέσουμε την μεταβλητή c ίση με αληθές. Αν γίνει αυτό θα ισχύουν όλες οι προτάσεις

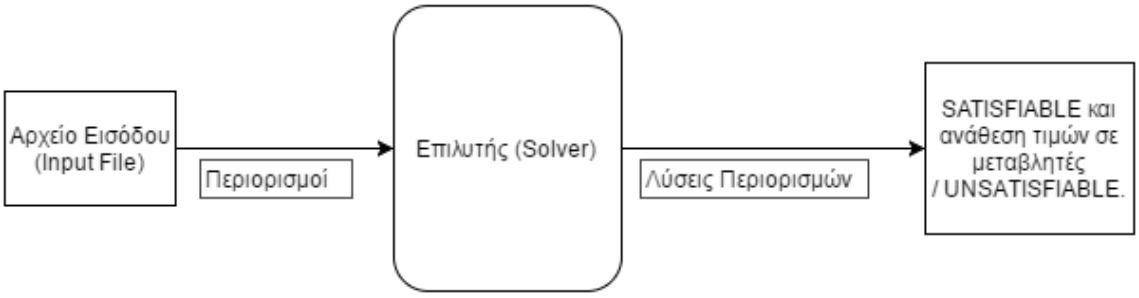
    Επομένως βρήκαμε ένα ζευγάρι τιμών για τις d και a που για κάθε τιμή της b υπάρχει μια τιμή για την c που ικανοποιεί την $(a \vee c) \wedge (d \vee \neg c) \wedge (\neg b \vee c) \wedge (d \vee \neg c)$.

    Άρα ικανοποιείται η έκφραση $\exists d, a\, \forall b\, \exists c\, (a \vee c) \wedge (d \vee \neg c) \wedge (\neg b \vee c) \wedge (d \vee \neg c)$.

Συνεπώς, με την χρήση ελαστικών μεταβλητών βρήκαμε μια ευέλικτη λύση που ικανοποίει την QBF έκφραση

3.4 Ο επιλυτής Quantom p

Οι επιλυτές συνήθως εμπεριέχουν αλγόριθμους επίλυσης περιορισμών. Πιο κάτω στο σχήμα 4.1 φαίνεται πως λειτουργεί ένας προτασιακός επιλυτής.



Σχήμα 4.1

Ο επιλυτής Quantom είναι ένας προτασιακός επιλυτής, ο οποίος επιλύνει προβλήματα ικανοποιησιμότητας. Πιο συγκεκριμένα, δέχεται σαν είσοδο ένα αρχείο (input file) από προτάσεις (clauses) σε κανονική διαζευκτική μορφή (DNF), δηλαδή προτάσεις που είναι της μορφής $d_0 \vee d_1 \vee \ldots \vee d_n$. Βάση των προτάσεων αυτών, αν υπάρχει μια ανάθεση τιμών των μεταβλητών που ικανοποιεί όλες τις προτάσεις του αρχείου εισόδου, επιστρέφει SATISFIABLE και την ανάθεση αυτή. Αλλιώς, επιστρέφει UNSATISFIABLE.

Το αρχείο εισόδου που δέχεται ο Quantom πρέπει να ξεκινά με την πιο κάτω γραμμή:

p cnf x y

x: ο αριθμός των μεταβλητών και

y ο αριθμός των προτάσεων (clauses) στο αρχείο εισόδου.

Στη συνέχεια του αρχείου θα πρέπει να καθοριστούν οι ελαστικές μεταβλητές του προβλήματος, δηλαδή οι μεταβλητές που μπορούν μετακινηθούν επίπεδο ποσοτικοποίησης(θα μελετηθεί στα επόμενα υποκεφάλαια) και αμέσως μετά οι προτάσεις(clauses) οι οποίες περιγράφουν τους περιορισμούς.

Στα επόμενα υποκεφάλαια θα ορισθεί το πρόβλημα επιχειρηματολογίας με ελλιπή γνώση και το πώς μετατρέπεται ένα τέτοιο πρόβλημα σε αρχείο εισόδου για να δοθεί στον solver για να μας βρει κάποια λύση ισχυρών επιχειρημάτων.

Κεφάλαιο 4

Ισχυρά σύνολα και Ποσοτικοποιημένος Δυαδικός Τύπος (QBF)

4.1 Το πρόβλημα των γενικευμένων ισχυρών συνόλων 29
4.2 Υπολογισμός γενικευμένων ισχυρών συνόλων με χρήση QBF 32

4.1 Το πρόβλημα των γενικευμένων ισχυρών συνόλων

Το πρόβλημα με το οποίο θα ασχοληθούμε είναι το εξής:
«Δίνεται μια σειρά από επιχειρήματα και ορισμένες υποθέσεις πάνω σε ένα μη-ολοκληρωμένο (incomplete) σύνολο επιθέσεων και επιχειρημάτων. Υπάρχει κάποιο σύνολο ισχυρών επιχειρημάτων που αν εισαχθεί στο αρχικό σύνολο των επιχειρημάτων θα επιτρέψει την αποδοχή ενός συγκεκριμένου επιχειρήματος που μέχρις στιγμής απορρίπτεται; "

Στη συνέχεια, πρώτα θα καθοριστεί το πρόβλημα των ισχυρών συνόλων λαμβάνοντας υπόψη ολοκληρωμένη (complete) γνώση. Πιο μετά θα γενικεύσουμε σε πρόβλημα με ελλιπή (incomplete) γνώση.

Ορισμός 3.1 [6]:
Σε ένα πρόβλημα ισχυρού συνόλου G = (T, S, L, q), μας δίνονται:

- ένα σύστημα επιχειρηματολογίας T = (A, R),
- ένα σύνολο επιχειρημάτων S, τέτοιο ώστε $S \cap A = \emptyset$ και μια η δυαδική σχέση $L \subseteq S \times (S \cup A)$, και
- ένα επιχείρημα q

και πρέπει να βρεθεί ένα σύνολο $S' \subseteq S$ τέτοιο ώστε το q είναι εύπιστο συμπέρασμα του συστήματος επιχειρηματολογίας $T' = (A \cup S', R \cup L_{S'})$, όπου $L_{S'} = \{(p,q) \mid (p,q) \in L$ και $p \in S', q \in S' \cup A\}$. Καλούμε το S' ισχυρό σύνολο του G.

Πρόταση 3.1 [6]: Το πρόβλημα απόφασης ισχυρού συνόλου είναι NP-complete.

Η έννοια ενός ολοκληρωμένου (complete) συστήματος επιχειρηματολογίας επεκτείνεται σε ένα μη-ολοκληρωμένο (incomplete) σύστημα επιχειρηματολογίας. Διαισθητικά, σε ένα μη-ολοκληρωμένο σύστημα, η σχέση επίθεσης είναι μερικώς γνωστή, με την έννοια ότι υπάρχουν ζεύγη επιχειρημάτων a_i a_j που έχουν σχέση επίθεσης, αλλά είναι άγνωστο ποιό από τα (a_i, a_j), $(a_{j,} a_i)$ ανήκει σε αυτή την σχέση.

Ορισμός 3.2 [6]:

Ένα μη-οκοληρωμένο σύστημα επιχειρηματολογίας είναι η τριάδα I = (A, R, U), τέτοιο ώστε το A να είναι ένα σύνολο από επιχειρήματα, $R \subseteq A \times A$, $U \subseteq A \times A$, τέτοια ώστε

- Αν $(a_i, a_j) \in U$ τότε $(a_{j,} a_i) \in U$
- Αν $(a_i, a_j) \in U$ τότε $\{(a_i, a_j), (a_{j,} a_i)\} \cap R = \emptyset$

Ένα σύστημα επιχειρηματολογίας $T = (A_T, R_T)$ είναι η συμπλήρωση ενός μη-ολοκληρωμένου συστήματος $I = (A_I, R_I, U_I)$ αν

- $A_T = A_I$, $R_I \subseteq R_T$
- Αν $(a_i, a_j) \in R_T \setminus R_I$ τότε $(a_i, a_j) \in U_I$
- Αν $(a_i, a_j) \in U_I$ τότε $\{(a_i, a_j), (a_{j,} a_i)\} \cap R_T \neq \emptyset$

Ένα επιχείρημα q είναι ένα εύπιστο συμπέρασμα ενός μη-ολοκληρωμένου συστήματος επιχειρηματολογίας I αν είναι εύπιστο συμπέρασμα όλων των συμπληρώσεων του I.

Με απλά λόγια, η συμπλήρωση ενός μη-ολοκληρωμένου συστήματος επιχειρηματολογίας I = (A, R, U) είναι ένα σύστημα που επεκτείνει το R με στοιχεία από το U.

Παράδειγμα:

Δίνεται το πιο κάτω πρόβλημα επιχειρηματολογίας $T = (A_T, R_T)$.

$A = \{a_1, a_2, a_3, a_{4,} a_5, a_6\}$,

$R = \{(a_1, a_2), (a_4, a_1), (a_1, a_5), (a_2, a_4), (a_3, a_2), (a_4, a_6), (a_5, a_3), (a_5, a_2), (a_6, a_1), (a_6, a_5)\}$

Και ένα ελλιπές πρόβλημα επιχειρηματολογίας $I = (A_I, R_I, U_I)$.

$A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$,

$R = \{(a_4, a_1), (a_1, a_5), (a_2, a_4), (a_3, a_2), (a_4, a_6), (a_5, a_2), (a_6, a_5)\}$

$U = \{(a_1, a_2), (a_2, a_1), (a_5, a_3), (a_3, a_5), (a_6, a_1), (a_1, a_6)\}$

Όπως μπορούμε να δούμε τα δύο προβλήματα επιχειρηματολογίας έχουν τα ίδια επιχειρήματα. Επίσης, βλέπουμε ότι οι επιθέσεις (a_1, a_2) , (a_5, a_3) και (a_6, a_1) που ανήκουν στη σχέση επίθεσης R του συστήματος επιχειρηματολογίας δεν ανήκουν στην σχέση επίθεσης R του ελλιπούς αλλά ανήκουν στην σχέση επίθεσης U του ελλιπούς. Επομένως, μπορούμε να πούμε ότι το πρόβλημα επιχειρηματολογίας $T = (A_T, R_T)$ είναι η συμπλήρωση του ελλιπούς προβλήματος επιχειρηματολογίας $I = (A_I, R_I, U_I)$.

Ορισμός 3.3 [6]:

Σε ένα γενικευμένο πρόβλημα ισχυρού συνόλου G = (T, S, L, q), μας δίνεται:

- ένα μη-ολοκληρωμένο σύστημα επιχειρηματολογίας I = (A, R, U),
- ένα σύνολο επιχειρημάτων S, τέτοιο ώστε $S \cap A = \emptyset$ και μια η δυαδική σχέση $L \subseteq S \times (S \cup A)$, και
- ένα επιχείρημα $q \in A$

και πρέπει να βρούμε ένα σύνολο $S' \subseteq S$ τέτοιο ώστε το q είναι ένα εύπιστο επιχείρημα ενός μη-ολοκληρωμένου συστήματος $I' = (A \cup S', R \cup L_{S'}, U)$, όπου $L_{S'} = \{(p,q) \mid (p,q) \in L$ και $p \in S', q \in S' \cup A\}$.

Παράδειγμα:

Δίνεται το πιο κάτω γενικευμένο πρόβλημα ισχυρού συνόλου G = (T, S, L, q).

$A = \{a_1, a_2, a_3, a_4\}$,

$R = \{(a_1, a_2), (a_4, a_1)\}$,

$U = \{(a_2, a_3), (a_3, a_2), (a_3, a_4), (a_4, a_3)\}$,

$S = \{a_5\}$,

$L = \{(a_5, a_4)\}$,

$q = a_3$

Το ισχυρό σύνολο στο πιο πάνω πρόβλημα είναι το $\{Xa_5, Xa_1, Xa_3\}$.

Πρόταση 3.1 [6]: Το γενικευμένο πρόβλημα απόφασης ισχυρού συνόλου είναι $\sum_2^p - complete.$

4.2 Υπολογισμός γενικευμένων ισχυρών συνόλων με χρήση QBF

Σε αυτή την υποενότητα θα παρουσιαστεί μια μέθοδος επίλυσης γενικευμένων προβλημάτων ισχυρών συνόλων, μεταφράζοντας τα σε QBF. Αυτή η προσέγγιση εμπλουτίζει την επιχειρηματολογία με προηγμένες τεχνικές για την επίλυση εξαιρετικά σύνθετων προβλημάτων που έχουν αναπτυχθεί όλα αυτά τα χρόνια από την κοινότητα της QBF. Υπάρχουν αρκετοί αποτελεσματικοί QBF επιλυτές(solvers), που μπορούν να χρησιμοποιηθούν για την επίλυση γενικευμένων προβλημάτων ισχυρών συνόλων.

Πιο κάτω θα εξηγηθεί πως θα γίνει η μετάφραση ενός γενικευμένου προβλήματος ισχυρού συνόλου σε QBF.

Δεδομένου ενός γενικευμένου προβλήματος ισχυρού συνόλου G, κατασκευάζουμε το $Q_{3,\exists}$

$$Q_G = \exists X_G \forall Y_G \exists Z_G \Phi_G$$

τέτοιο ώστε οι λύσεις του προβλήματος G να είναι σε άμεση αντιστοιχία με τις λύσεις του Q_G.

Οι μεταβλητές της QBF έκφρασης Q_G που θα δημιουργηθεί από το G=(I, S, L, q), με I=(A, R, U) είναι οι εξής:

- $X_G = \{X_{Si} \mid S_i \in S\}$. Δηλαδή, υπάρχει μια μεταβλητή για κάθε επιχείρημα από το σύνολο S. Κάθε αληθής ανάθεση V πάνω στο X_G επάγει ένα σύνολο S' από το S καθορίζεται ως $S' = \{S_i \mid S_i \in S\}$ και $V(X_{Si}) = True\}$.
- $Y_G = Y_G^1 \cup Y_G^2$, είναι το σύνολο των μεταβλητών που καταγράφει τα διαφορετικά δυνατά συμπληρώματα ενός μη-ολοκληρωμένου συστήματος επιχειρηματολογίας I του προβλήματος G. Αναλυτικότερα, κάθε αληθής ανάθεση πάνω στην μεταβλητή του Y_G επάγει την συμπλήρωση του I. Τα μέρη του Y_G ορίζονται ως εξής:

- Y_G^1 = {$b_{i,j}$ | i < j, (a_i,a_j) ∈ U}. Δηλαδή, για κάθε ζευγάρι από επιχειρήματα a_i,a_j που είναι αμοιβαίως συγκρουόμενα στο U, υπάρχει μια μεταβλητή στο $Y_{G.}$ Επιπρόσθετα, $b_{i,j}$ είναι true αν και τα δύο ζευγάρια (a_i,a_j) και (a_j,a_i) του U επιλέχθηκαν για να συμπεριληφθούν στην συμπλήρωση της θεωρίας I.
- Y_G^2 = {$d_{i,j}$ | i < j, (a_i,a_j) ∈ U}. Παρόμοια με το Y_G^1 , υπάρχει μεταβλητή για κάθε ζευγάρι επιχειρημάτων που σχετίζονται μέσω του U. Αν $d_{i,j}$ είναι true, τότε το ζευγάρι (a_i,a_j) συμπεριλαμβάνεται στην συμπλήρωση του μη-ολοκληρωμένου συστήματος επιχειρηματολογίας I του προβλήματος G. Αντιθέτως, αν το $d_{i,j}$ είναι false, το ζευγάρι (a_j,a_i) συμπεριλαμβάνεται στην συμπλήρωση.

- $Z_G = Z_G^1 \cup Z_G^2$, είναι το σύνολο των μεταβλητών που σχετίζονται με τις ευσταθής επεκτάσεις(stable extensions) των συμπληρώσεων ενός μη-ολοκληρωμένου συστήματος επιχειρηματολογίας I του προβλήματος G. Αναλυτικότερα, τα δύο σύνολα που περιλαμβάνουν Z_G ορίζονται ως εξής:
  - Z_G^1 = {$X_{\alpha i}$ | α_i ∈ A}. Δηλαδή, υπάρχει μια μεταβλητή για κάθε επιχείρημα του A. Η μεταβλητή $X_{\alpha i}$ είναι αληθής αν το επιχείρημα α_i ανήκει στις ευσταθής επεκτάσεις (stable extensions) της θεωρίας I, όπως καθορίζεται από την αληθής ανάθεση των μεταβλητών από το σύνολο $Y_{G.}$
  - Z_G^2 = {$t_{i,j}$ | (a_i,a_j) ∈ U}, είναι ένα σύνολο από βοηθητικές μεταβλητές, ένα για κάθε στοιχείο του U. Η μεταβλητή $t_{i,j}$ είναι αληθής αν το επιχείρημα α_i είναι ο λόγος απόρριψης του α_j για την συμπερίληψη του στην τρέχουσα ευσταθή επέκταση(stable extension). Σύμφωνα με την σημασιολογία(semantics) των ευσταθών επεκτάσεων(stable extensions), α_i είναι ο λόγος απόρριψης του α_j, αν το α_j βρίσκεται μέσα στην ευσταθή επέκταση και (a_i,a_j) ανήκει στην σχέση επίθεσης.

Η φόρμουλα Φ_G που σχετίζεται με το γενικευμένο πρόβλημα ισχυρού συνόλου G=(I,S,L,q), όπου I=(A,R,U), είναι ένα σύνολο από προτάσεις που ουσιαστικά κωδικοποιούν τη σημασιολογία των ευσταθών επεκτάσεων. Η διαίσθηση εδώ είναι ότι, δεδομένου μιας αληθής ανάθεσης πάνω στις μεταβλητές των συνόλων X_G και Y_G, οι ευσταθής επεκτάσεις της προκύπτουσας θεωρίας είναι ακριβώς τα σύνολα των επιχειρημάτων που ορίζονται ως $M = \{a_i \mid V(a_i) = True\}$, όπου V είναι η αληθής ανάθεση που ικανοποιεί την φόρμουλα Φ_G, οριζόμενη ως των συνδυασμών των ακόλουθων προτάσεων:

- $\neg X_{\alpha i} \vee X_{\alpha j,}$ για κάθε $(a_i,a_j) \in R \cup U$. Αυτό το σύνολο των προτάσεων(clauses) αναγκάζει τα επιχειρήματα που επιτίθενται το ένα του άλλου, να μην βρίσκονται στην ίδια ευσταθή επέκταση (conflict-free).
- $\neg X\alpha_i \rightarrow X\alpha j_1 \vee \ldots \vee X\alpha j_y \vee tm_{1\,i} \vee tm_{z\,i,}$ όπου για κάθε $1 \leq k \leq y$, $(\alpha j_{k,}\, a_i) \in R$ και για κάθε $1 \leq p \leq z$, $(\alpha m_{p,}\, a_i) \in U$. Αυτές οι προτάσεις καθορίζουν ότι αν ένα επιχείρημα δεν είναι μέσα στην ευσταθή επέκταση, τότε κάποια από τα επιχειρήματα που του επιτίθενται πρέπει να είναι στην επέκταση. Η επίθεση μπορεί να είναι είτε από επιχείρημα του συνόλου A(όταν μία από τις μεταβλητές $X\alpha j_\kappa$ είναι true) ή από επιχείρημα του συνόλου S (όταν μια από τις μεταβλητές $tm_{p\,i}$ είναι true).
- $t_{i,j} \rightarrow (b_{i,j} \vee d_{i,j}) \wedge X_{\alpha i}$ (ή $t_{i,j} \rightarrow (b_{i,j} \vee \neg d_{i,j}) \wedge X_{\alpha i}$, ανάλογα με την κατεύθυνση της ακμής), το οποίο είναι ισοδύναμο με: $(\neg t_{i,j} \vee b_{i,j} \vee d_{i,j}) \wedge (\neg t_{i,j} \vee X_{\alpha i})$. Αυτό το σύνολο των προτάσεων, συνδέουν τις μεταβλητές $t_{i,j}$, $b_{i,j}$ και $d_{i,j.}$ Προϋποθέτει ότι αν υπάρχει επίθεση από το a_i στο a_j στην τρέχουσα επέκταση, το ζεύγος $(a_i,a_j) \in U$ θα πρέπει να συμπεριληφθεί στην συμπλήρωση του μη-ολοκληρωμένου συστήματος επιχειρηματολογίας I, και το επιχείρημα a_i θα πρέπει να είναι στην ευσταθή επέκταση.
- X_q μια μοναδιαία πρόταση που εξασφαλίζει ότι η ευσταθής επέκταση περιέχει το q.

Μπορεί να αποδειχθεί ότι μια αληθής ανάθεση V στις μεταβλητές $X_G = \{X_{Si},\ldots, X_{Sc}\}$ ικανοποιεί Q_G αν και μόνο αν το σύνολο $S = \{S_i \mid V(Xs_i) = True\}$ είναι λύση του αντίστοιχου γενικευμένου προβλήματος ισχυρών συνόλων G. Η ικανοποίηση του Q_G σημαίνει ότι, δεδομένου των τιμών που ανατίθενται στο X_G από το V, για όλους τους πιθανούς συνδυασμούς των τιμών στο X_G, δηλαδή για όλες τις συμπληρώσεις του I, υπάρχει μια αληθής ανάθεση πάνω στις μεταβλητές του X_G όπου ικανοποιεί τις προτάσεις της Φ_G, δηλαδή υπάρχει κάποια ευσταθής επέκταση που περιέχει το q για συμπληρώσεις.

Πιο κάτω φαίνεται ένα παράδειγμα που απεικονίζει το QBF σχετιζόμενο με το γενικευμένο πρόβλημα ισχυρών συνόλων,

<u>Παράδειγμα Κατανόησης 4.1:</u>

Δίνετε το γενικευμένο πρόβλημα ισχυρών συνόλων G = (T, S, L, q) με I = (A, R, U), το οποίο ορίζεται ως εξής:

$A = \{a_1, a_2, a_3, a_4\}$,

$R = \{(a_1, a_2), (a_4, a_1)\}$,

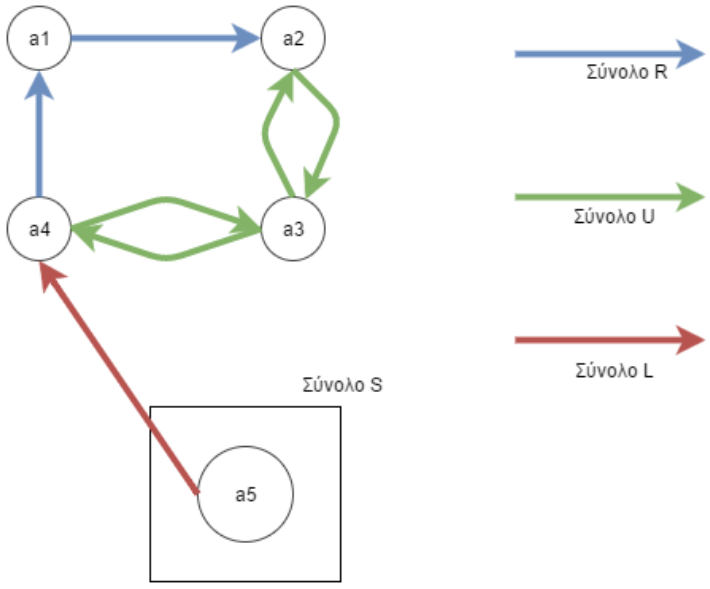
$U = \{(a_2, a_3), (a_3, a_2), (a_3, a_4), (a_4, a_3)\}$,

$S = \{a_5\}$,

$L = \{(a_5, a_4)\}$,

$q = a_3$

Σχηματική Απεικόνιση Προβλήματος:



Σχήμα 4.1

Από το G θα κατασκευάσουμε το QBF.

$Q_G = \exists X\alpha_5 \forall b_{2,3}\, b_{3,4}\, d_{2,3}\, d_{3,4} \exists X\alpha_1\, X\alpha_2\, X\alpha_3\, X\alpha_4\, t_{2,3}\, t_{3,2}\, t_{3,4}\, t_{4,3}\, \Phi_G$

Όπου το Φ_G περιέχει τις πιο κάτω προτάσεις:

$\neg X\alpha_1 \vee \neg X\alpha_2$

$\neg X\alpha_4 \vee \neg X\alpha_1$

$\neg X\alpha_2 \vee \neg X\alpha_3$

$\neg X\alpha_3 \vee \neg X\alpha_4$

$\neg X\alpha_5 \vee \neg X\alpha_4$

$X\alpha_1 \vee X\alpha_4$

$X\alpha_1 \vee X\alpha_2 \vee t_{3,2}$

$X\alpha_3 \vee t_{2,3} \vee t_{4,3}$

$X\alpha_5 \vee X\alpha_4 \vee t_{3,4}$

$\neg t_{2,3} \vee b_{2,3} \vee d_{2,3}$

$\neg t_{2,3} \vee X\alpha_2$

$\neg t_{3,2} \vee b_{2,3} \vee \neg d_{2,3}$

$\neg t_{3,2} \vee X\alpha_3$

$\neg t_{3,4} \vee b_{3,4} \vee d_{3,4}$

$\neg t_{3,4} \vee X\alpha_3$

$\neg t_{4,3} \vee b_{3,4} \vee \neg d_{3,4}$

$\neg t_{4,3} \vee X\alpha_4$

$X\alpha_3$

Όπως μπορούμε να δούμε η ανάθεση τιμής $V(X\alpha_5)$ = True ικανοποιεί την έκφραση Q_G. Για παράδειγμα, αν έχουμε την ανάθεση $b_{2,3}$= False, $d_{2,3}$= False, $b_{3,4}$= False, $d_{3,4}$= True, η οποία αντιστοιχεί στην συμπλήρωση $T' = (A, R \cup R')$ του I, με $R' = \{(a_3, a_2), (a_3, a_4)\}$. Τότε, η ανάθεση των άλλων μεταβλητών, $t_{2,3}$ = False, $t_{3,2}$ = True, $t_{3,4}$ = True, $t_{4,3}$ = False, $X\alpha_1$ = True, $X\alpha_2$ = False, $X\alpha_3$ = True, $X\alpha_4$ = False, ικανοποιεί όλες τις προτάσεις του Φ_G.

Στις περιπτώσεις όπου το γενικευμένο πρόβλημα ισχυρών συνόλων δεν έχει λύση, δηλαδή δεν έχει κάποιο σύνολο επιχειρημάτων από το S που καθιστά ένα επιχείρημα αποδεκτό για όλα τα πιθανά προφίλ του αντιπάλου, μπορεί να θέλουμε να αναζητήσουμε ένα υποσύνολο του S που μπορεί να το πετύχει αυτό για τα περισσότερα από αυτά τα χαρακτηριστικά. Οι πρόσφατες τεχνικές πάνω στο QBF με ελαστικές μεταβλητές μπορούν να χρησιμοποιηθούν για αυτό. Περισσότερα για αυτές τις τεχνικές βρίσκονται στο [8].

Το πλαίσιο του QBF με ελαστικές μεταβλητές που μελετήσαμε στο προηγούμενο κεφάλαιο μπορεί να χρησιμοποιηθεί στην περίπτωση που το Q_G είναι μη-ικανοποιήσιμο. Μετά, μπορούμε να ορίσουμε ένα σύνολο μεταβλητών $d_{i,j}$ και $b_{i,j}$, που αντιστοιχεί στις διαφορετικές κατευθύνσεις των επιθέσεων, ως ελαστικές μεταβλητές. Επιπρόσθετα, λόγω του ότι προτιμούμε η έκφραση να είναι ικανοποιήσιμη από όλες τις πιθανές τιμές των $d_{i,j}$ και $b_{i,j}$, καθορίζουμε κόστος 3 στο δεύτερο επίπεδο ποσοτικοποίησης (δηλαδή στο ∀ επίπεδο), κόστος 2 στο αριστερότερο (∃) επίπεδο και κόστος 1 στο δεξιότερο επίπεδο.

Πιο κάτω δίνετε ένα παράδειγμα για να κατανοηθεί καλύτερα η χρήση ελαστικών μεταβλητών.

Παράδειγμα Κατανόησης 4.2:

Δίνετε το γενικευμένο πρόβλημα ισχυρών συνόλων G = (T, S, L, q) με I = (A, R, U), το οποίο ορίζεται ως εξής:

$A = \{a_1, a_2, a_3, a_4, a_5\}$,

$R = \{(a_2, a_3), (a_3, a_4), (a_5, a_2), (a_5, a_4)\}$,

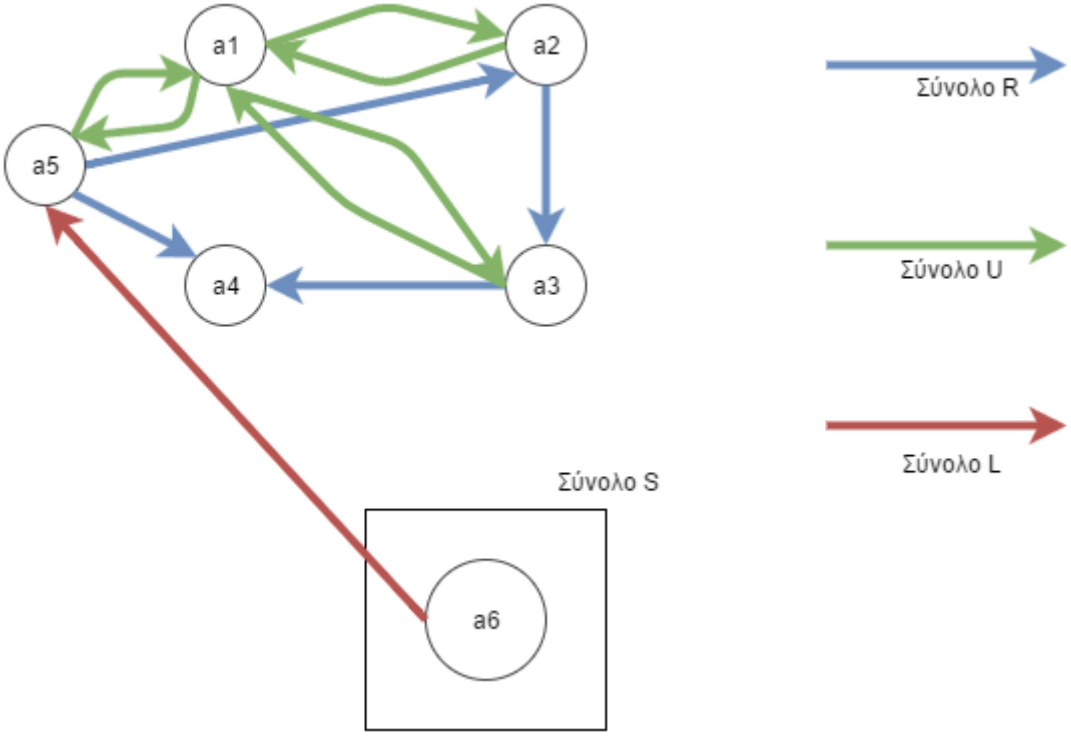
$U = \{(a_1, a_2), (a_2, a_1), (a_1, a_3), (a_3, a_1), (a_1, a_5), (a_5, a_1)\}$,

$S = \{a_6\}$,

$L = \{(a_6, a_5)\}$,

$q = a_4$

Σχηματική Απεικόνιση Προβλήματος:



Σχήμα 4.2

Από το G θα κατασκευάσουμε την QBF έκφραση:

$Q_G = \exists X\alpha_6 \, \forall b_{1,2}b_{1,3}b_{1,5}d_{1,2}d_{1,3}d_{1,5} \, \exists X\alpha_1 X\alpha_2 X\alpha_3 X\alpha_4 X\alpha_5 \, t_{1,2}t_{2,1}t_{1,3}t_{3,1}t_{1,5}t_{5,1} \, \Phi_G$

Όπου το Φ_G είναι τα αντίστοιχα clauses, τα οποία σε αυτό το παράδειγμα παραλείπονται αφού σκοπός του παραδείγματος είναι να δείξουμε πως θα γίνει η έκφραση με ελαστικές μεταβλητές.

Η έκφραση είναι μη-ικανοποιήσιμη, επομένως θα πρέπει να λυθεί με ελαστικές μεταβλητές.

Η αντίστοιχη QBF έκφραση με ελαστικές μεταβλητές είναι:

$$Q_G = \exists X\alpha_6 \, Q^{\forall, \exists}_{\{1, 2, 3\}} \, b_{1,2}b_{1,3}b_{1,5}d_{1,2}d_{1,3}d_{1,5} \, \exists X\alpha_1 X\alpha_2 X\alpha_3 X\alpha_4 X\alpha_5 \, t_{1,2}t_{2,1}t_{1,3}t_{3,1}t_{1,5}t_{5,1} \, \Phi_G$$

Μια λύση αυτής της έκφρασης αναθέτει στην μεταβλητή $X\alpha_6$ την τιμή True, όλες οι μεταβλητές $d_{i,j}$ και $b_{i,j}$ στο δεύτερο επίπεδο ποσοτικοποίησης, εκτός από την μεταβλητή $d_{1,3}$ που καθορίζεται στο πρώτο επίπεδο, με την τιμή True. Από την μεριά του αντίστοιχης θεωρίας της επιχειρηματολογίας, αυτό σημαίνει ότι το $\{a_6\}$ είναι ένα

ισχυρό σύνολο με την προϋπόθεση ότι όταν ένα επιχείρημα a_3 επιτίθεται στο a_1, τότε το a_1 πρέπει να επιτίθεται στο a_3. Πιο συγκεκριμένα , όταν το a_6 συμπεριλαμβάνεται στην θεωρία, το a_4 είναι εύπιστο επιχείρημα για όλα τα χαρακτηριστικά, εκτός αυτού που μεταξύ των υποσυνόλων $\{(a_3, a_1), (a_1, a_3)\}$, περιλαμβάνει μόνο το ζεύγος (a_3, a_1) στην σχέση επίθεσης.

Το πρόβλημα βελτιστοποίησης που αναφέραμε σε αυτό το υποκεφάλαιο μπορεί να χρησιμοποιηθεί για την επιλογή της επόμενης προσφοράς. Η γενική ιδέα είναι να προτιμηθεί η ιδέα που είναι πιο πιθανόν να γίνει αποδεκτή από τον αντίπαλο, όπου πιθανότερα καθορίζετε από το μέγεθος του συνόλου των μεταβλητών στο καθολικό(∀) επίπεδο ποσοτικοποίησης.

Κεφάλαιο 5

Σύστημα Υλοποίησης και Πειράματα

5.1 Επεξήγηση συστήματος υλοποίησης 40

5.2 Πειραματικά Αποτελέσματα 46

5.1 Επεξήγηση συστήματος υλοποίησης

Σε πρώτη φάση, υλοποιήθηκε πρόγραμμα σε γλώσσα προγραμματισμού C, το οποίο δεχόταν σαν είσοδο ένα γενικευμένο πρόβλημα επιχειρηματολογίας, και το μετάφραζε σε QBF. Πιο συγκεκριμένα, δημιουργούσε ένα output αρχείο με την μετάφραση αυτή. Έπειτα, αυτό το αρχείο που δημιουργήθηκε από το πρόγραμμα, δινόταν σαν είσοδο στον επιλυτή quantom, ο οποίος έβρισκε μια ανάθεση τιμών στις μεταβλητές η οποία ικανοποιούσε τους περιορισμούς του αρχείου.

Στη συνέχεια, το πιο πάνω πρόγραμμα επεκτάθηκε προσθέτοντας ελαστικές μεταβλητές, τις οποίες ο solver χρησιμοποιούσε σε περίπτωση που το γενικευμένο πρόβλημα επιχειρηματολογίας δεν έχει λύση. Πιο συγκεκριμένα, χρησιμοποιώντας ελαστικές μεταβλητές, ο solver αναζητούσε ένα υποσύνολο επιχειρημάτων που υπήρχε μεγάλη πιθανότητα να γίνουν αποδεκτά από τον αντίπαλο.

Πιο κάτω παρουσιάζετε ένα παράδειγμα αρχείου εξόδου που δημιουργούσε το πρόγραμμα, και δινόταν στον solver ως είσοδο.
Το παράδειγμα αυτό αναλύθηκε στο προηγούμενο κεφάλαιο.

Παράδειγμα Κατανόησης 5.1:
Δίνετε το γενικευμένο πρόβλημα ισχυρών συνόλων G = (T, S, L, q) με I = (A, R, U), το οποίο ορίζεται ως εξής:
$A = \{a_1, a_2, a_3, a_4\}$,
$R = \{(a_1, a_2), (a_4, a_1)\}$,

$U = \{(a_2, a_3), (a_3, a_2), (a_3, a_4), (a_4, a_3)\}$,

$S = \{a_5\}$,

$L = \{(a_5, a_4)\}$,

$q = a_3$

Από το G θα κατασκευάσουμε το QBF.

$Q_G = \exists X\alpha_5 \forall b_{2,3}\, b_{3,4}\, d_{2,3}\, d_{3,4} \exists X\alpha_1\, X\alpha_2\, X\alpha_3\, X\alpha_4\, t_{2,3}\, t_{3,2}\, t_{3,4}\, t_{4,3}\, \Phi_G$

<u>Αρχείο εισόδου του επιλυτή:</u>

```
p cnf 13 18
e 1 0
s [2,1;3,0] 2 0
s [2,1;3,0] 3 0
s [2,1;3,0] 4 0
s [2,1;3,0] 5 0
e 6 7 8 9 10 11 12 13 0
-6 -7 0
-9 -6 0
-7 -8 0
-8 -9 0
-1 -9 0
6 9 0
7 6 11 0
8 10 13 0
9 1 12 0
-10 2 4 0
-10 7 0
-11 2 -4 0
-11 8 0
-12 3 5 0
-12 8 0
-13 3 -5 0
-13 9 0
8 0
```

1ο επίπεδο ποσοτικοποίησης (quantification level) - $\exists X\alpha_5$

2ο επίπεδο ποσοτικοποίησης - Ελαστικές Μεταβλητές (Soft Variables) - $\forall b_{2,3}\, b_{3,4}\, d_{2,3}\, d_{3,4}$

3ο επίπεδο ποσοτικοποίησης (quantification level) - $\exists X\alpha_1\, X\alpha_2\, X\alpha_3\, X\alpha_4\, t_{2,3}\, t_{3,2}$

Προτάσεις/ Περιορισμοί (Clauses) - Φ_G

Αποτέλεσμα solver:

```
c allowed level for softvariable 5 is set to 3 with
c max varindex...........: 13
c #clauses...............: 18
c cpu limit..............: 0.000s
c decision heuristic.....: 3
c restart heuristic......: 1
c luby shift.............: 12
c initial phase..........: 2
c swap cache.............: 1
c switch Var sign .......: 1
c lhbr...................: 0
c subsumption........... : 1
c preprocessor.......... : 1
c variable elimination.. : 1
c upla.................. : 1
c solve................. : 1
c maxmode............... : 1
c solvemode............. : QBF with soft variables
c cpu time (loading CNF).: 0.000s
c ================================================

c #literals...........: 50
c #clauses............: 88
c #decisions..........: 21
c #BCP operations.....: 39
c #conflicts..........: 0
c #solutions..........: 16
c #pureLiterals.......: 4
c #dcLiterals.........: 4
c #restarts...........: 0
c #restartAttempts....: 0
c #clsimplifications..: 0
c #cusimplifications..: 0
c #lhbr clauses.......: 0
c #inprocessings......: 0
c cpu time............: 0.000s
s SATISFIABLE
v 1 6 -7 8 -9 -10 -11 -12 -13
o 0
sv 2 [2] 3 [2] 4 [2] 5 [2] 0
```

Όπως φαίνεται από την πιο πάνω εικόνα, solver βρήκε λύση που ικανοποιεί τις προτάσεις/περιορισμούς.

Η λύση αυτή είναι:

1 6 -7 8 -9 -10 -11 -12 -13

2 [2] 3[2] 4[2] 5[2]

Η οποία σημαίνει ότι:

Το σύνολο επιχειρημάτων {Xa_5, Xa_1, Xa_3} αποτελεί το ισχυρό σύνολο.

Η γραμμή 2 [2] 3[2] 4[2] 5[2] , δείχνει σε πιο επίπεδο ποσοτικοποίησης έγιναν αληθής οι μεταβλητές 2, 3, 4, 5. Σε αυτή την περίπτωση γίνονται όλες στο επίπεδο 2, άρα δεν χρειάστηκε επαναδιορισμός επιχειρημάτων.

Στο επόμενο παράδειγμα φαίνεται η περίπτωση που χρειάζεται η αλλαγή επιπέδου ποσοτικοποίησης ούτως ώστε να βρεθεί ένα σύνολο ισχυρών επιχειρημάτων.

Όπως και το προηγούμενο παράδειγμα, έτσι και αυτό αναλύθηκε στο προηγούμενο κεφάλαιο.

Παράδειγμα Κατανόησης 4.2:

Δίνετε το γενικευμένο πρόβλημα ισχυρών συνόλων G = (T, S, L, q) με I = (A, R, U), το οποίο ορίζεται ως εξής:

$A = \{a_1, a_2, a_3, a_4, a_5\}$,

$R = \{(a_2, a_3), (a_3, a_4), (a_5, a_2), (a_5, a_4)\}$,

$U = \{(a_1, a_2), (a_2, a_1), (a_1, a_3), (a_3, a_1), (a_1, a_5), (a_5, a_1)\}$,

$S = \{a_6\}$,

$L = \{(a_6, a_5)\}$,

$q = a_4$

Από το G θα κατασκευάσουμε την QBF έκφραση:

$Q_G = \exists X\alpha_6 \forall b_{1,2}b_{1,3}b_{1,5}d_{1,2}d_{1,3}d_{1,5} \exists X\alpha_1 X\alpha_2 X\alpha_3 X\alpha_4 X\alpha_5 t_{1,2}t_{2,1}t_{1,3}t_{3,1}t_{1,5}t_{5,1} \Phi_G$

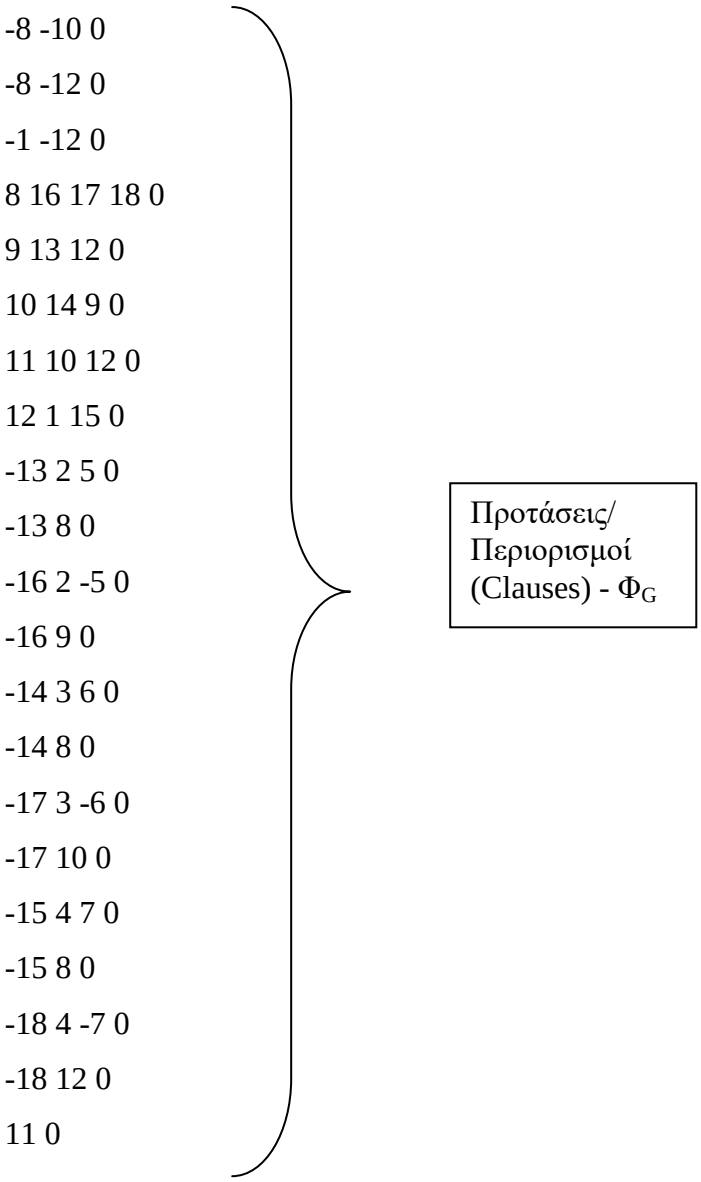
Αρχείο εισόδου του επιλυτή:

```
p cnf 18 26
e 1 0
s [2,1;3,0] 2 0
s [2,1;3,0] 3 0
s [2,1;3,0] 4 0
s [2,1;3,0] 5 0
s [2,1;3,0] 6 0
s [2,1;3,0] 7 0
e 8 9 10 11 12 13 14 15 16 17 18 0
-9 -10 0
-10 -11 0
-12 -9 0
-12 -11 0
-8 -9 0
```

1ο επίπεδο ποσοτικοποίησης (quantification level) - $\exists X\alpha_6$

2ο επίπεδο ποσοτικοποίησης - Ελαστικές Μεταβλητές (Soft Variables) - $\forall b_{1,2}b_{1,3}b_{1,5}d_{1,2}d_{1,3}d_{1,5}$

3ο επίπεδο ποσοτικοποίησης (quantification level) - $\exists X\alpha_1 X\alpha_2 X\alpha_3 X\alpha_4 X\alpha_5 t_{1,2}t_{2,1}t_{1,3}t_{3,1}t_{1,5}t_{5,1}$


-8 -10 0
-8 -12 0
-1 -12 0
8 16 17 18 0
9 13 12 0
10 14 9 0
11 10 12 0
12 1 15 0
-13 2 5 0
-13 8 0
-16 2 -5 0
-16 9 0
-14 3 6 0
-14 8 0
-17 3 -6 0
-17 10 0
-15 4 7 0
-15 8 0
-18 4 -7 0
-18 12 0
11 0
Προτάσεις/
Περιορισμοί
(Clauses) - Φ_G

Αποτέλεσμα solver:

```
c max varindex...........: 18
c #clauses..............: 26
c cpu limit.............: 0.000s
c decision heuristic.....: 3
c restart heuristic......: 1
c luby shift............: 12
c initial phase..........: 2
c swap cache............: 1
c switch Var sign .......: 1
c lhbr...................: 0
c subsumption........... : 1
c preprocessor.......... : 1
c variable elimination.. : 1
c upla.................. : 1
c solve................. : 1
c maxmode............... : 1
c solvemode............. : QBF with soft variables
c cpu time (loading CNF).: 0.001s
c ============================================================

c #literals...........: 81
c #clauses............: 159
c #decisions..........: 253
c #BCP operations.....: 325
c #conflicts..........: 4
c #solutions..........: 66
c #pureLiterals.......: 41
c #dcLiterals.........: 41
c #restarts...........: 0
c #restartAttempts....: 0
c #clsimplifications..: 0
c #cusimplifications..: 1
c #lhbr clauses.......: 0
c #inprocessings......: 0
c cpu time............: 0.001s
s SATISFIABLE
v 1 -8 9 -10 11 -12 -13 -14 -15 16 -17 -18
o 1
sv 2 [2] 3 [2] 4 [2] 5 [3] 6 [2] 7 [2] 0
103ws15:/home/students/cs/2012/rlouka01/Downloads>
```

Όπως φαίνεται από την πιο πάνω εικόνα, solver βρήκε λύση που ικανοποιεί τις προτάσεις/περιορισμούς.

Η λύση αυτή είναι:

1 -8 9 10 11 -12 -13 -14 -15 16 -17 -18

2 [2] 3[2] 4[2] 5[3] 6[3] 7[2]

Η οποία σημαίνει ότι:

Το σύνολο επιχειρημάτων {Xa_6, Xa_2, Xa_3, Xa_4} αποτελεί το ισχυρό σύνολο.

Όπως μπορούμε να δούμε στην γραμμή 2 [2] 3[2] 4[2] 5[3] 6[3] 7[2], οι μεταβλητές 2, 3, 4, 5, 7 καθορίζονται στο επίπεδο 2. Ενώ στη μεταβλητή 6 καθορίζεται στο επίπεδο 3.

5.2 Πειραματικά Αποτελέσματα

Στην τελευταία φάση έγινε μια πειραματική αξιολόγηση του συστήματος που υλοποιήθηκε. Δημιουργήθηκαν διάφορα προβλήματα τα οποία διέφεραν στον αριθμό επιχειρημάτων, στην πυκνότητα καθώς επίσης στο ποσοστό αβεβαιότητας(δηλαδή το ποσοστό από τις επιθέσεις που δεν ξέρουμε σίγουρα την κατεύθυνση).

Βάση της πυκνότητας και του ποσοστού αβεβαιότητας υπολογίζονται ανάλογα ο αριθμός των γνωστών επιθέσεων και ο αριθμός των ελλιπών επιθέσεων.

Η πυκνότητα των επιθέσεων υπολογίζεται ανάλογα με τον αριθμό των επιχειρημάτων (δηλαδή πυκνότητα * αρ. επιχειρημάτων * (αρ. επιχειρημάτων-1)).

Το ποσοστό αβεβαιότητας εκφράζει τον αριθμό από τις επιθέσεις που δεν γνωρίζουμε την ακριβή κατεύθυνση τους. Σχετίζεται άμεσα με την πυκνότητα των επιθέσεων. Αναλυτικότερα, αφού υπολογίσουμε των συνολικό αριθμό των επιθέσεων στο σύστημα με βάση την πυκνότητα, το ποσοστό αβεβαιότητας θα καθορίσει πόσες από αυτές δεν γνωρίζουμε την κατεύθυνση τους, δηλαδή δεν μπορούμε να πούμε με σιγουρία ποιο επιχείρημα επιτίθεται σε ποιο. ↔

Μέσα από αυτά τα πειράματα φτάσαμε σε κάποια συμπεράσματα κατά πόσο ο τρόπος εύρεσης των επιχειρημάτων είναι αποδοτικός. Τα συμπεράσματα αυτά θα παρουσιαστούν προς το τέλος του υποκεφαλαίου αυτού.

Πιο κάτω παρουσιάζονται τα διάφορα πειράματα που έγιναν και ένας μικρός σχολιασμός των αποτελεσμάτων.

1ο πείραμα: Κρατήσαμε σταθερά την πυκνότητα και το ποσοστό αβεβαιότητας και αλλάζαμε σταδιακά τον αριθμό των επιχειρημάτων. Για να μην επηρεάσουν σε μεγάλο βαθμό οι άλλοι δύο παράμετροι, τα ορίσαμε με χαμηλά ποσοστά. Πιο συγκεκριμένα, η πυκνότητα ήτα 10% και το ποσοστό αβεβαιότητας επίσης 10%.

Στον πίνακα 5.1 φαίνεται ο αριθμός των κόμβων και ο χρόνος σε δευτερόλεπτα που χρειάστηκε ο επιλυτής για να βρει ένα σύνολο ισχυρών επιχειρημάτων.

Αρ. Επιχειρημάτων	10	15	20	25	30	35	40
Χρόνος(sec)	0.0004	0.001	0.0013	0.0099	0.0195	0.0727	54.1528

Πίνακας 5.1 - Πυκνότητα 10% και Αβεβαιότητα 10%

Όπως μπορούμε να δούμε από τον πίνακα 5.1, όσο αυξάνονται τα επιχειρήματα, ο χρόνος εύρεσης λύσης μεγαλώνει αρκετά και μετά από ένα αριθμό επιχειρημάτων, ο χρόνος είναι πάρα πολύ μεγάλος αφού παίρνει ώρες για να βρει λύση.

2ο πείραμα: Κρατήθηκαν σταθερά ο αριθμός των επιχειρημάτων και το ποσοστό αβεβαιότητας και μεταβαλλόταν η πυκνότητα. Σε αυτό το πείραμα ορίσαμε τον αριθμό επιχειρημάτων ίσο με 20 και το ποσοστό αβεβαιότητας 10%.
Σημείωση: Όπως αναφέρθηκε και προηγουμένως, η πυκνότητα των επιθέσεων σχετίζεται άμεσα με τον αριθμό τον επιχειρημάτων.

Στον πίνακα 5.2 φαίνεται το ποσοστό της πυκνότητας και ο χρόνος σε δευτερόλεπτα που χρειάστηκε ο επιλυτής για να βρει ένα σύνολο ισχυρών επιχειρημάτων.

Πυκνότητα (%)	10	15	20	25	30	35	40
Χρόνος(sec)	0.0013	0.0022	0.0064	0.016	0.0567	0.9092	28.5096

Πίνακας 5.2 Αριθμός Επιχειρημάτων 20, Αβεβαιότητα 10%

Τώρα όσον αφορά τον πίνακα 5.2 και τον έλεγχο της πυκνότητας, βλέπουμε ότι ισχύει το ίδιο με τον πίνακα 5.1. Δηλαδή για μικρά ποσοστά πυκνότητας η εύρεση λύσης είναι

αρκετά αποδοτική. Όμως, για μεγάλα ποσοστά πυκνότητας, ο χρόνος εύρεσης λύσης στο πρόβλημα επιχειρηματολογίας είναι αρκετά μεγάλος.

3ο πείραμα: Σε αυτό το πείραμα κρατήσαμε σταθερά τον αριθμό των επιχειρημάτων και το ποσοστό πυκνότητας και αλλάξαμε το ποσοστό της αβεβαιότητας. Αρχικά ορίσαμε τον αριθμό επιχειρημάτων με 20, όπως και στο προηγούμενο πείραμα και το ποσοστό πυκνότητας ίσο με 15%. Στη συνέχεια, έγινε ακόμη μια δοκιμή με 20 επιχειρήματα ξανά και 20% πυκνότητα.

Στον πίνακα 5.3 φαίνεται το ποσοστό της αβεβαιότητας και ο χρόνος σε δευτερόλεπτα που χρειάστηκε ο επιλυτής για να βρει ένα σύνολο ισχυρών επιχειρημάτων, για 20 επιχειρήματα και 15% πυκνότητα. Ενώ, στον πίνακα 5.4 για 20 επιχειρήματα και 20% πυκνότητα.

Ποσοστό Αβεβαιότητας	10	15	20	25	30
Χρόνος(sec)	0.0021	0.0199	0.0672	11.6454	809.4552

Πίνακας 5.3 Επιχειρήματα 20, Πυκνότητα 15%

Ποσοστό Αβεβαιότητας	10	15	20
Χρόνος(sec)	0.0064	0.073	41.7025

Πίνακας 5.4 Επιχειρήματα 20, Πυκνότητα 20%

Στους πίνακες 5.3 και 5.4, μπορούμε να δούμε ότι το ποσοστό αβεβαιότητας επηρεάζει σε μεγάλο βαθμό τον χρόνο αφού ο επιλυτής προσπαθεί να βρει όλες τις πιθανές λύσεις και λόγω του ότι υπάρχουν πολλές επιθέσεις που δεν γνωρίζουμε τις κατευθύνσεις τους (αβεβαιότητα), δημιουργούνται μεγάλα προβλήματα στην εύρεση λύσης.

Συνοψίζοντας, μέσω των πειραμάτων που εξάγονται τα πιο κάτω συμπεράσματα:

1. Η μέθοδος που χρησιμοποιήσαμε για την εύρεση λύσης σε ένα πρόβλημα επιχειρηματολογίας είναι αποδοτική για μικρό αριθμό επιχειρημάτων. Για μεγάλο αριθμό επιχειρημάτων, ο χρόνος που χρειάζεται για να βρεθεί λύση είναι πολύ μεγάλος, δηλαδή δεν είναι αποδοτική η μέθοδος. Ο αριθμός αυτός εξαρτάται άμεσα από τα ποσοστά πυκνότητας και αβεβαιότητας.

2. Η μέθοδος που χρησιμοποιήσαμε για την εύρεση ενός συνόλου ισχυρών επιχειρημάτων σε ένα πρόβλημα επιχειρηματολογίας είναι αποδοτική για μικρό σχετικά ποσοστό πυκνότητας. Για μεγάλο ποσοστό πυκνότητας ο χρόνος που απαιτείται για την εύρεση λύσης είναι μεγάλος, δηλαδή η μέθοδος δεν είναι αποδοτική. Το ποσοστό αυτό εξαρτάται άμεσα από τον αριθμό επιχειρημάτων αλλά και από το ποσοστό αβεβαιότητας.

3. Όπως και στα προηγούμενα συμπεράσματα έτσι και σε αυτό, η μέθοδος είναι βοηθητική σε μικρά ποσοστά αβεβαιότητας. Αντιθέτως, σε μεγάλα ποσοστά δεν είναι καθόλου αποδοτική. Σε αυτή την περίπτωση όμως, παρατηρούμε ότι το ποσοστό αβεβαιότητας έχει χειρότερους χρόνους εύρεσης λύσης σε σχέση με τους άλλους δύο παραμέτρους. Το ποσοστό αυτό εξαρτάται άμεσα από τον αριθμό επιχειρημάτων καθώς και από τον ποσοστό πυκνότητας.

Κεφάλαιο 6

Συμπεράσματα και Μελλοντικές Επεκτάσεις

6.1 Συμπεράσματα 50

6.2 Μελλοντικές Επεκτάσεις 51

6.1 Συμπεράσματα

Συμπερασματικά, στην παρούσα διπλωματική εργασία μελετήσαμε το πρόβλημα επιχειρηματολογίας με ελλιπή γνώση. Πιο συγκεκριμένα, με βάση ενός ελλιπούς προβλήματος επιχειρηματολογίας προσπαθούσαμε να βρούμε ένα ισχυρό σύνολο επιχειρημάτων. Αν δεν υπήρχε ένα τέτοιο σύνολο, προσπαθούσαμε να βρούμε ένα άλλο σύνολο επιχειρημάτων το οποίο μας ικανοποιούσε όσο το δυνατόν περισσότερο, δηλαδή ήτ

Για την μελέτη του προβλήματος αυτού υλοποιήθηκε ένα σύστημα όπου με βάση ενός συστήματος επιχειρηματολογίας, μετάφραζε το πρόβλημα αυτό σε ποσοτικοποιημένο δυαδικό τύπο. Αφού ολοκληρωνόταν η μετάφραση, δινόταν στον επιλυτή quantom ο οποίος έβρισκε το ισχυρό σύνολο επιχειρημάτων του συγκεκριμένου προβλήματος και το παρουσίαζε στην οθόνη.

Ωστόσο, ο χρόνος εύρεσης ενός ισχυρού συνόλου επιχειρημάτων από τον επιλυτή απαιτούσε κάποιο χρόνο ανάλογα με το πρόβλημα επιχειρηματολογίας. Επομένως, για να εξετάσουμε κατά πόσο η μέθοδος που χρησιμοποιήσαμε θα μπορούσε να ήταν βοηθητική για την εύρεση ισχυρών συνόλων επιχειρημάτων σε προβλήματα επιχειρηματολογίας**,** πραγματοποιήσαμε κάποια πειράματα εξετάζοντας διάφορους σημαντικούς παράγοντες (πυκνότητα, αβεβαιότητα και αριθμός επιχειρημάτων).

Με βάση των πειραμάτων αυτών φτάσαμε σε ένα γενικό συμπέρασμα. Πιο συγκεκριμένα, καταλήξαμε στο ότι η μέθοδος είναι αποδοτική μόνο για συγκεκριμένο

και σχετικά μικρό αριθμό επιχειρημάτων, ποσοστό πυκνότητας και ποσοστό αβεβαιότητας. Όμως, για μεγάλη ποσότητα το προηγούμενων, ο χρόνος είναι απίστευτα μεγάλος και με καμία περίπτωση αποδοτικός. Δηλαδή για σχετικά μικρά προβλήματα επιχειρηματολογίας η μέθοδος θα μπορούσε να χαρακτηριστεί πολύ αποδοτική αλλά για μεγάλα προβλήματα επιχειρηματολογίας η μέθοδος δεν είναι καθόλου αποδοτική. Επομένως, μπορούμε να πούμε ότι η μέθοδος δεν είναι αρκετά βοηθητική.

6.2 Μελλοντικές Επεκτάσεις

Υπάρχουν διάφορες επεκτάσεις που μπορούν να γίνουν στη παρούσα διπλωματική εργασία καθώς επίσης και πολλές βελτιώσεις.

Αρχικά, όσο αφορά τις βελτιώσεις, θα μπορούσαν να γίνουν μερικές βελτιστοποιήσεις του κώδικα μετάφρασης ενός προβλήματος επιχειρηματολογίας σε QBF. Πιο συγκεκριμένα, θα μπορούσε να βελτιωθεί η υλοποίηση χρησιμοποιώντας πιο αποδοτικών δομών που θα μπορούσαν να έκαναν καλύτερη διαχείριση μνήμης.
Επίσης, θα μπορούσε να βελτιωθεί ο τρόπος δημιουργίας τυχαίων (random) γράφων, το οποίο θα μας οδηγούσε σε καλύτερα αποτελέσματα.

Τώρα, όσο αφορά τις επεκτάσεις που θα μπορούσαν να γίνουν. Μια ενδιαφέρουσα επέκταση θα ήταν η υλοποίηση αλγόριθμων που θα δέχονταν διάφορα προβλήματα επιχειρηματολογίας (πχ με πλήρη (complete) γνώση, με ελλιπή (incomplete) γνώση και με καθόλου γνώση). Θα μπορούσε να γίνει σύγκριση των αλγορίθμων αυτών.

Τέλος, ακόμη μια επέκταση που θα μπορούσε να πραγματοποιηθεί είναι η εύρεση ενός συνόλου ισχυρών επιχειρημάτων που θα μπορούσαν να αντισταθούν στα επιχειρήματα του προτείνοντα, ούτως ώστε να μπορεί να υποστηρίξει δική του άποψη του χωρίς να αλλάξει εύκολα θεωρία.

Βιβλιογραφία

[1] Phan Minh Dung, "On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games", Artif. Intell. 77, 1995

[2] Pietro Baroni, Massimiliano Giacomin, "Chapter 2: Semantics of Abstract Argument Systems"

[3] Ρεφανίδης Γιάννης, Προβλήματα Ικανοποίησης Περιορισμών, Λογικός Προγραμματισμός με Περιορισμούς

[4] Javier Larrosa, Joao Marques-Silva, Satisfiability: Algorithms, Applications and Extensions, 2010

[5] H. Kleine Bυning, U. Bubeck, Theory of quantified boolean formulas, in: Handbook of Satisfiability, 2009

[6] Yannis Dimopoulos, Pavlos Moriatis, Different People, Different Arguments

[7] https://en.wikipedia.org/wiki/True_quantified_Boolean_formula

[8] S. Reimer, M. Sauer, P. Marin, B. Becker, QBF with soft variables, in: 14th International Workshop on Automated Verification of Critical Systems(AVOCS), 2014

Παράρτημα Α

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

int main(){

   FILE *fp;
   fp = fopen("test.txt", "w");
   if(fp == NULL){
      printf("Error opening file!\n");
      exit(-1);
   }
   printf ("-------------------------\n");
   printf ("Dimiourgia I: \n");
   printf ("-------------------------\n");

   int argumentsI;
   int piknotitaR;
   int temp;
   int arithmosAkmwnR;
   double pososto;

   printf("Dwse ton arithmo twn arguments gia to I (>=0): ");
   scanf("%d",&argumentsI);
   printf("\n");
   while (argumentsI < 0){
      printf("Lathos!! O arithmos twn arguments prepei na einai megaliteros apo to
0!!\nKsanadwse ton arithmo twn arguments (>=0): ");
      scanf("%d",&argumentsI);
   }
   printf ("Arithmos twn arguments gia to I: %d\n", argumentsI);

   printf("Dwse tin piknotita twn komvwn(metaksi 0 kai 100 simperilamvanomenwn):
");
   scanf("%d",&piknotitaR);
   printf("\n");
   while ((piknotitaR<0) || (piknotitaR>100)){
      printf("Lathos!! I piknotita prepei na einai metaksi tou 0 kai tou 100!!\nKsanadwse
piknotita(0-100): ");
      scanf("%d",&piknotitaR);
   }

   temp = argumentsI * (argumentsI -1);
   arithmosAkmwnR = (piknotitaR/100.0) * temp;
   printf ("Arithmos Akmwn:  %d\n ", arithmosAkmwnR );
```

```
printf("Dwse to pososto avevaiotitas(%%%): ");
scanf("%lf",&pososto);
int arithmosAkmwnU = (arithmosAkmwnR * pososto)/100.0;
arithmosAkmwnR = arithmosAkmwnR - arithmosAkmwnU;

printf ("Arithmos Akmwn tou R:  %d\n ", arithmosAkmwnR );
printf ("Arithmos Akmwn tou U:  %d\n ", arithmosAkmwnU );

int q;

int R[argumentsI][argumentsI];
int U[argumentsI][argumentsI];

srand (time(NULL));

int i,j, n;
   for (i = 0; i < argumentsI; i++) {
      for ( j =0; j < argumentsI; j++){
         R[i][j] = 0;
      }
   }
int countOfOneR = 0;
      int n2;
do{
   for (i = 0; i < argumentsI; i++) {
      for (j =0; j < argumentsI; j++){
               if (R[i][j] != 1){
                        n = rand() % 100;
                        printf("%d\n",n);
                        if (n > 80){
                                 n2 = 1;
                                 countOfOneR++;
                                 if ((i == j)||(countOfOneR > arithmosAkmwnR)){
                                          n2 = 0;
                                          countOfOneR--;
                                 }
                        }else{
                                 n2 = 0;
                        }
                        R[i][j] = n2;
               }
      }
   }
}while(countOfOneR != arithmosAkmwnR);

   for (i = 0; i < argumentsI; i++) {
      for ( j =0; j < argumentsI; j++){
         U[i][j] = 0;
```

```
        }
      }

   int countOfOneU = 0;
   do{
      for (i = 0; i < argumentsI; i++) {
         for (j = i; j < argumentsI; j++){
            n = rand() % 100;
            if (n > 80){
                                          n2 = 1;
               if ((i == j) || (R[i][j] == 1) || (R[j][i] == 1) || (countOfOneU >
arithmosAkmwnU)){
                  n2 = 0;
               }
            }else{
               n2 = 0;
                                    }
            if ((n2 == 1) && (U[i][j] != 1)){
               U[i][j] = n2;
               U[j][i] = n2;
               countOfOneU = countOfOneU + 2;
            }
         }
      }
   }while(countOfOneU < arithmosAkmwnU);

   printf ("\n");
   printf ("R:\n");

   for (i = 0; i < argumentsI; i++) {
      for (j =0; j < argumentsI; j++){
         printf ("%d ", R[i][j]);
      }
      printf ("\n");
   }

   printf ("\n");
   printf ("U:\n");

   for (i = 0; i < argumentsI; i++) {
      for (j =0; j < argumentsI; j++){
         printf ("%d ", U[i][j]);
      }
      printf ("\n");
   }

   printf("Dwse to argument gia to q (metaksi 1 kai %d simperilamvanomenwn): " ,
argumentsI );
   scanf("%d",&q);
```

```
    printf("\n");
    while ((q<1) || (q>argumentsI)){
        printf("Den iparxei to argument p edwses! To argument prepei na einai metaksi tou
1 kai tou %d!!\nKsanadwse arithmo tou q: ", argumentsI);
        scanf("%d",&q);
    }

    printf ("-------------------------\n");
    printf ("Dimiourgia S: \n");
    printf ("-------------------------\n");

    int argumentsS;
    int countOfOneS= 0;

    printf("Dwse ton arithmo twn arguments gia to S (>=0): ");
    scanf("%d",&argumentsS);
    printf("\n");
    while (argumentsS < 0){
        printf("Lathos!! O arithmos twn arguments prepei na einai megaliteros apo to
0!!\nKsanadwse ton arithmo twn arguments (>=0): ");
        scanf("%d",&argumentsS);
    }
    printf ("Arithmos twn arguments gia to S: %d\n", argumentsS);

    float piknotitaS;
    int arithmosAkmwnS;

    printf("Dwse tin piknotita tou S(metaksi 0 kai 1 simperilamvanomenwn): ");
    scanf("%f",&piknotitaS);
    printf("\n");
    while ((piknotitaS<0) || (piknotitaS>1)){
        printf("Lathos!! I piknotita prepei na einai metaksi tou 0 kai tou 1!!\nKsanadwse
piknotita tou S(0-1): ");
        scanf("%f",&piknotitaS);
    }

    temp = argumentsS * (argumentsS -1);
    arithmosAkmwnS = piknotitaS * temp;

    printf ("Arithmos Akmwn tou S: %d\n", arithmosAkmwnS);

    printf ("-------------------------\n");
    printf ("Dimiourgia L: \n");
    printf ("-------------------------\n");

    int L1[argumentsS][argumentsS];
    int L2[argumentsS][argumentsI];
```

```
do {
        for (i = 0; i < argumentsS; i++) {
                for (j =0; j < argumentsI; j++){
                        n = rand() %2;
                        if (n == 1){
                                if ((i == j) || (countOfOneS > arithmosAkmwnS)){
                                        n = 0;
                                }else{
                                        countOfOneS++;
                                }
                        }
                        L2[i][j] = n;
                }
        }
   }while (countOfOneS < arithmosAkmwnS);

   printf ("\n");
   printf ("L1:(Komvoi apo to S se komvous tou S)\n");

   for (i = 0; i < argumentsS; i++) {
      for (j =0; j < argumentsS; j++){
         printf ("%d ", L1[i][j]);
      }
      printf ("\n");
   }

   printf ("L2:(Komvoi apo to S se komvous tou I)\n");

   for (i = 0; i < argumentsS; i++) {
      for (j =0; j < argumentsI; j++){
         printf ("%d ", L2[i][j]);
      }
      printf ("\n");
   }

   printf ("-------------------------\n");

   printf (" KATASKEUI QFA FORMULA \n");

   printf ("\n");
   printf (" Voithitiki \n");

   //arguments tou S (XG)
   if (argumentsS > 0)
      printf("e ");
   for (i=1; i<=argumentsS; i++)
      printf("Xa%d ", i);
   printf("0\n");
```

```
    // oles oi akmes pou einai sto U
    for (i=0; i<argumentsI; i++)
       for (j=i; j<argumentsI; j++)
          if (U[i][j] == 1)
             printf("s [2,1;3,0] b%d,%d \n" , i+1, j+1);

    for (i=0; i<argumentsI; i++)
       for (j=i; j<argumentsI; j++)
          if (U[i][j] == 1)
             printf(" d%d,%d" , i+1, j+1);
    printf(" 0\n");

    if ((arithmosAkmwnU > 0) || (argumentsI > 0))
       printf("e");

    //arguments tou I(ZG)
    for (i=1; i<=argumentsI; i++)
       printf(" x%d", i);
    //akmes tou U
    for (i=0; i<argumentsI; i++)
       for (j=0; j<argumentsI; j++)
          if (U[i][j] == 1){
             printf(" t%d,%d" , i+1, j+1);
          }

    printf(" 0\n");

/* NEW clauses 1 - START*/

    for (i=0; i<argumentsI; i++)
       for (j=0; j<argumentsI; j++)
          if (R[i][j] == 1){
             printf(" -X%d -X%d," , i+1, j+1);
          }
    printf("\n");

    for (i=0; i<argumentsI; i++)
       for (j=i; j<argumentsI; j++)
          if (U[i][j] == 1){
             printf(" -X%d -X%d," , i+1, j+1);
          }
    printf("\n");

    for (i=0; i<argumentsS; i++)
       for (j=0; j<argumentsI; j++)
          if (L2[i][j] == 1){
             printf(" -Xa%d -X%d," , i+1, j+1);
          }
```

```
    printf("\n");
    printf("\n");

/* NEW clauses 1 - END*/

/* NEW clauses 2 - START*/
    int k, l;
    printf("start \n");
    int argument = 0;

    int r = 0;
    int countClause2 = 0;
    for (i=0; i<argumentsI; i++){
        r = 0;
        for (j=0; j<argumentsI; j++){
            if (R[j][i]){
                if (r==0){
                    printf(" X%d", i+1);
                    r = 1;
                    countClause2++;
                }
                printf(" X%d", j+1);
            }

            if (j == 0){
                for (l=0; l<argumentsS; l++){
                    if (L2[l][i] == 1){
                        if (r == 0){
                            printf(" X%d", i+1);
                            r = 1;
                            countClause2++;
                        }
                        printf(" Xa%d", l+1);
                    }
                }
            }
                                if (U[j][i]){
                                        if (r == 0){
                                                printf(" X%d", i+1);
                                                r = 1;
                                                countClause2++;
                                        }
                                        printf(" t%d,%d", j+1, i+1);
                                }
                        }
                        printf(", ");
            }

            printf("\n end \n");
```

```
/* NEW clauses 2 - END*/

/* NEW clauses 3 - START*/
   printf("\n");

   for (i=0; i<argumentsI; i++){
      for (j=i; j<argumentsI; j++)
         if (U[i][j] == 1){
            printf("-t%d,%d" , i+1, j+1);
            printf(" b%d,%d", i+1, j+1);
            printf(" d%d,%d", i+1, j+1);

            printf(", ");

            printf("-t%d,%d" , i+1, j+1);
            printf(" X%d" , i+1);
            printf(", ");

            printf("-t%d,%d" , j+1, i+1);
            printf(" b%d,%d", i+1, j+1);

            printf(" -d%d,%d", i+1, j+1);

            printf(", ");

            printf("-t%d,%d" , j+1, i+1);
            printf(" X%d" , j+1);
            printf(", ");

         }
   }

/* NEW clauses 3 - END*/

/* NEW clauses 4 - START*/

   printf(" x%d", q);
/* NEW clauses 4 - END*/

//------------------------------------------------------------------------

//PRINT KANONIKA
         printf ("\n");
   printf ("\n");

   printf (" Kanoniki \n");
   int countU = 0;
   int countR = 0;
```

```
int countL = 0;
int countL1 = 0;
int countL2 = 0;

for (i=0;i<argumentsI;i++)
   for (j=0;j<argumentsI;j++){
      if (U[i][j] == 1)
         countU++;
      if (R[i][j] == 1)
         countR++;
   }
for (i=0;i<argumentsS;i++)
   for (j=0;j<argumentsS;j++)
      if (L1[i][j] == 1)
         countL1++;
 for (i=0;i<argumentsS;i++)
   for (j=0;j<argumentsI;j++)
      if (L2[i][j] == 1)
         countL2++;
countL = countL1 + countL2;

int num1 = argumentsS + countU + argumentsI + countU;
int num2 = countR + (countU/2) + countOfOneS;
int num3 = countClause2;
int num4 = 2 * countU;
int num = num2 + num3 + num4 + 1;
fprintf(fp, "p cnf %d %d\n", num1, num);
printf("p cnf %d %d\n", num1, num);

//arguments tou S (XG)
if (argumentsS > 0){
   fprintf(fp, "e");
   printf("e ");
}
for (i=1; i<=argumentsS; i++){
   printf("%d ", i);
   fprintf(fp, " %d" , i);

}
fprintf(fp, " 0\n");
printf("0\n");

int counter = argumentsS + 1;

int tempcB[argumentsI][argumentsI];
int tempcD[argumentsI][argumentsI];

for (i=0; i<argumentsI; i++)
   for (j=i; j<argumentsI; j++)
```

```
            if (U[i][j] == 1){
               fprintf(fp, "s [2,1;3,0] %d 0\n" , counter);
               printf("s [2,1;3,0] %d 0\n" , counter);
               tempcB[i][j] = counter;
               counter++;
            }

      for (i=0; i<argumentsI; i++)
         for (j=i; j<argumentsI; j++)
            if (U[i][j] == 1){
               fprintf(fp, "s [2,1;3,0] %d 0\n" , counter);
               printf("s [2,1;3,0] %d 0\n" , counter);
               tempcD[i][j] = counter;
               counter++;
            }

      if ((arithmosAkmwnU > 0) || (argumentsI > 0)){
         printf("e");
         fprintf(fp, "e ");
      }
      //arguments tou I(ZG)
      int tempcA = counter;
      for (i=1; i<=argumentsI; i++){
         printf(" %d", counter);
         fprintf(fp, "%d " , counter);
         counter++;
      }
      //akmes tou U
      int tempcT[argumentsI][argumentsI];
      for (i=0; i<argumentsI; i++){
         for (j=0; j<argumentsI; j++){
            tempcT[i][j] = 0;
         }
      }

      for (i=0; i<argumentsI; i++)
         for (j=0; j<argumentsI; j++)
            if (U[i][j] == 1){
               printf(" %d" , counter);
               fprintf(fp, "%d " , counter);
               tempcT[i][j] = counter;
               counter++;
            }

      printf(" 0\n");
      fprintf(fp, "0\n");

   /* NEW clauses 1 - START */
      for (i=0; i<argumentsI; i++)
```

```
        for (j=0; j<argumentsI; j++)
            if (R[i][j] == 1){
                printf("-%d -%d " , tempcA + i, tempcA + j);
                printf("0\n");
                fprintf(fp, "-%d -%d " , tempcA + i, tempcA + j);
                fprintf(fp, "0\n");
            }
    for (i=0; i<argumentsI; i++)
        for (j=i; j<argumentsI; j++)
            if (U[i][j] == 1){
                printf("-%d -%d " , tempcA + i, tempcA + j);
                printf("0\n");
                fprintf(fp, "-%d -%d " , tempcA + i, tempcA + j);
                fprintf(fp, "0\n");
            }

     for (i=0; i<argumentsS; i++)
        for (j=0; j<argumentsI; j++)
            if (L2[i][j] == 1){
                printf("-%d -%d " , i+1, tempcA + j);
                printf("0\n");
                fprintf(fp, "-%d -%d " , i+1, tempcA + j);
                fprintf(fp, "0\n");
            }
/* NEW clauses 1 - END*/

/* NEW clauses 2 - START*/
    int r1 = 0;
    for (i=0; i<argumentsI; i++){
        r1 = 0;
        for (j=0; j<argumentsI; j++){
            if (R[j][i]){
                if (r1 == 0){
                    printf("%d ", tempcA + i);
                    fprintf(fp, "%d ", tempcA + i);
                    r1 = 1;
                }
                printf("%d ", tempcA + j);
                fprintf(fp, "%d ", tempcA + j);
            }

            if (j == 0){
                for (l = 0; l<argumentsS; l++){
                    if (L2[l][i] == 1){
                        if (r1 == 0){
                            printf("%d ", tempcA + i);
                            fprintf(fp, "%d ", tempcA + i);
                            r1 = 1;
                        }
```

```
                printf("%d ", l+1);
                fprintf(fp, "%d ", l+1);
              }
            }
          }
          if (U[j][i]){
            if (r1 == 0){
              printf("%d ", tempcA + i);
              fprintf(fp, "%d ", tempcA + i);
              r1 = 1;
            }
            printf("%d ", tempcT[j][i]);
            fprintf(fp, "%d ", tempcT[j][i]);
          }
          }
        if (r1 == 1){
          printf(" 0\n");
          fprintf(fp, "0\n");
        }

    }

/* NEW clauses 2 - END*/

/* NEW clauses 3 - START*/

    for (i=0; i<argumentsI; i++)
      for (j=i; j<argumentsI; j++)
        if (U[i][j] == 1){
          printf("-%d" , tempcT[i][j]);
          fprintf(fp, "-%d" , tempcT[i][j]);
          printf(" %d", tempcB[i][j]);
          fprintf(fp, " %d", tempcB[i][j]);

          printf(" %d", tempcD[i][j]);
          fprintf(fp, " %d", tempcD[i][j]);

          printf(" 0\n");
          fprintf(fp, " 0\n");

          printf("-%d" , tempcT[i][j]);
          fprintf(fp, "-%d" , tempcT[i][j]);
          printf(" %d" , tempcA + i);
          fprintf(fp, " %d" , tempcA + i);
          printf(" 0\n");
          fprintf(fp, " 0\n");

          printf("-%d" , tempcT[j][i]);
          fprintf(fp, "-%d" , tempcT[j][i]);
```

```
            printf(" %d", tempcB[i][j]);
            fprintf(fp, " %d", tempcB[i][j]);

            printf(" -%d", tempcD[i][j]);
            fprintf(fp, " -%d", tempcD[i][j]);

            printf(" 0\n");
            fprintf(fp, " 0\n");

            printf("-%d" , tempcT[j][i]);
            fprintf(fp, "-%d" , tempcT[j][i]);
            printf(" %d" , tempcA + j);
            fprintf(fp, " %d" , tempcA + j);
            printf(" 0\n");
            fprintf(fp, " 0\n");
          }
 /* NEW clauses 3 - END*/

/* NEW clauses 4 - START*/

    printf(" %d 0", (tempcA - 1) + q);
    fprintf(fp, "%d" , (tempcA - 1) + q);
    fprintf(fp, " 0");

/* NEW clauses 4 - END*/
    fprintf(fp, " \n");
    fclose(fp);

    printf(" \n");

    printf("DIMIOURGITHIKE TO ARXEIO: test.txt");
    printf(" \n");

}
```