Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων
και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης
και Υπολογιστων

Γλώσσες Κβαντικού Προγραμματισμού
Θεωρία και Υλοποίηση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΙΧΑΗΛ Σ. ΛΑΜΠΗΣ

Επιβλέπων : Νικόλαος Παπασπύρου
Λέκτορας Ε.Μ.Π.

Αθήνα, Ιούλιος 2005

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων
και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης
και Υπολογιστων

Γλώσσες Κβαντικού Προγραμματισμού
Θεωρία και Υλοποίηση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΙΧΑΗΛ Σ. ΛΑΜΠΗΣ

Επιβλέπων : Νικόλαος Παπασπύρου
Λέκτορας Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 19 Ιουλίου 2005.

.....	.....	.....
Νικόλαος Παπασπύρου Λέκτορας Ε.Μ.Π.	Ευστάθιος Ζάχος Καθηγητής Ε.Μ.Π.	Ανδρέας-Γεώργιος Σταφυλοπάτης Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2005

Μιχαήλ Σ. Λαμπής

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Μιχαήλ Σ. Λαμπής, 2005.
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Σκοπός αυτής της διπλωματικής εργασίας είναι η μελέτη των γλωσσών προγραμματισμού που βασίζονται στις αρχές του κβαντικού μοντέλου υπολογισμών. Μελετήσαμε δύο γλώσσες που έχουν προταθεί στην πρόσφατη βιβλιογραφία: την QPL (Selinger, 2004) και την QML (Grattage and Altenkirch, 2005). Παραλλαγή της δεύτερης αποτελεί η NTUA-QML, μία νέα γλώσσα κβαντικού προγραμματισμού την οποία σχεδιάσαμε στο πλαίσιο αυτής της εργασίας. Κατασκευάσαμε μια υλοποίηση της QPL με εξομοίωση σε συμβατικούς υπολογιστές, γραμμένη στη γλώσσα Haskell, καθώς και μια υλοποίηση της NTUA-QML με μετασχηματισμό στην QPL. Για τον έλεγχο και την επίδειξη των υλοποιήσεων, μεταφράσαμε ορισμένα παραδείγματα γνωστών κβαντικών αλγορίθμων σε NTUA-QML.

Η δυνατότητα αξιοποίησης των χαρακτηριστικών της θεωρίας της κβαντομηχανικής για τη διενέργεια υπολογισμών κίνησε για πρώτη φορά το ενδιαφέρον των ερευνητών περίπου πριν από δύο δεκαετίες. Η ανακάλυψη όμως του αλγορίθμου παραγοντοποίησης ακεραίων του Shor, ενός αλγορίθμου πολυωνυμικής πολυπλοκότητας για ένα πρόβλημα το οποίο πιστεύεται πως δεν ανήκει στο P, έφερε τους κβαντικούς υπολογισμούς στο επίκεντρο εξαιτίας και των τεράστιων επιπτώσεων που θα είχε η ύπαρξη μιας πρακτικής υλοποίησης ενός τέτοιου αλγορίθμου στον τομέα της κρυπτανάλυσης. Από τότε η έρευνα έχει επικεντρωθεί στην προσπάθεια κατασκευής πρακτικών κβαντικών υπολογιστών.

Οι ιδιαιτερότητες του κβαντικού μοντέλου υπολογισμών, και η αντίθεσή του με την διαίσθηση του κλασικού προγραμματιστή είναι ένας από τους λόγους που δεν έχουν αναπτυχθεί ακόμα πολλοί ενδιαφέροντες κβαντικοί αλγόριθμοι. Η ύπαρξη γλωσσών υψηλού επιπέδου που θα επέτρεπαν στο χρήστη τους να σκέφτεται σε ένα πιο αφηρημένο επίπεδο χωρίς να απασχολείται με λεπτομέρειες υλοποίησης, όπως τα κβαντικά κυκλώματα ίσως να βοηθούσε αυτήν την κατάσταση. Μία καλή γλώσσα κβαντικού προγραμματισμού θα πρέπει να ενσωματώνει τους περιορισμούς και τις δυνάμεις των κβαντικών υπολογισμών διευκολύνοντας το χρήστη να εκφράσει προχωρημένους αλγορίθμους με φυσικό τρόπο.

Λέξεις κλειδιά

Κβαντικός προγραμματισμός, γλώσσες κβαντικού προγραμματισμού, αλγόριθμος του Deutsch, αλγόριθμος του Grover, αλγόριθμος του Shor.

Abstract

The purpose of this diploma dissertation is the study of programming languages which rely on the principles of the quantum computation model. We studied two such languages that have been proposed in recent literature: QPL (Selinger, 2004) and QML (Grattage and Altenkirch, 2005). A variation of the latter is NTUA-QML, a new quantum programming language designed as part of this dissertation. We built an implementation of QPL by simulation on conventional computers, written in Haskell, and an implementation of NTUA-QML by translation to QPL. For the purpose of checking and demonstrating our implementations, we translated several examples of well known quantum algorithms in NTUA- QML.

The possibility of exploiting the characteristics of quantum mechanics theory to perform calculation first attracted research interest two decades ago. But it was the discovery of Shor's integer factorization algorithm, an algorithm of polynomial complexity which solves a problem widely believed to be outside of P, which brought quantum computations in the spotlight, in part because of the huge implications the existence of a practical implementation of such an algorithm would have on the field of cryptanalysis. Research has focused on the effort to construct a practical quantum computer ever since.

The peculiarity of the quantum computation model and the fact that it often contradicts a classical programmer's intuition is one of the reasons that few interesting quantum algorithms have been developed so far. The existence of high level languages which allow the user to think at a more abstract level without worrying about implementation details such as quantum circuits, might help rectify this situation. A decent programming language would have to embody the limitations and strengths of quantum computations while making it easy for programmers to express advanced algorithms naturally.

Key words

Quantum programming, quantum programming language, Deutsch's algorithm, Grover's algorithm, Shor's algorithm.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής μου κ. Νίκο Παπασπύρου για την καθοδήγησή του και γιατί μου έδωσε την ευκαιρία να ασχοληθώ με ένα πολύ ενδιαφέρον θέμα. Επίσης θα ήθελα να ευχαριστήσω τους γονείς μου που με έστρεψαν από μικρό προς τις θετικές επιστήμες και είμαι σίγουρος πως θα κάνουν μια φιλότιμη προσπάθεια να διαβάσουν την εργασία αυτή. Τέλος θα ήθελα να ευχαριστήσω όλους τους συναδέλφους μου που περάσαμε μαζί πέντε όμορφα φοιτητικά χρόνια.

Μιχαήλ Σ. Λαμπής,

Αθήνα, 19η Ιουλίου 2005.

Η εργασία αυτή είναι επίσης διαθέσιμη ως Τεχνική Αναφορά CSD-SW-TR-3-05, Εθνικό Μετσόβιο Πολυτεχνείο, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών, Εργαστήριο Τεχνολογίας Λογισμικού, Ιούλιος 2005.

URL: `http://www.softlab.ntua.gr/techrep/`
FTP: `ftp://ftp.softlab.ntua.gr/pub/techrep/`

Περιεχόμενα

Περίληψη 5

Abstract 7

Ευχαριστίες 9

Περιεχόμενα 11

Κατάλογος Πινάκων 13

Κατάλογος Σχημάτων 15

1. Εισαγωγή 17
1.1 Ιστορία των κβαντικών υπολογισμών 17
1.2 Γλώσσες κβαντικού υπολογισμού 18
1.3 Στοιχεία κβαντικής θεωρίας πολυπλοκότητας 19
1.4 Οργάνωση της εργασίας 19

2. Κβαντικοί Υπολογισμοί 21
2.1 Εισαγωγή στους κβαντικούς υπολογισμούς 21
2.2 Κβαντικά bit 22
2.2.1 Γενικά 22
2.2.2 Κβαντικοί καταχωρητές 22
2.2.3 Entanglement 23
2.3 Ορθομοναδιαίοι μετασχηματισμοί 23
2.4 Μετρήσεις 25
2.5 Αγνές και Μικτές καταστάσεις 25
2.6 Πίνακες πυκνότητας 26
2.7 Κβαντικές λειτουργίες και πίνακες πυκνότητας 27
2.8 Κβαντικά κυκλώματα 28

3. Γλώσσες κβαντικού προγραμματισμού 31
3.1 Εισαγωγή 31
3.2 Η γλώσσα QPL 32
3.2.1 Σύνταξη 32
3.2.2 Λειτουργία 32
3.2.3 Παραδείγματα 34
3.3 Η γλώσσα QML 36
3.3.1 Συντακτικό 36
3.3.2 Σύστημα τύπων 36
3.3.3 Σημασιολογία 39
3.3.4 Παραδείγματα 42
3.4 Η γλώσσα NTUA-QML 43
3.4.1 Συντακτικό 43

3.4.2 Σύστημα τύπων 44
3.4.3 Σημασιολογία 45
3.4.4 Παραδείγματα 47

4. Υλοποίηση της NTUA-QML 49
4.1 Εισαγωγή 49
4.2 Υλοποίηση της QPL 49
4.3 Υλοποίηση της NTUA-QML 51

5. Παραδείγματα κβαντικών αλγορίθμων 55
5.1 Ο αλγόριθμος του Deutsch 55
5.1.1 Περιγραφή 55
5.1.2 Υλοποίηση σε NTUA-QML 56
5.2 Ο αλγόριθμος του Grover 57
5.2.1 Περιγραφή 57
5.2.2 Υλοποίηση σε NTUA-QML 59
5.3 Ο αλγόριθμος παραγοντοποίησης του Shor 60
5.3.1 Περιγραφή 60
5.3.2 Ο κβαντικός μετασχηματισμός Fourier 61
5.3.3 Υλοποίηση κβαντικής ύψωσης σε εκθέτη modulo N 63

6. Συμπεράσματα 67
6.1 Συνεισφορά 67
6.2 Μελλοντική έρευνα 67

Βιβλιογραφία 69

Κατάλογος Πινάκων

3.1 Σημασιολογία QPL κβαντικό μέρος. 33

4.1 Ορισμός της compute_state. 50

5.1 Εύρεση περιόδου συνάρτησης 3 bit. 63

Κατάλογος Σχημάτων

2.1 Ένα παράδειγμα κβαντικού κυκλώματος. 28

3.1 Κύκλωμα επιλογής υπό συνθήκη. 40
3.2 Κύκλωμα ϕ_δ 40
3.3 Κύκλωμα ϕ_C για context που μοιράζονται μεταβλητές. 40
3.4 Κύκλωμα ϕ_C για context που δεν μοιράζονται μεταβλητή. 41
3.5 Κύκλωμα γινομένου. 41
3.6 Κύκλωμα case. 41
3.7 Κύκλωμα $case^o$ 42

4.1 Κύκλωμα υπό συνθήκη αλλαγής θέσης δύο bit. 54

5.1 Κύκλωμα που υλοποιεί τον αλγόριθμο του Deutsch. 55

Κεφάλαιο 1

Εισαγωγή

Σκοπός αυτής της εργασίας είναι η μελέτη ενός συναρπαστικού νέου πεδίου της επιστήμης των υπολογισμών, του πεδίου των κβαντικών υπολογισμών. Η μελέτη αυτή γίνεται από τη σκοπιά των γλωσσών προγραμματισμού που έχουν προταθεί και ενσωματώνουν τα ιδιαίτερα χαρακτηριστικά του κβαντικού μοντέλου υπολογισμών που το κάνουν ισχυρό αλλά ταυτόχρονα και δύσχρηστο. Εκτός από τη μελέτη υπαρχουσών γλωσσών προτείνουμε και μία νέα γλώσσα κβαντικού προγραμματισμού η οποία αποτελεί παραλλαγή μίας από τις γλώσσες που μελετήθηκαν. Η νέα αυτή γλώσσα ονομάζεται NTUA-QML. Δημιουργήσαμε επίσης ένα σύστημα εξομοίωσης της εκτέλεσης προγραμμάτων της NTUA-QML σε έναν συμβατικό υπολογιστή χρησιμοποιώντας τη γλώσσα Haskell. Με τη βοήθειά του υλοποιήσαμε και κάποια κλασικά παραδείγματα κβαντικών αλγορίθμων τα οποία μεταφράσαμε στη γλώσσα NTUA-QML.

1.1 Ιστορία των κβαντικών υπολογισμών

Η ιδέα του κβαντικού υπολογιστή είναι μία σχετικά νέα ιδέα. Επισήμως προτάθηκε για πρώτη φορά στις αρχές της δεκαετίας του 80 από τον Richard Feynman. Το πρόβλημα που αντιμετώπιζε ο Feynman ήταν η ότι εξομοίωση κβαντομηχανικών συστημάτων με συμβατικούς υπολογιστές δεν γινόταν αρκετά αποδοτικά, και μάλιστα η πολυπλοκότητά της αυξανόταν εκθετικά με το μέγεθος του συστήματος που έπρεπε να εξομοιωθεί. Το γεγονός ότι δεν υπήρχε (και μέχρι σήμερα εξακολουθεί να μην υπάρχει) κάποια λύση στο πρόβλημα αυτό με κλασικά μέσα ώθησε τον Feynman να προτείνει τον κβαντικό υπολογιστή, δηλαδή έναν υπολογιστή ο οποίος θα βασιζόταν στη λειτουργία του ακριβώς σε αυτές τις φυσικές διαδικασίες οι οποίες ήταν δύσκολο να εξομοιωθούν. Με άλλα λόγια αυτό που υποστήριξε ο Feynman ήταν πως η ίδια η διεξαγωγή ενός κβαντικού πειράματος, αφού απαιτούσε τόσο επίπονους υπολογισμούς για να προβλεφθεί το αποτέλεσμά της, θα μπορούσει ίσως να θεωρηθεί από μόνη της ένα είδος υπολογισμού. Αυτή η σκέψη ήταν που έθεσε τα θεμέλια των κβαντικών υπολογισμών.

Από τη στιγμή που διατυπώθηκε για πρώτη φορά η ιδέα των κβαντικών υπολογισμών προσέλκυσε τεράστιο ερευνητικό ενδιαφέρον. Ήδη το 1985 ο David Deutsch ήταν αυτός που έθεσε τις θεωρητικές βάσεις για την ανάπτυξη των κβαντικών υπολογισμών μιλώντας για έναν παγκόσμιο κβαντικό υπολογιστή (universal quantum computer) αντίστοιχο με την μηχανή Turing στους κλασικούς υπολογισμούς. Ο παγκόσμιος κβαντικός υπολογιστής του Deutsch ήταν ικανός να εξομοιώσει οποιονδήποτε άλλον κβαντικό υπολογιστή με το πολύ πολυωνυμική επιβράδυνση.

Η μεγάλη έκρηξη όμως για το πεδίο των κβαντικών υπολογισμών ήρθε το 1994 όταν ο Peter Shor δημοσίευσε έναν κβαντικό αλγόριθμο ο οποίος μπορούσε σε πολυωνυμικό χρόνο να λύσει το πρόβλημα της παραγοντοποίησης ενός ακεραίου. Ως σήμερα δεν είναι γνωστός κάποιος αντίστοιχος κλασικός αλγόριθμος, χωρίς όμως να έχει αποδειχθεί ότι δεν μπορεί ένας τέτοιος αλγόριθμος να υπάρξει. Πάντως αποτελεί κοινή πεποίθηση πως το πρόβλημα της παραγοντοποίησης μάλλον δεν μπορεί να λυθεί σε πολυωνυμικό χρόνο με κλασικούς υπολογιστές, γι' αυτό άλλωστε και πολλά σύγχρονα κρυπτοσυστήματα δημοσίου κλειδιού βασίζονται σε αυτό. Είναι κατανοητό λοιπόν το γιατί η ανακάλυψη αυτή του Shor προκάλεσε μεγάλο εν-

διαφέρον όχι μόνο στην επιστημονική κοινότητα αλλά και σε κυβερνήσεις, μυστικές υπηρεσίες και γενικότερα όποιον ασχολείται με την κρυπτογραφία.

Δύο χρόνια μετά τη δημοσίευση του αλγορίθμου του Shor εμφανίστηκε και ο αλγόριθμος του Lov Grover για αναζήτηση σε αταξινόμητες βάσεις δεδομένων. Αν και ο αλγόριθμος του Grover δεν λύνει ένα πρόβλημα τόσο ευαίσθητο όσο αυτός του Shor, ούτε προσφέρει μία τόσο μεγάλη επιτάχυνση έχει το εξής μεγάλο πλεονέκτημα: είναι αποδεδειγμένα πιο αποδοτικός από τον καλύτερο κλασικό αλγόριθμο που μπορεί να υπάρξει για τη λύση του ίδιου προβλήματος. Η ύπαρξη λοιπόν του αλγορίθμου του Grover είναι και μία θεωρητική επιβεβαίωση για την υπεροχή της απόδοσης των κβαντικών αλγορίθμων απέναντι στους κλασικούς.

Δυστυχώς η έρευνα πάνω στο ζήτημα της υλοποίησης ενός κβαντικού υπολογιστή βρίσκεται ακόμα σε πρώιμο στάδιο. Διάφορες αρχιτεκτονικές έχουν προταθεί, κάθε μία εκ των οποίων χρησιμοποιεί και μία άλλη κβαντική ιδιότητα των σωματιδίων που απαρτίζουν την ύλη για να διεξάγει υπολογισμούς. Καμία όμως δεν έχει καταφέρει ως τώρα να μας δώσει έναν πρακτικό τρόπο για τη δημιουργία πολύπλοκων κβαντικών υπολογισμών. Παρ' όλα αυτά τα πρώτα πειράματα με κβαντικούς υπολογισμούς έχουν γίνει και η θεωρία φαίνεται σωστή. Το 2001 ένας κβαντικός υπολογιστής των 7 qbit κατάφερε να αναλύσει τον αριθμό 15 σε 3×5. Ίσως βρισκόμαστε ακόμα μακριά από την πραγματοποίηση ενός πρακτικού κβαντικού υπολογιστή που να μπορεί να χειριστεί ποσότητα πληροφορίας αντίστοιχη με τους σημερινούς κλασικούς, είμαστε όμως σε θέση να πούμε πως οι κβαντικοί υπολογισμοί είναι κάτι παραπάνω από ένα πνευματικό παιχνίδι: είναι μια θεωρία πάνω στην οποία μπορούν ίσως να στηριχτούν οι υπολογιστές του μέλλοντος.

1.2 Γλώσσες κβαντικού υπολογισμού

Ο κβαντικός προγραμματισμός συνηθίζεται να μελετάται σε ένα χαμηλό επίπεδο πολύ κοντά στο hardware. Αυτό είναι λογική συνέπεια της μη ύπαρξης ως τώρα κβαντικών υπολογιστών, κάτι που δεν δίνει στον προγραμματιστή την πολυτέλεια να ασχοληθεί μόνο με τη λογική του αλγορίθμου αγνοώντας τελείως την υλοποίηση. Έχει ως αποτέλεσμα οι περισσότεροι αλγόριθμοι να αναλύονται σε χρήση κβαντικών πυλών. Είναι σαφές ότι κάτι τέτοιο περιορίζει την εκφραστική δύναμη, όπως άλλωστε θα συνέβαινε και αν ήμασταν υποχρεωμένοι να εκφράζουμε τους κλασικούς αλγορίθμους με λογικά κυκλώματα. Για το λόγο αυτό έχουν γίνει προσπάθειες για την εισαγωγή και γλωσσών κβαντικού προγραμματισμού υψηλού επιπέδου.

Μία γλώσσα κβαντικού προγραμματισμού θα πρέπει να λαμβάνει υπ' όψιν της τα ιδιαίτερα χαρακτηριστικά των κβαντικών υπολογισμών που τους δίνουν το συγκριτικό πλεονέκτημα απέναντι στους κλασικούς, αλλά και τους επιπλέον περιορισμούς που επιβάλλονται όταν κανείς χειρίζεται πληροφορίες σε κβαντικό επίπεδο. Ο πιο σημαντικός ίσως από αυτούς τους περιορισμούς είναι η κβαντική κατάρρευση, η οποία υπαγορεύει πως μία ανάγνωση της πληροφορίας που περιέχει ένα κβαντικό σύστημα απαλείφει την ασάφεια φέρνοντας το σύστημα σε μία κλασική κατάσταση. Αυτό είναι καθοριστικό διότι οι κβαντικοί υπολογισμοί βασίζονται σε πολύ μεγάλο βαθμό στη δυνατότητα της υπέρθεσης, δηλαδή τη δυνατότητα να χρησιμοποιούν ένα σύστημα το οποίο βρίσκεται ταυτόχρονα σε πολλές καταστάσεις.

Οι περισσότερες γλώσσες κβαντικού προγραμματισμού που έχουν προταθεί βασίζονται στην αρχή 'quantum data, classical control'. Με άλλα λόγια βασίζονται στην ιδέα ότι η εκτέλεση ενός κβαντικού προγράμματος ακολουθεί μία συγκεκριμένη ροή, όπως και η εκτέλεση ενός προγράμματος σε έναν κλασικό υπολογιστή. Η προσθήκη που κάνουν είναι ότι δίνουν τη δυνατότητα στον χρήστη να χειριστεί και κβαντικά δεδομένα, εκτός απο κλασικά, και μέσα από το χειρισμό τους να πετύχει τη διενέργεια κβαντικών υπολογισμών. Ως παράδειγμα τέτοιων γλωσσών αναφέρουμε ενδεικτικά την QPL του Peter Selinger, η οποία παρουσιάζεται στο κεφάλαιο 3, καθώς και την QCL του Bernard Oemer ([Oeme03]) η οποία συνοδεύεται και από μία πολύ καλή υλοποίηση.

Στον αντίποδα βρίσκονται οι γλώσσες της λογικής 'quantum data and control'. Σε αυτές

δίνεται η δυνατότητα για κατασκευή προγραμμάτων τα οποία θα βρίσκονται ταυτόχρονα σε πολλές καταστάσεις, όπως και τα κβαντικά δεδομένα τα οποία χειρίζονται. Τέτοια γλώσσα είναι η QML του Thorsten Altenkirch η οποία επίσης παρουσιάζεται στο κεφάλαιο 3, αλλά και η παραλλαγή της NTUA-QML η οποία δημιουργήθηκε για τους σκοπούς αυτής της εργασίας.

1.3 Στοιχεία κβαντικής θεωρίας πολυπλοκότητας

Προτού προχωρήσουμε παρακάτω με τις γλώσσες κβαντικού προγραμματισμού ας πούμε και δυο λόγια για τις θεωρητικές δυνατότητες που πιστεύεται πως έχουν οι κβαντικοί υπολογιστές.

Η κλάση των κβαντικών αλγορίθμων οι οποίοι μπορούν να εκτελεστούν σε πολυωνυμικό χρόνο ονομάζεται BQP από τα αρχικά Bounded Error, Quantum, Polynomial. Όπως φαίνεται και από το όνομα μιλάμε για μία κλάση αλγορίθμων που αποτελεί το κβαντικό ανάλογο της κλάσης BPP.

Η σχέση της κλάσης προβλημάτων BQP με τις NP,NP-complete και P δεν έχει μέχρι σήμερα αποδειχθεί. Πιστεύεται όμως ότι το P είναι υποσύνολο του BQP, καθώς και ότι το NP-complete είναι ξένο με το BQP. Η παραγοντοποίηση ακεραίων είναι ένα παράδειγμα προβλήματος που ανήκει στο NP και στο BQP αλλά πιστεύεται πως δεν ανήκει στο BPP. Μέχρι στιγμής δεν έχει βρεθεί πολυωνυμικός κβαντικός αλγόριθμος ο οποίος να λύνει κάποιο NP-complete πρόβλημα.

Παρά το ότι οι κβαντικοί υπολογιστές είναι κάποιες φορές πιο αποδοτικοί από τους κλασικούς αυτό δεν σημαίνει πως μπορούν να λύσουν προβλήματα που οι κλασικοί δεν μπορούν να λύσουν, όπως για παράδειγμα το halting problem. Κάθε υπολογισμός που μπορεί να γίνει κβαντικά μπορεί να γίνει και κλασικά δοθέντος αρκετού χρόνου και μνήμης. Με άλλα λόγια η ύπαρξη των κβαντικών υπολογιστών δεν αναιρεί την Church-Turing thesis.

Συνήθως όταν θέλουμε να μελετήσουμε την πολυπλοκότητα ενός κβαντικού αλγορίθμου χρησιμοποιούμε το μοντέλο της κβαντικής μηχανής Turing, η οποία χρησιμοποιεί μία κβαντική συνάρτηση μετάβασης, δηλαδή μπορεί να βρίσκεται σε πολλές διαφορετικές καταστάσεις ταυτόχρονα. Κάτι τέτοιο όμως συνήθως είναι αρκετά πολύπλοκο και ξεφεύγει από τους σκοπούς αυτής της εργασίας, γι' αυτό τα μοντέλα κβαντικών υπολογισμών που θα χρησιμοποιήσουμε θα είναι αρκετά πιο απλά (και εύχρηστα).

1.4 Οργάνωση της εργασίας

Το υπόλοιπο της εργασίας οργανώνεται ως εξής:

- Στο κεφάλαιο 2 δίνουμε ένα μαθηματικό μοντέλο για τους κβαντικούς υπολογισμούς το οποίο αποτελεί τη βάση της υπόλοιπης εργασίας. Επίσης εξηγούμε πώς τα κβαντικά κυκλώματα συνδέονται με αυτό το μοντέλο.

- Στο κεφάλαιο 3 παρουσιάζουμε τρία παραδείγματα γλωσσών κβαντικού προγραμματισμού. Εξετάζουμε τις γλώσσες QPL και QML και εισάγουμε την NTUA-QML, μία παραλλαγή της QML που δημιουργήθηκε για τους σκοπούς αυτής της εργασίας.

- Στο κεφάλαιο 4 ασχολούμαστε με την υλοποίηση ενός εξομοιωτή της NTUA-QML σε έναν σημερινό υπολογιστή. Η υλοποίησή μας βασίζεται στην ύπαρξη ενός ενδιάμεσου επιπέδου το οποίο εξομοιώνει ένα υποσύνολο της QPL.

- Στο κεφάλαιο 5 δίνουμε τρία παραδείγματα κβαντικών αλγορίθμων. Παρουσιάζονται ο αλγόριθμος του Deutsch, ο αλγόριθμος του Grover και ο αλγόριθμος του Shor. Για τους πρώτους δύο δίνουμε επίσης ολοκληρωμένες υλοποιήσεις σε NTUA-QML ενώ για τον τρίτο δίνουμε μια υλοποίηση ενός σημαντικού μέρους του (του διακριτού μετασχηματισμού Fourier).

- Στο κεφάλαιο 6 παρουσιάζουμε τα συμπεράσματα που προέκυψαν από αυτήν την εργασία.

Για την εξακρίβωση της ορθής λειτουργίας των αλγορίθμων του κεφαλαίου 5 χρησιμοποιήθηκε η υλοποίηση της γλώσσας NTUA-QML που παρουσιάζεται στο κεφάλαιο 4.

Κεφάλαιο 2

Κβαντικοί Υπολογισμοί

2.1 Εισαγωγή στους κβαντικούς υπολογισμούς

Οι κβαντικοί υπολογισμοί είναι ένα νέο πεδίο το οποίο, όπως και η θεωρία της κβαντομηχανικής στην οποία στηρίζεται, πολλές φορές εμφανίζει αντιφάσεις με την κοινή λογική και τον τρόπο που έχουμε συνηθίσει να σκεφτόμαστε. Ένας τρόπος για να κατανοήσουμε καλύτερα πώς λειτουργούν οι κβαντικοί υπολογισμοί είναι να προσπαθήσουμε να βρούμε αναλογίες με τον τρόπο λειτουργίας ενός συμβατικού υπολογιστή.

Η λειτουργία ενός συμβατικού υπολογιστή όπως ξέρουμε βασίζεται στην ικανότητα διενέργειας υπολογισμών στο δυαδικό σύστημα αρίθμησης, δηλαδή υπολογισμών στους οποίους χρησιμοποιούνται bit. Αυτό προϋποθέτει την ύπαρξη αντίστοιχων φυσικών διεργασιών οι οποίες να φέρνουν κάποιο σύστημα από μία κατάσταση η οποία αντιστοιχεί στο 1 σε μία που αντιστοιχεί στο 0, και αντίστροφα, όσες φορές χρειάζεται για να ολοκληρωθεί ο υπολογισμός. Στην πράξη τα φυσικά μεγέθη που χρησιμοποιούνται σχεδόν χωρίς εξαίρεση για το σχεδιασμό και την κατασκευή ενός υπολογιστή είναι ηλεκτρικά, δεν υπάρχει όμως κάτι που να εμποδίζει θεωρητικά την κατασκευή π.χ. ενός μηχανικού ή υδραυλικού υπολογιστή, αρκεί αυτός να έχει κάποια αντίστοιχη δυνατότητα χειρισμού των bit.

Ποιες είναι όμως αυτές οι βασικές δυνατότητες που έχει ένας συμβατικός υπολογιστής ως προς τον χειρισμό των bit; Θα πρέπει φυσικά να μπορεί κατά βούληση να θέσει την τιμή ενός bit σε 0 ή 1, να υπολογίσει κάποιες απλές συναρτήσεις των bit, καθώς και να τα αποθηκεύει και να τα διαβάζει όταν χρειάζεται κατά τη διάρκεια των υπολογισμών.

Μέχρι στιγμής περιγράψαμε σε πολύ αδρές γραμμές ένα υπολογιστικό μοντέλο, το οποίο θυμίζει τους σημερινούς υπολογιστές. Ας αλλάξουμε κάποιους από τους θεμελιώδεις αυτούς κανόνες του παιχνιδιού των υπολογισμών:

- Έστω πως κάθε bit δεν παίρνει τιμές στο πεδίο $\{0, 1\}$ αλλά μπορεί να έχει σαν τιμή οποιαδήποτε υπέρθεση (superposition) αυτών των δύο καταστάσεων. Επίσης κάθε ομάδα από n bit δεν παίρνει απλώς μία τιμή στο $\{0, \cdots, 2^n - 1\}$ αλλά οποιαδήποτε υπέρθεσή τους.

- Έστω πως το διάβασμα ενός bit δίνει πάντα μια απάντηση που ανήκει στο $\{0, 1\}$. Η απάντηση αυτή είναι μη ντετερμινιστική και οι πιθανότητες να είναι 0 ή 1 εξαρτώνται από την κατάσταση στην οποία βρισκόταν το bit. Επίσης το διάβασμα (το οποίο στο εξής θα λέμε *μέτρηση*) έχει σαν παρενέργεια το bit να έρθει σε μια κατάσταση τέτοια ώστε επόμενο διάβασμα του ίδιου bit να δίνει το ίδιο αποτέλεσμα με πιθανότητα 1.

Οι παραπάνω κανόνες ίσως φαίνονται περίεργοι, όμως προκύπτουν από (εξίσου περίεργα) συμπεράσματα της κβαντικής φυσικής. Όπως ένα σωματίδιο αντί για πλήρως καθορισμένη θέση έχει μια συνάρτηση πυκνότητας πιθανότητας, η οποία προσδιορίζει την πιθανότητα το σωματίδιο να βρεθεί σε κάθε θέση, έτσι και ένα κβαντικό bit (qbit) δεν έχει τιμή 0 ή 1 αλλά έχει κάποια πιθανότητα να μετρηθεί 0 ή 1.

Η χρήση της κβαντικής υπέρθεσης είναι αυτή που δίνει στους κβαντικούς αλγόριθμους το πλεονέκτημά τους απέναντι στους αντίστοιχους κλασικούς. Η πραγματοποίηση ενός υπολογισμού με 1 qbit το οποίο είναι ταυτόχρονα 0 και 1 θα μας δώσει μια υπέρθεση των αποτελεσμάτων

που θα μας είχε δώσει ο ίδιος υπολογισμός με είσοδο 0 και 1. Ομοίως η πραγματοποίηση ενός υπολογισμού n qbit θα μας δώσει μια υπέρθεση των 2^n αποτελεσμάτων.

Από την άλλη υπάρχουν περιορισμοί που επιβάλλονται πάνω στους υπολογισμούς ώστε να μπορούμε να εκμεταλλευτούμε τον κβαντικό παραλληλισμό, όπως η αντιστρεψιμότητα. Ακόμα πιο σημαντικό είναι ότι παρ'οτι το αποτέλεσμά μας είναι θεωρητικά μια υπέρθεση όλων των δυνατών αποτελεσμάτων, στην πράξη μπορούμε να δούμε (μετρώντας) μόνο ένα από τα δυνατά αποτελέσματα. Έτσι οι κβαντικοί αλγόριθμοι είναι αναγκασμένοι να καταφεύγουν σε τεχνάσματα ώστε να καταφέρνουν να προσδιορίσουν από την υπέρθεση των αποτελεσμάτων το αποτέλεσμα που πραγματικά ενδιαφέρει.

Όπως κανείς δεν θα προσπαθούσε ποτέ να προγραμματίσει έναν υπολογιστή χρησιμοποιώντας τους νόμους του Kirchoff, έτσι κανείς δεν προσπαθεί να διεξάγει κβαντικούς υπολογισμούς χρησιμοποιώντας μερικές διαφορικές εξισώσεις όπως η εξίσωση του Schroedinger. Στο υπόλοιπο αυτού του κεφαλαίου προσπαθούμε να δώσουμε τις λεπτομέρειες ενός μαθηματικού μοντέλου κβαντικών υπολογισμών το οποίο τελικά θα βασιστεί στην άλγεβρα των ερμιτιανών, θετικά ορισμένων πινάκων με μιγαδικά στοιχεία. Το μοντέλο αυτό θα είναι για τους κβαντικούς υπολογισμούς ότι η άλγεβρα Boole για τους κλασικούς: ένα στρώμα χαμηλού επιπέδου που θα έχει σαν στόχο να κρύψει τις φυσικές λεπτομέρειες της υλοποίησης χωρίς να μας στερεί υπολογιστική ικανότητα.

2.2 Κβαντικά bit

2.2.1 Γενικά

Ο βασικός τύπος δεδομένων κάθε κβαντικού υπολογισμού είναι τα κβαντικά bit. Όπως γνωρίζουμε οι δυνατές καταστάσεις ενός κλασικού bit b είναι $b = \mathbf{0}$ και $b = \mathbf{1}$. Οι δυνατές καταστάσεις στις οποίες μπορεί να βρίσκεται ένα κβαντικό bit q είναι οποιοσδήποτε γραμμικός συνδυασμός $q = \alpha\mathbf{0} + b\mathbf{1}$, όπου $\alpha, \beta \in \mathbb{C}$ και τα α, β δεν μπορεί να είναι ταυτόχρονα μηδέν. Οι συντελεστές α, β λέγονται πλάτη της κβαντικής κατάστασης. Αν για δύο κβαντικά bit q_1, q_2 έχουμε $q_i = \alpha_i\mathbf{0} + \beta_i\mathbf{1}, i = 1, 2$ και $\alpha_1 = \gamma\alpha_2$, $\beta_1 = \gamma\beta_2$ για κάποια μη μηδενική σταθερά $\gamma \in \mathbb{C}$ τότε οι καταστάσεις των δύο bit θεωρούνται ισοδύναμες. Συνήθως υποθέτουμε πως τα α, β είναι κανονικοποιημένα έτσι ώστε $|\alpha|^2 + |\beta|^2 = 1$. Αυτό μας επιτρέπει να αποδώσουμε μια φυσική σημασία στα α και β: είναι αριθμοί τέτοιοι ώστε $|\alpha|^2$ είναι η πιθανότητα μετά από μία μέτρηση να παρατηρηθεί η τιμή $\mathbf{0}$ στο q και $|\beta|^2$ η πιθανότητα να μετρηθεί $\mathbf{1}$.

Έτσι, στους κβαντικούς υπολογισμούς οι γνωστές από την άλγεβρα Boole σταθερές $\mathbf{0}$ και $\mathbf{1}$ δίνουν τη θέση τους σε δύο διανύσματα c_0 και c_1 τα οποία αποτελούν τη βάση ενός μιγαδικού διανυσματικού χώρου δύο διαστάσεων. Στη βιβλιογραφία τα δύο αυτά διανύσματα εμφανίζονται πολύ συχνά με το λεγόμενο 'ket' notation ως εξής: $c_0 = |\mathbf{0}\rangle$ και $c_1 = |\mathbf{1}\rangle$. Έτσι η κατάσταση στην οποία βρίσκεται ένα κβαντικό bit μπορεί να γραφτεί ως $\alpha|\mathbf{0}\rangle + \beta|\mathbf{1}\rangle$. Σημειώνουμε πως η λέξη ket προέρχεται από το δεύτερο μισό της λέξης bracket και εκτός από τα ket υπάρχουν και τα bra.

Οι καταστάσεις $1|\mathbf{0}\rangle + 0|\mathbf{1}\rangle$ και $0|\mathbf{0}\rangle + 1|\mathbf{1}\rangle$ λέγονται κλασικές καταστάσεις. Οποιαδήποτε άλλη κατάσταση λέγεται κβαντική υπέρθεση των $|\mathbf{0}\rangle$ και $|\mathbf{1}\rangle$.

2.2.2 Κβαντικοί καταχωρητές

Όπως γνωρίζουμε η κατάσταση στην οποία βρίσκεται ένα σύνολο από n κλασικά bit (ένας καταχωρητής των n bit) μπορεί να περιγραφεί πλήρως αν περιγράψουμε την κατάσταση στην οποία βρίσκεται κάθε ένα από τα n bit. Ένα από τα πιο ενδιαφέροντα χαρακτηριστικά του κβαντικού προγραμματισμού είναι ότι κάτι αντίστοιχο δεν ισχύει απαραίτητα για έναν καταχωρητή κβαντικών bit.

Ας πάρουμε για παράδειγμα έναν καταχωρητή δύο bit. Στην κλασική περίπτωση οι δυνατές καταστάσεις του είναι $|\mathbf{00}\rangle$,$|\mathbf{01}\rangle$,$|\mathbf{10}\rangle$ και $|\mathbf{11}\rangle$. Στην κβαντική περίπτωση οι δυνατές καταστάσεις

είναι ένας γραμμικός συνδυασμός της μορφής $\alpha_{00}|\mathbf{00}\rangle + \alpha_{01}|\mathbf{01}\rangle + \alpha_{10}|\mathbf{10}\rangle + \alpha_{11}|\mathbf{11}\rangle$. Όπως προηγουμένως υποθέτουμε πως κάποιο από τα α_{ij} είναι μη μηδενικό και ορίζουμε την ισοδυναμία ανάμεσα σε δύο καταστάσεις με ανάλογο τρόπο. Ορίζουμε λοιπόν την κατάσταση ενός καταχωρητή 2 bit σαν ένα διάνυσμα ενός μιγαδικού διανυσματικού χώρου τεσσάρων διαστάσεων με βάση $\{|\mathbf{00}\rangle, |\mathbf{01}\rangle, |\mathbf{10}\rangle, |\mathbf{11}\rangle\}$.

Παίρνουμε τώρα δύο κβαντικά bit q και p. Αν $p = \alpha|\mathbf{0}\rangle + \beta|\mathbf{1}\rangle$ και $q = \gamma|\mathbf{0}\rangle + \delta|\mathbf{1}\rangle$, ο συνδυασμός των δύο, δηλαδή η κατάσταση στην οποία βρίσκεται ένας καταχωρητής δύο bit ο οποίος αποτελείται από τα p και q είναι $pq = \alpha\gamma|\mathbf{00}\rangle + \alpha\delta|\mathbf{01}\rangle + \beta\gamma|\mathbf{10}\rangle + \beta\delta|\mathbf{11}\rangle$.

Από τα παραπάνω μπορούμε να συμπεράνουμε πως υπάρχουν πιθανές καταστάσεις για έναν καταχωρητή των 2 bit οι οποίες δεν μπορούν να περιγραφούν πλήρως σαν συνδυασμός των καταστάσεων δύο ανεξάρτητων κβαντικών bit. Για παράδειγμα η κατάσταση $\frac{1}{\sqrt{2}}|\mathbf{00}\rangle + \frac{1}{\sqrt{2}}|\mathbf{11}\rangle$ αφού θα πρέπει $\alpha\gamma \neq 0$, $\beta\delta \neq 0$ αλλά και $\alpha\delta = \beta\gamma = 0$.

Γενικεύοντας τα παραπάνω μπορούμε να πούμε πως η κατάσταση ενός συστήματος n qbit γράφεται ως $\alpha_0|\mathbf{00\cdots00}\rangle + a_1|\mathbf{00\cdots01}\rangle + \cdots + a_{2^n-1}|\mathbf{11\cdots11}\rangle$. Από εδώ και πέρα θα χρησιμοποιούμε για λόγους ευκολίας το διάνυσμα $[a_0, a_1, \cdots, a_{2^n-1}] \in \mathbb{C}^{2^n}$ για να αναπαραστήσουμε την ίδια κατάσταση.

2.2.3 Entanglement

Entanglement (συμπλοκή), ονομάζουμε το φαινόμενο κατά το οποίο η κατάσταση δύο ή περισσοτέρων κβαντικών bit δεν μπορεί να περιγραφεί σαν συνδυασμός των καταστάσεων του κάθε bit ξεχωριστά. Με άλλα λόγια κάποια bit είναι κατά κάποιο τρόπο συμπλεγμένα με κάποια άλλα. Όπως θα δούμε αργότερα entanglement μπορεί να δημιουργηθεί από διάφορους κβαντικούς μετασχηματισμούς που διενεργούνται σε περισσότερα του ενός bit.

Με απλά λόγια θα μπορούσε κανείς να πει πως όταν τα bit p και q είναι συμπλεγμένα ο προσδιορισμός της τιμής του ενός μας δίνει πληροφορία και για την κατάσταση του άλλου. Για παράδειγμα αν η κατάσταση των p και q είναι $\frac{1}{\sqrt{2}}|\mathbf{00}\rangle + \frac{1}{\sqrt{2}}|\mathbf{11}\rangle$ και μία μέτρηση μας δώσει την τιμή $|\mathbf{0}\rangle$ για το p αυτόματα γνωρίζουμε πως το q έχει επίσης την ίδια τιμή.

Όταν δύο κβαντικά bit δεν είναι συμπλεγμένα λέμε πως είναι ανεξάρτητα. Σε αυτή την περίπτωση η κατάσταση ενός καταχωρητή που τα περιέχει μπορεί να δοθεί αν πολλαπλασιάσουμε τους αντίστοιχους συντελεστές των δύο qbit. Η αντίστοιχη πράξη από τη γραμμική άλγεβρα είναι το τανυστικό γινόμενο (tensor product). Το tensor product ορίζεται για διανύσματα ως εξής: αν $u \in \mathbb{C}^n$ και $v \in \mathbb{C}^m$ τότε $w = u \otimes v \in \mathbb{C}^{nm}$ και $w_{(i,j)} = u_i v_j$.

2.3 Ορθομοναδιαίοι μετασχηματισμοί

Υπάρχουν δύο τρόποι με τους οποίους μπορούμε να χειριστούμε μια κβαντική κατάσταση ενός ή περισσοτέρων bit: οι ορθομοναδιαίοι (unitary) μετασχηματισμοί και οι μετρήσεις.

Η κατάσταση ενός κβαντικού συστήματος μετασχηματίζεται εφαρμόζοντας πάνω του έναν ορθομοναδιαίο μετασχηματισμό. Έστω πως ο S είναι ένας 2×2 ορθομοναδιαίος πίνακας, δηλαδή ισχύει $SS^* = I$ όπου S^* ο αναστροφοσυζυγής του S. Τότε η κατάσταση ενός qbit μετασχηματίζεται ως εξής:

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \mapsto S \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

Ομοίως χρησιμοποιώντας έναν ορθομοναδιαίο πίνακα μεγέθους $2^n \times 2^n$ μπορούμε να μετασχηματίσουμε μία ομάδα από n qbit. Ένας τέτοιος πίνακας καλείται και κβαντική πύλη n bit.

Μερικές από τις πιο χρήσιμες πύλες είναι οι εξής:

$$N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad V = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \quad W = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{i} \end{pmatrix}$$

$$N_c = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & N \end{array}\right) \quad H_c = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & H \end{array}\right) \quad V_c = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & V \end{array}\right) \quad W_c = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & W \end{array}\right)$$

$$X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Η πύλη N καλείται και πύλη not αφού $N|\mathbf{0}\rangle = |\mathbf{1}\rangle$ και $N|\mathbf{1}\rangle = |\mathbf{0}\rangle$. Η πύλη H λέγεται πύλη Hadamard και είναι μία από τις πιο χρήσιμες πύλες στους κβαντικούς υπολογισμούς. Με είσοδο ένα bit στην κατάσταση $|\mathbf{0}\rangle$ δίνει ένα bit που βρίσκεται σε μια υπέρθεση των $|\mathbf{0}\rangle$ και $|\mathbf{1}\rangle$ με τις ίδιες πιθανότητες. Οι πύλες V και W αντιπροσωπεύουν μιγαδικές αλλαγές φάσης.

Οι πύλες 2 bit στη δεύτερη σειρά αποτελούν τις controlled εκδοχές αυτών στην πρώτη. Δέχονται σαν είσοδο δύο bit και αν το πρώτο είναι στην κατάσταση $|\mathbf{0}\rangle$ επιστρέφουν το ίδιο αποτέλεσμα, ενώ αν είναι στην κατάσταση $|\mathbf{1}\rangle$ εκτελούν τον αντίστοιχο μετασχηματισμό στο άλλο bit. Προφανώς αν το πρώτο bit είναι σε μια υπέρθεση των $|\mathbf{0}\rangle$ και $|\mathbf{1}\rangle$ τότε το δεύτερο θα έρθει σε μια υπέρθεση της τωρινής του κατάστασης και εκείνης που προκύπτει από εφαρμογή της αντίστοιχης πυλης πάνω του. Μάλιστα η υπέρθεση αυτή εξαρτάται από την ακριβή υπέρθεση στην οποία βρίσκεται το πρώτο. Αυτό έχει σαν αποτέλεσμα να δημιουργηθει entanglement και τα δύο bit να πάψουν να είναι ανεξάρτητα.

Για παράδειγμα έστω πως έχουμε ένα bit στην κατάσταση $\frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle)$ και ένα στην κατάσταση $|\mathbf{0}\rangle$. Η συνολική κατάσταση είναι

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}$$

Εφαρμόζοντας την πύλη N_c έχω:

$$N_c \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

Όπως βλέπουμε η κατάσταση που προκύπτει είναι $\frac{1}{\sqrt{2}}(|\mathbf{00}\rangle + |\mathbf{11}\rangle)$, η οποία όπως έχουμε ήδη πει είναι ένα παράδειγμα entanglement.

Η πύλη X αποτελεί απλώς ένα swap των δύο bit στα οποία εφαρμόζεται.

Σε περίπτωση που το σύστημά μας έχει περισσότερα bit από όσα χρειάζεται ως είσοδο η πύλη που θέλουμε να υλοποιήσουμε μπορούμε να χρησιμοποιήσουμε το tensor product ώστε να κατασκευάσουμε μία πύλη η οποία έχει το σωστό αριθμό εισόδων και αφήνει τα υπόλοιπα bit απείραχτα. Για παράδειγμα, έστω πως έχω 4 bit και θέλω να εφαρμόσω στο δεύτερο την πύλη Hadamard, θα εφαρμόσω στο σύστημα την πύλη $I \otimes H \otimes I \otimes I$, όπου I ο 2×2 μοναδιαίος πίνακας. Το tensor product σε πίνακες ορίζεται ως εξής: $A \in \mathbb{C}^{n \times n}$ και $B \in \mathbb{C}^{m \times m}$ τότε $C = A \otimes B \in \mathbb{C}^{nm \times nm}$ και $c_{(i,j),(i',j')} = a_{ii'} b_{jj'}$

Θεωρητικά, κάθε ορθομοναδιαίος πίνακας μπορεί να υλοποιηθεί σε έναν κβαντικό υπολογιστή, όμως όπως στην πράξη τα ψηφιακά κυκλώματα υλοποιούνται χρησιμοποιώντας ένα μικρό (αλλά πλήρες) σύνολο από πύλες, έτσι και τα κβαντικά κυκλώματα ενός κβαντικού υπολογιστή θα υλοποιούνται με ένα μικρό σύνολο από πύλες. Υπάρχει ένα σύνολο από κβαντικές πύλες

τέτοιο ώστε κάθε κβαντική πύλη να μπορεί να προσεγγιστεί με οσοδήποτε μεγάλη ακρίβεια από αυτές τις πύλες. Ένα τέτοιο σύνολο σχηματίζουν οι πύλες $\{H, V_c, X\}$ ([Clev00]).

2.4 Μετρήσεις

Η δεύτερη θεμελιώδης πράξη με την οποία μπορούμε να χειριστούμε ένα κβαντικό σύστημα είναι η μέτρηση. Η μέτρηση συμβαίνει όταν προσπαθούμε να παρατηρήσουμε την τιμή ενός qbit και να το μετατρέψουμε σε κλασικό bit. Κατά κανόνα η μέτρηση είναι το τελευταίο βήμα της εκτέλεσης ενός κβαντικού αλγορίθμου, χωρίς όμως αυτό να είναι απαραίτητο.

Ας θεωρήσουμε την κατάσταση ενός απομονωμένου qbit $\alpha|\mathbf{0}\rangle + \beta|\mathbf{1}\rangle$. Έστω επίσης πως τα α και β είναι κανονικοποιημένα με τον τρόπο που έχουμε ήδη αναφέρει, δηλαδή $|\alpha|^2 + |\beta|^2 = 1$. Τότε η μέτρηση του qbit θα μας δώσει σαν αποτέλεσμα $\mathbf{0}$ με πιθανότητα $|\alpha|^2$ ή $\mathbf{1}$ με πιθανότητα $|\beta|^2$. Η μέτρηση έχει σαν παρενέργεια την κατάρρευση (collapse) της κβαντικής κατάστασης, ώστε μια αμέσως επόμενη μέτρηση δίνει το ίδιο αποτέλεσμα με την πρώτη με πιθανότητα 1. Με άλλα λόγια μία μέτρηση έχει ως παρενέργεια τη μετάβαση της κατάστασης σε μία εκ των $1|\mathbf{0}\rangle + 0|\mathbf{1}\rangle$ ή $0|\mathbf{0}\rangle + 1|\mathbf{1}\rangle$.

Τα πράγματα περιπλέκονται όταν έχουμε μία μέτρηση στην οποία εμπλέκονται περισσότερα του ενός κβαντικά bit. Θεωρούμε ένα σύστημα 2 qbit με κατάσταση $\alpha|\mathbf{00}\rangle + \beta|\mathbf{01}\rangle + \gamma|\mathbf{10}\rangle + \delta|\mathbf{11}\rangle$. Υποθέτουμε και πάλι ότι τα πλάτη είναι κανονικοποιημένα. Μετράμε το πρώτο qbit. Με πιθανότητα $|\alpha|^2 + |\beta|^2$ το αποτέλεσμα είναι $\mathbf{0}$ και η κατάσταση του συστήματος καταρρέει στην $\alpha|\mathbf{00}\rangle + \beta|\mathbf{01}\rangle$. Με πιθανότητα $|\gamma|^2 + |\delta|^2$ το αποτέλεσμα είναι $\mathbf{1}$ και η κατάσταση του συστήματος γίνεται $\gamma|\mathbf{10}\rangle + \delta|\mathbf{11}\rangle$.

Έστω πως η μέτρηση του πρώτου qbit έδωσε $\mathbf{0}$. Τότε μια μέτρηση του δεύτερου μας δίνει με πιθανότητα $\frac{|\alpha|^2}{|\alpha|^2+|\beta|^2}$ αποτέλεσμα $\mathbf{0}$ και νέο state $\alpha|\mathbf{00}\rangle$ και με πιθανότητα $\frac{|\beta|^2}{|\alpha|^2+|\beta|^2}$ αποτέλεσμα $\mathbf{1}$ και νέο state $\beta|\mathbf{01}\rangle$. Αντίστοιχα ισχύουν και στην περίπτωση που η μέτρηση του πρώτου qbit έδωσε $\mathbf{1}$.

Από τα παραπάνω συμπεραίνουμε πως με δύο διαδοχικές μετρήσεις του πρώτου και του δεύτερου qbit καταλήγουμε σε μία από τις τέσσερις καταστάσεις $|\mathbf{00}\rangle, |\mathbf{01}\rangle, |\mathbf{10}\rangle, |\mathbf{11}\rangle$ με πιθανότητα $|\alpha|^2, |\beta|^2, |\gamma|^2, |\delta|^2$ αντίστοιχα. Είναι εύκολο να διαπιστωθεί πως με τις ίδιες πιθανότητες καταλήγουμε στις ίδιες καταστάσεις αν αντιστρέψουμε τη σειρά των μετρήσεων. Στην ορολογία της κβαντομηχανικής αυτό σημαίνει πως οι δύο μετρήσεις συγκοινωνούν (commute). Γενικά στην κβαντομηχανική αυτό δεν είναι απαραίτητο να ισχύει για δύο μετρήσεις, στους κβαντικούς υπολογισμούς όμως ισχύει πάντα.

Εισάγουμε τώρα μία νέα κανονικοποίηση για τις κβαντικές καταστάσεις: κανονικοποιούμε τα πλάτη μίας κατάστασης έτσι ώστε το άθροισμα των τετραγώνων των πλατών να είναι ίσο με τη συνολική πιθανότητα να φτάσουμε σε μία κατάσταση. Είναι προφανές πως όταν στους υπολογισμούς μας δεν περιέχονται μετρήσεις η κανονικοποίηση αυτή είναι η ίδια με αυτή που έχουμε ήδη προτείνει. Όταν όμως πραγματοποιηθεί μία μέτρηση η σύμβαση αυτή μας απαλλάσσει από την ανάγκη να επανακανονικοποιήσουμε τα πλάτη των νέων καταστάσεων που προκύπτουν και απλοποιεί τους υπολογισμούς μας.

2.5 Αγνές και Μικτές καταστάσεις

Όπως έχουμε πει η κατάσταση ενός κβαντικού συστήματος απο n qbit μπορεί να περιγραφεί από ένα διάνυσμα $u \in \mathbb{C}^{2^n}$. Παρ'ολα αυτά ένας εξωτερικός παρατηρητής του συστήματος μπορεί να μην έχει πλήρη γνώση της κατάστασής του. Έστω για παράδειγμα ένα σύστημα το οποίο βρίσκεται στην κατάσταση $\frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle)$. Αν κάποιος διενεργήσει μία μέτρηση πάνω στο μοναδικό qbit του συστήματος και κρατήσει μυστικό το αποτέλεσμα δεν μπορούμε με βεβαιότητα να πούμε ποια είναι η νέα κατάσταση του συστήματος. Γνωρίζουμε όμως ότι είναι μία εκ των $|\mathbf{0}\rangle$ και $|\mathbf{1}\rangle$ με ίση πιθανότητα. Γράφουμε λοιπόν ότι η κατάσταση του συστήματος

είναι $\frac{1}{2}\{|\mathbf{0}\rangle\} + \frac{1}{2}\{|\mathbf{1}\rangle\}$. Γενικότερα, όταν ξέρουμε ότι ένα σύστημα μπορεί να βρίσκεται σε μία κατάσταση u_i με πιθανότητα λ_i και όλες οι πιθανές καταστάσεις είναι m γράφουμε ότι η κατάστασή του είναι $\lambda_1\{u_1\} + \cdots + \lambda_m\{u_m\}$. Θα πρέπει να ισχύει $\sum \lambda_i = 1$. Μία τέτοια κατανομή πιθανότητας ανάμεσα σε κβαντικές καταστάσεις λέγεται μικτή κατάσταση (mixed state). Οι καταστάσεις u_i λέγονται αγνές καταστάσεις (pure states).

Είναι σημαντικό να σημειώσουμε πως οι mixed states δεν έχουν κάποιο ιδιαίτερο φυσικό νόημα, αλλά αντιπροσωπεύουν τη γνώση κάποιου παρατηρητή για την κατάσταση του συστήματος. Υποτίθεται πως κάθε φυσικό σύστημα βρίσκεται ανά πάσα στιγμή σε μία κατάσταση η οποία μπορεί να περιγραφεί από μία pure state αλλά από την άποψη ενός παρατηρητή το σύστημα ίσως βρίσκεται σε μία mixed state και πιθανώς από την άποψη ενός άλλου παρατηρητή ίσως βρίσκεται σε μία διαφορετική mixed state.

Οι ορθομοναδιαίοι μετασχηματισμοί μετασχηματίζουν τις mixed states δρώντας πάνω στις pure states που αποτελούν τα συστατικά τους. Δηλαδή ο μετασχηματισμός S φέρνει την mixed state $\sum \lambda_i\{u_i\}$ στην $\sum \lambda_i\{Su_i\}$.

2.6 Πίνακες πυκνότητας

Θα εισάγουμε τώρα την έννοια των πινάκων πυκνότητας οι οποίοι αποτελούν έναν πολύ πιο αποτελεσματικό τρόπο αναπαράστασης των pure και mixed states.

Αρχικά ας πάρουμε ένα σύστημα του οποίου η κατάσταση μπορεί να περιγραφεί με το διάνυσμα $u \in \mathbb{C}^{2^n}$. Είναι σαφές ότι μιλάμε για μία αγνή κατάσταση. Μία ισοδύναμη περιγραφή του συστήματος είναι από τον πίνακα uu^*, ο οποίος καλείται πίνακας πυκνότητας (density matrix). Για παράδειγμα το σύστημα $\frac{1}{\sqrt{2}}(|\mathbf{0}\rangle - |\mathbf{1}\rangle)$ έχει πίνακα πυκνότητας τον

$$uu^* = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Ας σημειώσουμε εδώ πως αν γνωρίζουμε τον πίνακα πυκνότητας μιας κατάστασης δεν μπορούμε να προσδιορίσουμε μοναδικά το διάνυσμα u από το οποίο αυτός προήλθε αφού αν $\gamma \in \mathbb{C}$ και $|\gamma| = 1$ τότε ο ίδιος πίνακας πυκνότητας προκύπτει και από κάθε διάνυσμα γu. Αυτό όμως στην πραγματικότητα είναι πλεονέκτημα της χρήσης πινάκων πυκνότητας αντί για διανυσμάτων γιατί μας δίνει πιο εύκολα τη δυνατότητα να προσδιορίσουμε την ισοδυναμία δύο καταστάσεων: θα πρέπει αυτές να έχουν τον ίδιο πίνακα πυκνότητας.

Το μεγάλο πλεονέκτημα όμως των πινάκων πυκνότητας βρίσκεται στον τρόπο αναπαράστασης των mixed states. Το mixed state $\sum \lambda_i\{u_i\}$ παριστάνεται από τον πίνακα $\sum \lambda_i(u_i u_i^*)$. Για παράδειγμα το mixed state $\frac{1}{2}\{|\mathbf{0}\rangle\} + \frac{1}{2}\{|\mathbf{1}\rangle\}$ έχει αντίστοιχο πίνακα πυκνότητας

$$\frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$$

Αξίζει να πούμε πως η μετάβαση από την προηγούμενη αναπαράσταση mixed stated σε πίνακες πυκνότητας προκαλεί κάποια απώλεια πληροφορίας. Για παράδειγμα είναι εύκολο να δούμε πως το mixed state $\frac{1}{2}\{\frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle)\} + \frac{1}{2}\{\frac{1}{\sqrt{2}}(|\mathbf{0}\rangle - |\mathbf{1}\rangle)\}$ έχει πίνακα πυκνότητας τον

$$\frac{1}{2}\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} + \frac{1}{2}\begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$$

Ο πίνακας αυτός είναι ίδιος με αυτόν που υπολογίσαμε στο προηγούμενο παράδειγμα. Όπως θα δούμε αργότερα δεν υπάρχει παρατηρήσιμη διαφορά ανάμεσα σε δύο μικτές καταστάσεις οι οποίες μας δίνουν τον ίδιο πίνακα πυκνότητας, και επομένως δεν υπάρχει πρόβλημα αν οι δύο αυτές καταστάσεις θεωρούνται ταυτόσημες. Η διαφορά που υπάρχει ανάμεσα σε δύο mixed states με τον ίδιο πίνακα πυκνότητας μπορεί να γίνει πιο κατανοητή με μελέτη του παραδείγματος 3.2.3

Από τα παραπάνω προκύπτει ότι για έναν πίνακα A έχω $A = \sum \lambda_i (u_i u_i^*)$ όπου τα λ_i είναι μη αρνητικές πραγματικές σταθερές με $\sum \lambda_i \leq 1$ και τα u_i μοναδιαία διανύσματα αν και μόνο αν ο A είναι ερμιτιανός, θετικά ορισμένος με $tr(A) \leq 1$. Το ευθύ της πρότασης αυτής προκύπτει εύκολα από το γεγονός ότι $tr(u_i u_i^*) = u_i^* u_i = 1$ για μοναδιαία διανύσματα. Το ανάστροφο από το γεγονός ότι κάθε ερμιτιανός θετικά ορισμένος πίνακας A μπορεί να διαγωνοποιηθεί ως $A = SDS^*$ με $D = \sum \lambda_i e_i e_i^*$ όπου e_i τα διανύσματα της κανονικής βάσης του αντίστοιχου χώρου. Τότε θα ισχύει $A = \sum \lambda_i (Se_i)(Se_i)^*$. Τα λ_i είναι οι ιδιοτιμές του A. Προφανώς αν ο A είναι πίνακας που αντιστοιχεί σε pure state τότε όλες οι ιδιοτιμές είναι ίσες με το 0 εκτός από μία.

Δίνουμε λοιπόν τον εξής ορισμό: Πίνακας πυκνότητας λέγεται ένας θετικά ορισμένος ερμιτιανός πίνακας για τον οποίο ισχύει $tr(A) \leq 1$. Το σύνολο των πινάκων αυτών με διάσταση n το συμβολίζουμε με $D_n \subseteq \mathbb{C}^{n \times n}$.

2.7 Κβαντικές λειτουργίες και πίνακες πυκνότητας

Έχουμε ορίσει δύο ήδη κβαντικών λειτουργιών: τους ορθομοναδιαίους μετασχηματισμούς και τις μετρήσεις. Οι ορθομοναδιαίοι μετασχηματισμοί εύκολα επεκτείνονται στη χρήση των πινάκων πυκνότητας. Αν ο μετασχηματισμός S αντιστοιχεί την αγνή κατάσταση u στην κατάσταση Su τότε αντιστοιχεί τον αγνό πίνακα πυκνότητας uu^* στον Suu^*S^*. Σε σχέση με τις μικτές καταστάσεις, έχουμε πει ότι ο ορθομοναδιαίος μετασχηματισμός εφαρμόζεται ξεχωριστά σε κάθε μία από τις αγνές συνιστώσες της μικτής κατάστασης. Επομένως ο mixed πίνακας πυκνότητας A αντιστοιχεί στον SAS^*.

Ας ασχοληθούμε τώρα με το αποτέλεσμα της μέτρησης σε έναν πίνακα πυκνότητας. Ας θεωρήσουμε αρχικά μία αγνή κατάσταση u με τον αντίστοιχο πίνακα uu^*. Έστω πως

$$u = \left(\frac{v}{w}\right) \Rightarrow uu^* = \left(\begin{array}{c|c} vv^* & vw^* \\ \hline wv^* & ww^* \end{array}\right)$$

Μία μέτρηση του πρώτου bit της κατάστασης θα δώσει αποτέλεσμα $\left(\frac{v}{0}\right)$ με πιθανότητα $\|v\|^2$ ή $\left(\frac{0}{w}\right)$ με πιθανότητα $\|w\|^2$. Δηλαδή σε πίνακες πυκνότητας το αποτέλεσμα θα είναι $\left(\begin{array}{c|c} vv^* & 0 \\ \hline 0 & 0 \end{array}\right)$ ή $\left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & ww^* \end{array}\right)$.

Παρατηρούμε ότι ο πρώτος πίνακας προκύπτει με πιθανότητα $\|v\|^2 = tr(vv^*)$ και ο δεύτερος με $\|w\|^2 = tr(ww^*)$. Έτσι η σύμβαση της κανονικοποίησης επεκτείνεται φυσικά σους πίνακες πυκνότητας: το ίχνος ενός πίνακα πυκνότητας είναι ίσο με την πιθανότητα να φτάσουμε στην κατάσταση που αντιπροσωπεύει ο πίνακας.

Η λειτουργία της μέτρησης επεκτείνεται γραμμικά και στις μικτές καταστάσεις. Έτσι η πραγματοποίηση μιας μέτρησης σε μια μικτή κατάσταση της μορφής $\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$ θα μας δώσει σαν αποτέλεσμα την κατάσταση με πίνακα πυκνότητας $\left(\begin{array}{c|c} A & 0 \\ \hline 0 & 0 \end{array}\right)$ με πιθανότητα $tr(A)$ ή την κατάσταση με πίνακα πυκνότητας $\left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & D \end{array}\right)$ με πιθανότητα $tr(D)$.

Μία σημαντική παρατήρηση είναι η εξής: έστω πως διεξάγουμε μία μέτρηση και μετά ξεχνάμε το αποτέλεσμα. Τότε ο νέος πίνακας πυκνότητας είναι ο

$$\left(\begin{array}{c|c} A & 0 \\ \hline 0 & D \end{array}\right)$$

Δηλαδή η μέτρηση έχει πάντα σαν παρενέργεια τον μηδενισμό κάποιων περιοχών του πίνακα πυκνότητας.

Είδαμε τα αποτελέσματα των θεμελιωδών λειτουργιών των κβαντομηχανικής, των ορθομοναδιαίων μετασχηματισμών και των μετρήσεων, όπως εκφράζονται μέσα από τα αποτελέσματά τους πάνω στους πίνακες πυκνότητας. Αφού δεν έχουμε στη διάθεσή μας άλλο τρόπο αλληλεπίδρασης με ένα κβαντικό σύστημα πέρα από αυτές τις λειτουργίες συμπεραίνουμε πως δεν υπάρχει παρατηρήσιμη διαφορά ανάμεσα σε δύο μικτές καταστάσεις που έχουν τον ίδιο πίνακα πυκνότητας.

2.8 Κβαντικά κυκλώματα

Όσα έχουμε πει εδώ συνιστούν ένα μοντέλο κβαντικών υπολογισμών. Θα μπορούσαμε να συνεχίσουμε, θα κάνουμε όμως μια μικρή παράκαμψη για να αναφερθούμε στα κβαντικά κυκλώματα.

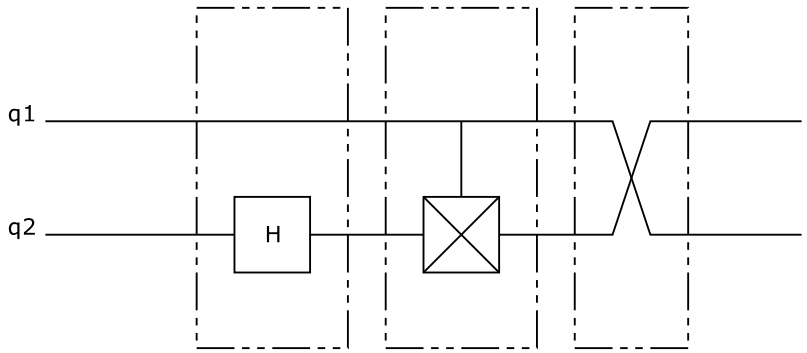
Στους κλασικούς υπολογισμούς υπάρχουν τα λογικά κυκλώματα: κυκλώματα τα οποία αποτελούνται από ένα πεπερασμένο πλήθος από πύλες, η έξοδος των οποίων μας δίνει το αποτέλεσμα ενός υπολογισμού. Ήδη και στους κβαντικούς υπολογισμούς έχουμε υπονοήσει την ύπαρξη μιας αντίστοιχης ιδέας καλώντας τους ορθομοναδιαίους μετασχηματισμούς κβαντικές πύλες. Είναι εύκολο λοιπόν να προχωρήσουμε στον ορισμό ενός άλλου μοντέλου υπολογισμών, αυτού των κβαντικών κυκλωμάτων.

Ένα κβαντικό κύκλωμα μας επιτρέπει να εκφράσουμε ένα σύνολο διαδοχικών ορθομοναδιαίων μετασχηματισμών οι οποίοι διενεργούνται πάνω στα qbits της εισόδου. Δεν μας επιτρέπει να εκφράσουμε τη λειτουργία της μέτρησης η οποία υποτίθεται πως διενεργείται πάντα στην έξοδο του κυκλώματος, δηλαδή στο αποτέλεσμα των υπολογισμών. Οι περιορισμοί που συναντάμε στο σχεδιασμό ενός κβαντικού κυκλώματος σε σχέση με ένα κλασικό είναι οι εξής:

- Όλες οι πύλες έχουν τόσες εισόδους όσες και εξόδους. Αυτό προκύπτει από το γεγονός ότι οι ορθομοναδιαίοι μετασχηματισμοί είναι αντιστρέψιμοι, άρα και οι πύλες θα πρέπει να είναι αντιστρέψιμες.

- Όλο το κύκλωμα έχει τόσες εισόδους όσες και εξόδους. Αυτό προκύπτει από το γεγονός ότι όλο το κύκλωμα, από τη στιγμή που δεν περιέχει μετρήσεις, είναι στην ουσία ένας ορθομοναδιαίος μετασχηματισμός. Επομένως κάθε κβαντικό κύκλωμα δουλεύει και προς τις δύο κατευθύνσεις.

- Δεν μπορούμε να χρησιμοποιήσουμε 'διακλαδώσεις'. Με άλλα λόγια αν ένα κβαντικό bit χρησιμοποιείται στην είσοδο μιας πύλης, δεν μπορεί να χρησιμοποιείται στην είσοδο κάποιας άλλης, ούτε και σε διαφορετική είσοδο της ίδιας. Και αυτός ο περιορισμός προκύπτει από την αντιστρεψιμότητα (η διακλάδωση είναι ουσιαστικά μία συνάρτηση που παίρνει ένα bit και δίνει δύο bit ίσα με αυτό της εισόδου).



Σχήμα 2.1: Ένα παράδειγμα κβαντικού κυκλώματος.

Ένα παράδειγμα κβαντικού κυκλώματος μπορούμε να δούμε στο σχήμα 2.1. Το κύκλωμα έχει δύο εισόδους $q1, q2$. Μπορεί να χωριστεί σε τρεις φάσεις. Στην πρώτη το bit $q1$ δεν επηρεάζεται ενώ στο bit $q2$ εφαρμόζεται η πύλη Hadamard. Συνολικά λοιπόν στο σύστημα των δύο qbit εφαρμόζεται ο μετασχηματισμός $I \otimes H$. Στην δεύτερη φάση βλέπουμε το συμβολισμό που χρησιμοποιείται στα κβαντικά κυκλώματα για τις ελεγχόμενες πύλες. Η πύλη μέσα από την οποία περνά το $q2$ είναι η γνωστή πύλη NOT. Η σύνδεση όμως της πύλης με το $q1$ σημαίνει πως ο μετασχηματισμό ελέγχεται από το $q1$, επομένως το σύστημα περνά από την πύλη N_c. Ουσιαστικά ο συμβολισμός αυτός είναι μια συντομογραφία η οποία μας διευκολύνει να καταλάβουμε τη λειτουργία μιας ελεγχόμενης πύλης. Στην τρίτη φάση τα δύο bit αλλάζουν θέση. Αυτό ισοδυναμεί με αλλαγή βάσης ή με χρήση της πύλης X. Καταλήγουμε λοιπόν πως το κύκλωμα αυτό είναι ισοδύναμο με τον μετασχηματισμό $XN_c(I \otimes H)$. Αν η κατάσταση του συστήματος ήταν αρχικά $\alpha|\mathbf{00}\rangle + \beta|\mathbf{01}\rangle + \gamma|\mathbf{10}\rangle + \delta|\mathbf{11}\rangle$ θα γίνει

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} \alpha+\beta \\ \gamma-\delta \\ \alpha-\beta \\ \gamma+\delta \end{pmatrix}$$

.

Αν για παράδειγμα και τα δύο qbit αρχικά ήταν false. Τότε $\alpha = 1$ και $\beta = \gamma = \delta = 0$. Σε αυτήν την περίπτωση ο ελεγχόμενος μετασχηματισμός δεν πραγματοποιείται, έτσι το κύκλωμα απλώς περνά το δεύτερο bit από την πύλη Hadamard και τους αλλάζει θέση. Το τελικό σύστημα θα είναι το tensor product του $\frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle)$ με το $|\mathbf{0}\rangle$, αφού τα δύο bits άλλαξαν θέση. Θα είναι δηλαδή $\frac{1}{\sqrt{2}}(|\mathbf{00}\rangle + |\mathbf{10}\rangle)$. Το ίδιο αποτέλεσμα παίρνουμε και με αντικατάσταση των $\alpha, \beta, \gamma, \delta$ παραπάνω.

Κεφάλαιο 3

Γλώσσες κβαντικού προγραμματισμού

3.1 Εισαγωγή

Η γλώσσα την οποία κανείς χρησιμοποιεί για να εκφράσει συλλογισμούς και συμπεράσματα πολύ συχνά παίζει σημαντικό ρόλο στη σύλληψη των ίδιων αυτών συλλογισμών. Είναι μέσα στα χαρακτηριστικά μιας καλής γλώσσας, είτε μιλάμε για μια γλώσσα φυσική είτε για μια γλώσσα προγραμματισμού, ο πλούτος και η ευκολία της εκφραστικότητας. Θα πρέπει δηλαδή μια γλώσσα να επιτρέπει στον χρήστη της να μεταφέρει τις σκέψεις του σε αυτή χωρίς δυσκολία, ή ίσως και να είναι έτσι δομημένη που να προσφέρει από μόνη της στο χρήστη προφανείς δημιουργικές διεξόδους σε προβλήματα που για άλλες γλώσσες ίσως να θεωρούνταν δύσκολα.

Η συναρπαστική πρόοδος της τεχνολογίας των υπολογιστών τις τελευταίες δεκαετίες, και μαζί της η πρόοδος των συστημάτων λογισμικού βάδισαν χέρι-χέρι με την πρόοδο στις γλώσσες προγραμματισμού. Είναι αδύνατο να φανταστούμε το πώς θα μπορούσαν σήμερα να υπάρξουν συστήματα λογισμικού τόσο πολύπλοκα όσο αυτά που βλέπουμε γύρω μας καθημερινά χωρίς την ύπαρξη γλωσσών υψηλού επιπέδου. Οι γλώσσες υψηλού επιπέδου έχουν στόχο παρόμοιο με αυτόν που πετυχαίνουν αντίστοιχες αφαιρέσεις σε χαμηλότερα επίπεδα στο σχεδιασμό ενός υπολογιστή: όπως η χρήση λογικών πυλών και καταχωρητών καταφέρνει να δώσει στο σχεδιαστή ένα απλό και εύχρηστο σύνολο κανόνων χωρίς να του στερεί σημαντικές δυνατότητες υπολογισμού που θα του προσέφερε η σκέψη κατευθείαν με όρους ηλεκτρικών κυκλωμάτων, έτσι και η γλώσσα ωψηλού επιπέδου δίνει στον προγραμματιστή ένα σύνολο από στοιχειώδη εργαλεία με τα οποία τον προστατεύει από την υπερβολική λεπτομέρεια της χρήσης απευθείας του υλικού.

Ο παραλληλισμός των παραπάνω με την κβαντική περίπτωση είναι σαφής. Θα πρέπει και για τους κβαντικούς υπολογισμούς, όπως και για τους κλασικούς, να δημιουργηθούν γλώσσες οι οποίες θα αποκαλύπτουν στον προγραμματιστή όλα τα χαρακτηριστικά του κβαντικού μοντέλου υπολογισμού κρύβοντάς του τις αποπροσανατολιστικές λεπτομέρειες. Μέχρι σήμερα έχουν γίνει πολλές και αξιόλογες προσπάθειες προς αυτήν την κατεύθυνση, κάποιες εκ των οποίων παρουσιάζονται παρακάτω. Οι προσπάθειες αυτές όμως σκοντάφτουν συνήθως σε ένα θεμελιώδες πρόβλημα: ότι ο ίδιος ο κβαντικός προγραμματισμός είναι δυσνόητος από τη φύση του.

Η δημιουργία των κλασικών υπολογιστών ήταν μια φυσική εξέλιξη του τρόπου σκέψης με τον οποίο η ανθρωπότητα ήταν εξοικειωμένη για αιώνες. Ήδη πριν φτιαχτούν οι πρώτοι υπολογιστές οι άνθρωποι ήξεραν να λύνουν απλά προβλήματα όπως η ταξινόμηση και η εύρεση του μέγιστου κοινού διαιρέτη δύο ακεραίων με τον ίδιο τρόπο που αργότερα προγραμμάτισαν τους υπολογιστές να το κάνουν. Ουσιαστικά το μοντέλο λειτουργίας ενός σημερινού υπολογιστή είναι αυτό ενός 'χαζού' ανθρώπου, ενός ανθρώπου δηλαδή ο οποίος μπορεί να ακολουθεί απλές οδηγίες και να κάνει πράξεις. Η ιδέα της ύπαρξης ενός υπολογιστή δεν ξεπήδησε από την παρατήρηση μιας ενδιαφέρουσας ηλεκτρικής συμπεριφοράς των ημιαγωγών, αλλά υπήρχε ήδη αιώνες πριν. Η ανακάλυψη ενός κατάλληλου φυσικού ισοδύναμου ήταν απλώς το τελευταίο συστατικό που χρειαζόταν για να πάρει το όραμα των υπολογιστικών μηχανών σάρκα και οστά.

Ο κβαντικός προγραμματισμός αντιθέτως είναι κάτι ξένο από τον τρόπο που οι άνθρωποι συνηθίζουν να σκέφτονται όταν λύνουν μαθηματικά ή λογικά προβλήματα. Πώς γίνεται ένα bit

να είναι ταυτόχρονα 0 και 1; Πώς γίνεται δύο bit να είναι συμπλεγμένα και η μέτρηση του ενός να επηρεάζει το άλλο; Γιατί το διάβασμα ενός bit είναι μία μη αναστρέψιμη διαδικασία; Μια κλασική γλώσσα έχει απλώς σαν αποστολή να δώσει στον προγραμματιστή δομές ελέγχου τέτοιες ώστε η καθοδήγηση του 'χαζού ανθρώπου' που κάνει τις πράξεις να γίνει πιο άνετη. Μια κβαντική γλώσσα θα πρέπει να γεφυρώσει το χάσμα ανάμεσα σε έναν προγραμματιστή και ένα εξίσου χαζό αλλά εντελώς ασυνήθιστο ον.

Η νεότητα του πεδίου των κβαντικών υπολογισμών δημιουργεί ένα ακόμη πρόβλημα που δυσχεραίνει την κατάσταση: την σπανιότητα ενδιαφέροντων αλγορίθμων. Παρ'οτι υπάρχουν λαμπρές εξαιρέσεις, οι οποίες είναι άλλωστε και αυτές που ώθησαν στην εξερεύνηση του πεδίου, όπως ο αλγόριθμος του Shor ο οποίος υποψιαζόμαστε ότι είναι ποιοτικά καλύτερος από οποιονδήποτε κλασικό αλγόριθμο που λύνει το πρόβλημα της παραγοντοποίησης ακεραίων, είναι δύσκολο να βρεις κανείς στη βιβλιογραφία έναν χρήσιμο κβαντικό αλγόριθμο. Αυτό βέβαια είναι ένα φυσικό επακόλουθο της ανυπαρξίας μέχρι αυτή τη στιγμή και κβαντικών υπολογιστών. Όμως μια καλή γλώσσα προγραμματισμού θα πρέπει να επιτρέπει στο χρήστη να εκφράσει εύκολα τα στοιχεία εκείνα που συνθέτουν έναν καλό αλγόριθμο και η αλήθεια είναι πως μέχρι σήμερα δεν είμαστε σίγουροι ποια είναι τα στοιχεία που συνθέτουν έναν καλό κβαντικό αλγόριθμο, γι' αυτό άλλωστε γνωρίζουμε και τόσο λίγους.

3.2 Η γλώσσα QPL

Στο [Seli04] ο Peter Selinger περιγράφει μία γλώσσα κβαντικού προγραμματισμού που προκύπτει αβίαστα από το μοντέλο προγραμματισμού που δώσαμε στο κεφάλαιο 2.

Χαρακτηριστικά της γλώσσας είναι:

- Υποστήριξη κλασικών bit και σαφής διαχωρισμός κλασικών και κβαντικών bit.
- Υποστήριξη οποιουδήποτε ορθομοναδιαίου μετασχηματισμού
- Υποστηριξη loops
- Υποστήριξη αναδρομικών συναρτήσεων

Η σημασιολογία κάθε προγράμματος δίνεται σαν μία συνάρτηση $D_n^\alpha \to D_m^\beta$ όπου τα n, m εξαρτώνται από το πλήθος των qbit της εισόδου και της εξόδου και τα α, β από το πλήθος των κλασικών bit εισόδου και εξόδου. Δίνουμε παρακάτω τη σύνταξη της γλώσσας και εξηγούμε με συντομία τη σημασιολογία της. Περισσότερες λεπτομέρειες δίνονται στο κεφάλαιο 4.

3.2.1 Σύνταξη

Η γραμματική των όρων της QPL είναι η εξής:

$$
\begin{array}{rcl}
P & ::= & \textbf{new bit}\ \mathrm{b} := 0 \mid \textbf{new qbit}\ \mathrm{q} := 0 \mid \textbf{discard}\ \mathrm{x} \\
 & \mid & b := 0 \mid b := 1 \mid q_1, q_2, \cdots, q_n * = S \\
 & \mid & \textbf{skip} \mid P;Q \\
 & \mid & \textbf{if}\ b\ \textbf{then}\ P\ \textbf{else}\ Q \mid \textbf{measure}\ q\ \textbf{then}\ P\ \textbf{else}\ Q \mid \textbf{while}\ b\ \textbf{do}\ P \\
 & \mid & \textbf{proc}\ X : \Gamma \to \Gamma'\ \{P\}\ \textbf{in}\ Q \mid y_1, y_2, \cdots, y_m = X(x_1, x_2, \cdots, x_n)
\end{array}
$$

Η σημασία των παραπάνω όρων αναλύεται στην επόμενη υποενότητα.

3.2.2 Λειτουργία

Το μοντέλο λειτουργίας της QPL είναι ουσιαστικά αυτό ενός κλασικού υπολογιστή ο οποίος έχει πρόσβαση σε κβαντική μνήμη. Ο έλεγχος ροής του προγράμματος είναι παραδοσιακός (μάλιστα ο Selinger δίνει μια εναλλακτική 'παιδαγωγική' μορφή της σύνταξης η οποία βασίζεται σε

διαγράμματα ροής). Κατά τη διάρκεια της εκτέλεσής του ένα πρόγραμμα έχει τη δυνατότητα να ζητήσει από το 'λειτουργικό' να του παραχωρηθεί ένα επιπλέον qbit. Επίσης έχει τη δυνατότητα να ελευθερώσει ένα qbit που δεν χρειάζεται πλέον (εντολές new και discard). Από άποψη χειρισμού των qbits έχει τη δυνατότητα να εφαρμόσει έναν ορθομοναδιαίο μετασχηματισμό ή να μετρήσει κάποιο qbit και να προβεί σε διαφορετικές ενέργειες, ανάλογα με το αποτέλεσμα της μέτρησης. Ο χειρισμός των κλασικών bit δεν εμφανίζει διαφορά σε σχέση με ότι θα περιμέναμε από μια κλασική γλώσσα, όμως η ύπαρξη του κβαντικού μέρους δυσκολεύει τον ορισμό της σημασιολογίας.

Κβαντικό μέρος

Για λόγους ευκολίας, θα ασχοληθούμε πρώτα με το κβαντικό μέρος της γλώσσας, το οποίο είναι και το πιο ενδιαφέρον. Παρακάτω εξηγούμε πώς επεκτείνεται η σημασιολογία με την προσθήκη κλασικών bits.

Η σημασία κάθε προγράμματος είναι μία συνάρτηση $D_{2^n} \to D_{2^m}$, αν το πρόγραμμα δέχεται σαν είσοδο n qbit, δίνει έξοδο m qbit και D_n το σύνολο των πινάκων πυκνότητας διαστάσεων $n \times n$. Θα δώσουμε τη σημασία οποιουδήποτε προγράμματος επαγωγικά, ορίζοντας τη σημασία κάθε όρου της γραμματικής που δώσαμε παραπάνω. Τα αποτελέσματα φαίνονται στον πίνακα 3.1 όπου με $\|P\|$ συμβολίζουμε τη σημασία που αποδίδουμε στον όρο P.

Πίνακας 3.1: Σημασιολογία QPL κβαντικό μέρος.

Όρος	Πεδίο ορισμού	Συνάρτηση
new qbit	$D_n \to D_{2n}$	$A \to \left(\begin{array}{c\|c} A & 0 \\ \hline 0 & 0 \end{array}\right)$
discard	$D_n \to D_{n/2},\ n > 1$	$\left(\begin{array}{c\|c} A & B \\ \hline C & D \end{array}\right) \to A + D$
$q_1, q_2, \cdots, q_n* = S$	$D_{2^n} \to D_{2^n}$	$A \to SAS^*$
measure q **then** P **else** Q	$D_n \to D_m$	$\left(\begin{array}{c\|c} A & B \\ \hline C & D \end{array}\right) \to \|P\|(\left(\begin{array}{c\|c} A & 0 \\ \hline 0 & 0 \end{array}\right)) +$
	με $\|P\|, \|Q\| : D_n \to D_m$	$+\|Q\|(\left(\begin{array}{c\|c} 0 & 0 \\ \hline 0 & D \end{array}\right))$
skip	$D_n \to D_n$	$A \to A$
proc X: $\{P\}$ **in** Q	ίδιο με της $\|Q\|$	$\|Q\|$ με $\|\text{call } X\| := \|P\|$
$y_1, \cdots, y_m = X(x_1, \cdots, x_n)$	$D_{2^n} \to D_{2^m}$	Προκύπτει από προηγούμενο **proc**
$P; Q$	$D_n \to D_m$	$\|Q\|(\|P\|)$ με $\|Q\| : D_k \to D_m$, $\|P\| : D_n \to D_k$

Η σημασιολογία αυτή με απλά λόγια μπορεί να οριστεί ως εξής:

- Η δέσμευση ενός νέου qbit γίνεται πάντα έτσι ώστε το νέο qbit να είναι το πιο σημαντικό του καταχωρητή της συνολικής κατάστασης. Έτσι το νέο state είναι $|\mathbf{0}\rangle \otimes s_p$ όπου s_p το προηγούμενο.

- Η απελευθέρωση ενός qbit μπορεί εύκολα να οριστεί όταν απελευθερώνουμε το πιο σημαντικό qbit. Είναι ισοδύναμη με τη μέτρησή του και την απώλεια του αποτελέσματος. Ο λόγος που γίνεται αυτό είναι ότι το bit που απελευθερώνουμε μπορεί να είναι entangled με κάποιο άλλο το οποίο κρατάμε και μια πιθανή μέτρησή του να επιφέρει αλλαγές και σε αυτό. Αν το απελευθερώσουμε χωρίς να προκαλέσουμε άμεσα αυτές τις αλλαγές είναι αδύνατο να προβλέψουμε πότε θα προκληθούν κι έτσι δεν μπορούμε να δώσουμε μια σωστή σημασιολογία. Σε περίπτωση που θέλουμε να απελευθερώσουμε κάποιο άλλο qbit (όπως άλλωστε επιτρέπει η σύνταξη) μπορούμε να κάνουμε μία αλλαγή βάσης ώστε το ζητούμενο qbit να γίνει το πιο σημαντικό, και μετά το discard μία δεύτερη αλλαγή βάσης ώστε το qbit το οποίο ήταν αρχικώς πιο σημαντικό να επιστρέψει στη θέση του.

- Η μέτρηση ενός qbit μας δίνει την δυνατότητα να εκτελέσουμε ένα μέρος του προγράμματος υπό συνθήκη. Η σημασιολογία ορίζεται μόνο σε περίπτωση που τα new και discard είναι ισορροπημένα στα δύο branches του measure. Παρατηρούμε πως η εντολή measure q_0 then skip else skip έχει την ίδια σημασιολογία με το discard , όπως είδαμε και παραπάνω.

- Η χρήση ενός ορθομοναδιαίου μετασχηματισμού έχει την ιδιαιτερότητα πως μπορεί να γίνει σε ένα αυθαίρετο σύνολο από qbit. Για να χρησιμοποιήσουμε όσα ξέρουμε ήδη για τους μετασχηματισμούς θα πρέπει να κάνουμε κατάλληλες αλλαγές βάσεις, ενώ αν οι τελεστέοι είναι λιγότεροι από το πλήθος των qbits που αποτελούν το state ο S θα πρέπει να αντικατασταθεί φυσικά από το τανυστικό γινόμενό του με τον αντίστοιχο μοναδιαίο.

Ιδιαίτερο ενδιαφέρον παρουσιάζει η χρήση procedures και μάλιστα αναδρομικών. Η χρήση μιας διαδικασίας η οποία δεν είναι αναδρομική,έμμεσα ή άμεσα, μπορεί ουσιαστικά να αντικατασταθεί στο σημείο όπου χρησιμοποιείται, σαν μακροεντολή. Σε περίπτωση όμως που έχουμε αναδρομή θα ακολουθήσουμε την εξής διαδικασία. Έστω X η αναδρομική διαδικασία. Τότε θα λέμε $X(Y)$ την διαδικασία που είναι ίδια με την X μόνο που η αναδρομική κλήση έχει αντικατασταθεί από τον όρο Y. Έστω Y_0 ένα πρόγραμμα που δεν τερματίζεται. Επίσης έστω $Y_{i+1} = X(Y_i)$. Τότε χρησιμοποιώντας το γεγονός ότι $\|Y_0\| = 0$ υπολογίζουμε το όριο $\lim_{i\to\infty} \|Y_i\|$, το οποίο είναι και το $\|X\|$. Είναι πιθανό το trace του πίνακα που δίνεται σαν είσοδο σε μία αναδρομική διαδικασία να είναι μεγαλύτερο από αυτό που προκύπτει σαν έξοδος. Δηλαδή η χρήση αναδρομικών συναρτήσεων είναι το μόνο κομμάτι της γλώσσας που δεν διατηρεί αναγκαστικά ανέπαφο το trace (οι ορθομοναδιαίοι μετασχηματισμοί το διατηρούν λόγω της αντίστοιχης ιδιότητας των πινάκων όπως το διατηρούν και οι μετρήσεις). Αυτό απεικονίζει το γεγονός πως μπορούμε να έχουμε και προγράμματα τα οποία δεν τερματίζουν ποτέ.

Κλασικό μέρος

Η ύπαρξη ενός κλασικού μέρους στη γλώσσα QPL αντιμετωπίζεται ως εξής: Η σημασία των προγραμμάτων δεν δίνεται πλέον με συναρτήσεις από το D_n στο D_m αλλά με συναρτήσεις από το $D_{n_1} \times D_{n_2} \times \cdots \times D_{n_a}$ στο $D_{m_1} \times D_{m_2} \times \cdots \times D_{m_b}$. Αντί λοιπόν για πίνακες πυκνότητας έχουμε πλέον διατεταγμένα σύνολα πινάκων πυκνότητας.

Τα σύνολα αυτά μεγαλώνουν με την εντολή δέσμευσης νέου bit και μικραίνουν με την εντολή απελευθέρωσής του. Αν σε κάποια στιγμή το πρόγραμμα έχει δεσμεύσει n bit θα έχει ένα σύνολο από 2^n πίνακες πυκνότητας. Με άλλα λόγια θα έχει έναν πίνακα πυκνότητας για κάθε πιθανή τιμή που μπορεί να πάρουν τα n bit.

Η $\|$new bit$\|$ λοιπόν είναι μία συνάρτηση που παίρνει ένα διατεταγμένο σύνολο $(A_1, \cdots, A_n)$ και δίνει σαν αποτέλεσμα $(A_1, \cdots, A_n, 0, \cdots, 0)$ ($2n$ στοιχεία στο αποτέλεσμα). Η $\|$discard bit$\|$ παίρνει το $(A_1, A_2, \cdots, A_{2n})$ και δίνει σαν αποτέλεσμα το $(A_1+A_{n+1}, A_2+A_{n+2}, \cdots, A_n+A_{2n})$. Όταν αναφέρεται στο πιο σημαντικό bit η $\|b := 1\|$ είναι η $(A_1, A_2, \cdots, A_{2n}) \to (0, 0, \cdots, 0, A_1 + A_{n+1}, A_2 + A_{n+2}, \cdots, A_n + A_{2n})$, ενώ αντίστοιχα η $\|b := 0\|$ είναι η $(A_1, A_2, \cdots, A_{2n}) \to (A_1 + A_{n+1}, , A_2 + A_{n+2}, \cdots, A_n + A_{2n}, 0, 0, \cdots, 0)$. Για άλλα bit οι αντίστοιχες συναρτήσεις μπορούν εύκολα να προκύψουν.

Τέλος για τον ορισμό της σημασιολογίας του loop ακολουθούμε μία διαδικασία αντίστοιχη με αυτή που ακολουθούμε για να ορίσουμε τη σημασιολογία των αναδρομικών προγραμμάτων, υπολογίζοντας τελικά ένα όριο.

3.2.3 Παραδείγματα

Ρίψη κέρματος

Έστω πως θέλουμε να εκτελέσουμε ένα από δύο τμήματα ενός προγράμματος με πιθανότητα 50%. Έστω P και Q οι αντίστοιχοι όροι των προγραμμάτων αυτών. Ένα πρόγραμμα που πετυχαίνει αυτό είναι

```
new qbit q
q* = H
measure q   then
                discard q ; P
            else
                discard q ; Q
```

Πράγματι, αν A ο πίνακας πυκνότητας που έχει σαν είσοδο το παραπάνω πρόγραμμα. Η προσθήκη ενός qbit μας δίνει τον πίνακα $\left(\begin{array}{c|c} A & 0 \\ \hline 0 & 0 \end{array}\right)$. Η εφαρμογή του H στο νέο qbit ισοδυναμεί με εφαρμογή του $H \otimes I = \frac{1}{\sqrt{2}} \left(\begin{array}{c|c} I & I \\ \hline I & -I \end{array}\right)$ σε όλο το state. Επομένως το νέο state είναι

$$\frac{1}{2} \left(\begin{array}{c|c} A & A \\ \hline A & A \end{array}\right)$$

Το measure δίνει σαν είσοδο στο ένα branch $\frac{1}{2} \left(\begin{array}{c|c} A & 0 \\ \hline 0 & 0 \end{array}\right)$ και στο άλλο $\frac{1}{2} \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & A \end{array}\right)$. Η εφαρμογή όμως του discard καταλήγει στην εκτέλεση του P ή του Q με είσοδο το $\frac{1}{2}A$. Επομένως καταφέραμε το ζητούμενο που είναι η δημιουργία ενός προγράμματος που δίνει έξοδο $P(\frac{1}{2}A) + Q(\frac{1}{2}A) = \frac{1}{2}(P(A) + Q(A))$. Το τελευταίο προκύπτει από τη γραμμικότητα της σημασιολογίας της γλώσσας μας.

Ρίψη κέρματος ή μέτρηση

Ας συνεχίσουμε το προηγούμενο παράδειγμα. Έστω πως $Q :=$ **skip** και $P := \quad q_0* = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. Ας υποθέσουμε πως το πρόγραμμα έχει είσοδο $\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$. Τότε έχω

$$P(\frac{1}{2} \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)) = \frac{1}{2} \left(\begin{array}{c|c} A & -B \\ \hline -C & D \end{array}\right)$$

Επομένως το αποτέλεσμα όλου του προγράμματος είναι

$$P(\frac{1}{2} \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)) + Q(\frac{1}{2} \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)) = \left(\begin{array}{c|c} A & 0 \\ \hline 0 & D \end{array}\right)$$

Το αποτέλεσμα αυτό όμως είναι το ίδιο με το αποτέλεσμα μίας μέτρησης πάνω στο πρώτο qbit του state!

Για να καταλάβουμε καλύτερα τη διαφορά ανάμεσα στο πρόγραμμά μας και τη διενέργεια κατευθείαν μιας μέτρησης ας πάρουμε σαν παράδειγμα την είσοδο $\frac{3}{5}|\mathbf{0}\rangle + \frac{4}{5}|\mathbf{1}\rangle$. Το πρόγραμμά μας, με πιθανότητα $\frac{1}{2}$ αφήνει την είσοδο ανέπαφη ή πολλαπλασιάζει με -1 τον συντελεστή του $|\mathbf{1}\rangle$. Οδηγεί λοιπόν στη μικτή κατάσταση $\frac{1}{2}\{\frac{3}{5}|\mathbf{0}\rangle + \frac{4}{5}|\mathbf{1}\rangle\} + \frac{1}{2}\{\frac{3}{5}|\mathbf{0}\rangle - \frac{4}{5}|\mathbf{1}\rangle\}$. Αντιθέτως μία μέτρηση οδηγεί στη μικτή κατάσταση $\frac{9}{25}\{|\mathbf{0}\rangle\} + \frac{16}{25}\{|\mathbf{1}\rangle\}$.

Οι δύο αυτές καταστάσεις όμως έχουν τον ίδιο πίνακα πυκνότητας. Από μία άποψη αυτό είναι θετικό γιατί τα δύο προγράμματα είναι όντως ισοδύναμα με την έννοια ότι θα συμπεριφερθούν με τον ίδιο ακριβώς τρόπο ως μέρη ενός μεγαλύτερου προγράμματος. Ας μην ξεχνάμε όμως ότι μία μικτή κατάσταση είναι απλώς μία αναπαράσταση της γνώσης κάποιου παρατηρητή για ένα σύστημα. Έτσι στην περίπτωση του προγράμματός μας, αν ένας παρατηρητής γνωρίζει το αποτέλεσμα της μέτρησης είναι σε θέση, χρησιμοποιώντας έναν αντίστροφο μετασχηματισμό να ξαναφέρει το state εκεί που ήταν πριν την εκτέλεση του προγράμματος. Αντιθέτως στη δεύτερη περίπτωση η κατάσταση που υπήρχε πριν τη μέτρηση έχει χαθεί.

Η συζήτηση αυτή, παρ'οτι πολύ ενδιαφέρουσα είναι έξω από τους σκοπούς αυτής της εργασίας αφού αφορά το πεδίο της κβαντικής κρυπτογραφίας. Στο εξής θα μελετάμε προγράμματα

χωρίς να μας απασχολεί η ύπαρξη τρίτων παρατηρητών και θα θεωρούμε mixed states με τον ίδιο πίνακα πυκνότητας ισοδύναμες.

3.3 Η γλώσσα QML

Στο [Grat05b] ο Thorsten Altenkirch και ο Jonathan Grattage παρουσιάζουν μία συναρτησιακή γλώσσα κβαντικού προγραμματισμού. Η QML εισάγει πέρα από κβαντικές δομές δεδομένων και κβαντικές δομές ελέγχου. Βασίζεται στην αυστηρή γραμμική λογική, επιτηρώντας προσεκτικά τα μέρη της γλώσσας που υπονοούν τη διενέργεια μέτρησης. Η λειτουργική σημασιολογία της δίνεται με κβαντικά κυκλώματα.

Η γλώσσα QML αποτελεί ένα πολύ ουσιαστικό πρώτο βήμα προς την κατεύθυνση της δημιουργίας μιας κβαντικής γλώσσας ωψηλού επιπέδου. Επιχειρεί να ενσωματώσει έννοιες εγγενείς στο κβαντικό μοντέλο, όπως η υπέρθεση και οι ελεγχόμενοι μετασχηματισμοί, με τρόπο τέτοιο ώστε ο προγραμματιστής να διευκολύνεται στη χρήση τους. Ταυτόχρονα επιχειρεί να διατηρήσει ένα συναρτησιακό στυλ, ξεχωρίζοντας μέσα στο σύστημα τύπων τα μέρη της γλώσσας που έχουν παρενέργειες (δηλαδή τις μετρήσεις).

Παρακάτω παρουσιάζουμε το συντακτικό και το σύστημα τύπων της QML, και δίνουμε μια σύντομη περιγραφή της σημασιολογίας της με κυκλώματα. Έπειτα, βασιζόμενοι πάνω στις ιδέες και το συντακτικό της QML δημιουργούμε μια νέα γλώσσα κβαντικού προγραμματισμού την NTUA-QML. Σκιαγραφούμε τη σημασιολογία της στο μοντέλο κβαντικών υπολογισμών που έχουμε ήδη αναπτύξει, και δίνουμε μία αναλυτική περιγραφή της στο κεφάλαιο 4 το οποίο ασχολείται με την υλοποίησή της.

3.3.1 Συντακτικό

Η γραμματική των όρων της QML είναι:

$$
\begin{array}{rcl}
t & ::= & x \mid \textbf{let}\ x = t\ \textbf{in}\ u \\
 & \mid & x^{\vec{y}} \mid () \\
 & \mid & (t, u) \mid \textbf{let}\ (x, y) = t\ \textbf{in}\ u \\
 & \mid & \textbf{qinl}\ t \mid \textbf{qinr}\ t \\
 & \mid & \textbf{case}\ t\ \textbf{of}\ \{\textbf{qinl}\ x \Rightarrow u \mid \textbf{qinr}\ y \Rightarrow u'\} \\
 & \mid & \textbf{case}^{o}\ t\ \textbf{of}\ \{\textbf{qinl}\ x \Rightarrow u \mid \textbf{qinr}\ y \Rightarrow u'\} \\
 & \mid & \{(\kappa)t \mid (\iota)u\} \\
 & \mid & f\ \vec{t}
\end{array}
$$

όπου τα διανύσματα $\vec{a}$ αντιπροσωπεύουν μία ακολουθία από συντακτικά αντικείμενα δηλαδή

$$\vec{\alpha} = \epsilon \mid \alpha\vec{\alpha}$$

3.3.2 Σύστημα τύπων

Το σύστημα τύπων της QML έχει την πρωτοτυπία ότι αποτελείται ουσιαστικά από δύο διαφορετικά συστήματα τύπων: ένα αυστηρό και ένα γενικό. Όταν ένα πρόγραμμα d κάνει typecheck αυστηρά στο context Γ και έχει τύπο τ θα γράφουμε $\Gamma \vdash^{o} d : \tau$ ενώ όταν κάνει απλώς typecheck θα γράφουμε $\Gamma \vdash d : \tau$. Το context Γ είναι μία δομή η οποία αντιστοιχεί σε κάθε όνομα μεταβλητής έναν τύπο.

Εισάγουμε τώρα τον τελεστή $\otimes$, ο οποίος αντιστοιχεί ζευγάρια από context σε context:

$$
\begin{array}{rcl}
\Gamma, x : \sigma \otimes \Delta, x : \sigma & = & (\Gamma \otimes \Delta), x : \sigma \\
\Gamma, x : \sigma \otimes \Delta & = & (\Gamma \otimes \Delta), x : \sigma \quad \text{Αν } x \notin \text{ dom } \Delta \\
\bullet \otimes \Delta & = & \Delta
\end{array}
$$

όπου με $\bullet$ συμβολίζουμε το κενό context.

Η γραμματική των τύπων της γλώσσας μας είναι

$$\tau \ ::= \ Q_1 \ | \ \tau \otimes \sigma \ | \ \tau \oplus \sigma$$

Δηλαδή η QML υποστηρίζει ένα μόνο θεμελιώδη τύπο (ο οποίος όπως θα δούμε δεν μπορεί να κρατήσει καθόλου πληροφορία), γινόμενα και αθροισματα τύπων. Ο τύπος του Qbit δίνεται ως $Q_1 \oplus Q_1$, ενώ δεν υποστηρίζονται κλασικά bit.

Ας δούμε τώρα τους κανόνες τύπων:

$$\frac{\Gamma \vdash^o t : \sigma}{\Gamma \vdash t : \sigma}\text{emb}$$

$$\frac{}{x : \sigma \vdash^o x : \sigma}\text{var} \qquad \frac{\Gamma \vdash t : \sigma}{\Gamma \otimes \Delta \vdash t^{dom\ \Delta} : \sigma}\text{weak}$$

Ο κανόνας emb μας δείχνει ότι όταν ένα πρόγραμμα κάνει typecheck αυστηρά, κάνει typecheck. Ο κανόνας var μας επιτρέπει να χρησιμοποιήσουμε μία μεταβλητή που βρίσκεται στο context. Ο κανόνας weak επιτρέπει την αποδυνάμωση, δηλαδή την απελευθέρωση των μεταβλητών του context Δ.

$$\frac{\begin{array}{c}\Gamma \vdash^\alpha t : \sigma \\ \Delta, x : \sigma \vdash^\alpha u : \tau\end{array}}{\Gamma \otimes \Delta \vdash^\alpha \textbf{let}\ x = t\ \textbf{in}\ u : \tau}\text{let}$$

Χρησιμοποιούμε $\vdash^\alpha$ όταν ο ίδιος κανόνας ισχύει για το $\vdash$ και το $\vdash^o$ για λόγους συντομίας. Χρησιμοποιώντας του κανόνες των let και emb συμπεραίνουμε πως

$$\frac{\begin{array}{c}\Gamma \vdash^\alpha t : \sigma \\ \Delta, x : \sigma \vdash^\beta u : \tau\end{array}}{\Gamma \otimes \Delta \vdash^{\alpha \sqcap \beta} \textbf{let}\ x = t\ \textbf{in}\ u : \tau}$$

όπου μόνο $o \sqcap o = o$.

Η αποδυνάμωση μπορεί να επηρεάσει το νόημα ενός προγράμματος. Για παράδειγμα το

$$y : Q_2 \vdash \textbf{let}\ x = y\ \textbf{in}\ x^{\{\}} : Q_2$$

έχει διαφορετικό νόημα από το

$$y : Q_2 \vdash \textbf{let}\ x = y\ \textbf{in}\ x^{\{y\}} : Q_2$$

διότι το πρώτο θα ερμηνευθεί σαν το ταυτοτικό κύκλωμα ενώ το δεύτερο υπονοεί μέτρηση του y η οποία θα γίνει κατά την απελευθέρωσή του.

Γινόμενα

Οι κανόνες για τα Q_1 και $\otimes$ είναι κλασικοί:

$$\frac{}{\bullet \vdash^o () : Q_1}Q_1\text{−intro} \qquad \frac{\Gamma, x : Q_1 \vdash^\alpha t : \sigma}{\Gamma \vdash^\alpha t : \sigma}Q_1\text{−weak}$$

Σημειώνουμε πως ο κανόνας του Q_1−weak διατηρεί την αυστηρότητα αφού δεν υπάρχει απώλεια πληροφορίας.

Το γινόμενο εισάγεται ως εξής:

$$\frac{\Gamma \vdash^\alpha t : \sigma \quad \Delta \vdash^\alpha u : \tau}{\Gamma \otimes \Delta \vdash^\alpha (t, u) : \sigma \otimes \tau} \otimes\text{−intro}$$

Η εξουδετέρωση του γινομένου γίνεται με το συνηθισμένο pattern-matching let:

$$\frac{\begin{array}{c}\Gamma \vdash^{\alpha} t : \sigma \otimes \tau \\ \Delta, x : \sigma, y : \tau \vdash^{\alpha} u : \rho\end{array}}{\Gamma \otimes \Delta \vdash^{\alpha} \textbf{let } (x, y) = t \textbf{ in } u : \rho} \otimes -\text{elim}$$

Ας δούμε και πάλι ένα παράδειγμα:

$$p : Q_2 \otimes Q_2 \vdash \textbf{let } (x, y) = p \textbf{ in } (y^{\{\}}, x^{\{\}}) : Q_2 \otimes Q_2$$

και το αντίστοιχο

$$p : Q_2 \otimes Q_2 \vdash \textbf{let } (x, y) = p \textbf{ in } (y^{\{p\}}, x^{\{p\}}) : Q_2 \otimes Q_2$$

Το πρώτο απλώς αλλάζει θέση στα δύο qbit που αποτελούν το p, ενώ το δεύτερο τα μετράει προτού τους αλλάξει θέση.

Αθροίσματα

Η εισαγωγή του $\oplus$ γίνεται επίσης με τους κλασικούς κανόνες:

$$\frac{\Gamma \vdash^{\alpha} s : \sigma}{\Gamma \vdash^{\alpha} \textbf{inl } s : \sigma \oplus \tau} \oplus -\text{intro}_1 \qquad \frac{\Gamma \vdash^{\alpha} t : \tau}{\Gamma \vdash^{\alpha} \textbf{inr } t : \sigma \oplus \tau} \oplus -\text{intro}_2$$

Ορίζουμε τα qbit $Q_2 = Q_1 \oplus Q_1$ και χρησιμοποιούμε τις συντομογραφίες $\text{qtrue} = \text{qinl } ()$ και $\text{qfalse} = \text{qinr } ()$.

Έχουμε δύο κανόνες εξουδετέρωσης του $\oplus$. Ας εξετάσουμε πρώτα αυτόν που προκαλεί μέτρηση:

$$\frac{\begin{array}{c}\Gamma \vdash c : \sigma \oplus \tau \\ \Delta, x : \sigma \vdash t : \rho \\ \Delta, y : \tau \vdash u : \rho\end{array}}{\Gamma \otimes \Delta \vdash \textbf{case } c \textbf{ of } \{\textbf{inl } x \Rightarrow t \mid \textbf{inr } y \Rightarrow u\} : \rho} \oplus -\text{elim}$$

Μπορούμε να εξάγουμε από αυτόν το if-then-else ως

$$\textbf{if } b \textbf{ then } t \textbf{ else } u = \textbf{case } b \textbf{ of } \{\textbf{inl } _ \Rightarrow t \mid \textbf{inr } _ \Rightarrow u\}$$

Η έκδοση αυτή του if θα μετρήσει το b προκαλώντας έτσι πιθανώς ανεπιθύμητες παρενέργειες στο σύστημά μας. Εισάγουμε τώρα μία δεύτερη μορφή του κανόνα $\oplus - \text{elim}$ η οποία δεν έχει παρενέργειες. Θα χρειαστούμε γι' αυτό την έννοια της ορθογωνιότητας (orthogonality) την οποία ορίζουμε τυπικά σε έναν από τους επόμενους κανόνες.

$$\frac{\begin{array}{c}\Gamma \vdash^{\alpha} c : \sigma \oplus \tau \\ \Delta, x : \sigma \vdash^{o} t : \rho \\ \Delta, y : \tau \vdash^{o} u : \rho \qquad t \perp u\end{array}}{\Gamma \otimes \Delta \vdash^{\alpha} \textbf{case}^{o} \; c \textbf{ of } \{\textbf{inl } x \Rightarrow t \mid \textbf{inr } y \Rightarrow u\} : \rho} \oplus -\text{elim}^{o}$$

Υπερθέσεις

Η QML περιέχει μία δομή για δημιουργία υπερθέσεων ανάμεσα σε οποιεσδήποτε δύο ορθογώνιες καταστάσεις ίδιου τύπου. Ο αντίστοιχος κανόνας είναι:

$$\frac{\begin{array}{c}\Gamma \vdash^{o} t, u : \sigma \qquad t \perp u \\ |\lambda|^2 + |\lambda'|^2 = 1\end{array}}{\Gamma \vdash^{o} \{(\lambda)t \mid (\lambda')u\} : \sigma} \text{sup}$$

Πολύ συχνά θα παραλείπουμε τους παράγοντες κανονικοποίησης αν αυτοί μπορούν εύκολα να υπολογιστούν. Για παράδειγμα γράφουμε $\{\text{qtrue} \mid \text{qfalse}\}$ αντί για $\{\frac{1}{\sqrt{2}}\text{qtrue} \mid \frac{1}{\sqrt{2}}\text{qfalse}\}$.

Ορθογωνιότητα

Η ορθογωνιότητα δύο όρων σημαίνει πως υπάρχει μία μέτρηση η οποία μπορεί να τους ξεχωρίσει. Πιο αναλυτική συζήτηση περί ορθογωνιότητας θα ακολουθήσει στον ορισμό της NTUA-QML. Μέχρι τότε θα αρκεστούμε στους εξής κανόνες:

$$\frac{\Gamma \vdash^o t : \sigma \qquad \Gamma \vdash^o u : \tau}{\textbf{inl}\ t \perp \textbf{inr}\ u \qquad \textbf{inr}\ t \perp \textbf{inl}\ u}\text{Oinlr, Oinrl}$$

Σημειώνουμε ότι από τον παραπάνω κανόνα προκύπτει και το πολύ σημαντικό συμπέρασμα $\mathrm{qtrue} \perp \mathrm{qfalse}$.

Οι constructors διατηρούν την ορθογωνιότητα:

$$\frac{t \perp u}{\textbf{inl}\ t \perp \textbf{inl}\ u \qquad \textbf{inr}\ t \perp \textbf{inr}\ u}\text{Oinl, Oinr}$$

$$\frac{t \perp u}{(t, v) \perp (u, w) \qquad (v, t) \perp (w, u)}\text{Opairl, Opairr}$$

Τέλος οι υπερθέσεις μπορούν να είναι ορθογώνιες:

$$\frac{t \perp u \qquad \lambda_0^* \kappa_0 + \lambda_1^* k_1 = 0}{\{(\lambda_0)t \mid (\lambda_1)u\} \perp \{(\kappa_0)t \mid (\kappa_1)u\}}\text{Osup}$$

3.3.3 Σημασιολογία

Η σημασιολογία της QML δίνεται στο [Grat05a] με όρους κβαντικών κυκλωμάτων. Σε κάθε όρο αντιστοιχίζεται ένα κβαντικό κύκλωμα το οποίο σχηματίζεται επαγωγικά συνδέοντας κατάλληλα τα κυκλώματα που αντιστοιχούν στους υποόρους του.

Το πλήθος των εισόδων και εξόδων ενός κυκλώματος μπορεί να καθοριστεί από τον τύπο του όρου από τον οποίο προέρχεται καθώς και τους τύπους των υποόρων του. Ορίζουμε λοιπόν για το σύστημα τύπων την πράξη $|\tau|$ η οποία για κάθε τύπων μας δίνει το αντίστοιχο πλήθος. Έχουμε: $|Q_1| = 0$, $|\tau \otimes \sigma| = |\tau| + |\sigma|$, $|\tau \oplus \sigma| = (|\tau| \sqcup |\sigma|) + 1$ όπου με $\sqcup$ συμβολίζουμε το μέγιστο δύο αριθμών. Έτσι ένας όρος που έχει τύπο Q_2 θα μας δίνει ένα κύκλωμα με $|Q_2| = |Q_1 \oplus Q_1| = |Q_1| + 1 = 1$ έξοδο.

Τα στοιχειώδη κυκλώματα, δηλαδή οι στοιχειώδεις αντιστρέψιμοι κβαντικοί υπολογισμοί που θα χρησιμοποιήσουμε είναι:

- Περιστροφή: Η περιστροφή είναι ένας υπολογισμός μίας εισόδου και μίας εξόδου που χαρακτηρίζεται από τον ορθομοναδιαίο πίνακα

  $$\begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}$$

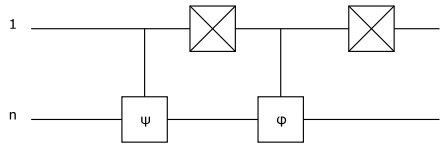
  Σημειώνουμε πως και η άρνηση είναι μία ειδική περίπτωση περιστροφής με $u_{00} = u_{11} = 0$ και $u_{10} = u_{01} = 1$.

- Αναδιάταξη: Ένας κβαντικός υπολογισμός n εισόδων και εξόδων που αντιπροσωπεύει οποιαδήποτε μετάθεση των εισόδων του

- Διαδοχική σύνθεση: Χρησιμοποιώντας δύο κυκλώματα με τον ίδιο αριθμό εισόδων και εξόδων σχηματίζουμε ένα κύκλωμα όπου η έξοδος του πρώτου είναι είσοδος του δεύτερου.

- Παράλληλη σύνθεση: Χρησιμοποιώντας δύο κυκλώματα, το πρώτο με n εισόδους και εξόδους και το δεύτερο με m σχηματίζουμε ένα κύκλωμα με $n + m$ εισόδους και εξόδους, όπου οι πρώτες n είσοδοι πάνε στο πρώτο συστατικό κύκλωμα και οι υπόλοιπες στο άλλο.

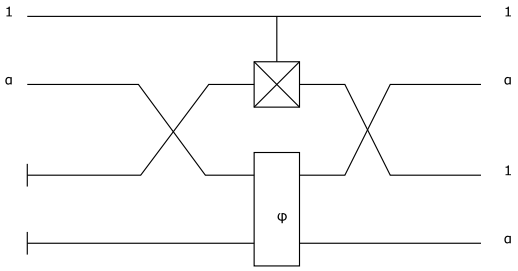
- Συνθήκη: Δοθέντων δύο κυκλωμάτων ϕ, ψ ίδιου πλήθος εισόδων και εξόδων n σχεδιάζουμε ένα κύκλωμα με $n + 1$ εισόδους και εξόδους όπως φαίνεται στο σχήμα 3.1.



Σχήμα 3.1: Κύκλωμα επιλογής υπό συνθήκη.

Όλα τα παραπάνω κυκλώματα έχουν το ίδιο πλήθος εισόδων και εξόδων λόγω της αντιστρεψιμότητάς τους. Μπορούμε από αυτά να δημιουργήσουμε μη αντιστρέψιμους υπολογισμούς διαφορετικού πλήθους εισόδων και εξόδων θεωρώντας πως ένα μέρος των εισόδων είναι αρχικοποιημένες στην τιμή qfalse και πως ένα μέρος των εξόδων είναι σκουπίδια. Οι είσοδοι που ξεκινάνε αρχικοποιημένες θα συμβολίζονται με $\vdash$ και τα σκουπίδια θα καταλήγουν σε $\dashv$ στα κυκλώματά μας.

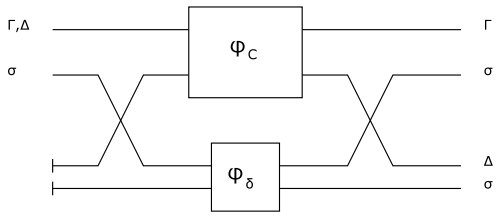
Ας ορίσουμε τώρα κάποια βοηθητικά κυκλώματα τα οποία θα μας βοηθήσουν να ξεχωρίσουμε τα context που απαιτούνται από κάθε όρο. Στο σχήμα 3.2 φαίνεται ο επαγωγικός ορισμός ενός κυκλώματος $\phi_\delta\ \alpha$ για κάθε α. Συγκεκριμένα για $\alpha = 0$ το κύκλωμα γίνεται μία ελεγχόμενη πύλη not. Για $\alpha > 0$ το κύκλωμα χρησιμοποιεί το $\phi_\delta\ \alpha$ για το σχηματισμό του $\phi_\delta(\alpha + 1)$.



Σχήμα 3.2: Κύκλωμα ϕ_δ .

Ο ρόλος του κυκλώματος ϕ_δ είναι το μοίρασμα των μεταβλητών ανάμεσα στα context. Έστω πως έχουμε δύο όρους οι οποίοι μοιράζονται μία μεταβλητή. Είναι γνωστό πως σε ένα κβαντικό κύκλωμα δεν μπορούμε να κάνουμε διακλάδωση και να δώσουμε την ίδια είσοδο στα κυκλώματα των δύο όρων. Το καλύτερο που μπορούμε να κάνουμε είναι να χρησιμοποιήσουμε την πύλη cnot ώστε να δώσουμε στα δύο κυκλώματα ένα entangled ζευγάρι από qbit όπου τα δύο qbit έχουν τις ίδιες πιθανότητες μετά από μία μέτρηση να δώσουν αποτέλεσμα $|\mathbf{0}\rangle$ ή $|\mathbf{1}\rangle$.

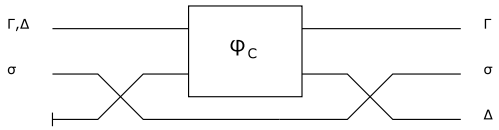
Χρησιμοποιώντας το κύκλωμα ϕ_δ κατασκευάζουμε το κύκλωμα ϕ_C στην περίπτωση που δύο context Γ, Δ μοιράζονται την ίδια μεταβλητή. Το κύκλωμα φαίνεται στο σχήμα 3.3.



Σχήμα 3.3: Κύκλωμα ϕ_C για context που μοιράζονται μεταβλητές.

Το κύκλωμα αυτό ορίζεται επίσης επαγωγικά. Σε περίπτωση που τα Γ, Δ μοιράζονται μία μεταβλητή $x : \sigma$ χρησιμοποιούμε το ϕ_δ για να μοιράσουμε δύο φορές αυτή τη μεταβλητή και το ϕ_C στις υπόλοιπες μεταβλητές τους. Σε περίπτωση που τα Γ, Δ δεν μοιράζονται μεταβλητή το ϕ_C έχει τη μορφή του σχήματος 3.4, ενώ σε περίπτωση που το ένα είναι το $\bullet$ το ϕ_C είναι απλώς το ταυτοτικό κύκλωμα, δηλαδή δίνει απλώς το άλλο.

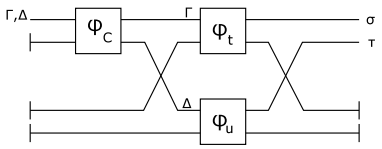
Με βάση τα παραπάνω η ερμηνεία του weak είναι απλή: περνάμε τα context Γ, Δ από το



Σχήμα 3.4: Κύκλωμα ϕ_C για context που δεν μοιράζονται μεταβλητή.

ϕ_C και στέλνουμε την έξοδο που αντιστοιχεί στο Δ στα σκουπίδια.

Ας δούμε τώρα τους πιο ενδιαφέροντες σημασιολογικούς κανόνες της γλώσσας. Ξεκινάμε με τα γινόμενα, όπως φαίνονται στο σχήμα 3.5

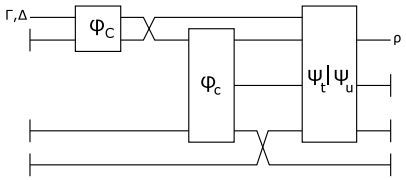


Σχήμα 3.5: Κύκλωμα γινομένου.

Όπως βλέπουμε ο όρος (t, u) εμηνεύεται ουσιαστικά σαν η παράλληλη σύνθεση δύο κυκλωμάτων: αυτού που αντιστοιχεί στο t και αυτού που αντιστοιχεί στο u. Το context μοιράζεται κατάλληλα στα δύο κυκλώματα σε περίπτωση που μοιράζονται μεταβλητές.

Τα κυκλώματα που αντιστοιχουν στους δύο constructors των αθροισμάτων είναι στοιχειώδη. Έστω πως έχουμε τον όρο $t : \tau$ και θέλουμε να κατασκευάσουμε το κύκλωμα για το **inl** $t : \tau \oplus \sigma$. Αν $|\tau| > |\sigma|$ το κύκλωμα είναι απλώς το κύκλωμα που αντιστοιχεί στο t με την προσθήκη στην έξοδο ενός qbit στην κατάσταση qtrue. Αλλιώς προσθέτουμε ένα padding από όσα χρειάζεται qbit ώστε η έξοδος να έχει μέγεθος $|\sigma|+1$. Αντίστοιχα ισχύουν και για το inr μόνο που το qbit που προσθέτουμε είναι qfalse. Βλέπουμε λοιπόν πως τα αθροίσματα υλοποιούνται με έναν τρόπο παρόμοιο με αυτόν που θα χρησιμοποιούσαμε και σε μία κλασική γλώσσα: δημιουργούμε έναν τύπο αρκετά μεγάλο για να χωρέσει και τους δύο τύπους του αθροίσματος και προσθέτουμε ένα qbit για να ξεχωρίσουμε ανάμεσα στους δύο τύπους που μπορεί να περιέχονται.

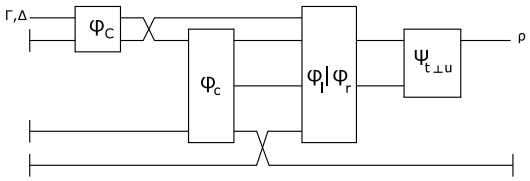
Ας δούμε τώρα την περίπτωση του case, η οποία φαίνεται στο σχήμα 3.6. Πρώτα θα εκτελεστεί ο υπολογισμός που αντιστοιχεί στο πρώτο τελούμενο του case, ο οποίος λόγω του ότι είναι τύπου $\oplus$ θα μας δώσει σίγουρα ένα Q_2. Το qbit αυτό χρησιμοποιείται στον έλεγχο της υπό συνθήκη αποτίμησης των t, u και στο τέλος καταλήγει στα σκουπίδια (κάτι που υπονοεί μέτρηση).



Σχήμα 3.6: Κύκλωμα case.

Το μεγαλύτερο ενδιαφέρον παρουσιάζει η περίπτωση του case^o. Το αντίστοιχο κύκλωμα φαίνεται στο σχήμα 3.7. Η υπό συνθήκη αποτίμηση δεν παράγει σκουπίδια, διότι οι αντίστοιχοι όροι αποδεικνύονται αυστηρά. Επίσης χρησιμοποιούνται τρία κυκλώματα τα ϕ_l, ϕ_r και $\psi_{t\perp u}$, τα οποία προκύπτουν από την ερμηνεία της ορθογωνιότητας των t, u στον κανόνα του $case^o$. Χωρίς να μπούμε σε λεπτομέρειες μπορούμε να πούμε πως τα κυκλώματα αυτά ορίζονται πάλι επαγωγικά από απλούστερα αποτελούμενα μόνο από μεταθέσεις και πύλες not. Μόνη εξαίρεση ο κανόνας ορθογωνιότητας των υπερθέσεων ο οποίος προκαλεί την περιστροφή $\psi = \begin{pmatrix} \lambda_0 & \lambda_1 \\ \kappa_0 & \kappa_1 \end{pmatrix}$. Αξίζει πάντως να σημειώσουμε πως αν τα t, u είναι $\mathrm{qtrue}, \mathrm{qfalse}$ με αυτή τη σειρά τότε το $\psi_{t\perp u}$

είναι το ταυτοτικό κύκλωμα, ενώ αν είναι με την αντίθετη σειρά το $\psi_{t\perp u}$ είναι μια πύλη not.



Σχήμα 3.7: Κύκλωμα $case^o$.

Τέλος το κύκλωμα που αντιστοιχεί στην υπέρθεση υπολογίζεται εύκολα αν λάβουμε υπ'οψιν πως $\{(\lambda)t \mid (\lambda')u\} \equiv \textbf{if}^o\{(\lambda)qtrue \mid (\lambda')qfalse\}$ **then** t **else** u. Η υπέρθεση ενός μόνο qbit μπορεί να δημιουργηθεί με την κατάλληλη περιστροφή του qfalse.

3.3.4 Παραδείγματα

Το πιο απλό παράδειγμα είναι η πραγματοποίηση μιας πύλης μίας εισόδου όπως η πύλη not:

$$\textbf{if}^o\ x\ \textbf{then}\ \text{qfalse}\ \textbf{else}\ \text{qtrue}$$

Αυτό θα έχει σαν αποτέλεσμα την είσοδο του x σε μία πύλη not. Η πύλη Hadamard υλοποιείται ως εξής:

$$\textbf{if}^o\ x\ \textbf{then}\ \{\text{qfalse} \mid -\text{qtrue}\}\ \textbf{else}\ \{\text{qfalse} \mid \text{qtrue}\}$$

Σε αυτήν την περίπτωση η περιστροφή που είναι ερμηνεία της ορθογωνιότητας είναι αυτή ακριβώς η πύλη Hadamard και μέσα από αυτή περνά το x.

Ένα στοιχείο με το οποίο πρέπει να είμαστε αρκετά προσεκτικοί στην QML είναι το μοίρασμα των context. Ας δούμε ένα παράδειγμα: έστω had x η εφαρμογή της πύλης Hadamard, όπως την είδαμε πιο πάνω, στο x. Προφανώς έχουμε had $\{\text{qfalse} \mid \text{qtrue}\} = \text{qfalse}$. Έστω το πρόγραμμα

$$\textbf{let}\ x = \text{qfalse}\ \textbf{in}\ (\text{had}\ x, \text{had}\ x)$$

Το μοίρασμα των context θα δημιουργήσει άλλο ένα qbit ώστε το x να μπορεί να μοιραστεί ανάμεσα στους δύο όρους του ζευγαριού. Επειδή το x είναι qfalse και το νέο qbit είναι qfalse άρα το αποτέλεσμα είναι όπως θα περιμέναμε δύο qbit στην κατάσταση $\{\text{qfalse} \mid \text{qtrue}\}$. Έστω τώρα το

$$\textbf{let}\ x = \{\text{qfalse} \mid \text{qtrue}\}\ \textbf{in}\ (\text{had}\ x, \text{had}\ x)$$

Λογικά θα αναμέναμε το αποτέλεσμα να είναι $(\text{qfalse}, \text{qfalse})$. Για να καταλάβουμε ακριβώς τι γίνεται ας σκεφτούμε με όρους διανυσμάτων. Το x έχει κατάσταση $\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$. Με το μοίρασμά του έχουμε πλέον δύο qbit με κατάσταση $\frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$. Η εφαρμογή της πύλης Hadamard στο καθένα, δηλαδή η εφαρμογή της $(I \otimes H)(H \otimes I) = H \otimes H$ έχει σαν αποτέλεσμα την κατάσταση $\frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$. Βλέπουμε λοιπόν πως ενώ περάσαμε και τα δύο αντίγραφα του x από πύλη Hadamard δεν παίρνουμε το αναμενόμενο αποτέλεσμα.

Γενικώς μπορούμε να πούμε πως η τακτική του μοιράσματος μεταβλητών μειώνει αρκετά την αξία του συναρτησιακού μέρους της γλώσσας, διότι ένας όρος που έχει μια δεδομένη συμπεριφορά όταν χειρίζεται μία ανεξάρτητη μεταβλητή αποκτά μία άλλη, διαφορετική συμπεριφορά όταν η μεταβλητή αυτή χρησιμοποιείται και από έναν άλλο όρο και μάλιστα αυτό εξαρτάται

και από τον τρόπο που ο άλλος όρος χειρίζεται το δικό του αντίγραφο. Αυτό αντιπροσωπεύει μία εγγενή αδυναμία του κβαντικού υπολογιστικού μοντέλου, η οποία προκύπτει ακριβώς από το γεγονός ότι είναι αδύνατο να δημιουργήσουμε αντίγραφα ενός qbit. Δεν είναι όμως σαφές κάποιο πλεονέκτημα της λύσης που υλοποιήθηκε στην QML, λαμβάνοντας μάλιστα υπ'οψιν πως ο σχεδιασμός της γλώσσας κάνει μεγάλο κόπο για να απομονώσει τους όρους που έχουν παρενέργειες με μέτρηση. Ίσως θα ήταν πιο συνετό το βάρος της δημιουργίας ψευδοαντιγράφων μιας μεταβλητής να πέφτει στον προγραμματιστή, όπου αυτό είναι αναγκαίο.

3.4 Η γλώσσα NTUA-QML

Η γλώσσα NTUA-QML είναι μία παραλλαγή της QML που προσπαθεί να προχωρήσει ένα βήμα παραπέρα. Για το σκοπό αυτό

- Δεν δημιουργεί αυτόματα ψευδοαντίγραφα μεταβλητών αλλά αφήνει τον προγραμματιστή να το κάνει μόνο όπου εκείνος κρίνει ότι χρειάζεται. Οι μεταβλητές θεωρούνται πλέον references σε κάποια bit του κβαντικού state του συστήματος και όχι αυτόνομοι φορείς πληροφορίας.
- Χρησιμοποιεί διαφορετική σημασιολογία για το if^o ώστε να μην δημιουργεί παρενέργειες στο πρώτο τελούμενο.
- Εισάγει μία ακόμη έκδοση του τελεστή if το if^-. Το if^- προσπαθεί να παίξει το ρόλο που παίζει το if^o στην QML, με πιο περιορισμένα όμως καθήκονται και ταυτόχρονα αυξημένες δυνατότητες.
- Απλοποιεί το σύστημα τύπων, καταργώντας τα αθροίσματα και εισάγοντας τα qtrue,qfalse κατευθείαν στη γλώσσα, ως σταθερές
- Επεκτείνει τους κανόνες ορθογωνιότητας, ώστε σε συνδυασμό με το if^- να είναι δυνατός οποιοσδήποτε ορθομοναδιαίος μετασχηματισμός

Η σημασιολογία της QML δίνεται σε επίπεδο πινάκων πυκνότητας και όχι κβαντικών κυκλωμάτων. Θα αναλυθεί σε βάθος στο κεφάλαιο 4 όπου θα εξεταστεί και μια υλοποίησή της με ενδιάμεση γλώσσα την QPL.

3.4.1 Συντακτικό

Το συντακτικό της NTUA-QML έχει ως εξής:

$$
\begin{array}{rcl}
t & ::= & x \mid \textbf{let}\ x = t\ \textbf{in}\ u \\
 & \mid & (t, u) \mid \textbf{let}\ (x, y) = t\ \textbf{in}\ u \\
 & \mid & \textbf{if}\ t\ \textbf{then}\ u\ \textbf{else}\ v \\
 & \mid & \textbf{if}^o\ t\ \textbf{then}\ u\ \textbf{else}\ v \\
 & \mid & \textbf{if}^-\ t\ \textbf{then}\ u\ \textbf{else}\ v \\
 & \mid & \{(\kappa)t \mid (\iota)u\} \\
 & \mid & \textbf{qtrue} \mid \textbf{qfalse}
\end{array}
$$

Σε σχέση με την γραμματική της QML, έχουν αφαιρεθεί η αποδυνάμωση, καθώς και οι constructors των αθροισμάτων. Η αφαίρεση της κλήσης συναρτήσεων δεν είναι ουσιαστική αφου η QML ούτως ή άλλως δεν υποστήριζε αναδρομικές συναρτήσεις κι έτσι οι συναρτήσεις της δεν ήταν παρά μακροεντολές. Η αποδυνάμωση, η κατεύθυνση δηλαδή κάποιων μεταβλητών στα σκουπίδια θεωρήθηκε πως δεν είναι ουσιώδες μέρος της γλώσσας και μπορεί να παραλειφθεί.

Η ύπαρξη αθροισμάτων δεν είναι πλέον κάτι απαραίτητο στη γλώσσα μας, λόγω της αλλαγής στο σύστημα τύπων την οποία θα αναλύσουμε παρακάτω. Έτσι, χάριν απλούστευσης κρατάμε μόνο τα γινόμενα.

3.4.2 Σύστημα τύπων

Οι τύποι της NTUA-QML ακολουθούν την εξής γραμματική:

$$\tau \ ::= \ Q_2 \ \mid \ Q_2 \otimes \tau$$

Επίσης ορίζουμε ότι $\tau \otimes \sigma = Q_2 \otimes (\tau' \otimes \sigma)$ για $\tau = Q_2 \otimes \tau'$.

Για τα μεγέθη των τύπων έχουμε ότι $|Q_2| = 1$ και $|Q_2 \otimes \tau| = |\tau| + 1$ οπότε $|\tau \otimes \sigma| = |\tau| + |\sigma|$ όπως και στην QML.

Το σύστημα τύπων είναι βασισμένο στους τύπους της QML για τους όρους που υπάρχουν και στις δύο γλώσσες. Παραθέτουμε μόνο τους κανόνες που έχουν αλλαγές, καθώς και τους κανόνες του if^-.

$$\frac{}{\bullet \vdash^o \textbf{qtrue}, \textbf{qfalse} : Q_2}\text{constants} - \text{intro}$$

Εισάγουμε τις δύο νέες σταθερές στη γλώσσα μας με τύπο Q_2.

$$\frac{\begin{array}{c}\Gamma \vdash c : \tau \\ \Delta \vdash t : \rho \\ \Delta \vdash u : \rho\end{array}}{\Gamma \otimes \Delta \vdash \textbf{if}\ c\ \textbf{then}\ t\ \textbf{else}\ u : \rho}\text{if}$$

$$\frac{\begin{array}{c}\Gamma \vdash^\alpha c : \tau \\ \Delta \vdash^o t : \rho \\ \Delta \vdash^o u : \rho\end{array}}{\Gamma \otimes \Delta \vdash^\alpha \textbf{if}^o\ c\ \textbf{then}\ t\ \textbf{else}\ u : \rho}\text{if}^o$$

Βλέπουμε και τους κανόνες για τα if και if^o οι οποίοι είναι ουσιαστικά ίδιοι. Μόνη σημαντική διαφορά η απουσία της απαίτησης για ορθογωνιότητα στο if^o. Δίνουμε και τον κανόνα για το if^-.

$$\frac{\begin{array}{c}\Gamma \vdash^\alpha c : \tau \\ \Delta \vdash^o t : \rho \\ \Delta \vdash^o u : \rho \\ t \perp u \qquad |\rho| = \langle t \rangle\end{array}}{\Gamma \otimes \Delta \vdash^\alpha \textbf{if}^-\ c\ \textbf{then}\ t\ \textbf{else}\ u : \rho}\text{if}^-$$

Το $\langle t \rangle$ ορίζεται ως εξής: $\langle \textbf{if}^-\ c\ \textbf{then}\ t\ \textbf{else}\ u \rangle = 1 + \langle t \rangle$ και $\langle t \rangle = 0$ σε κάθε άλλη περίπτωση. Ο λόγος που χρησιμοποιούμε αυτό το τέχνασμα θα γίνει σαφής όταν μιλήσουμε για τη σημασιολογία του if^-.

Εισάγουμε τώρα και μερικούς νέους κανόνες για το orthogonality.

$$\frac{}{\text{qtrue} \perp \text{qfalse}}\text{Oconst}$$

$$\frac{t \perp v \quad t \perp w \qquad u \perp v \quad u \perp w}{\{(\lambda)t \mid (\kappa)u\} \perp \{(\lambda')v \mid (\kappa')w\}}\text{Osup2}$$

$$\frac{t \perp v \quad t \perp w \qquad u \perp v \quad u \perp w}{\textbf{if}^-\ c\ \textbf{then}\ t\ \textbf{else}\ u \perp \textbf{if}^-\ c\ \textbf{then}\ v\ \textbf{else}\ w}\text{Oif}^-$$

Ο κανόνας Osup2 είναι αρκετά διασθητικός: μας λέει πως αν ένας όρος είναι υπέρθεση ανάμεσα στο $(\text{qfalse}, \text{qfalse})$ και το $(\text{qfalse}, \text{qtrue})$ και ένας άλλος ανάμεσα στα $(\text{qtrue}, \text{qfalse})$, $(\text{qtrue}, \text{qtrue})$ τότε οι δύο υπερθέσεις είναι ορθογώνιες, αφού η πρώτη έχει το πρώτο qbit qfalse και η δεύτερη το έχει qtrue. Ο Oif^- βασίζεται κι αυτός σε μια παρόμοια λογική. Η τεκμηρίωσή του θα πρέπει να περιμένει την εισαγωγή της σημασιολογίας της γλώσσας.

3.4.3 Σημασιολογία

Θα δώσουμε τώρα τη σημασιολογία της NTUA-QML χρησιμοποιώντας πίνακες πυκνότητας. Για να το κάνουμε αυτό θεωρούμε πως το πρόγραμμά μας λειτουργεί πάνω σε ένα κβαντικό state που αρχικά αποτελείται από 0 bit. Καθώς εκτελείται δεσμεύονται qbit, και κάθε μεταβλητή του αντιστοιχεί σε κάποια από αυτά.

Η σημασιολογία ενός όρου σε συνδυασμό με κάποιο context είναι μία συνάρτηση από πίνακες πυκνότητας σε πίνακες πυκνότητας. Δηλαδή αν έχουμε έναν όρο και για κάθε μεταβλητή η οποία χρησιμοποιείται σε αυτόν τη λίστα από qbit στα οποία αντιστοιχεί θα πρέπει να μπορούμε να προσδιορίσουμε τι μεταβολές επιφέρει ο όρος αυτός στο συνολικό state.

Ας ξεκινήσουμε με τις δύο σταθερές της γλώσσας μας. Η qfalse προκαλεί απλώς τη δέσμευση μίας νέας μεταβλητής αρχικοποιημένης στο $|\mathbf{0}\rangle$. Έτσι η χρήση του όρου qfalse προκαλεί τη μετάβαση από το state s στο state $\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes s$. Ομοίως η χρήση του όρου qtrue ισοδυναμεί με μετάβαση στο state $\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes s$

Η χρήση μίας μεταβλητής δεν επιφέρει από μόνη της κάποια μεταβολή στο state.

Ο όρος **let** $x = t$ **in** u μπορεί να αναλυθεί ουσιαστικά σε διαδοχικές εφαρμογές των t και u όπου οι μεταβολές που επιφέρει ο u στο state υπολογίζονται λαμβάνοντας υπ'οψιν το εμπλουτισμένο με την x context. Όμοια ισχύουν και για τον όρο **let** $(x, y) = t$ **in** u με τη διαφορά ότι τα x, y παίρνουν το καθένα ένα μέρος των bit του t ανάλογα με τον τύπο τους.

Ο όρος (t, u) αναλύεται απλώς σε μία διαδοχική αποτίμηση των t και u.

Το ενδιαφέρον φυσικά βρίσκεται στη σημασιολογία των if. Για το **if** c **then** t **else** u θεωρούμε ως πρώτο τελούμενο το πρώτο bit του c. Έτσι η μεταβολή που επιφέρει το if στο state είναι μία διαδοχική αποτίμηση πρώτα του c και μετά μία μέτρηση πάνω στο πρώτο bit που αντιστοιχεί στο c. Έπειτα αποτιμάμε το t για το state που προκύπτει αν το bit που μετρήθηκε ήταν qtrue, και το u για το state που προκύπτει αν το bit που μετρήθηκε ήταν qfalse, και τα προσθέτουμε. Δηλαδή, αν το state μετά τον υπολογισμό του c είναι $\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$ και το πρώτο bit του c είναι το πιο σημαντικό bit του state τότε υπολογίζουμε τη μεταβολή που επιφέρει ο t στο state $\left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & D \end{array}\right)$ και ο u στο $\left(\begin{array}{c|c} A & 0 \\ \hline 0 & 0 \end{array}\right)$ και τις προσθέτουμε. Αν το πρώτο bit του c δεν είναι το πιο σημαντικό θα πρέπει να εκτελέσουμε προφανώς κάποια κατάλληλη μετάθεση. Η σημασιολογία αυτή για το if δεν είναι τίποτα άλλο από μια μεταγραφή της σημασιολογίας του measure της QPL.

Το if^o θα αποτελέσει για τη γλώσσα μας τον κβαντικό τελεστή αποτίμησης υπο συνθήκη. Ουσιαστικά δηλαδή δεν θα είναι παρά μία γενικευμένη εκδοχή των ελεγχόμενων πυλών που έχουμε ήδη δει. Η σημασιολογία του προκύπτει άμεσα από το κύκλωμα 3.1, όπου το Q_2 είναι το πρώτο bit που προκύπτει από την αποτίμηση του c. Ας υποθέσουμε και πάλι πως το πρώτο bit του c είναι το πιο σημαντικό του state. Λόγω της απαίτησης από το σύστημα τύπων $\vdash^o t, u$ γνωρίζουμε πως οι όροι t και u δεν προκαλούν μέτρηση. Ως εκ τούτου, μπορούμε να θεωρήσουμε πως η αποτίμησή τους δεν είναι παρά η εφαρμογή ενός ορθομοναδιαίου μετασχηματισμού στο σύστημα που προκύπτει από τον υπολογισμό του c και την προσθήκη επιπλέον bit που τυχόν δεσμεύουν οι t και u. Έστω T ο μετασχηματισμός που αντιστοιχεί στο t και U αυτός που αντιστοιχεί στον u. Οι δύο αυτοί μετασχηματισμοί ενεργούν σε όλο το state εκτός από το πρώτο bit του c. Τότε, σύμφωνα και με το κύκλωμα έχουμε πως το state υφίσταται με τη σειρά τους εξής μετασχηματισμούς: $\left(\begin{array}{c|c} I & 0 \\ \hline 0 & T \end{array}\right)$,$N \otimes I$, $\left(\begin{array}{c|c} I & 0 \\ \hline 0 & U \end{array}\right)$,$N \otimes I$. Οι μετασχηματισμοί αυτοί συνολικά μπορούν να γραφούν ως

$$\left(\begin{array}{c|c} U & 0 \\ \hline 0 & T \end{array}\right)$$

Προσοχή απαιτείται ώστε το πρώτο bit του c να μην περιλαμβάνεται στους t,u γιατί δεν γίνεται ένα qbit να ελέγχει έναν μετασχηματισμό στον οποίο συμμετέχει και το ίδιο. Ένα τέτοιο λάθος είναι δύσκολο να αποκλειστεί από το σύστημα τύπων, είναι εύκολο όμως να διαπιστωθεί στη μετάφραση.

Το if^o είναι ένα χρήσιμο εργαλείο, όμως δεν είναι αρκετό, γιατί μπορεί να εκφράσει μετασχηματισμούς με δύο τουλάχιστον bit και μόνο μιας συγκεκριμένης μορφής. Για το λόγο αυτό εισάγουμε το if$^-$. Ο στόχος του είναι να μπορεί να εκφράσει οποιονδήποτε ορθομοναδιαίο μετασχηματισμό. Ας ξεκινήσουμε με την απλή περίπτωση όπου τα δύο branches έχουν τύπο Q_2 κι έπειτα γενικεύουμε. **if$^-$** c **then** t **else** u. Όπως και προηγουμένως το πρώτο bit του c είναι αυτό που μας ενδιαφέρει. Από τους κανόνες του συστήματος τύπων έχουμε πως $t \perp u$. Έχουμε υποθέσει πως $t, u : Q_2$ άρα τα t, u δεν μπορεί να είναι ζευγάρια. Επίσης δεν μπορεί να είναι if^- γιατί $\langle t \rangle + 1 = |Q_2| = 1 \rightarrow \langle t \rangle = 0$. Επομένως τα t, u είναι και τα δύο σταθερές ή superposition από τις δύο σταθερές. Θεωρώντας πως και τα qtrue,qfalse είναι υπερθέσεις με τους αντίστοιχους συντελεστές 1 και 0 παρατηρώ πως σε οποιαδήποτε περίπτωση αν γράψω τα t, u σαν υπερθέσεις της μορφής $\{(\lambda_0)\text{qfalse} \mid (\lambda_1)\text{qtrue}\}$ και $\{(\kappa_0)\text{qfalse} \mid (\kappa_1)\text{qtrue}\}$ θα ισχύει $\lambda_0^* \kappa_0 + \lambda_1^* \kappa_1 = 0$. Όμως αυτό, μαζί με τη συνθήκη $|\lambda_0|^2 + |\lambda_1|^2 = |\kappa_0|^2 + |\kappa_1|^2$ μας εξασφαλίζει πως ο πίνακας $\begin{pmatrix} \kappa_0 & \lambda_0 \\ \kappa_1 & \lambda_1 \end{pmatrix}$ είναι ορθομοναδιαίος. Ορίζουμε λοιπόν ότι η εφαρμογή αυτής της πύλης πάνω στο πρώτο bit του c είναι η σημασία του if$^-$ σε αυτήν την περίπτωση.

Ας δούμε τώρα τη γενική περίπτωση. Τα t, u έχουν τύπο $\tau \neq Q_2$ επομένως δεν μπορεί παρά να είναι και τα ίδια if$^-$. Γνωρίζουμε το βάθος $\langle t \rangle$ έστω $d > 0$. Καταλήγουμε λοιπόν πως θα έχουμε τελικά 2^{d+1} branches στο δέντρο των if$^-$ τα οποία δεν είναι if$^-$ και τα οποία έχουν τύπο τ. Επίσης, όπως εύκολα προκύπτει από τον κανόνα ορθογωνιότητας των if$^-$ όλα τα branches θα πρέπει να είναι ανά δύο ορθογώνια.

Ας ασχοληθούμε εδώ με την έννοια της ορθογωνιότητας. Είναι εύκολο με όσα έχουμε πει ως τώρα να αντιστοιχήσουμε σε κάθε όρο ο οποίος αποτελείται μόνο απο ζευγάρια, σταθερές και υπερθέσεις τους έναν πίνακα πυκνότητας ή ισοδύναμα ένα μοναδιαίο διάνυσμα. Έτσι π.χ. στο qtrue αντιστοιχεί το $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ στο $\{(a)\text{qtrue} \mid (b)\text{qfalse}\}$ το $\begin{pmatrix} b \\ a \end{pmatrix}$ κλπ. Όταν δύο τέτοιο όροι αποδεικνύεται με τους κανόνες που έχουμε ήδη δώσει ότι είναι ορθογώνιοι, τα διανύσματά τους έχουν εσωτερικό γινόμενο 0, δηλαδή αν u, t τα δύο διανύσματα έχουμε $u^* t = 0$ ή σε braket notation $\langle u | t \rangle = 0$. Αυτό μπορεί να δειχθεί επαγωγικά ως εξής:

- Είναι προφανές για τα διανύσματα που αντιστοιχούν στα qtrue,qfalse.

- Αν $t \perp u$ και για τα αντίστοιχα διανύσματα ισχύει $t^* u = 0$. Τότε στον (t, v) αντιστοιχεί το διάνυσμα $t \otimes v$ και στον (u, w) το $u \otimes w$. Από ιδιότητα του τανυστικού γινομένου προκύπτει το ζητούμενο

- Το διάνυσμα που προκύπτει από τον όρο $\{(a)t \mid (b)u\}$ είναι το $at + bu$. Έτσι αν πάρουμε και τον όρο $\{(c)t \mid (d)u\}$ με $a^* c + b^* d = 0$, ώστε να είναι ορθογώνιοι, το γινόμενο των δύο διανυσμάτων είναι $(at + bu)^*(ct + du) = a^* c t^* t + b^* d u^* u$ αφού λόγω ορθογωνιότητας $t^* u = 0$. Όμως τα t, u είναι μοναδιαία διανύσματα άρα το άθροισμα είναι 0.

- Αν έχουμε τους όρους $\{(a)t \mid (b)u\}$,$\{(c)v \mid (d)w\}$ που είναι κάθετοι ανά δύο τότε το γινόμενο $(at + bu)^*(cv + dw) = 0$

Επανερχόμαστε στο if$^-$. Έχουμε δείξει ότι έχουμε 2^{d+1} ορθογώνιους ανά δύο όρους με μέγεθος $d + 1$ qbit. Για να ισχύει αυτό θα πρέπει και τα $d + 1$ qbit να χρησιμοποιούνται σε κάποιον κανόνα ορθογωνιότητας γιατί με d qbit δεν μπορώ να σχηματίσω 2^{d+1} ανά δύο ορθογώνιους όρους. Αυτό γιατί το διάνυσμα που προκύπτει έχει μέγεθος 2^d και αντίστοιχος χώρος έχει μία κανονική βάση μεγέθους 2^d.

Συμπεραίνουμε λοιπόν ότι τα branches αποτελούνται μόνο από σταθερές και ζευγάρια ή υπερθέσεις σταθερών. Όπως είδαμα μπορούμε σε κάθε branch να αντιστοιχήσουμε ένα διάνυσμα και όλα τα διανύσματα να έχουν ανά δύο εσωτερικό γινόμενο μηδέν και να είναι μοναδιαία. Αυτό σημαίνει πως ένας πίνακας με στήλες τα διανύσματα αυτά είναι ορθομοναδιαίος. Ορίζουμε λοιπόν τη σημασιολογία του if^- να είναι ο ορθομοναδιαίος μετασχηματισμός που περιγράφεται από αυτόν τον πίνακα, αν η διάταξη των διανυσμάτων είναι τέτοια ώστε στην πρώτη στήλη να βρίσκεται αυτό που προκύπτει από τον όρο που βρίσκεται ακολουθώντας όλα τα κλαδιά else του δέντρου. Ο μετασχηματισμός αυτός εφαρμόζεται στα πρώτα bit του πρώτου τελούμενου κάθε if^- με τη σειρά.

3.4.4 Παραδείγματα

Ο τελεστής not στην NTUA-QML είναι:

$$\textbf{if}^- \; x \; \textbf{then} \; \text{qfalse} \; \textbf{else} \; \text{qtrue}$$

Παρομοίως ο τελεστής Hadamard είναι:

$$\textbf{if}^- \; x \; \textbf{then} \; \{\text{qfalse} \mid -\text{qtrue}\} \; \textbf{else} \; \{\text{qfalse} \mid \text{qtrue}\}$$

Ο τελεστής ελεγχόμενου not είναι

$$\textbf{if}^o \; x \; \textbf{then} \; \text{not} \; y \; \textbf{else} \; y$$

Έκπληξη ίσως προκαλεί το γεγονός ότι το πρόγραμμα

$$\textbf{if}^o \; x \; \textbf{then} \; \text{qtrue} \; \textbf{else} \; \text{qfalse}$$

είναι στην πραγματικότηα άλλο ένα παράδειγμα τελεστή cnot. Θα πρέπει να θυμηθούμε πως η χρήση των σταθερών δεσμεύει μια επιπλέον μεταβλητή. Αν το x είναι qtrue η μεταβλητή θα αντισταφεί για να προκύψει το qtrue, εξ' ου και η ερμηνεί του προγράμματος ως cnot.

Ας δούμε κι ένα πιο περίπλοκο παράδειγμα:

$$\begin{array}{l} \textbf{if}^o \; x \; \textbf{then} \\ \qquad\qquad \textbf{if}^o \; y \; \textbf{then} \; \text{qtrue} \; \textbf{else} \; \text{qfalse} \\ \qquad \textbf{else} \\ \qquad\qquad \textbf{if}^o \; y \; \textbf{then} \; \text{qfalse} \; \textbf{else} \; \text{qtrue} \end{array}$$

Όπως είπαμε το αποτέλεσμα του $\textbf{if}^o \; y \; \textbf{then}$ qtrue **else** qfalse είναι μια πύλη N_c. Το αποτέλεσμα του $\textbf{if}^o \; y \; \textbf{then}$ qfalse **else** qtrue μπορεί εύκολα να διαπιστωθεί πως είναι ο πίνακας $(N \otimes I) \left(\begin{array}{c|c} I & 0 \\ \hline 0 & N \end{array}\right) (N \otimes I) = \left(\begin{array}{c|c} N & 0 \\ \hline 0 & I \end{array}\right)$. Ο μετασχηματισμός λοιπόν είναι

$$\left(\begin{array}{c|c|c|c} N & 0 & 0 & 0 \\ \hline 0 & I & 0 & 0 \\ \hline 0 & 0 & I & 0 \\ \hline 0 & 0 & 0 & N \end{array}\right)$$

Σε περίπτωση που τα x, y έχουν κλασικές τιμές το πρόγραμμα συμπεριφέρεται όπως θα περιμέναμε, μας δίνει δηλαδή qtrue όταν είναι ίσα. Σε περίπτωση που είναι στην ίδια κβαντική

κατάσταση π.χ. $\begin{pmatrix} a \\ b \end{pmatrix}$. Η εφαρμογή του μας δίνει την κατάσταση

$$\left(\begin{array}{c|c|c|c} N & 0 & 0 & 0 \\ \hline 0 & I & 0 & 0 \\ \hline 0 & 0 & I & 0 \\ \hline 0 & 0 & 0 & N \end{array}\right) \begin{pmatrix} a^2 \\ 0 \\ ab \\ 0 \\ ab \\ 0 \\ b^2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ a^2 \\ ab \\ 0 \\ ab \\ 0 \\ 0 \\ b^2 \end{pmatrix}$$

Δηλαδή η πιθανότητα το bit του αποτελέσματος να είναι 1 είναι $|a^2|^2 + |b^2|^2$ και 0 $2|ab|^2$. Επομένως η πιθανότητα να μετρήσουμε 0 όταν τα x, y είναι σίγουρα στην ίδια κατάσταση είναι μέγιστη όταν τα x, y έχουν τη μέγιστη δυνατή αβεβαιότητα δηλαδή $|a|^2 = |b|^2 = \frac{1}{2}$. Όταν κάποιο από τα a, b έχει μεγαλύτερο μέτρο η πιθανότητα να μετρήσουμε 1 αυξάνεται. Φυσικά η πιθανότητα να μετρήσουμε 1 ποτέ δεν πέφτει κάτω από $\frac{1}{2}$, κάτι φυσικό αν αναλογιστούμε πως έχουμε σαν είσοδο δύο ίδια qbit. Καλό είναι πάντως να έχουμε υπ' όψιν πως ενδεχόμενη μέτρηση θα επηρεάσει ανεπανόρθωτα την κατάσταση των άλλων δύο bit. Έτσι είναι εύκολο να διαπιστώσουμε πως στη γενική περίπτωση, μία μέτρηση του qbit αποτελέσματος που έχει αποτέλεσμα 1 φέρνει τα άλλα δύο qbit σε καταστάσεις στις οποίες είναι σίγουρα ίσα ενώ μία μέτρηση με αποτέλεσμα 0 τα φέρνει σε καταστάσεις που είναι σίγουρα διαφορετικά.

Κεφάλαιο 4

Υλοποίηση της NTUA-QML

4.1 Εισαγωγή

Στο κεφάλαιο αυτό θα ασχοληθούμε με μία υλοποίηση της NTUA-QML σε γλώσσα Haskell. Φυσικά μία τέτοια υλοποίηση δεν θα μπορούσε να μας δώσει μια καλή εικόνα για την πραγματική ισχύ του κβαντικού προγραμματισμού αφού η εξομοίωση των κβαντικών διαδικασιών σε έναν σημερινό υπολογιστή έχει εκθετική πολυπλοκότητα σε σχέση με τον αριθμό των εμπλεκόμενων κβαντικών bit. Παρ' όλα αυτά, μια τέτοια υλοποίηση έχει μεγάλη εκπαιδευτική χρησιμότητα και θα μας επιτρέψει να δούμε τα αποτελέσματα μερικών κβαντικών αλγορίθμων χωρίς να είμαστε αναγκασμένοι να κάνουμε πράξεις με πίνακες πυκνότητας.

Η υλοποίηση αποτελείται από δύο μέρη: έναν μεταφραστή κάποιου χρήσιμου υποσυνόλου της QPL, το οποίο θα παίξει το ρόλο ενδιάμεσης γλώσσας, κι έναν μεταφραστή από την NTUA-QML στην QPL. Ο χωρισμός σε δύο μέρη έχει το πλεονέκτημα πως μας επιτρέπει να ασχοληθούμε με τις καθαρά τεχνικές λεπτομέρειες, όπως οι μεταθέσεις του state, στην γλώσσα χαμηλού επιπέδου κι έτσι μπορούμε πιο εύκολα να επικεντρωθούμε σε μια σωστή μετάφραση της NTUA-QML.

Μετά από την εκτέλεση κάθε αλγορίθμου το τελικό αποτέλεσμα θα είναι ένας πίνακας πυκνότητας ο οποίος θα μας δίνει το συνολικό state που προέκυψε.

4.2 Υλοποίηση της QPL

Η χρήση της QPL ως ενδιάμεσης γλώσσας καθιστά λιγότερο χρήσιμα τα πιο προηγμένα χαρακτηριστικά της όπως τα loop και η αναδρομή. Επίσης δεν μας ενδιαφέρουν τα κλασικά bit αφού η NTUA-QML δεν τα υποστηρίζει. Αυτό που χρειαζόμαστε είναι μια γλώσσα η οποία να μπορεί να διενεργεί ορθομοναδιαίους μετασχηματισμούς και μετρήσεις στα qbit του state που θα της καθορίζουμε. Ο ορισμός του τύπου των όρων της γλώσσας μας είναι:

```
data TermS a =     New
                 | Discard Int
                 | Measure Int (TermS a) (TermS a)
                 | Transform [Int] [[Complex a]]
                 | More (TermS a) (TermS a)
                 | Skip
        deriving (Show,Eq)
```

Ο τύπος αυτός είναι παραμετρικός ως προς το a ώστε να επιτρέπονται πίνακας μιγαδικών Float ή Double. Οι πίνακες ορίζονται σαν λίστες από λίστες μιγαδικών αριθμών. Το νόημα των όρων είναι:

- New: δεσμεύει ένα νέο qbit
- Discard: απελευθερώνει το qbit του με state που βρίσκεται στη δοθείσα θέση

- Measure: Μετρά το qbit που βρίσκεται στη δοθείσα θέση. Αποτιμά τον πρώτο από τους δύο όρους σε περίπτωση που το αποτέλεσμα είναι qtrue, και τον δεύτερο σε περίπτωση που είναι qfalse
- Transform: Μετασχηματίζει τα qbit της λίστας με βάση τον μετασχηματισμό του πίνακα
- More: Σειριακή αποτίμηση των δύο όρων
- Skip: Καμία μεταβολή

Η αρίθμηση των qbit του state ξεκινά από το 1, το οποίο είναι το πιο σημαντικό.

Ο μεταφραστής μας βασίζεται σε μια συνάρτηση η οποία δοθέντος ενός όρου της γλώσσας του παραπάνω τύπου και του υπάρχοντος state θα υπολογίζει το νέο. Ο τύπος της συνάρτησης αυτής είναι:

```
compute_state :: (RealFloat a) => TermS a -> [[Complex a]] -> [[Complex a]]
```

Είναι πολύ εύκολο να ορίσουμε τη συνάρτηση αυτή όταν το qbit που δίνεται ως παράμετρος είναι το πρώτο, ή αντίστοιχα είναι μια λίστα όλων των φυσικών από το 1 ως το n στην περίπτωση του Transform. Σε περίπτωση όμως που αυτό δεν συμβαίνει θα πρέπει να υλοποιήσουμε κατάλληλες μεταθέσεις του state καθώς και τις αντίστροφές τους. Για παράδειγμα η μέτρηση του τρίτου qbit είναι ισοδύναμη με μία αλλαγή βάσης τέτοια ώστε το τρίτο qbit να γίνει το πιο σημαντικό, μέτρηση του πρώτου qbit και εφαρμογή της αντίστροφης αλλαγής βάσης.

Στον πίνακα 4.1 δίνουμε τη συνάρτηση compute_state για τους διάφορους όρους της γλώσσας. Με $\leftrightarrow$ συμβολίζουμε την αλλαγή θέσης δύο qbit, ενώ με λίστα ακεραίων δίνουμε μία μετάθεση. Το αρχικό state συμβολίζεται με s ή με $\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$.

Πίνακας 4.1: Ορισμός της compute_state.

Όρος	Μετάθεση	Μεταβολή state	Μετάθεση
New		$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes s$	
Discard n	$1 \leftrightarrow n$	$A + D$	$1 \leftrightarrow (n-1)$
Measure n t_1 t_2	$1 \leftrightarrow n$	$\left(\begin{array}{c\|c} A & 0 \\ \hline 0 & D \end{array}\right) \rightarrow$	$1 \leftrightarrow n$
		$\rightarrow$compute_state $\left(\begin{array}{c\|c} 0 & 0 \\ \hline 0 & D \end{array}\right)$ t_1 +	
		+ compute_state $\left(\begin{array}{c\|c} A & 0 \\ \hline 0 & 0 \end{array}\right)$ t_2	
Transform l S	$l + ([1 \cdots n)] - l)$	$(S \otimes I)s(S \otimes I)^*$	ο αντίστροφος του προηγουμένου
More t_1 t_2		compute_state (compute_state s t_1) t_2	
Skip		s	

Η μετάθεση που εννοείται στο Transform είναι μία λίστα που δημιουργείται με συγχώνευση της λίστας ορισμάτων και όλων των υπολοίπων qbit του state, τα οποία φυσικά θα μείνουν ανεπηρέαστα από το μετασχηματισμό αφού θα χρησιμοποιήσουμε το tensor product του S με τον κατάλληλο μοναδιαίο πίνακα.

Όπως βλέπουμε η δυσκολία παρουσιάζεται κυρίως στην υλοποίηση των μεταθέσεων, αφού η υλοποίηση των λειτουργιών της γλώσσας μπορεί να γίνει άμεσα, αν έχουμε ορίσει βασικές λειτουργίες πινάκων.

Για να λύσουμε το πρόβλημα των μεταθέσεων αρχικά αντιμετωπίζουμε το θέμα της απλής αλλαγής θέσης δύο bit. Στη συνέχεια θα μπορέσουμε να υλοποιήσουμε οποιαδήποτε μετάθεση σαν διαδοχικές αλλαγές bit. Όταν θέλουμε να αλλάξουμε θέσεις στα bit n_1,n_2, αυτό θα προκαλέσει μία μετάθεση στο διάνυσμα κατάστασης ή αντίστοιχα στον πίνακα πυκνότητας.

Ας ασχοληθούμε πρώτα με το διάνυσμα κατάστασης. Ως γνωστόν, κάθε στοιχείο του είναι συντελεστής κάποιου $|\mathbf{i}\rangle$. Αν γράψουμε όλα τα $|\mathbf{i}\rangle$ στο δυαδικό σύστημα θα δούμε ότι σε κάποια οι τιμές των bit n_1 και n_2 είναι ίδιες. Προφανές παράδειγμα είναι το $|\mathbf{0}\cdots\mathbf{0}\rangle$. Οι συντελεστές αυτών δεν θα επηρεαστούν από την αλλαγή. Αν το bit n_1 είναι 1 και το bit n_2 0 σε κάποιο $|\mathbf{i}\rangle$ τότε ο συντελεστής του θα πρέπει να αλλάξει θέση με αυτόν του $|\mathbf{j}\rangle$ το οποίο έχει όλα τα άλλα bit ίσα με το $|\mathbf{i}\rangle$ αλλά το n_1 είναι 0 και το n_2 1. Έτσι αν a το αρχικό διάνυσμα κατάστασης τότε το νέο διάνυσμα θα είναι

$$a'_i = \begin{cases} a_i & \text{, τα } n_1, n_2 \text{ είναι ίσα στο } i \\ a_j & \text{, τα } n_1, n_2 \text{ άνισα και } j \text{ προκύπτει από } i \text{ με αντιστροφή των } n_1, n_2 \end{cases}$$

Για παράδειγμα σε ένα state τριών bit, η αλλαγή του πρώτου και του τρίτου θα οδηγήσει στη μετάθεση

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} \rightarrow \begin{pmatrix} a_0 \\ a_4 \\ a_2 \\ a_6 \\ a_1 \\ a_5 \\ a_3 \\ a_7 \end{pmatrix}$$

Η λογική αυτή επεκτείνεται εύκολα και σε πίνακες πυκνότητας, οι οποίοι προφανώς προκύπτουν από πολλαπλασιασμό δύο διανυσμάτων που έχουν υποστεί την ίδια μετάθεση. Ως εκ τούτου θα πρέπει να εφαρμόσουμε αυτήν τη μετάθεση σε κάθε στήλη ξεχωριστά του πίνακα, κι έπειτα να μεταθέσουμε τις στήλες με τον ίδιο τρόπο.

Έχοντας υλοποιήσει την αλλαγή βάσης είναι πλέον απλή υπόθεση η υλοποίηση οποιασδήποτε μετάθεσης. Δίνουμε εδώ την υλοποίηση σε Haskell μίας συνάρτησης η οποία εντοπίζει τα swaps που πρέπει να γίνουν ώστε από μία λίστα να μεταβούμε στην άλλη. Προφανώς το τυπικό πρώτο όρισμα θα είναι η λίστα $[1\cdots n]$.

```
findswaps :: [Int] -> [Int] -> [(Int,Int)]
findswaps l1 l2 | l1==l2 = []
                | (head l1) == (head l2) = findswaps (tail l1) (tail l2)
                | otherwise = ((head l1,head l2):
                        (findswaps l1 (swap (head l1) (head l2) l2)))
```

Η συνάρτηση swap απλώς επιστρέφει τη λίστα l2 με τα δύο στοιχεία που δίνονται στα πρώτα ορίσματα να έχουν αλλάξει θέση.

Έχοντας υλοποιήσει μία ενδιάμεση γλώσσα που αποτελεί ένα καλό επίπεδο αφαίρεσης θα προχωρήσουμε στην υλοποίηση της NTUA-QML.

4.3 Υλοποίηση της NTUA-QML

Ξεκινάμε με τον ορισμό του τύπου των όρων της γλώσσας μας:

```
data TermA =      Var String
                | Let String TermA TermA
                | Qtrue | Qfalse
                | Pair TermA TermA
                | LetP String TypeA String TypeA TermA TermA
                | If TermA TermA TermA
                | If0 TermA TermA TermA
                | Ifm TermA TermA TermA
```

```
                | Super (Complex Double) TermA (Complex Double) TermA
        deriving (Show,Eq)
```

και με τον ορισμό του συστήματος τύπων:

```
data TypeA = Qbit | Mult TypeA
        deriving (Show,Eq)
```

Ως τύπο του context θα χρησιμοποιούμε τον [(**String**, TypeA)] όταν ασχολούμαστε με το typechecking και τον [(**String**,[**Int**])] όταν ασχολούμαστε με την μετάφραση.

Προτού προσπαθήσουμε να μεταφράσουμε οποιοδήποτε πρόγραμμα θα πρέπει να δειχθεί ότι αυτό υπακούει στους κανόνες του συστήματος τύπων μας. Για το σκοπό αυτό χρησιμοποιούμε δύο συναρτήσεις, μία για τους γενικούς κανόνες τύπων και μία για τους αυστηρούς. Οι τύποι τους είναι

```
typecheck :: TermA -> [(String,TypeA)] -> Bool
typecheck0 :: TermA -> [(String,TypeA)] -> Bool
```

Η υλοποίηση των συναρτήσεων αυτών είναι σχετικά απλή, με τη βοήθεια φυσικά μερικών στοιχειωδών βοηθητικών συναρτήσεων οι οποίες υλοποιούν την πράξη $\otimes$ για context, υπολογίζουν τα μεγέθη των τύπων κλπ. Μία επίσης απαραίτητη βοηθητική συνάρτηση είναι αυτή που αποφαίνεται πότε δύο όροι είναι ορθογώνιοι, και η υλοποίησή της μπορεί επίσης να προκύψει από τους κανόνες που έχουμε δώσει.

Το κύριο βάρος της μετάφρασης πέφτει στην συνάρτηση trans με τύπο

```
trans :: TermA -> [(String,[Int])] -> (TermS Double,[Int])
```

Η συνάρτηση αυτή δέχεται σαν ορίσματα έναν όρο της γλώσσας και ένα context και μας επιστρέφει τη μετάφραση του όρου σε QPL καθώς και μία λίστα ακεραίων που είναι τα bit στα οποία αντιστοιχεί ο όρος. Η λίστα αυτή των ακεραίων επιτρέπεται να περιέχει το ίδιο στοιχείο πολλές φορές. Για παράδειγμα ο όρος

$$\textbf{let } x = \text{qtrue } \textbf{in } (x, x)$$

θα επιστρέψει τη λίστα $[1, 1]$. Αυτό είναι μια επιλογή στον ορισμό της γλώσσας η οποία μας απομακρύνει από την QML, αφού πλέον ο τύπος ενός όρου δεν μας δίνει απαραίτητα πληροφορίες για το πλήθος των qbit που αυτός καταλαμβάνει (π.χ. ο παραπάνω όρος έχει τύπο $Q_2 \otimes Q_2$ ενώ χρησιμοποιεί ένα μόνο qbit).

Ο πιο εύκολος τρόπος να ορίσουμε με σαφήνεια μία συνάρτηση όπως η trans είναι να δώσουμε απλώς την υλοποίησή της σε Haskell. Ας πούμε όμως δυο λόγια για τον τρόπο που μεταφράζει η trans κάθε όρο της γλώσσας μας.

- Για τους όρους qtrue και qfalse, η μετάφραση είναι απλώς έναν New που στην περίπτωση του qtrue ακολουθείται και από μία εφαρμογή της πύλης not. Και στις δύο περιπτώσεις το αποτέλεσμα είναι το bit 1. Το γεγονός αυτό έχει την επιπλοκή ότι όταν μεταφράζουμε έναν όρο που περιέχει δύο υποόρους θα πρέπει να λάβουμε υπ'οψιν ότι τα πιθανά qtrue, qfalse που περιέχει ο πρώτος από αυτούς επηρεάζουν το context με βάση το οποίο πρέπει να μεταφραστεί ο δεύτερος. Η τακτική όμως το New να δίνει πάντα το πιο σημαντικό bit της κατάστασης, η οποία προέρχεται από την υλοποίηση της QPL που δώσαμε, μας βολεύει διότι απλοποιεί τη μετάφραση των δύο σταθερών.

- Η μετάφραση του ζευγαριού (t, u) δεν είναι παρά η μετάφραση του t ακολουθούμενη από τη μετάφραση του u. Όπως είπαμε πρέπει να είμαστε προσεκτικοί ώστε το u να μεταφραστεί με βάση το σωστό context, ανάλογα του πόσα New περιέχει η μετάφραση του t. Τα bit του αποτελέσματος είναι η συγχώνευση των bit των αποτελεσμάτων των δύο μεταφράσεων, όπου και πάλι θα πρέπει να είμαστε προσεκτικοί διότι η πιθανή ύπαρξη New στην μετάφραση του u θα πρέπει να έχει ως συνέπεια κατάλληλη μεταβολή των bit αποτελέσματος του t.

- Η μετάφραση του **let**, και στις δύο εκδόσεις του, ακολουθεί την ίδια λογική με τη μετάφραση των ζευγαριών. Μεταφράζουμε τις δύο υποεκφράσεις στη σειρά, λαμβάνοντας υπ' όψιν τις μεταβολές που η κάθε μία προκαλεί στο context

- Η μετάφραση μίας μεταβλητής είναι μία εντολή skip. Τα bit αποτελέσματος είναι αυτά που υπαγορεύει το context

- Η μετάφραση μίας υπέρθεσης των qtrue, qfalse είναι ένα New ακολουθούμενο από τον κατάλληλο μετασχηματισμό εφαρμοσμένο πάνω στο νέο qbit. Συγκεκριμένα, ο πίνακας που χρησιμοποιείται για την $\{(\kappa_1)\mathrm{qfalse} \mid (\kappa_2)\mathrm{qtrue}\}$ είναι $\begin{pmatrix} \kappa_1 & \kappa_2 \\ \kappa_2 & -\kappa_1 \end{pmatrix}$ ο οποίος όπως εύκολα διαπιστώνεται είναι ορθομοναδιαίος. Σε περίπτωση που οι εκφράσεις που υπερτίθενται δεν είναι οι δύο σταθερές χρησιμοποιούμε τη σχέση $\{(\kappa_1)t_1 \mid (\kappa_2)t_2\} \equiv$ **if**$^o\{(\kappa_1)\mathrm{qfalse} \mid (\kappa_2)\mathrm{qtrue}\}$ **then** t_1 **else** t_2, λαμβάνοντας φυσικά υπ' όψιν ότι η μετάφρασή της θα δημιουργήσει ένα ακόμη qbit ως τελούμενο του if^o.

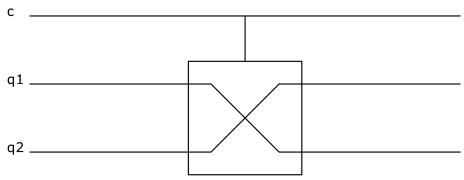
- Για τη μετάφραση του if^o αρχικά υπολογίζουμε τη μετάφραση του πρώτου τελούμενου, η οποία θα είναι και το πρώτο μέρος της συνολικής μετάφρασης. Έπειτα, λαμβάνοντας υπ'οψιν και τις μεταβολές του context υπολογίζουμε τη μετάφραση καθενός από τα δύο branches ξεχωριστά. Μετράμε τα New που προκύπτουν από το καθένα και αν δεν είναι ίσα τα ισορροπούμε προσθέτοντας στο τέλος του branch με τα λιγότερα όσα χρειάζονται. Στη συνέχεια αφαιρούμε τα New και από τις δύο μεταφράσεις. Για να το κάνουμε αυτό εκμεταλλευόμαστε το γεγονός πως οι μεταφράσεις μπορεί να περιέχουν μόνο όρους New, Transform και Skip, λόγω των κανόνων που επιβάλλει το αυστηρό σύστημα τύπων. Έτσι, εφαρμόζουμε διαδοχικά τον μετασχηματισμό **Transform** $l\ S$; **New** $\rightarrow$ **New**; **Transform** $l'\ (I \otimes S)$, όπου το l' προκύπτει από αύξηση όλων των στοιχείων του l κατά 1. Με άλλα λόγια διαμορφώνουμε το πρόγραμμα ώστε όλες οι δεσμεύσεις bit να γίνονται στην αρχή, κάτι που δεν αλλάζει το τελικό αποτέλεσμα διότι οι μετασχηματισμοί μεταβάλλονται κατάλληλα. Έτσι η μετάφραση του κάθε branch μετατρέπεται σε μία σειρά από New ακολουθούμενη από μία σειρά από Transform. Τα New εισάγονται στην τελική μετάφραση μετά από τη μετάφραση της συνθήκης. Στη συνέχεια υλοποιούμε ουσιαστικά το κύκλωμα του conditional, δηλαδή κάθε μετασχηματισμό των δύο branches **Transform** $l\ S$ τον μετατρέπουμε σε **Transform** $n : l\ \left(\begin{array}{c|c} I & 0 \\ \hline 0 & S \end{array}\right)$ όπου n το bit της συνθήκης. Εισάγουμε λοιπόν την ελεγχόμενη μετάφραση του όρου του then, ακολουθούμενη από μία αντιστροφή του bit συνθήκης, την ελεγχόμενη μετάφραση του όρου του else, και τέλος μία ακόμη αντιστροφή του bit συνθήκης. Τα πράγματα όμως δεν τελειώνουν εδώ. Αν τα bit αποτελέσματος των δύο branches είναι τα ίδια τότε έχουμε ήδη την ολοκληρωμένη μετάφραση και τα bits αποτελέσματος του if^o είναι επίσης τα ίδια. Αν όμως διαφέρουν επιλέγουμε για bit αποτελέσματος του if^o τα bit του then και σε εφαρμόζουμε ένα conditional swap ανάμεσα στα bit που διαφέρουν με ελεγκτή το bit συνθήκης. Το conditional swap υλοποιείται με τον πίνακα $\left(\begin{array}{c|c} X & 0 \\ \hline 0 & I \end{array}\right)$. Έτσι σε περίπτωση που το bit ελέγχου είναι true θα έχουμε όντως το σωστό αποτέλεσμα, ενώ σε περίπτωση που είναι false το αποτέλεσμα θα έρθει στα bit του then και θα είναι πάλι σωστό. Η αλλαγή θέσης δύο bit φαίνεται σχηματικά στο κύκλωμα 4.1. Η χρήση του χαρακτηριστικού αυτού της γλώσσας καλό είναι να γίνεται με προσοχή, διότι είναι πιθανό να επηρεάσει άλλα μέρη του προγράμματος που μοιράζονται bits με τα branches του if^o.

- Η μετάφραση του if$^-$ είναι σχετικά απλή υπόθεση. Αρχικά εντοπίζουμε τους τελεστέους, τα bit δηλαδή στα οποία θα εφαρμοστεί ο μετασχηματισμός. Αυτά είναι το πρώτο bit της έκφρασης συνθήκης του if$^-$ και τα πρώτα bit των if$^-$ που αποτελούν πιθανώς τα branches. Στη συνέχει εξετάζουμε κάθε φύλλο και υπολογίζουμε το διάνυσμα που του αντιστοιχεί ως εξής:

- Αν ο όρος είναι qtrue το διάνυσμα είναι το $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Αν είναι qfalse είναι $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$.
- Αν ο όρος είναι (t, u) τότε το διάνυσμα είναι των tensor product των διανυσμάτων των t, u
- Αν ο όρος είναι $\{ \kappa t \mid \lambda u \}$ το διάνυσμα είναι $\kappa t + \lambda u$ όπου με το ίδιο γράμμα έχουμε συμβολίσει τα αντίστοιχα διανύσματα των όρων

Σημειώνουμε πως όλα τα διανύσματα που προκύπτουν από τους παραπάνω κανόνες είναι προφανώς μοναδιαία, κάτι που θεωρήσαμε δεδομένο και στη συζήτηση περί ορθογωνιότητας που έγινε στο προηγούμενο κεφάλαιο. Η μετάφραση του if^- είναι ένα Transform πάνω στα bits που υπολογίστηκαν, με πίνακα τον πίνακα που έχει στήλες τα διανύσματα που υπολογίστηκαν, με πρώτη στήλη αυτή που αντιστοιχεί στο φύλλο που προκύπτει ακολουθώντας συνέχεια τα else.

- Για το if η μετάφραση είναι επίσης σχετικά απλή διότι υπάρχει ξεκάθαρη αντιστοιχία με το measure. Έτσι και εδώ ακολουθούμε τις διαδικασίες για swap των bit που δεν συμφωνούν στο τέλος αλλά και για ισορρόπηση των New, χωρίς όμως να κάνουμε κάποια επιπλέον αλλαγή στις μεταφράσεις των δύο branches, οι οποίες γίνονται τα δύο branches του Measure



Σχήμα 4.1: Κύκλωμα υπό συνθήκη αλλαγής θέσης δύο bit.

Κεφάλαιο 5

Παραδείγματα κβαντικών αλγορίθμων

Στο κεφάλαιο αυτό θα παρουσιάσουμε κάποια παραδείγματα μερικών γνωστών κβαντικών αλγορίθμων, καθώς και την υλοποίησή τους σε QML-NTUA. Ο έλεγχος της ορθής λειτουργίας της υλοποίησης έγινε με τη βοήθεια του interpreter της QML-NTUA τη δημιουργία του οποίου καλύψαμε στο κεφάλαιο 4

5.1 Ο αλγόριθμος του Deutsch

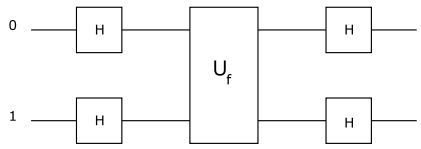
5.1.1 Περιγραφή

Ένας από τους πρώτους κβαντικούς αλγορίθμους που προτάθηκαν είναι ο αλγόριθμος του Deutsch και η επέκτασή του, ο αλγόριθμος των Deutsch-Jozsa. Το πρόβλημα το οποίο λύνει ο αλγόριθμος του Deutsch μπορεί να περιγραφεί ως εξής: Έστω πως έχουμε μία συνάρτηση $f(x) : \{0,1\} \rightarrow \{0,1\}$. Πώς μπορούμε να πούμε με σιγουριά αν η συνάρτηση αυτή είναι σταθερή διεξάγοντας μόνο έναν υπολογισμό της;

Είναι σαφές πως με κλασικά μέσα το παραπάνω πρόβλημα δεν έχει λύση. Υπάρχουν τέσσερις συναρτήσεις του τύπου της f και είναι οι $f(x) = \neg x, f(x) = x$ οι οποίες δεν είναι σταθερές και οι $f(x) = 0, f(x) = 1$ οι οποίες είναι σταθερές. Αφού έχουμε τη δυνατότητα για υπολογισμό της συνάρτησης μία μόνο φορά μπορούμε να γνωρίζουμε είτε το $f(0)$ είτε το $f(1)$. Γνωρίζοντας όμως μόνο το ένα από αυτά δεν είμαστε σε θέση να προσδιορίσουμε το ζητούμενο. Π.χ. αν γνωρίζουμε ότι $f(0) = 1$ τότε μπορεί $f(x) = \neg x$ ή $f(x) = 1$.

Η κβαντική λύση χρησιμοποιεί το τρικ του υπολογισμού της συνάρτησης με είσοδο μία υπέρθεση των 0 και 1. Έτσι, καταφέρνουμε εμμέσως να διαπιστώσουμε αν το αποτέλεσμα των $f(1)$ και $f(0)$ είναι το ίδιο. Σημειώνουμε πως ακόμα κι έτσι δεν μπορούμε να είμαστε σίγουροι με μία μόνο μέτρηση για το ποια συνάρτηση είναι η f.

Ένα κύκλωμα που υλοποιεί τον αλγόριθμο φαίνεται στο σχήμα 5.1.



Σχήμα 5.1: Κύκλωμα που υλοποιεί τον αλγόριθμο του Deutsch.

Η υλοποίηση του αλγορίθμου του Deutsch βασίζεται στη δημιουργία μία βοηθητικής συνάρτησης U_f. Η συνάρτηση αυτή θα δημιουργεί την αντιστοιχία $|\mathbf{i}, \mathbf{j}\rangle \rightarrow |\mathbf{i}, \mathbf{f}(\mathbf{i}) \oplus \mathbf{j}\rangle$. Έτσι όταν $f(x) = 0$ η U_f είναι ο ταυτοτικός μετασχηματισμός. Όταν $f(x) = 1$ η U_f είναι η πύλη cnot. Για $f(x) = x$ και $f(x) = \neg x$ μπορούμε εύκολα να εξάγουμε τους αντίστοιχους μετασχηματισμούς που αντιστοιχούν στη U_f. Όταν η f είναι μία σταθερή συνάρτηση το πρώτο bit του αποτελέσματος του αλγορίθμου είναι $|\mathbf{0}\rangle$ ενώ σε αντίθετη περίπτωση είναι $|\mathbf{1}\rangle$. Το δεύτερο bit είναι σε κάθε περίπτωση $|\mathbf{1}\rangle$.

Μία λογική ένσταση που μπορεί να έχει κανείς σε αυτό το σημείο είναι ότι ακόμα κι αν το κύκλωμα του σχήματος 5.1 λειτουργεί σωστά (κάτι που θα δείξουμε παρακάτω) η κατασκευή του

είναι αδύνατη βάση των περιορισμών του προβλήματος αφού για να φτιάξουμε την U_f θα πρέπει να ξέρουμε την f. Κάτι τέτοιο όμως δεν ισχύει, όπως θα φανεί ξεκάθαρα από το πρόγραμμα NTUA-QML που θα δώσουμε.

Η απόδειξη της ορθότητας του αλγορίθμου του Deutsch για συναρτήσεις ενός qbit είναι σχετικά απλή και μπορεί να γίνει με έλεγχο της εξόδου του κυκλώματος για τις τέσσερις συναρτήσεις ενός bit. Ας πάρουμε εδώ ως παράδειγμα την $f(x) = \neg x$. Τότε ο η U_f εκτελεί την διαδικασία $|\mathbf{i}, \mathbf{j}\rangle \rightarrow |\mathbf{i}, (\neg \mathbf{i}) \oplus \mathbf{j}\rangle$ δηλαδή

$$\begin{array}{lcl} |\mathbf{00}\rangle & \rightarrow & |\mathbf{01}\rangle \\ |\mathbf{01}\rangle & \rightarrow & |\mathbf{00}\rangle \\ |\mathbf{10}\rangle & \rightarrow & |\mathbf{10}\rangle \\ |\mathbf{11}\rangle & \rightarrow & |\mathbf{11}\rangle \end{array}$$

Είναι σαφές λοιπόν πως ο πίνακας του μετασχηματισμού της U_f είναι

$$U_f = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Οι άλλοι δύο μετασχηματισμοί του κυκλώματος είναι $H \otimes H$ οπότε το κύκλωμα υλοποιεί συνολικά τον μετασχηματισμό $(H \otimes H)U_f(H \otimes H)$ ο οποίος είναι ίσος με

$$(H \otimes H)U_f(H \otimes H) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}$$

Η εφαρμογή του πίνακα αυτού σε αρχική κατάσταση $|\mathbf{01}\rangle$ μας δίνει

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

το οποίο ήταν και το αναμενόμενο αποτέλεσμα αφού η $f(x)$ δεν είναι σταθερή συνάρτηση. Η εξακρίβωση μπορεί να γίνει με παρόμοιο τρόπο και για τις υπόλοιπες συναρτήσεις.

5.1.2 Υλοποίηση σε NTUA-QML

Όπως ξέρουμε η NTUA-QML δεν υποστηρίζει συναρτήσεις υψηλής τάξης, οπότε είναι δύσκολο να υλοποιήσουμε έναν αλγόριθμο ο οποίος έχει σαν είσοδο μία συνάρτηση, όπως ο αλγόριθμος του Deutsch. Μπορούμε όμως να χρησιμοποιήσουμε δυνατότητες της Haskell για να γεμίσουμε το κενό, μια που δεν ζητάμε κάτι ιδιαίτερα εξεζητημένο. Ορίζουμε λοιπόν μία συνάρτηση της Haskell με τον εξής τύπο

```
deutsch :: (TermA -> TermA) -> TermA
```

Η συνάρτηση deutsch θα παίρνει ως παράμετρο μία συνάρτηση της Haskell η οποία θα υλοποιεί την f. Για παράδειγμα η $f(x) = \neg x$ είναι η

```
qnot :: TermA -> TermA
qnot x = Ifm x Qfalse Qtrue
```

Η υλοποίηση της U_f σε NTUA-QML μπορεί να γίνει πολύ εύκολα ως εξής:

$$\mathbf{if}^{o}\ (f\ i)\ \mathbf{then}\ (\mathbf{if}^{-}\ j\ \mathbf{then}\ \mathrm{qfalse}\ \mathbf{else}\ \mathrm{qtrue})\ \mathbf{else}\ j$$

όπου i, j μεταβλητές που αναφέρονται στα δύο bit που λαμβάνει ως είσοδο η U_f.

Έτσι ο κώδικας σε Haskell της υλοποίησής μας είναι

```
deutsch :: (TermA -> TermA) -> TermA
deutsch f = Let "i" (had Qfalse)
              (Let "j" (had Qtrue)
               (Let "uf2" (If0 (f (Var "i")) (qnot (Var "j")) (Var "j"))
                (Let "uf1" (Var "i")
                 (Pair (had (Var "uf1")) (had (Var "uf2"))))))
```

Ο όρος had παραπάνω είναι απλώς η χρήση της πύλης Hadamard η οποία υλοποιείται με το if^{-}.

Μία σημαντική παρατήρηση που πρέπει να γίνει εδώ είναι ότι το πλήθος των bit του τελικού state εξαρτάται από την f. Συγκεκριμένα όταν η f είναι μία αντιστρέψιμη συνάρτηση το κύκλωμα 5.1 απεικονίζει όλη την αλήθεια. Όταν όμως η f είναι σταθερή, και επομένως μη αντιστρέψιμη, η f δεν μπορεί να υλοποιηθεί με έναν απλό ορθομοναδιαίο μετασχηματισμό. Επομένως θα χρειαστεί ένα qbit ακόμα ώστε να διατηρηθεί η αντιστρεψιμότητα. Ως εκ τούτου το κύκλωμά μας θα έχει στην πραγματικότητα τρεις εισόδους και εξόδους, όπου η τρίτη είσοδος θα ξεκινάει από $\vdash$ και θα περνά από την U_f και θα καταλήγει σε $\dashv$.

Θα μπορούσε κανείς να υποστηρίξει ότι αυτό αποτελεί μία αδυναμία της μεθόδου μας, αφού προσπαθούμε να λύσουμε ένα πρόβλημα ήδη λυμένο. Ο διαχωρισμός των σταθερών συναρτήσεων από τις αντιστρέψιμες μπορεί να γίνει από το πλήθος των εισόδων-εξόδων που χρειάζονται για να υλοποιηθούν. Στην πραγματικότητα όμως αυτό δεν αφαιρεί από την αξία του αλγορίθμου του Deutsch, αφού για παράδειγμα αντί για την προηγούμενη qnot θα μπορούσαμε να χρησιμοποιήσουμε την

```
qnot2 x = Pair (Ifm x Qfalse Qtrue) Qfalse
```

Η qnot2 παράγει επίσης ένα επιπλέον bit, επομένως δεν μπορούμε να την ξεχωρίσουμε μόνο από τον αριθμό εισόδων και εξόδων από μία σταθερή συνάρτηση. Ο αλγόριθμος του Deutsch όμως εξακολουθεί να δουλεύει όπως πριν.

Μία επέκταση του αλγορίθμου του Deutsch αποτελεί ο αλγόριθμος των Deutsch-Jozsa. Στην περίπτωση αυτή η συνάρτηση f είναι μία συνάρτηση $f(x) : \{0,1\}^n \rightarrow \{0,1\}$ με τον περιορισμό ότι θα είναι είτε σταθερή είτε ισορροπημένη, δηλαδή θα δίνει αποτέλεσμα 0 για τις μισές πιθανές εισόδους και 1 για τις άλλες μισές. Για $n = 1$ όλες οι συναρτήσεις έχουν αυτήν την ιδιότητα και ο αλγόριθμος εκφυλίζεται στον αλγόριθμο που ήδη παρουσιάσαμε. Για $n > 1$ και πάλι υπολογίζουμε την U_f με τον ίδιο τρόπο δηλαδή $|\mathbf{i}, \mathbf{j}\rangle \rightarrow |\mathbf{i}, \mathbf{f}(\mathbf{i}) \oplus \mathbf{j}\rangle$, όπου i τα πρώτα n bit του state. Με άλλα λόγια δίνουμε σαν είσοδο στη U_f n bits τα οποία βρίσκονται στην κατάσταση $(\frac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle))^n$ και 1 bit που βρίσκεται στην κατάσταση $\frac{1}{\sqrt{2}}(|\mathbf{0}\rangle - |\mathbf{1}\rangle)$.

Η επέκταση του κώδικα που παραθέσαμε παραπάνω ώστε να υλοποιηθεί ο αλγόριθμος των Deutsch-Jozsa για κατάλληλη συνάρτηση f συνίσταται απλώς στην αλλαγή του **let** $i =$ had qfalse σε (had qfalse, had qfalse) για δύο bit ή κάτι αντίστοιχο για περισσότερα.

5.2 Ο αλγόριθμος του Grover

5.2.1 Περιγραφή

Ο αλγόριθμος του Grover ([Grov96]) λύνει ένα αρκετά σημαντικό πρόβλημα πετυχαίνοντας μάλιστα απόδοση η οποία είναι αποδεδειγμένα καλύτερη από την απόδοση που μπορεί να έχει ο καλύτερος κλασικός αλγόριθμος. Το εν λόγω πρόβλημα είναι αυτό της αναζήτησης σε έναν μη ταξινομημένο πίνακα. Η πολυπλοκότητα του καλύτερου κλασικού αλγορίθμου είναι ως γνωστόν $O(\frac{n}{2})$ στη μέση περίπτωση και $O(n)$ στη χειρότερη. Ο αλγόριθμος του Grover είναι πιθανοτικός με πολυπλοκότητα $O(\sqrt{n})$.

Μπορούμε να εκφράσουμε γενικότερα το πρόβλημα ως εξής: έστω πως έχουμε μία συνάρτηση $f(x) : \{0, \cdots, 2^n - 1\} \rightarrow \{0,1\}$, για την οποία ισχύει για ένα μοναδικό x_0 $f(x_0) = 1$ καθώς και $f(x) = 0, \forall x \neq x_0$. Αυτό είναι ισοδύναμο με το να έχουμε έναν πίνακα 2^n στοιχείων και να

αναζητούμε ένα στοιχείο το οποίο υπάρχει ακριβώς μία φορά στον πίνακα. Τότε η συνάρτηση f είναι απλώς μία συνάρτηση που συγκρίνει το στοιχείο που βρίσκεται στη θέση x με το ζητούμενο.

Αν υποθέσουμε πως μπορούμε να υλοποιήσουμε κβαντικά την f αυτό σημαίνει πως μπορούμε να υλοποιήσουμε κβαντικά και την εξής συνάρτηση $g(x) : |\mathbf{x}\rangle \to |\mathbf{x}\rangle, \forall x \neq x_0$ και $|\mathbf{x_0}\rangle \to -|\mathbf{x_0}\rangle$. Με άλλα λόγια η g είναι ο ταυτοτικός μετασχηματισμός όταν έχει είσοδο οτιδήποτε εκτός από τη λύση του προβλήματος x_0, ενώ όταν έχει είσοδο x_0 επιστρέφει πάλι x_0 αλλά έχοντας πολλαπλασιάσει το πλάτος με -1. Ο πίνακας που αντιστοιχεί λοιπόν στη g δεν είναι παρά ο μοναδιαίος πίνακας με μόνη διαφορά το στοιχείο της διαγωνίου που αντιστοιχεί στο x_0 το οποίο είναι -1. Προφανώς ο πίνακας αυτός είναι ορθομοναδιαίος.

Ορίζουμε τώρα τον diffusion operator (τελεστή διάχυσης) ο οποίος έχει πίνακα $I-2P$ όπου P ένας $2^n \times 2^n$ πίνακας που όλα τα στοιχεία του είναι ίσα με 2^{-n}. Είναι εύκολο να διαπιστώσουμε πως $P^2 = P$, όπως και ότι $P^* = P$. Επομένως έχουμε $(I-2P)(I-2P)^* = (I-2P)(I-2P) = I-4P+4P^2 = I-4P+4P = I$. Δείξαμε λοιπόν πως ο diffusion operator είναι ορθομοναδιαίος και επομένως μπορούμε να τον χρησιμοποιήσουμε.

Ο diffusion operator λέγεται και inversion about average operator. Αν έχει είσοδο μία κβαντική κατάσταση όπου ο μέσος όρος των πλατών είναι μ, τότε κάθε ιδιοδιάνυσμα που είχε πλάτος μεγαλύτερο από μ θα αποκτήσει πλάτος μικρότερο από μ και μάλιστα η διαφορά του από το μ θα παραμείνει αναλλοίωτη. Ομοίως αν ένα ιδιοδιάνυσμα είχε πλάτος μικρότερο από μ θα αποκτήσει με τον ίδιο τρόπο πλάτος μεγαλύτερο από μ. Είναι σαφές ότι ο μέσος όρος μ δεν μεταβάλλεται με χρήση του diffusion operator. Φυσικά αναφερόμαστε σε πραγματικά πλάτη (ο αλγόριθμος του Grover δεν ασχολείται με μιγαδικά πλάτη).

Ο δρόμος τώρα είναι ανοικτός για τη λύση του προβλήματος. Έστω πως θέλουμε να κάνουμε αναζήτηση σε έναν πίνακα με $N = 2^n$ στοιχεία. Χρησιμοποιούμε n qbit τα οποία αρχικοποιούμε σε $|\mathbf{0}\rangle$. Χρησιμοποιώντας την πύλη Hadamard φέρνουμε τα n qbit σε μια υπέρθεση όλων των πιθανών καταστάσεων με ίση πιθανότητα. Έπειτα χρησιμοποιούμε τη συνάρτηση g που ορίσαμε πιο πάνω. Μετά την εφαρμογή της όλα τα ιδιοδιανύσματα έχουν πλάτος $\frac{1}{N}$ εκτός από το $|\mathbf{x_0}\rangle$ που έχει πλάτος $-\frac{1}{N}$. Όλα λοιπόν τα ιδιοδιανύσματα βρίσκονται πάνω από τον μέσο όρο εκτός από το $|\mathbf{x_0}\rangle$. Ως εκ τούτου εφαρμογή του diffusion operator θα αυξήσει σημαντικά την πιθανότητα μια μέτρηση να μας δώσει αποτέλεσμα x_0 μειώνοντας ταυτόχρονα την πιθανότητα όποιουδήποτε άλλου αποτελέσματος. Στη συνέχεια μπορούμε να εφαρμόσουμε ξανά την g και τον diffusion operator όσες φορές χρειάζεται για να αυξήσουμε ικανοποιητικά την πιθανότητα μέτρησης του x_0.

Μπορεί να αποδειχθεί πως με $O(\sqrt{N})$ διαδοχικές εφαρμογές των δύο παραπάνω τελεστών καταλήγουμε σε κατάσταση που δίνει το επιθυμητό αποτέλεσμα με πιθανότητα μεγαλύτερη του $\frac{1}{2}$. Ο ακριβής αριθμός των επαναλήψεων που χρειάζεται εξαρτάται από το N, όμως έχει επίσης δειχθεί ότι η πολυπλοκότητα κάθε δυνατού κβαντικού αλγορίθμου που λύνει το πρόβλημα είναι $\Omega(\sqrt{N})$, δηλαδή ο αλγόριθμος αυτός είναι βέλτιστος.

Η ορθότητα του αλγορίθμου μπορεί να δειχθεί ως εξής: έστω πως η κατάσταση του συστήματος είναι τέτοια ώστε το ιδιοδιάνυσμα $|\mathbf{x_0}\rangle$ έχει πιθανότητα k και όλα τα υπόλοιπα l. Θα δείξουμε πως αν $0 < k < \frac{1}{\sqrt{2}}$ και $l > 0$ τότε η μεταβολή Δk του k η οποία προκύπτει από διαδοχική εφαρμογή των δύο τελεστών που δώσαμε είναι φραγμένη κάτω από το $\frac{1}{2\sqrt{N}}$. Επίσης ότι το l εξακολουθεί να είναι θετικό.

Από τον ορισμό του diffusion operator είναι εύκολο να δούμε πως αν εφαρμοστεί σε μία κατάσταση που ένα ιδιοδιάνυσμα έχει πλάτος k_1 και όλα τα άλλα l_1 θα μας δώσει μία κατάσταση όπου το ίδιο ιδιοδιάνυσμα έχει πλάτος $k_2 = (\frac{2}{N} - 1)k_1 + 2\frac{N-1}{N}l_1$ και τα υπόλοιπα πλάτη θα γίνουν $l_2 = (\frac{2}{N}k_1 + \frac{N-2}{N}l_1$. Αυτό προκύπτει από το ότι ο προηγούμενος μέσος όρος είναι $\mu = \frac{k_1+(N-1)l_1}{N}$ και ότι ο diffusion operator όταν εφαρμοστεί μετασχηματίζει το πλάτος k στο $\mu - (k - \mu)$ όπως είπαμε όταν δείξαμε πως είναι ισοδύναμος με τη λειτουργία inversion about average. Μία ενδιαφέρουσα ιδιότητα είναι πως $k_2^2 + (N-1)l_2^2 = k_1^2 + (N-1)l_1^2$, η οποία μπορεί να αποδειχθεί και με πράξεις, αλλά προκύπτει επίσης από το γεγονός ότι τα k, l υψωμένα στο τετράγωνο εκφράζουν πιθανότητες και η συνολική πιθανότητα δεν μεταβάλλεται.

Επίσης από τα παραπάνω συμπεραίνουμε πως αν $k_1 < 0$ και $l_1 > 0$, τότε μετά την εφαρμογή του diffusion operator $k_2 > 0$ και αν επιπλέον ίσχυε $|\frac{k_1}{l_1}| < \sqrt{N}$ τότε και $l_2 > 0$. Αναλυτικά, αφού $k_2 = (\frac{2}{N} - 1)k_1 + 2\frac{N-1}{N}l_1$ για $N > 2 \Rightarrow \frac{2}{N} - 1 < 0$ και αφού $k_1 < 0$ έχουμε $(\frac{2}{N} - 1)k_1 > 0$. Επίσης από $l_1 > 0$ έχουμε $2\frac{N-1}{N}l_1 > 0$ άρα $k_2 > 0$. Επίσης αν $|\frac{k_1}{l_1}| < \sqrt{N}$ τότε για $N \geq 4$ έχω $|\frac{k_1}{l_1}| < \frac{N-2}{2}$ κάτι που εξασφαλίζει πως $l_2 > 0$.

Θα δείξουμε τώρα το ζητούμενο. Έστω πως πριν την εφαρμογή της g έχουμε πλάτος $k > 0$ για τη λύση και $l > 0$ για όλα τα υπόλοιπα ιδιοδιανύσματα. Τότε μετά την g έχουμε $k_1 < 0$ για τη λύση και $l_1 < 0$ για τα υπόλοιπα. Αν $0 < k < \frac{1}{\sqrt{2}} \Rightarrow l_1 > \frac{1}{2\sqrt{N}}$ διότι $k_1^2 + (N-1)l_1^2$. Έχω $\Delta k = k_2 - k = (\frac{2}{N} - 1)k_1 + 2\frac{N-1}{N}l_1 - k = -(\frac{2}{N} - 1)k + 2\frac{N-1}{N}l_1 - k = -\frac{2}{N}k + 2\frac{N-1}{N}l_1$ το οποίο για $N \geq 16$ μας δίνει το ζητούμενο $\Delta k > \frac{1}{2\sqrt{N}}$. Επίσης από $k < \frac{1}{\sqrt{2}}$ και $l > \frac{1}{N\sqrt{2}}$ προκύπτει ότι $|\frac{k_1}{l_1}| < \sqrt{N}$ κάτι που μας δίνει και ότι $l_2 > 0$.

Δείξαμε λοιπόν πως σε περίπτωση που βρισκόμαστε σε μία κατάσταση όπου το πλάτος της λύσης μας δίνει πιθανότητα μικρότερη από $\frac{1}{2}$ η εφαρμογή του αλγορίθμου θα αυξήσει το πλάτος κατά τουλάχιστον $\frac{1}{2\sqrt{N}}$. Φυσικά αν το πλάτος είναι ήδη μεγαλύτερο του $\frac{1}{\sqrt{2}}$ η εφαρμογή του αλγορίθμου πιθανώς θα το μειώσει, οπότε είναι καλό να είμαστε προσεκτικοί όσον αφορά τον ακριβή αριθμό επαναλήψεων που απαιτείται για συγκεκριμένο N.

5.2.2 Υλοποίηση σε NTUA-QML

Παρουσιάζουμε τώρα μία υλοποίηση που επιδεικνύει τη σωστή λειτουργία του αλγορίθμου του Grover σε NTUA-QML. Για λόγους απλότητας χρησιμοποιούμε την ειδική περίπτωση που $N = 4$. Σε αυτήν την περίπτωση μία και μόνη εφαρμογή του αλγορίθμου είναι αρκετή ώστε να πάρουμε το σωστό αποτέλεσμα με πιθανότητα 1.

Αρχικά υλοποιούμε τον diffusion operator. Αρκούν δύο if$^-$ ως εξής:

```
if⁻ x1 then
    if⁻ x2 then
        {{−(qfalse, qfalse) | − (qfalse, qtrue)}
          | {−(qtrue, qfalse) | (qtrue, qtrue)}}
        else
        {{−(qfalse, qfalse) | − (qfalse, qtrue)}
          | − {−(qtrue, qfalse) | (qtrue, qtrue)}}
    else
    if⁻ x2 then
        {−{(qfalse, qfalse) | − (qfalse, qtrue)}
          | {−(qtrue, qfalse) | − (qtrue, qtrue)}}
        else
        {{(qfalse, qfalse) | − (qfalse, qtrue)}
          | {−(qtrue, qfalse) | − (qtrue, qtrue)}}
```

Όπως μπορούμε εύκολα να διαπιστώσουμε το παραπάνω αντιστοιχεί στο μετασχηματισμό

$$\frac{1}{2}\begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{pmatrix}$$

που είναι η μορφή που παίρνει ο diffusion operator όταν $N = 4$.

Κατόπιν υλοποιούμε τη συνάρτηση g η οποία θα πραγματοποιεί το conditional phase

change

$$
\begin{array}{lll}
\textbf{if}^{-}\ x1 & \textbf{then} & \\
 & \textbf{if}^{-}\ x2 & \textbf{then} \\
 & & (\mathrm{qtrue}, \mathrm{qtrue}) \\
 & & \textbf{else} \\
 & & -(\mathrm{qtrue}, \mathrm{qfalse}) \\
 & \textbf{else} & \\
 & \textbf{if}^{-}\ x2 & \textbf{then} \\
 & & (\mathrm{qfalse}, \mathrm{qtrue}) \\
 & & \textbf{else} \\
 & & (\mathrm{qfalse}, \mathrm{qfalse})
\end{array}
$$

όπου έχουμε υποθέσει πως η σωστή απάντηση είναι το $|\mathbf{10}\rangle$. Με $-(\mathrm{qtrue}, \mathrm{qfalse})$ συντομογραφούμε το $(\{\mathrm{qfalse} \mid -\mathrm{qtrue}\}, \mathrm{qfalse})$.

Δεν μένει παρά να συναρμολογήσουμε το τελικό πρόγραμμα, το οποίο όπως είπαμε στη συγκεκριμένη περίπτωση χρειάζεται μόνο ένα loop του αλγορίθμου.

$$
\begin{array}{lll}
\textbf{let}\ x1 & (\mathrm{had\ qfalse})\ \textbf{in} & \\
 & \textbf{let}\ x2 & (\mathit{had}\ \mathrm{qfalse})\ \textbf{in} \\
 & \textbf{let}\ \mathit{tmp} & (\mathrm{check}\ x1\ x2)\ \textbf{in} \\
 & & \mathrm{average}\ x1\ x2
\end{array}
$$

Όπως και πριν, με had συντομογραφούμε τη χρήση της πύλης Hadamard, ενώ με check και average εννοούμε τις δύο διαδικασίες που δώσαμε προηγουμένως. Μία εκτέλεση του προγράμματος μας δίνει το αναμενόμενο αποτέλεσμα φέρνοντας τα bit του αποτελέσματος στην κατάσταση $|\mathbf{10}\rangle$.

5.3 Ο αλγόριθμος παραγοντοποίησης του Shor

5.3.1 Περιγραφή

Θα εξετάσουμε τώρα τον πιο σημαντικό ίσως αλγόριθμο που έχει παρουσιαστεί ως τώρα, τον αλγόριθμο του Shor ([Shor97]). Μιλώντας με κλασικούς όρους θα μπορούσαμε να πούμε πως είναι ένας αλγόριθμος ο οποίος λύνει σε πολυωνυμικό χρόνο χρησιμοποιώντας ένα μαντείο το οποίο μπορεί να υπολογίσει την περίοδο μιας συνάρτησης επίσης σε πολυωνυμικό χρόνο. Η καινοτομία έγκειται στην κβαντική υλοποίηση του μαντείου.

Το πρόβλημα μπορεί να διατυπωθεί ως εξής: έστω ένας φυσικός αριθμός N, να βρεθεί ένας μη τετριμμένος διαιρέτης του (δηλαδή ένας διαιρέτης του εκτός των $1, N$). Έχουν προταθεί πολλοί αλγόριθμοι που λύνουν αυτό το πρόβλημα, όπως η μέθοδος βάσεων παραγοντοποίησης και η μέθοδος Number Field Sieve. Όλες όμως έχουν εκθετική ως προς το μήκος της εισόδου πολυπλοκότητα. Ο αλγόριθμος του Shor είναι ο μόνος ως τώρα γνωστός πιθανοτικός αλγόριθμος πολυωνυμικής πολυπλοκότητας που λύνει το πρόβλημα της παραγοντοποίησης.

Τα βήματα του αλγορίθμου είναι τα εξής:

1. Επιλέγουμε έναν τυχαίο αριθμό $\alpha < N$

2. Υπολογίζουμε το $\gcd(\alpha, N)$. Μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο του Ευκλείδη. Αν ο μέγιστος κοινός διαιρέτης είναι $\neq 1$ τότε βρήκαμε έναν μη τετριμμένο διαιρέτη και το πρόβλημα λύθηκε

3. Υπολογίζουμε την περίοδο της συνάρτησης $f(x) = \alpha^x \bmod N$. Στο βήμα αυτό χρησιμοποιούμε το κβαντικό μαντείο το οποίο θα περιγραφεί παρακάτω. Έστω πως η περίοδος που υπολογίστηκε είναι r.

4. Αν ο r είναι περιττός επιστρέφουμε στο βήμα 1

5. Αν $\alpha^{\frac{r}{2}} \equiv -1(\mathrm{mod} N)$ επιστρέφουμε στο βήμα 1

6. Σε αντίθετη περίπτωση έχω $\alpha^{\frac{r}{2}} \equiv 1(\mathrm{mod}\ N) \Rightarrow \alpha^r \equiv 1(\mathrm{mod}\ N) \Rightarrow \alpha^r - 1 = kN \Rightarrow (\alpha^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) = kN$. Χρησιμοποιώντας τον αλγόριθμο του μέγιστου κοινού διαιρέτη μπορούμε να βρούμε έναν μη τετριμμένο διαιρέτη του N, διότι $\alpha^{\frac{r}{2}} + 1 \neq mN$ όπως εξασφαλίσαμε στο προηγούμενο βήμα

Θα ασχοληθούμε τώρα με το κβαντικό μέρος του αλγορίθμου, το οποίο είναι και το πιο ενδιαφέρον. Θέλουμε να υπολογίσουμε την περίοδο της συνάρτησης $f(x) = \alpha^x \mathrm{mod}\ N$. Ακολουθούμε τα εξής βήματα:

1. Ξεκινάμε με δύο καταχωρητές των $n = \log_2 N$ qbits ο καθένας. Ο πρώτος αρχικοποιείται σε μία υπέρθεση όλων των δυνατών καταστάσεων με την ίδια πιθανότητα (κάτι τέτοιο μπορεί να επιτευχθεί με πύλες Hadamard) ενώ ο δεύτερος αρχικοποιείται στο $|\mathbf{0}\rangle$.

2. Με την προϋπόθεση ότι έχουμε υλοποιήσει την f ως κβαντική συνάρτηση την εφαρμόζουμε στο σύστημα. Τώρα ο δεύτερος καταχωρητής περιέχει το $f(x)$ όπου x η κατάσταση στην οποία ήταν ο πρώτος

3. Εφαρμόζουμε τον κβαντικό μετασχηματισμό Fourier στον πρώτο καταχωρητή και μετράμε το περιεχόμενό του δεύτερου. Τώρα μία μέτρηση του πρώτου έχει μεγάλη πιθανότητα να μας δώσει την περίοδο της συνάρτησης

Ας προσπαθήσουμε να εξηγήσουμε λίγο πιο αναλυτικά γιατί ο παραπάνω αλγόριθμος δουλεύει. Μετά το βήμα 1 η κατάσταση του συστήματος είναι $\frac{1}{\sqrt{N}}(1\ 1\ \cdots 1)^T \otimes (1\ 0\ \cdots 0)^T$. Η εφαρμογή της f θα δημιουργήσει entanglement ανάμεσα στους δύο καταχωρητές. Έτσι αν μετά την εφαρμογή της f αποπειραθούμε μία μέτρηση στον δεύτερο καταχωρητή και πάρουμε αποτέλεσμα y_0 τότε ο πρώτος καταχωρητής θα έρθει σε μία κατάσταση η οποία θα είναι υπέρθεση των x για τα οποία ισχύει $f(x) = y_0$. Δηλαδή αν s το μικρότερο τέτοιο x θα έρθει σε μία υπέρθεση των $|\mathbf{s} + \mathbf{rj}\rangle,\ 0 \leq j \leq \lceil \frac{N}{r} \rceil - 1$. Δυστυχώς δεν είναι δυνατόν από μία τέτοια υπέρθεση να συμπεράνουμε το r γιατί το s είναι τυχαίο. Αν όμως εφαρμόσουμε έναν μετασχηματισμό Fourier στον πρώτο καταχωρητή το φάσμα πιθανοτήτων που θα προκύψει θα είναι ανεξάρτητο του s. Συγκεκριμένα αν το r διαιρεί το N τότε θα έχουμε μη μηδενική πιθανότητα να μετρήσουμε μόνο πολλαπλάσια του N/r. Αλλά ακόμα και σε περίπτωση που το r δεν διαιρεί το N το φάσμα θα εμφανίσει κορυφές στα πολλαπλάσια του N/r. Έτσι μία μέτρηση θα μας δώσει με μεγάλη πιθανότητα μία τιμή c κοντά στο $\lambda N/r$. Φυσικά το N είναι γνωστό, οπότε ουσιαστικά αποκτούμε μία προσέγγιση του λ/r. Για να μπορέσουμε από αυτό το κλάσμα να προσδιορίσουμε το r θα πρέπει οι λ, r να είναι πρώτοι μεταξύ τους, κάτι που συμβαίνει με πιθανότητα μεγαλύτερη του $1/lnr$, οπότε θα χρειαστούμε το πολύ $O(n)$ προσπάθειες για να βρούμε το r με πιθανότητα που μπορεί να πλησιάσει όσο θέλουμε το 1. Φυσικά η επαλήθευση μπορεί να γίνει απλά με τον έλεγχο για το αν $f(x) = f(x + r)$.

5.3.2 Ο κβαντικός μετασχηματισμός Fourier

Όπως είδαμε ο μετασχηματισμός Fourier είναι ένα κρίσιμο κομμάτι του αλγορίθμου του Shor. Σε αυτήν την ενότητα ορίζουμε αναλυτικά τον μετασχηματισμό και παρουσιάζουμε ένα παράδειγμα υλοποίησής του σε NTUA-QML.

Ο κβαντικός μετασχηματισμός σε braket notation, με είσοδο μία κατάσταση $|\mathbf{x}\rangle$ δίνεται ως

$$|\mathbf{x}\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i x k/N} |\mathbf{k}\rangle$$

Σε μορφή πίνακα είναι

$$\frac{1}{\sqrt{N}}[\omega^{kl}], \omega = e^{2\pi i/N}$$

Με τη βοήθεια του θεωρήματος Plancherel μπορεί να δειχθεί πως ο μετασχηματισμός αυτός είναι ορθομοναδιαίος.

Ας δούμε αναλυτικά τη μορφή του πίνακα του μετασχηματισμού ώστε να βοηθηθούμε στη μετάφρασή του σε NTUA-QML. Για δύο bit ο πίνακας παίρνει τη μορφή

$$\frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{\pi i/2} & e^{\pi i} & e^{3\pi i/2} \\ 1 & e^{\pi i} & e^{2\pi i} & e^{3\pi i} \\ 1 & e^{3\pi i/2} & e^{3\pi i} & e^{9\pi i/2} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

Παρατηρούμε πως τα στοιχεία της πρώτης στήλης μπορούν να γραφούν ως $\frac{1}{2}\begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Ομοίως τα στοιχεία της δεύτερης γράφονται ως $\frac{1}{2}\begin{pmatrix} 1 \\ e^{\pi i} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ e^{\pi i/2} \end{pmatrix}$. Συνολικά τα στοιχεία της k στήλης γράφονται ως $\frac{1}{2}\begin{pmatrix} 1 \\ e^{k\pi i} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ e^{k\pi i/2} \end{pmatrix}$ αν η αρίθμηση των στηλών ξεκινά από το 0. Το σκεπτικό αυτό μπορεί να επεκταθεί και στον μετασχηματισμό Fourier n bits. Έτσι τα στοιχεία της k στήλης είναι

$$2^{-n/2}\begin{pmatrix} 1 \\ e^{k\pi i} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ e^{k\pi i/2} \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} 1 \\ e^{k\pi i/2^{n-1}} \end{pmatrix}$$

Ορίζουμε τώρα σε Haskell τη βοηθητική συνάρτηση make_ifms η οποία παίρνει ως παράμετρο τους όρους που δίνουν τα bit στα οποία θα εφαρμοστεί ο μετασχηματισμός, μία συνάρτηση που με είσοδο έναν ακέραιο μας δίνει τον όρο του αντίστοιχου branch, καθώς και ένα πεδίο αποδεκτών ακεραίων για τη συνάρτηση αυτή. Αποτέλεσμα είναι το δέντρο από if$^-$ που επιτελούν τον επιθυμητό μετασχηματισμό.

```
make_ifms :: [TermA] -> (Int -> TermA) -> [Int] -> TermA
make_ifms [] f dom | length dom == 1 =  f (head dom)
make_ifms l f dom  | (2^(length l)) == (length dom) =
                 let branch1= make_ifms (tail l) f
                         (take (div (length dom) 2) dom) in
                 let branch2= make_ifms (tail l) f
                         (dom \\ (take (div (length dom) 2) dom)) in
                 Ifm (head l) branch2 branch1
```

Για να χρησιμοποιήσουμε την make_ifms θα πρέπει να ορίσουμε μία συνάρτηση η οποία με όρισμα έναν ακέραιο n θα μας δίνει τον όρο που αντιστοιχεί στην n στήλη του μετασχηματισμού Fourier. Αυτό είναι εύκολο να γίνει με την ανάλυση του μετασχηματισμού σε γινόμενα που κάναμε παραπάνω. Ας δούμε την υλοποίηση σε Haskell αυτής της συνάρτησης για 3 qbit.

```
dft_branch :: Int -> TermA
dft_branch n = Pair (Pair
                 (Super con2 Qfalse (con2*(coef (4*n))) Qtrue)
                 (Super con2 Qfalse (con2*(coef (2*n))) Qtrue))
                 (Super con2 Qfalse (con2*(coef n)) Qtrue)
```

Στον παραπάνω κώδικα con2 είναι η σταθερά $\frac{1}{\sqrt{2}}$ και coef είναι μία συνάρτηση που επιστρέφει το $e^{n\pi i/4}$.

Μπορούμε να χρησιμοποιήσουμε τα παραπάνω για να ελέγξουμε μία εύκολη περίπτωση του μετασχηματισμού Fourier. Παρακάτω βλέπουμε τον κώδικα Haskell που παράγει έναν όρο ο οποίος εφαρμόζει τον μετασχηματισμό Fourier σε τρία bit.

```
test_dft = Let "a" (Super con2 Qfalse con2 Qtrue)
           (Let "b" (Super con2 Qfalse con2 Qtrue)
           (Let "c" (Qtrue)
             (make_ifms [(Var "a"),(Var "b"),(Var "c")] dft_branch [0..7])))
```

Αρχικά τα a,b,c σχηματίζουν μία κατάσταση με περίοδο 2, αφού έχουμε μία υπέρθεση των $|\mathbf{xy1}\rangle$. Η εφαρμογή του μετασχηματισμού Fourier δίνει μία υπέρθεση των $|\mathbf{100}\rangle$ και $|\mathbf{000}\rangle$. Προφανώς αν μία μέτρηση δώσει αποτέλεσμα $|\mathbf{000}\rangle$ δεν έχουμε αποτέλεσμα και πρέπει να ξαναπροσπαθήσουμε. Όμως με πιθανότητα $1/2$ μία μέτρηση θα μας δώσει $|\mathbf{100}\rangle = 4$. Άρα έχουμε $\frac{\lambda}{r} = 4/8 = 1/2 \rightarrow r = 2$ που είναι και το ζητούμενο. Χρήσιμο είναι να παρατηρήσουμε πως σε περίπτωση που το c είναι Qfalse το αποτέλεσμα δεν αλλοιώνεται.

Ας δούμε τώρα και την περίπτωση που το b είναι Qfalse και το c Qtrue, δηλαδή έχουμε $|\mathbf{001}\rangle + |\mathbf{101}\rangle$. Η εφαρμογή του Fourier μας δίνει $|\mathbf{000}\rangle + |\mathbf{010}\rangle + |\mathbf{100}\rangle + |\mathbf{110}\rangle$. Ας δούμε τα πιθανά αποτελέσματα για την κάθε κατάσταση στον πίνακα 5.1.

Πίνακας 5.1: Εύρεση περιόδου συνάρτησης 3 bit.

Αποτέλεσμα μέτρησης	$c/N = \lambda/r$	r
0	0	-
2	$1/4$	4
4	$1/2$	2
6	$3/4$	4

Παρατηρούμε πως και πάλι με πιθανότητα $1/2$ μετράμε ένα αποτέλεσμα που μας οδηγεί στη σωστή περίοδο. Για άλλη μια φορά να σημειώσουμε πως αν αλλάξουμε τις τιμές των b,c το αποτέλεσμα αυτό δεν αλλάζει αρκεί αυτές να παραμείνουν κλασικές (δηλαδή Qtrue ή Qfalse).

5.3.3 Υλοποίηση κβαντικής ύψωσης σε εκθέτη modulo N

Το άλλο θεμελιώδες πρόβλημα που πρέπει να λυθεί ώστε να υλοποιηθεί το κβαντικό μέρος του αλγορίθμου του Shor είναι η κβαντική υλοποίηση της $f(x) = \alpha^x \bmod N$. Αυτό το πρόβλημα, παρ' ότι στους κλασικούς υπολογισμούς είναι μάλλον απλό, στους κβαντικούς εμφανίζει αρκετά μεγάλη δυσκολία. Ας μην ξεχνάμε πως οι κβαντικές συναρτήσεις πρέπει να τηρούν κάποιες αρκετά αυστηρές προϋποθέσεις όπως να είναι αντιστρέψιμες καθώς και το ότι μέσα στις στοιχειώδεις λειτουργίες της NTUA-QML αλλά και των περισσοτέρων άλλων κβαντικών γλωσσών που υπάρχουν στη βιβλιογραφία δεν εμφανίζονται στοιχειώδεις λειτουργίες όπως πρόσθεση ή πολλαπλασιασμός κβαντικών ποσοτήτων (πόσο μάλλον υπολογισμός δύναμης).

Παρ' όλα αυτά η υλοποίηση της $f(x)$ είναι δυνατή, αλλά θα πρέπει να ξεκινήσουμε από τα βασικά: πρώτα θα κατασκευάσουμε μία διαδικασία που κάνει πρόσθεση, έπειτα μία που κάνει πολλαπλασιασμό και τέλος μία που υλοποιεί την ζητούμενη συνάρτηση. Στο υπόλοιπο της ενότητας δίνουμε ένα παράδειγμα υλοποίησης της πρόσθεσης σε NTUA-QML κι έπειτα σκιαγραφούμε την πορεία που θα πρέπει κανείς να ακολουθήσει ώστε να φτάσει από εκεί στην υλοποίηση της $f(x)$.

Πρόσθεση

Ας ασχοληθούμε αρχικά με το πρόβλημα της πρόσθεσης δύο δυαδικών αριθμών. Έστω πως σε έναν καταχωρητή των n bit θέλουμε να προσθέσουμε τον αριθμό α. Ο μετασχηματισμός που θα εφαρμόσουμε θα είναι τέτοιος ώστε να δημιουργείται η αντιστοιχία $|\mathbf{i}\rangle \rightarrow |\mathbf{i} + \alpha \bmod \mathbf{2^n}\rangle$. Φυσικά η αντιστοιχία αυτή είναι αντιστρέψιμη και μπορεί να πραγματοποιηθεί με έναν ορθομοναδιαίο πίνακα. Αν $e_0, e_1, \cdots, e_{2^n-1}$ η κανονική βάση του χώρου διαστάσεων 2^n, τότε ο πίνακας που πραγματοποιεί αυτόν το μετασχηματισμό είναι αυτός που έχει στήλες $e_\alpha, e_{\alpha+1}, \cdots, e_{\alpha+2^n-1 \bmod 2^n}$.

Ας δούμε τον κώδικα Haskell ο οποίος παράγει όρους οι οποίοι διεξάγουν προσθέσεις:

```
add_to_terms:: TermA -> Int -> TermA
add_to_terms t 0 = t
add_to_terms Qfalse 1 = Qtrue
add_to_terms Qtrue 1 = Qfalse
add_to_terms (Pair t1 Qfalse) n = add_to_terms (Pair t1 Qtrue) (n-1)
add_to_terms (Pair t1 Qtrue) n = add_to_terms
                                 (Pair (add_to_terms t1 1) Qfalse) (n-1)

false_terms:: Int -> TermA
false_terms 1 = Qfalse
false_terms n = Pair (false_terms (n-1)) Qfalse

make_byte:: Int -> Int -> TermA -- how many bits, which number
make_byte n 0 = false_terms n
make_byte n m | m<(2^n)= add_to_terms (false_terms n) m
              | otherwise = make_byte n (mod m (2^n))

addition_term:: [TermA] -> Int -> TermA
addition_term l n = make_ifms l (make_byte (length l))
                                [n..n+(2^(length l))-1 ]

adder:: [TermA] -> [TermA] -> [Int] -> TermA
adder [] b n | length n==1 = addition_term b (head n)
adder a b n = If0 (head a)
                (adder (tail a) b (n \\ (take (div (length n) 2) n)))
                (adder (tail a) b (take (div (length n) 2) n) )
```

Η κυριότερη συνάρτηση εδώ είναι η adder. Παίρνει δύο λίστες από όρους, η πρώτη εκ των οποίων χρησιμοποιείται ως ελεγκτής για τον μετασχηματισμό της δεύτερης. Η λίστα ακεραίων είναι το πεδίο από 0 έως $2^n - 1$ στο οποίο μπορούν να βρεθούν τα αποτελέσματα της πρόσθεσης. Πλεονέκτημα της συνάρτησης αυτής είναι ότι μπορούν στην πρώτη λίστα να χρησιμοποιηθούν είτε μεταβλητές είτε σταθερές ανάλογα αν θέλουμε να κάνουμε πρόσθεση ανάμεσα σε δύο καταχωρητές ή αύξηση ενός καταχωρητή κατά μία γνωστή (κλασική) σταθερά. Έτσι για παράδειγμα αν η πρώτη λίστα είναι $[\mathrm{qtrue}, \mathrm{qfalse}]$ θα μετασχηματιστούν κατάλληλα τα bit που αντιπροσωπεύονται από όρους της δεύτερης ώστε η συνολική κατάστασή τους να αυξηθεί κατά 2.

Η υπόθεση της πρόσθεσης όμως δεν τελειώνει εδώ. Για να υλοποιήσουμε τον αλγόριθμο του Shor χρειαζόμαστε υλοποίηση της πρόσθεσης modulo N, όπου N ο αριθμός τον οποίο προσπαθούμε να παραγοντοποιήσουμε ο οποίος φυσικά δεν είναι της μορφής 2^n. Δυστυχώς η πρόσθεση module N είναι μη αντιστρέψιμη διαδικασία. Για παράδειγμα ας πάρουμε την πρόσθεση modulo 5 του αριθμού 3 σε έναν κβαντικό καταχωρητή τριών bit. Προφανώς αν η κατάσταση του κβαντικού καταχωρητή είναι $|\mathbf{5}\rangle$ το αποτέλεσμα θα είναι το ίδιο με αυτό που θα παίρναμε αν ήταν $|\mathbf{0}\rangle$. Μία λύση είναι η χρήση ενός επιπλέον flag qbit το οποίο θα τίθεται στην τιμή 1 μετά την πρόσθεση όταν η αρχική κατάσταση αντιστοιχούσε σε ιδιοδιάνυσμα μεγαλύτερο ή ίσο του $|\mathbf{N}\rangle$. Χρησιμοποιώντας αυτήν την τακτική μπορούμε να δημιουργήσουμε μία αντιστρέψιμη διαδικασία η οποία με είσοδο έναν καταχωρητή στην κατάσταση x, μας δίνει έναν καταχωρητή στην κατάσταση $x + \alpha \bmod N$.

Πολλαπλασιασμός και ύψωση σε εκθέτη

Για να λύσουμε τώρα το πρόβλημα του πολλαπλασιασμού modulo N θα θυμηθούμε πως ο πολλαπλασιασμός ενός αριθμού modulo N είναι αντιστρέψιμος, αν ο πολλαπλασιαστής α εί-

ναι σχετικά πρώτος με το N. Αυτό σημαίνει πως χρησιμοποιώντας την πρόσθεση modulo N μπορούμε να κατασκευάσουμε με αντίστοιχο τρόπο και τον πολλαπλασιασμό modulo N. Έτσι χρησιμοποιώντας σαν ελεγκτές τα κβαντικά bit ενός καταχωρητή x μπορούμε να υπολογίσουμε το αx ως εξής: Αρχικοποιούμε έναν καταχωρητή στο $|\mathbf{0}\rangle$ κι έπειτα με ελεγκτή το λιγότερο σημαντικό bit του x του προσθέτουμε α, με ελεγκτή το δεύτερο λιγότερο σημαντικό bit του προσθέτουμε 2α κοκ. Επειδή η διαδικασία αυτή είναι αντιστρέψιμη μπορεί να γίνει με τρόπο τέτοιο ώστε το αποτέλεσμα να μένει στο x δηλαδή να πραγματοποιείται η $|\mathbf{x}\rangle \rightarrow |\alpha\mathbf{x} \bmod \mathbf{N}\rangle$.

Η ύψωση σε εκθέτη μπορεί να πραγματοποιηθεί με έναν εντελώς όμοιο τρόπο μόνο που αυτή τη φορά θα χρησιμοποιήσουμε τον αλγόριθμο του επαναλαμβανόμενου τετραγωνισμού. Με άλλα λόγια ουσιαστικά η αλλαγή με την υλοποίηση του πολλαπλασιασμού είναι ότι όπου πριν κάναμε πρόσθεση τώρα κάνουμε πολλαπλασιασμό και όπου πριν κάναμε διπλασιασμό τώρα κάνουμε ύψωση στο τετράγωνο.

Κεφάλαιο 6

Συμπεράσματα

6.1 Συνεισφορά

Η συνεισφορά και τα συμπεράσματα της εργασίας μπορούν να συνοψιστούν ως εξής

- Προτάθηκε μία νέα γλώσσα κβαντικού προγραμματισμού, η NTUA-QML. Η γλώσσα αυτή προέκυψε από την QML, εκμεταλλευόμενη τις καλύτερες ιδέες της για κβαντικό έλεγχο. Οι διαφορές της με την QML έχουν σαν σκοπό να διορθώσουν αδυναμίες της QML και να βελτιώσουν την εκφραστική δύναμη της γλώσσας, ενώ ταυτόχρονα η διατύπωση της σημασιολογίας με όρους πινάκων πυκνότητας αντί για κυκλώματα είχε στόχο τη σαφέστερη εξήγηση της λειτουργίας της.

- Δημιουργήθηκε μία υλοποίηση μίας κβαντικής γλώσσας προγραμματισμού, της NTUA-QML. Η υλοποίηση αυτή μας έδωσε τη δυνατότητα να πάρουμε μία γεύση από τις δυνατότητες του κβαντικού μοντέλου υπολογισμών. Εκτός από την NTUA-QML, υλοποιήθηκε και ένα στοιχειώδες υποσύνολο της QPL.

- Χρησιμοποιώντας την παραπάνω υλοποίηση υλοποιήθηκαν και στην γλώσσα NTUA-QML γνωστοί κβαντικοί αλγόριθμοι. Αυτό συνεισέφερε και στην καλύτερη κατανόηση της λειτουργίας τους, και στη βελτίωση της NTUA-QML και της υλοποίησής της.

6.2 Μελλοντική έρευνα

Η κυριότερη κατεύθυνση μελλοντικής έρευνας πάνω στις γλώσσες κβαντικού προγραμματισμού είναι, όπως υπογραμμίστηκε και στο κεφάλαιο 3, αυτή που στρέφεται στο ξεκαθάρισμα των εννοιών που κάνουν τον κβαντικό προγραμματισμό ιδιαίτερο. Μία καλή γλώσσα θα πρέπει να δίνει τη δυνατότητα στο χρήστη της να δημιουργεί υπερθέσεις και entanglement, και να εκμεταλλεύεται τα οφέλη του κβαντικού παραλληλισμού με την ίδια φυσικότητα που μια κλασική γλώσσα επιτρέπει στο χρήστη της να χειρίζεται στοιχειώδη χαρακτηριστικά των κλασικών υπολογισμών. Θα μπορούσε εδώ κανείς να ισχυριστεί πως αυτός είναι ένας πολύ δύσκολος στόχος, λαμβάνοντας υπ' όψιν πως η ίδια η φυσική θεωρία στην οποία βασίζονται οι κβαντικοί υπολογισμοί είναι για πολλούς ακατανόητη. Μια τέτοια ηττοπάθεια φυσικά δεν είναι λύση: η πρόοδος μπορεί πάντα να επιτευχθεί και μάλιστα δεν πρέπει να θεωρείται απίθανο οι κβαντικοί υπολογισμοί να γίνουν κάποτε τόσο σημαντικοί που θα ενσωματωθούν στην κοινή λογική του μέσου προγραμματιστή και θα δημιουργείται ίσως η απορία γιατί κάποτε οι άνθρωποι τους θεωρούσαν κάτι περίεργο και εξωτικό. Φυσικά μέχρι τότε έχουμε μάλλον πάρα πολύ δρόμο.

Μία γλώσσα όπως η NTUA-QML προσπαθεί να συνεισφέρει στην κατεύθυνση της απλοποίησης του κβαντικού προγραμματισμού με διάφορους τρόπους:

- Με το if^o δημιουργεί την αίσθηση της κβαντικής ροής ελέγχου. Η πολύ καλή αυτή ιδέα, η οποία ξεκινά φυσικά από την QML, έχει στόχο να κάνει πιο άνετη τη χρήση μιας κλάσης ορθομοναδιαίων μετασχηματισμών.

- Με το if^{-} προσπαθεί να κάνει τον προγραμματιστή να αντιληφθεί την εφαρμογή ενός ορθομοναδιαίου μετασχηματισμού με όρους του αποτελέσματος που θα είχε αυτός ο μετασχηματισμός αν εφαρμοζόταν σε μία κλασική κατάσταση. Η λογική αυτή δεν είναι φυσικά κάποια μεγάλη καινοτομία και θα πρέπει να σκεφτούμε αν υπάρχουν κι άλλες κλάσεις μετασχηματισμών που θα μπορούσαμε να εξηγήσουμε πιο απλά από το να δώσουμε το αποτέλεσμα τους σε κάθε κλασική κατάσταση, εκτός από τους ελεγχόμενους μετασχηματισμούς τους οποίους καλύπτει το if^{o}.

- Με τη χρήση του if δίνει στο χρήστη να καταλάβει ακριβώς το πότε γίνεται η μέτρηση και έτσι τον προετοιμάζει για τις συνέπειες που έχει κάτι τέτοιο στην κατάσταση του όλου συστήματος.

Στοιχεία της NTUA-QML που θα μπορούσαν να βελτιωθούν είναι:

- Προσθήκη γνωστών από τον κλασικό προγραμματισμό δομών όπως τα loop, και η αναδρομή. Είδαμε πως στην υλοποίησή μας χρησιμοποιήσαμε πολλές φορές τη Haskell για να παρακάμψουμε αυτές τις αδυναμίες, όμως η υλοποίησή τους μέσα στη γλώσσα θα ήταν φυσικά προτιμότερη και θα μας έδινε νέες δυνατότητες.

- Εμπλουτισμός του συστήματος τύπων. Για παράδειγμα θα μπορούσαν να επανέλθουν τα αθροίσματα της QML ή να εισαχθεί ο τύπος συνάρτησης ώστε να μπορούμε να ορίζουμε και προγράμματα υψηλής τάξης. Η σημασιολογία τέτοιων προγραμμάτων θα μπορούσε να βασιστεί ίσως και σε δουλειά που έχει γίνει πάνω στον κβαντικό λ-λογισμό ([vTon04]).

- Εισαγωγή αριθμητικών πράξεων, με τρόπο συμβατό με το κβαντικό μοντέλο (σεβασμός στην αντιστρεψιμότητα) αλλά και εύκολο στη χρήση.

Το κυριότερο όμως που παρατηρήθηκε κατά τη διάρκεια της δημιουργίας της NTUA-QML είναι ότι η προσπάθεια υλοποίησης ενός αλγορίθμου σε μία γλώσσα είναι αποκαλυπτική των εκφραστικών κενών και αδυναμιών που αυτή περιέχει. Έτσι ίσως μια προϋπόθεση για την δημιουργία καλύτερων γλωσσών κβαντικού προγραμματισμού να είναι η δημιουργία μιας καλής βιβλιοθήκης αλγορίθμων. Όταν το πεδίο των κβαντικών υπολογισμών ωριμάσει είναι σίγουρο πως επιτέλους θα δούμε πραγματικά εύχρηστες γλώσσες κβαντικού προγραμματισμού.

Βιβλιογραφία

[Clev00] R. Cleve, "An Introduction to Quantum Complexity Theory", *Collected Papers on Quantum Computation and Quantum Information*, pp. 103–127, 2000.

[Deut95] David Deutsch, Adriano Barenco and Artur Ekert, "Universality in quantum computation", 1995.

[Grat05a] Jonathan Grattage and Thorsten Altenkirch, "A compiler for a functional quantum programming language". submitted for publication, January 2005.

[Grat05b] Jonathan Grattage and Thorsten Altenkirch, "QML: Quantum data and control". submitted for publication, February 2005.

[Grov96] Lov K. Grover, "A fast quantum mechanical algorithm for database search", 1996.

[Knil96] E. Knill, "Conventions for quantum pseudocode", 1996.

[Oeme03] B. Oemer, "Structured Quantum Programming, PhD Thesis, Institue for Theoretical Physics, Technical University of Vienna", 2003.

[Sand00] J. Sanders and P. Zulianni, "Quantum Programming", *Mathematics of Program Construction*, vol. 1837, pp. 80–99, 2000.

[Seli04] Peter Selinger, "Towards a Quantum Programming Language", *Mathematical Structures in Computer Science*, vol. 14, no. 4, pp. 527–586, 2004.

[Shor97] Peter W. Shor, "Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM J. Sci. Statist. Comput.*, vol. 26, p. 1484, 1997.

[vTon04] Andre van Tonder, "A Lambda Calculus for Quantum Computation", *Siam J. Comput.*, vol. 33, pp. 1109–1135, 2004.