Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής
και Υπολογιστών

Στατικές Εγγυήσεις Ασφάλειας για Γλώσσες με Ταυτόχρονο Προγραμματισμό

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

ΠΡΟΔΡΟΜΟΣ ΓΕΡΑΚΙΟΣ

Επιβλέπων : Κωνσταντίνος Σαγώνας
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2012

.....

Πρόδρομος Γερακιός

Διδάκτωρ Πληροφορικής Ε.Μ.Π.

Copyright © Πρόδρομος Γερακιός, 2012.
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

Τα σύγχρονα υπολογιστικά συστήματα από επιτραπέζιους υπολογιστές μέχρι υπερυπολογιστές φέρουν πολυπύρηνους (multicore) επεξεργαστές. Οι πολυπύρηνοι επεξεργαστές αποδίδουν περισσότερη υπολογιστική ισχύ σε ένα τσιπ και το λογισμικό διαδραματίζει καθοριστικό ρόλο στην αξιοποίηση της. Ο πολυνηματικός προγραμματισμός είναι ένα ευρέως χρησιμοποιούμενο πρότυπο προγραμματισμού για την αξιοποίηση του ταυτοχρονισμού σε πολυπύρηνα συστήματα. Μια εγγενής παρενέργεια αυτού του προτύπου είναι η μη ντετερμινιστική εκτέλεση των νημάτων, η οποία επιτρέπει να συμβούν ανεπιθύμητες νηματικές διεμπλοκές (thread interleavings). Τέτοιες διεμπλοκές δεν μπορούν να αναπαραχθούν εύκολα αφού εξαρτώνται από τον χρόνο, καθιστώντας δύσκολη την αξιόπιστη ανάπτυξη ταυτόχρονου λογισμικού. Οι συνθήκες ανταγωνισμού (data races) και τα αδιέξοδα (deadlocks) είναι δύο γνωστές συνέπειες της μη ντετερμινιστικής εκτέλεσης. Η συνθήκη ανταγωνισμού είναι μία κατάσταση κατά την οποία πολλαπλά νήματα έχουν ταυτόχρονη πρόσβαση στην ίδια θέση μνήμης χωρίς συγχρονισμό. Ένα σύνολο από νήματα φτάνει σε αδιέξοδο, όταν κάθε νήμα βρίσκεται σε αναμονή για μια κλειδαριά, η οποία είναι κλειδωμένη από κάποιο άλλο νήμα του ίδιου συνόλου. Σε αυτή την διατριβή δείχνουμε ότι είναι δυνατό να αυξηθεί η αξιοπιστία των ταυτόχρονων προγραμμάτων, με την εξάλειψη των παραβιάσεων πρόσβασης μνήμης, συνθηκών ανταγωνισμού και αδιεξόδων από πολυνηματικά προγράμματα. Συγκεκριμένα, παρουσιάζουμε την θεωρία και υλοποίηση συστημάτων τύπων (type systems) και στατικών αναλύσεων (static analyses) που παρέχουν κάποιες εγγυήσεις αξιοπιστίας σχετικά με την πολυνηματική εκτέλεση προγραμμάτων. Στα θεωρητικά τμήματα της διατριβής παρουσιάζουμε την τυπική σημασιολογία (formal semantics) των συστημάτων τύπων και των γλωσσών και δίνουμε τυπικές αποδείξεις ορθότητας. Στα πρακτικά τμήματα αναλύουμε την ενσωμάτωση των τυπικών συστημάτων που αναπτύξαμε σε πραγματικές γλώσσες προγραμματισμού και τις προκλήσεις που συναντήσαμε. Επίσης παρουσιάζουμε εκτεταμένες μετρήσεις για την απόδοση των αναλύσεων μας και τις συγκρίνουμε με εναλλακτικές τεχνικές.

Ευχαριστίες

Θέλω να ευχαριστήσω από τα βάθη της καρδιάς μου τους επιβλέποντες μου, Νίκο Παπασπύρου και Κωστή Σαγώνα, για την καθοδήγηση και την υποστήριξη που μου παρείχαν συνεχώς. Θα ήθελα να ευχαριστήσω τα μέλη του Εργαστηρίου Τεχνολογίας Λογισμικού: Παναγιώτη Βεκρή, Γιώργο Κορφιάτη, Μιχάλη Παπακυριάκου, Μαρία Χριστάκη, Χάρη Νάκο, Φάνη Τζίασιο και τον Άγγελο Μανουσαρίδη για την εξαιρετική υποστήριξή τους, τις εποικοδομητικές συζητήσεις και τις καλές στιγμές που έχουμε μοιραστεί. Πολλά ευχαριστώ στην οικογένειά μου, ειδικά στον αδελφό μου Κώστα Γερακιό, για ατέλειωτη υποστήριξή τους με πολλούς τρόπους. Η έρευνά μου χρηματοδοτήθηκε εν μέρει από το πρόγραμμα για τη στήριξη της βασικής έρευνας (ΠΕΒΕ 2010) του Εθνικού Μετσόβιου Πολυτεχνείου στο πλαίσιο του έργου "Μελέτη Ιδιοτήτων Ασφαλείας και Προόδου σε Γλώσσες Ταυτόχρονου Προγραμματισμού".

Πρόδρομος Γερακιός,

Αθήνα, 19η Ιουλίου 2012

Η εργασία αυτή είναι επίσης διαθέσιμη ως Τεχνική Αναφορά CSD-SW-TR-1-12, Εθνικό Μετσόβιο Πολυτεχνείο, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών, Εργαστήριο Τεχνολογίας Λογισμικού, Ιούλιος 2012.

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

Πίνακας περιεχομένων

Περίληψη 3

Ευχαριστίες 5

Πίνακας περιεχομένων 7

Σχήματα 9

Πίνακες 11

Κώδικες 13

1. Εισαγωγή 15
1.1 Παρακίνηση 15
1.2 Επισκόπηση της διατριβής 15
1.2.1 Ιεραρχικές περιοχές και κλειδαριές για ασφαλή πολυνηματικά προγράμματα 16
1.2.2 Ασφάλεια από αδιέξοδα: Μια υβριδική προσέγγιση. 18

2. Συμπερασμός εφέ ιεραρχικών περιοχών με κλειδαριές ανάγνωσης/εγγραφής 21
2.1 Επισκόπηση 21
2.2 Τυπική γλώσσα 22
2.3 Λειτουργική σημασιολογία 22
2.4 Στατική σημασιολογία 22
2.5 Ασφάλεια τύπων 23

3. Ταυτόχρονη Cyclone 29
3.1 Επισκόπηση 29
3.2 Cyclone: Μια ασφαλής διάλεκτος της C 29
3.2.1 Διαχείρηση μνήμης στη Cyclone 29
3.2.2 Ταυτοχρονισμός στη Cyclone 30
3.3 Ταυτόχρονη Cyclone 31
3.3.1 Εκτεταμένες περιοχές και σύστημα Ειδών 31
3.3.2 Εξαιρέσεις 31
3.3.3 Επανεισερχόμενες συναρτήσεις 31
3.3.4 Πολυμορφισμός τύπων 31
3.3.5 Λειτουργία cap. 31
3.3.6 Εκτεταμένες λειτουργίες περιοχών. 32
3.3.7 Δημιουργία νημάτων. 32
3.4 Υλοποίηση 32
3.4.1 Μεταγλωττιστής 32
3.4.2 Παραγωγή κώδικα και σύστημα χρόνου εκτέλεσης 32
3.5 Αποτίμηση απόδοσης 33

4. Εξαγωγή εφέ για αποφυγή αδιεξόδων 35
4.1 Επισκόπηση 35
4.2 Τυπική γλώσσα 35
4.3 Λειτουργική σημασιολογία 35
4.4 Στατική σημασιολογία 36
4.5 Ασφάλεια τύπων και απουσία αδιεξόδων 37
4.6 Καταληκτικές παρατηρήσεις 40

5. Εργαλείο αποφυγής αδιεξόδων 43
5.1 Επισκόπηση 43
5.2 Ανάλυση αποφυγής αδιεξόδων 43
5.2.1 Στατική ανάλυση 43
5.2.2 Μετασχηματισμός κώδικα 44
5.2.3 Περιορισμοί 44
5.2.4 Σύστημα χρόνου εκτέλεσης 45
5.3 Αποτίμηση επιδόσεων 45
5.4 Καταληκτικές παρατηρήσεις 47

6. Σχετικές εργασίες 49
6.1 Ασφαλείς γλώσσες συστημάτων 49
6.2 Διαχείριση μνήμης με περιοχές 50
6.3 Ασφαλής Ταυτοχρονισμός 50
6.3.1 Απουσία συνθηκών ανταγωνισμού 50
6.3.2 Απουσία αδιεξόδων 52
6.3.3 Διαφορετικές προσεγγίσεις για ασφαλή ταυτοχρονισμό 53

7. Συμπεράσματα 55

Βιβλιογραφία 59

Σχήματα

2.1 Σύνταξη γλώσσας. 22
2.2 Βοηθητική σύνταξη για την λειτουργική σημασιολογία. 22
2.3 Σχέση αποτίμησης $C \rightsquigarrow C'$. 23
2.4 Βοηθητικές συναρτήσεις και κατηγορήματα. 24
2.5 Σύνταξη τύπων, εφέ και περιβαλλόντων. 24
2.6 Κανόνες τύπων. 25
2.7 Κανόνες επικύρωσης και μετασχηματισμού εφέ. 26
2.8 Συνοψισμένα εφέ αναδρομικών συναρτήσεων. 27
2.9 Κανόνες τύπων για το πλαίσιο αποτίμησης. 28

4.1 Σύνταξη γλώσσας και τύπων. 36
4.2 Λειτουργική σημασιολογία και περιβάλλον αποτίμησης. 36
4.3 Λειτουργική σημασιολογία. 37
4.4 Βοηθητικές συναρτήσεις λειτουργικής σημασιολογίας. 38
4.5 Κανόνες τύπων. 39
4.6 Κανόνες καλοσχηματισμένων τύπων. 40
4.7 Κανόνες τύπων για το περιβάλλον αποτίμησης. 41

5.1 Συγκριτικά αποτελέσματα για το πρόβλημα των *συνδαιτυμόνων φιλοσόφων*. Μετράμε τον συνολικό αριθμό (n) που έφαγαν οι φιλόσοφοι. 46

Πίνακες

3.1 Συγκριτικά αποτελέσματα προγραμμάτων Ταυτόχρονης Cyclone και C/pthreads προερχόμενα από το “The Computer Language Benchmarks Game.” 34

5.1 Απόδοση αρχικής C με μετασχηματισμένη C (C+da). 45

Κώδικες

1.1 Κώδικας από το EFS του Linux (`linux/fs/efs/namei.c`) 19

Κεφάλαιο 1

Εισαγωγή

1.1 Παρακίνηση

Η ορθή, αξιόπιστη και συνεχής λειτουργία σύνθετων και ποικιλόμορφων υπολογιστικών συστημάτων ή συστημάτων που σε μεγάλο βαθμό στηρίζονται σε προγράμματα λογισμικού, όπως τηλεπικοινωνιακών συστημάτων, αυτοκινήτων, αεροσκαφών, συστημάτων ανταλλαγής πληροφοριών, κ.λπ. είναι εξαιρετικά σημαντική για τις ανάγκες της κοινωνίας μας. Όταν τα υπολογιστικά συστήματα αποτύχουν ή παραβιαστούν λόγω σφαλμάτων λογισμικού, οι επιπτώσεις μπορεί να είναι σημαντικές.

Οι πρόσφατες εξελίξεις στον τομέα του υλικού των υπολογιστών, με την όλο και πιο διαδεδομένη χρήση των λεγόμενων πολυπύρηνων αρχιτεκτονικών, όπου πολλαπλοί πυρήνες ή/και επεξεργαστές επικοινωνούν άμεσα μέσω μοιραζόμενης μνήμης, καθιστούν την ανάπτυξη αποδοτικών και αξιόπιστων πολυνηματικών προγραμμάτων ακόμα πιο δύσκολη. Συνήθως, μαζί με την αύξηση της επίδοσης των προγραμμάτων, η παράλληλη ή ταυτόχρονη εκτέλεση πολλών νημάτων εισάγει νέα προβλήματα όσον αφορά στην αξιοπιστία λογισμικού.

Γλώσσες προγραμματισμού που διατηρούν τη διαφάνεια και τον έλεγχο της μνήμης, όπως η C, είναι κατάλληλες για πολυνηματικό προγραμματισμό και την εκμετάλλευση της επίδοσης των πολυπύρηνων αρχιτεκτονικών. Τα προγράμματα που είναι γραμμένα σε τέτοιες γλώσσες επιτρέπουν συχνά εσφαλμένες προσβάσεις μνήμης, διαρροές μνήμης, κλπ., και γίνονται επιρρεπή σε συνθήκες ανταγωνισμού και αδιέξοδα από την απρόσεκτη χρήση κλειδαριών. Έτσι, μια πρόκληση για την έρευνα είναι η σχεδίαση και υλοποίηση πολυνηματικών γλωσσών χαμηλού επιπέδου, οι οποίες παρέχουν στατικές εγγυήσεις ασφάλειας, όπως η απουσία παραβιάσεων μνήμης, και σφαλμάτων ταυτοχρονισμού, όπως οι συνθήκες ανταγωνισμού και τα αδιέξοδα.

1.2 Επισκόπηση της διατριβής

Ο βασικός σκοπός της διατριβής είναι η παρουσίαση των εργαλείων και τεχνικών που αναπτύχθηκαν για την βελτίωση της αξιοπιστίας προγραμμάτων με ταυτοχρονισμό. Συνεπώς, αναπτύσσουμε στατικές αναλύσεις που εγγυώνται την απουσία σφαλμάτων μνήμης, συνθηκών ανταγωνισμού και αδιεξόδων από ταυτόχρονα προγράμματα με ελάχιστη επιβάρυνση του χρόνου εκτέλεσης τους. Η παρούσα εργασία εστιάζει σε γλώσσες χαμηλού επιπέδου με μη δομημένο κλείδωμα (unstructured locking),[1]. Εφαρμόζουμε την θεωρία σε πολυνηματικά προγράμματα Cyclone και C/pthreads. Οι φορμαλισμοί μας είναι απλές επεκτάσεις του λάμβδα λογισμού, επιτρέποντας στα βασικά αποτελέσματα μας να είναι *γενικά* και *ανεξάρτητα* από τις προαναφερθείσες γλώσσες.

Προτείνουμε ένα σύστημα τύπων που εγγυάται την απουσία σφαλμάτων μνήμης και συνθηκών ανταγωνισμού για το λάμβδα λογισμό με διαχείριση μνήμης βασισμένη σε περιοχές (region-based memory management) και μη δομημένο κλείδωμα. Το προτεινόμενο σύστημα τύπων εξάγει τις επισημειώσεις τύπων αυτόματα και επιτρέπει την χρήση κλειδαριών ανάγνωσης/εγγραφής (reader-writer locks).

[1] Στο υπόλοιπο του κειμένου εναλλάσσουμε τους όρους “μη δομημένο κλείδωμα”, “αδόμητο κλείδωμα”, και “κλείδωμα εκτός λεξικής εμβέλειας”.

Στη συνέχεια προτείνουμε ένα σύστημα τύπων που εγγυάται την απουσία αδιεξόδων για τον λάμβδα λογισμό με μη δομημένο κλείδωμα, την απουσία αδιεξόδων από πολυνηματικά προγράμματα και επιτρέπει αυτόματο συμπερασμό επισημειώσεων τύπων. Επίσης σχολιάζουμε την ενσωμάτωση των προτεινόμενων συστημάτων τύπων σε γλώσσες χαμηλού επιπέδου όπως την Cyclone και την C/pthreads. Τέλος αξιολογούμε την αποτελεσματικότητα των συστημάτων τύπων δοκιμάζοντας τις υλοποιήσεις μας με διάφορα μη τετριμμένα προγράμματα αξιολόγησης.

1.2.1 Ιεραρχικές περιοχές και κλειδαριές για ασφαλή πολυνηματικά προγράμματα

Τα πολυνηματικά προγράμματα μοιραζόμενης μνήμης επιτρέπουν την ταυτόχρονη πρόσβαση μνήμης σε ένα ενιαίο χώρο διευθύνσεων. Τα νήματα που χρονοδρομολογούνται ασθενώς δίκαια και προεκτοπιστικά παράγουν τυχαίες διεμπλοκές. Μόνο ένα υποσύνολο των πιθανών διεμπλοκών οδηγεί το πρόγραμμα σε συνεπείς καταστάσεις. Ασυνεπείς καταστάσεις προκύπτουν όταν ένα νήμα έχει πρόσβαση σε μια θέση μνήμης την ίδια στιγμή που ένα άλλο νήμα προσπαθεί να γράψει σε αυτήν. Σε τέτοιου είδους διεμπλοκές έχουμε *συνθήκη ανταγωνισμού*. Μια βασική εγγύηση για τα πολυνηματικά προγράμματα είναι η απουσία συνθηκών ανταγωνισμού. Σε αρχιτεκτονικές χωρίς μνήμη συναλλαγών [Harr03] (transactional memory), τα νήματα πρέπει να συμφωνήσουν στην σειρά εκτέλεσης συγκεκριμένων διεμπλοκών έτσι ώστε να αποφευχθούν οι συνθήκες ανταγωνισμού. Συγκεκριμένα, τα νήματα πρέπει να συγχρονίσουν τις ενέργειες τους. Στην παρούσα εργασία εξετάζουμε αποκλειστικά τον συγχρονισμό μέσω κλειδαριών (lock-based blocking synchronization). Υποθέτουμε επίσης ότι οι προσπελάσεις μνήμης είναι διαδοχικά συνεπείς (sequentially consistent). Εκτός των ασυνεπών καταστάσεων που προκύπτουν από συνθήκες ανταγωνισμού, ασυνεπείς καταστάσεις προκύπτουν από προσβάσεις (πιθανώς ταυτόχρονες) σε ξεκρέμαστους δείκτες (dangling pointers). Συνεπώς η απουσία παραβιάσεων μνήμης (*ασφάλεια μνήμης*) είναι ακόμα μια βασική εγγύηση ασφάλειας σε πολυνηματικά προγράμματα.

Παρουσιάζουμε την τυπική σημασιολογία, μεταθεωρία και υλοποίηση μιας γλώσσας με ισχυρό σύστημα τύπων, διαχείριση μνήμης με περιοχές και ιεραρχικό συγχρονισμό με κλειδαριές. Προτείνουμε ένα σύστημα τύπων για την γλώσσα μας με στατικές εγγυήσεις ασφάλειας μνήμης και εγγυήσεις απουσίας συνθηκών ανταγωνισμού.

Η περιοχή [Toft94] (μνήμης) μπορεί να θεωρηθεί ως ένα τμήμα μνήμης. Η βασική ιδέα της διαχείρισης μνήμης με περιοχές είναι ότι κάθε αντικείμενο δεδομένων είναι δεσμευμένο σε κάποια περιοχή μνήμης. Όταν μια περιοχή αποδεσμεύεται, όλα τα αντικείμενα που περιέχονται σε αυτήν αποδεσμεύονται ταυτόχρονα. Η διαχείριση μνήμης με περιοχές έχει πολλαπλά οφέλη. Ο προγραμματιστής έχει ρητό έλεγχο της θέσης και της διάρκειας ζωής των αντικειμένων μνήμης δεδομένου ότι είναι δυνατό να επιλέξει ρητά την περιοχή στην οποία θα δεσμευτούν. Σε ορισμένες γλώσσες είναι επίσης δυνατή η ρητή και πρόωρη αποδέσμευση περιοχών μνήμης (early region deallocation). Επιπλέον, είναι πιο αποτελεσματικό να δεσμεύονται τα αντικείμενα σε υπάρχουσες περιοχές μνήμης από το να δεσμεύονται καινούργια τμήματα μνήμης για κάθε αντικείμενο ξεχωριστά.

Οι περιοχές μνήμης μας οργανώνονται με ένα ιεραρχικό τρόπο [Gay01], όπου κάθε περιοχή είναι φυσικά δεσμευμένη εντός κάποιας περιοχής-γονέα και μπορεί να περιέχει πολλές περιοχές-παιδιά. Προτείνουμε μια γλώσσα που επιτρέπει την ασφαλή αποδέσμευση ολόκληρων υποδένδρων ακόμα όταν αυτά μοιράζονται μεταξύ νημάτων. Η αποδέσμευση μιας περιοχής επιτρέπεται σε οποιοδήποτε σημείο ενός προγράμματος. Κάθε περιοχή σχετίζεται με μια κρυφή κλειδαριά. Συνεπώς, οι κλειδαριές ακολουθούν την ιεραρχική δομή των περιοχών. Κάθε περιοχή προστατεύεται από την κλειδαριά της και όλες τις κλειδαριές των προγόνων της. Σε αντίθεση με την πλειοψηφία των συστημάτων τύπων και αναλύσεων που έχουν προταθεί στη βιβλιογραφία για απουσία συνθηκών ανταγωνισμού σε προγράμματα με κλειδώματα λεξικής εμβέλειας [Flan99a, Gros03, Boya02], η προσέγγιση μας είναι πιο γενική, υποστηρίζοντας κλειδώματα μη λεξικής εμβέλειας τα οποία είναι πιο κατάλληλα για γλώσσες στο επίπεδο αφαίρεσης της C. Ακόμα, η προσέγγιση μας επιτρέπει σε περιοχές και κλειδαριές να αντιγράφονται με ασφάλεια και να ξεφεύγουν από την λεξική εμβέλεια των νημάτων όταν διαμοιράζονται. Επίσης, έχουμε ενσωματώσει την ανάλυση μας στην Cyclone, μια γλώσσα με ισχυρό σύστημα τύπων, η οποία διατηρεί ρητό έλεγχο και αναπαράσταση μνήμης χωρίς να θυσιάζει την ασφάλεια της.

Παρουσιάζουμε εν συντομία τους βασικούς στόχους του σχεδιασμού της γλώσσα μας, καθώς επίσης και μερικές από τις βασικές αποφάσεις σχεδίασης που εξυπηρετούν τους σκοπούς αυτούς.

Χαμηλού επιπέδου και ταυτόχρονη. Η γλώσσα πρέπει να είναι στο επίπεδο αφαίρεσης της C και να παρέχει ενσωματωμένες κατασκευές για ταυτοχρονισμό παρόμοιες με εκείνες που χρησιμοποιούνται σήμερα π.χ. C/pthreads (δηλαδή, θα πρέπει να υποστηρίζει τον αδόμητο τρόπο κλειδώματος της C).

Μοιραζόμενη μνήμη. Η γλώσσα θα πρέπει να χρησιμοποιεί την κοινόχρηστη μνήμη ως μέσο επικοινωνίας μεταξύ νημάτων. Αυτό είναι θεμιτό τόσο για λόγους αποτελεσματικότητας όσο και για την ευκολία ενσωμάτωσης του συστήματος επικοινωνίας/συγχρονισμού με μοιραζόμενη μνήμη στο υπάρχον σύστημα περιοχών μνήμης της Cyclone.

Συμβατότητα. Ο ακολουθιακός κώδικας Cyclone πρέπει να λειτουργεί όπως αναμένεται χωρίς τροποποιήσεις.

Στατικές εγγυήσεις ασφάλειας. Το προτεινόμενο σύστημα τύπων πρέπει να διασφαλίζει την απουσία παραβιάσεων μνήμης και συνθηκών ανταγωνισμού με ελάχιστη επιβάρυνση στον χρόνο εκτέλεσης τον προγραμμάτων.

Ασφαλής και αποδοτική διαχείριση μνήμης με περιοχές. Η παραδοσιακή προσέγγιση διαχείρισης μνήμης με περιοχές μνήμης σε μορφή στοίβας [Toft94] δεν επιτρέπει την πρόωρη αποδέσμευση τους και δεν μπορεί να μοντελοποιήσει την διάρκεια ζωής μοιραζόμενων περιοχών μνήμης, η οποία εξαρτάται από την διάρκεια ζωής του μακροβιότερου νήματος που έχει πρόσβαση στις εν λόγω περιοχές. Αντίθετα, θέλουμε περιοχές μνήμης που να μπορούν να αποδεσμευτούν πρόωρα και να μοιράζονται με ασφάλεια μεταξύ ταυτόχρονων νημάτων. Έχουμε επιλέξει μια *ιεραρχική* οργάνωση περιοχών μνήμης: κάθε περιοχή είναι δεσμευμένη σε κάποια περιοχή-γονέα και μπορεί να περιέχει πολλές περιοχές-παιδιά. Η πρόωρη αποδέσμευση μιας περιοχής στην πολυεπίπεδη ιεραρχία που περιγράφουμε παραπάνω συνεπάγεται αυτόματη αποδέσμευση του υποδέντρου της χωρίς να είναι απαραίτητη η ρητή αποδέσμευση όλων των κόμβων του υποδέντρου της. Η ιεραρχική δομή επιβάλλει τον περιορισμό ότι μια περιοχή-παιδί είναι δεσμευμένη μόνο όταν οι πρόγονοί της είναι ζωντανοί (δεν έχουν αποδεσμευτεί).

Ασφάλεια από συνθήκες ανταγωνισμού. Για να αποφευχθούν οι συνθήκες ανταγωνισμού χρησιμοποιούμε κλειδαριές έτσι ώστε μόνο ένα νήμα να έχει πρόσβαση σε μοιραζόμενα δεδομένα κάθε φορά (αμοιβαίος αποκλεισμός). Αντί να έχουμε ένα ξεχωριστό μηχανισμό για κλειδαριές, επιλέγουμε ομοιόμορφη μεταχείριση των περιοχών και των κλειδαριών: Οι κλειδαριές τοποθετούνται στην ίδια ιεραρχία με τις περιοχές και έχουν παρόμοιες ιδιότητες. Κάθε περιοχή προστατεύεται από τη δική της κλειδαριά και κληρονομεί τα δικαιώματα πρόσβασης των προγόνων της. Η σημασιολογία του κλειδώματος μιας περιοχής είναι ότι κλειδώνουμε *ατομικά* την περιοχή καθώς και ολόκληρο το υποδέντρο της. Το ιεραρχικό κλείδωμα μπορεί να μοντελοποιήσει σύνθετες στρατηγικές συγχρονισμού χωρίς να είναι απαραίτητο το ρητό κλείδωμα πολλαπλών κλειδαριών. Αν και αδιέξοδα είναι πιθανά στο σύστημα μας, μπορούν να εξαλειφθούν από την απόκτηση μιας ενιαίας κλειδαριάς-πατέρα για μια ομάδα περιοχών αποφεύγοντας το κλείδωμα πολλαπλών κλειδαριών, το οποίο μπορεί να οδηγήσει σε κυκλικές εξαρτήσεις και αδιέξοδα. Επιπλέον, επιτρέπουμε κλειδώματα εκτός λεξικής εμβέλειας τα οποία είναι πιο γενικά και χρήσιμα σε γλώσσες χαμηλού επιπέδου.

Πολυμορφισμός και αντίγραφα περιοχών μνήμης. Υποστηρίζουμε πολυμορφικές συναρτήσεις (ως προς τις περιοχές μνήμης) όπως ακριβώς και στην αρχική γλώσσα Cyclone. Ο πολυμορφισμός περιοχών μνήμης επιτρέπει την επαναχρησιμοποιήση συναρτήσεων και το πέρασμα πολλαπλών αντιγράφων περιοχών μνήμης (*region aliasing*) (π.χ, μια πραγματική περιοχή θα μπορούσε να να περαστεί στη θέση δύο ξεχωριστών επίσημων παραμέτρων). Η ύπαρξη πολλαπλών αντιγράφων περιοχών

σε γενικές συναρτήσεις στις οποίες επιτρέπεται η πρόωρη αποδέσμευση μνήμης ή κλειδαριών είναι επικίνδυνη. Επιτρέπουμε την ασφαλή ύπαρξη αντιγράφων περιοχών σε γενικές συναρτήσεις με ελάχιστους περιορισμούς. Οι μηχανισμοί που χρησιμοποιούμε για το σκοπό αυτό μας επιτρέπει να κωδικοποιήσουμε χρήσιμα ιδιώματα ταυτόχρονου προγραμματισμού με ελάχιστους περιορισμούς όπως μετανάστευση περιοχών, μεταφορά κλειδαριών, τοπικές και μοιραζόμενες περιοχές μνήμης.

Στο Κεφάλαιο 2 παρουσιάζουμε ένα λάμβδα λογισμό με διαχείριση μνήμης με περιοχές, αδόμητες κατασκευές κλειδώματος, κλειδαριές ανάγνωσης/εγγραφής, ο οποίος δεν απαιτεί ρητές επισημειώσεις τύπων αφού είναι δυνατό να εξαχθούν αυτόματα από το σύστημα τύπων. Παρουσιάζουμε επίσης την λειτουργική σημασιολογία του, το σύστημα τύπων, τα κύρια θεωρήματα που εγγυώνται απουσία παραβιάσεων μνήμης και συνθηκών ανταγωνισμού από έγκυρα (well-typed) προγράμματα. Στο Κεφάλαιο 3, παρουσιάζουμε την ενσωμάτωση του προτεινόμενου συστήματος τύπων στον μεταγλωττιστή της γλώσσας Cyclone και συγκρίνουμε την απόδοση των ταυτόχρονων προγραμμάτων Cyclone με την απόδοση την αντίστοιχων προγραμμάτων C/pthreads.

1.2.2 Ασφάλεια από αδιέξοδα: Μια υβριδική προσέγγιση.

Τα αδιέξοδα εμφανίζονται σε ταυτόχρονα προγράμματα με μοιραζόμενη μνήμη ως συνέπεια της κυκλικής εξάρτησης μεταξύ των νημάτων του προγράμματος. Δύο ή περισσότερα νήματα βρίσκονται σε αδιέξοδο όταν καθένα από αυτά είναι σε αναμονή για μια κλειδαριά που έχει αποκτηθεί από κάποιο άλλο νήμα. Τα αδιέξοδα συνιστούν ένα σοβαρό πρόβλημα και έχουν προταθεί διάφορες μέθοδοι για να επιτευχθεί απουσία αδιεξόδων από πολυνηματικά προγράμματα. Ειδικότερα, προσεγγίσεις που βασίζονται σε συστήματα τύπων παρέχουν στατικές εγγυήσεις ασφάλειας. Τα περισσότερα από τα προτεινόμενα συστήματα τύπων σε αυτή την κατηγορία [Flan99b, Koba06, Suen08, Vasc10] *προλαμβάνουν* τα αδιέξοδα με την επιβολή καθολικής, αυστηρής και μη κυκλικής διάταξης απόκτησης των κλειδαριών, η οποία πρέπει να τηρείται σε όλο το πρόγραμμα. Ωστόσο, η καθολική διάταξη απόκτησης των κλειδαριών περιορίζει την εκφραστικότητα της γλώσσας και απορρίπτει άσκοπα πολλά σωστά προγράμματα. Μια εναλλακτική λύση για να διασφαλίσουμε την απουσία αδιεξόδων είναι η αποφυγή τους δυναμικά (κατά τον χρόνο εκτέλεσης), αξιοποιώντας πληροφορίες σχετικά με τη μελλοντική χρήση κλειδαριών, η οποίες παρέχονται από την στατική ανάλυση προγραμμάτων. Μια πρόσφατη δουλειά προς αυτή την κατεύθυνση είναι του Boudol [Boud09] ο οποίος παρουσίασε ένα σύστημα τύπων για αποφυγή αδιεξόδων σε γλώσσες με κλειδώματα λεξικής εμβέλειας, όπως η Java. Δυστυχώς, το σύστημα του Boudol δεν είναι εφαρμόσιμο σε προγράμματα χαμηλού επιπέδου με αδόμητο κλείδωμα. Για παράδειγμα, το πρόγραμμα του Σχήματος 1.1 αποτελεί χαρακτηριστικό παράδειγμα προγράμματος συστήματος (systems code), το οποίο εμπεριέχει αδόμητο κλείδωμα. Επιπλέον, ακόμα όταν ένα χαμηλού επιπέδου πρόγραμμα τηρεί δομημένο κλείδωμα, αυτό είναι τίποτα περισσότερο από μια σύμβαση: σε επίπεδο πηγαίου κώδικα, οι αναλύσεις λειτουργούν με την υπόθεση ότι το κλείδωμα είναι αδόμητο και υπάρχουν ρητές λειτουργίες ξεκλειδώματος. Τέλος, σε όλες σχεδόν τις γλώσσες με δομημένο κλείδωμα, οι μεταγλωττιστές μεταφράζουν το δομημένο κλείδωμα σε αδόμητο για λόγους βελτιστοποίησης του τελικού κώδικα.

Παρουσιάζουμε μια μέθοδο που αποφεύγει δυναμικά τα αδιέξοδα και καθοδηγείται από πληροφορίες σχετικά με τη σειρά των λειτουργιών κλειδώματος και ξεκλειδώματος κάθε νήματος, η οποία υπολογίζεται μέσω μιας στατικής ανάλυσης. Η ανάλυση αυτή βασίζεται σε ένα σύστημα τύπων και εφέ (type and effect system) και είναι αρκετά γενική ώστε να ισχύει σε οποιαδήποτε γλώσσα με ρητές λειτουργίες κλειδώματος και ξεκλειδώματος. Οι αδόμητες λειτουργίες κλειδώματος και η δυνατότητα περάσματος πολλαπλών αντιγράφων κλειδαριών εισάγουν σημαντική πολυπλοκότητα στο σύστημα τύπων μας σε σχέση με αντίστοιχο σύστημα για δομημένα κλειδώματα, όπου οι λειτουργίες κλειδώματος-ξεκλειδώματος είναι πάντα σε σωστή σειρά και μάλιστα στην ίδια λεξική εμβέλεια. Το προτεινόμενο σύστημα τύπων εγγυάται ασφάλεια από αδιέξοδα, ξεκρέμαστες λειτουργίες κλειδώματος υπό την ύπαρξη απεριόριστων αντιγράφων κλειδαριών στο πρόγραμμα.

Στο Κεφάλαιο 4 παρουσιάζουμε έναν λάμβδα λογισμό με αδόμητες λειτουργίες κλειδώματος, ο οποίος δεν απαιτεί ρητές επισημειώσεις τύπων. Στη συνέχεια παρουσιάζουμε την δυναμική σημασιολογία των προγραμμάτων, την στατική σημασιολογία τους και τα κύρια θεωρήματα που εγγυώνται

```
59 struct dentry * efs_lookup(struct inode *dir, struct dentry *dentry) {
60   efs_ino_t inodenum;
61   struct inode * inode = NULL;
62   lock_kernel();
63   inodenum = efs_find_entry(dir, dentry->d_name.name, dentry->d_name.len);
64   if (inodenum) {
65     if (!(inode = iget(dir->i_sb, inodenum))) {
66       unlock_kernel();
67       return ERR_PTR(-EACCES);
68     }
69   }
70   unlock_kernel();
71   d_add(dentry, inode);
72   return NULL;
73 }
```

Κώδικας. 1.1: Κώδικας από το EFS του Linux (`linux/fs/efs/namei.c`)

την ελευθερία αδιέξοδων από έγκυρα προγράμματα. Στο Κεφάλαιο 5 παρουσιάζουμε ένα εργαλείο το οποίο υλοποιεί την προτεινόμενη στατική σημασιολογία (Κεφάλαιο 4), το οποίο ενσωματώνει προγράμματα C/pthreads με πληροφορίες σχετικά με την σειρά των λειτουργιών κλειδώματος και ξεκλειδώματος κάθε συνάρτησης. Επίσης, ενσωματώνουμε σύστημα χρόνου εκτέλεσης το οποίο αποφεύγει πιθανά αδιέξοδα δυναμικά, χρησιμοποιώντας τις πληροφορίες της στατικής ανάλυσης. Αναφέρουμε επίσης κάποια πολύ ελπιδοφόρα συγκριτικά αποτελέσματα, τα οποία δείχνουν ότι όλα τα πιθανά αδιέξοδα μπορούν αυτόματα να αποφευχθούν μόνο με μια μικρή επιβάρυνση στον χρόνο εκτέλεσης των προγραμμάτων C/pthreads. Ένα από τα πιο σημαντικά οφέλη είναι ότι δεν χρειάζεται να τροποποιηθεί το πηγαίο πρόγραμμα μεταβάλλοντας την σειρά των λειτουργιών κλειδώματος ή προσθέτοντας ρητές επισημειώσεις τύπων.

Κεφάλαιο 2

Συμπερασμός εφέ ιεραρχικών περιοχών με κλειδαριές ανάγνωσης/εγγραφής

2.1 Επισκόπηση

Σε αυτό το κεφάλαιο παρουσιάζουμε μια πολυνηματική γλώσσα με προηγμένη διαχείριση μνήμης με περιοχές. Οι περιοχές οργανώνονται σε μια ιεραρχική δομή και κάθε περιοχή είναι δεσμευμένη σε μια συγκεκριμένη περιοχή-γονέα και μπορεί να περιέχει πολλαπλές περιοχές-παιδιά. Κάθε περιοχή προστατεύεται από μια έμμεση κλειδαριά ανάγνωσης/εγγραφής, η οποία συγχρονίζει τις ταυτόχρονες προσβάσεις μνήμης. Η ιεραρχική οργάνωση επιβάλει μια σχέση "ιεραρχικής" πρόσβασης σε περιοχές. Συγκεκριμένα, κάθε περιοχή προστατεύεται από τη δική της κλειδαριά καθώς και τις κλειδαριές όλων των προγόνων της. Επιπλέον, όταν μια περιοχή αποδεσμευτεί, οι υποπεριοχές της αποδεσμεύονται αυτόματα. Η γλώσσα επιτρέπει την πρόωρη αποδέσμευση περιοχών και κλειδαριών, δίνοντας στον προγραμματιστή ρητό έλεγχο της διάρκειας ζωής των περιοχών και των κλειδωμάτων. Η προτεινόμενη γλώσσα υποστηρίζει ιδιώματα πολυνηματικού προγραμματισμού, όπως η μετανάστευση και η μεταφορά κλειδωμένων δεδομένων μεταξύ νημάτων. Επιπλέον, η κατάσταση των δεδομένων μπορεί να εναλλάσσεται από τοπική και μοιραζόμενη σε οποιοδήποτε σημείο του προγράμματος. Αξίζει να σημειώσουμε ότι δεν απαιτούνται ρητές επισημειώσεις τύπων στον ορισμό συναρτήσεων. Το στατικό σύστημα τύπων και εφέ για αυτή τη γλώσσα εγγυάται απουσία παραβιάσεων μνήμης και συνθηκών ανταγωνισμού από έγκυρα προγράμματα. Επίσης, το προτεινόμενο σύστημα τύπων συμπεραίνει αυτόματα τις επισημειώσεις εφέ διευκολύνοντας έτσι την συγγραφή προγραμμάτων.

Εν ολίγοις, τα κύρια χαρακτηριστικά του προτεινόμενου συστήματος τύπων είναι:

Ιεραρχικές περιοχές με κλειδαριές ανάγνωσης-εγγραφής. Έχουμε αναπτύξει ένα λάμβδα λογισμό με πολυμορφικές περιοχές, οι οποίες οργανώνονται σε ιεραρχίες και προστατεύονται με κλειδαριές ανάγνωσης/εγγραφής. Όταν μια περιοχή κλειδωθεί για ανάγνωση/εγγραφή, τότε οι υποπεριοχές της κληρονομούν αυτόματα τα ίδια δικαιώματα πρόσβασης. Επιπλέον, υπάρχει δυνατότητα μετανάστευσης ή μοιράσματος κλειδωμένων ιεραρχιών σε νέα νήματα.

Εξαγωγή εφέ. Οι συναρτήσεις δεν επισημειώνονται με ρητά εφέ και έτσι οι πολυμορφικές συναρτήσεις της γλώσσα μας γίνονται πιο γενικές και επαναχρησιμοποιήσιμες.

Φορμαλισμοί και ορθότητα. Παρουσιάζουμε τη λειτουργική και την στατική σημασιολογία της γλώσσας μας. Παρουσιάζουμε συνοπτικά θεώρημα ασφάλειας τύπων, το οποίο εγγυάται απουσία παραβιάσεων μνήμης και συνθηκών ανταγωνισμού από έγκυρα προγράμματα.

Οι επόμενες ενότητες παρουσιάζουν την τυπική γλώσσα, την λειτουργική και την στατική της σημασιολογία καθώς και τα βασικά θεωρήματα που διασφαλίζουν την απουσία συνθηκών ανταγωνισμού και παραβιάσεων μνήμης από έγκυρα προγράμματα.

Expression	e	$::= x \mid f \mid () \mid \texttt{true} \mid \texttt{false} \mid e\ e \mid e\,[r] \mid \texttt{if}\ e\ \texttt{then}\ e\ \texttt{else}\ e$
		$\mid \texttt{newrgn}\ \rho, x\ @\ e\ \texttt{in}\ e \mid \texttt{new}\ e\ @\ e \mid e := e \mid \texttt{deref}\ e \mid \texttt{cap}_\eta\ e$
		$\mid \texttt{spawn}_\xi\ e \mid \texttt{loc}_\ell \mid \texttt{rgn}_\imath$
Function	f	$::= \lambda x.\, e \mid \Lambda\rho.\, f \mid \texttt{fix}\ x.\, f$
Value	v	$::= f \mid () \mid \texttt{true} \mid \texttt{false} \mid \texttt{loc}_\ell \mid \texttt{rgn}_\imath$
Region	r	$::= \rho \mid \imath$
Count vector	η	$::= (n, n, n)$
Spawn effect	ξ	$::= \emptyset \mid \xi, r \mapsto \eta$

Σχήμα. 2.1: Σύνταξη γλώσσας.

Hierarchy	θ	$::= \emptyset \mid \theta, \imath \mapsto (\eta, \imath)$
Heap	H	$::= \emptyset \mid H, \ell \mapsto v$
Store	S	$::= \emptyset \mid S, \imath \mapsto H$
Threads	T	$::= \emptyset \mid T, \langle \theta; e \rangle$
Configuration	C	$::= S; T$
Stack	E	$::= \square \mid E[F]$
Frame	F	$::= \square\ e \mid v\ \square \mid \square\,[r] \mid \texttt{if}\ \square\ \texttt{then}\ e\ \texttt{else}\ e \mid \texttt{newrgn}\ \rho, x\ @\ \square\ \texttt{in}\ e$
		$\mid \texttt{new}\ \square\ @\ e \mid \texttt{new}\ v\ @\ \square \mid \square := e \mid v := \square \mid \texttt{deref}\ \square \mid \texttt{cap}_\eta\ \square$

Σχήμα. 2.2: Βοηθητική σύνταξη για την λειτουργική σημασιολογία.

2.2 Τυπική γλώσσα

Η σύνταξη της γλώσσας μας παρουσιάζεται στο Σχήμα 2.1. Η βασική γλώσσα συμπεριλαμβάνει μεταβλητές (x), σταθερές (`true`, `false` and ()), συναρτήσεις (f), εφαρμογές συναρτήσεων ($e_1\, e_2$), και εκφράσεις ελέγχου (`if` e `then` e_1 `else` e_2). Οι συναρτήσεις είναι μονομορφικές ($\lambda x.\, e$), πολυμορφικές ως προς τις περιοχές ($\Lambda\rho.\, f$), όπου ρ είναι μεταβλητή περιοχής, και αναδρομικές (`fix` $x.\, f$). Η εφαρμογή πολυμορφικών συναρτήσεων σε μεταβλητές περιοχών είναι ρητή. Η έκφραση `newrgn` ρ, x @ e_1 `in` e_2 δεσμεύει μια καινούργια περιοχή ρ, στα πλαίσια μιας άλλης περιοχής με χειριστήριο e_1, και θέτει την μεταβλητή x ως χειριστήριο της περιοχής ρ. Οι εκφράσεις για την δέσμευση, ανάθεση και εγγραφή μνήμης είναι τυπικές. Η έκφραση $\texttt{cap}_\eta\ e$, μεταβάλει την κατάσταση μια περιοχής (π.χ., αποδέσμευση ή κλείδωμα).

2.3 Λειτουργική σημασιολογία

Η λειτουργική σημασιολογία (*μικρών βημάτων*) της γλώσσας μας παρουσιάζεται στα Σχήματα 2.2 και 2.3. Η σχέση αποτίμησης μετασχηματίζει δυαδικές πλειάδες αποτελούμενες από έναν αφηρημένο σωρό και μια λίστα νημάτων. Κάθε νήμα βλέπει μια τοπική ιεραρχία περιοχών θ. Το νήμα συνδέεται με κάποια έκφραση e που εκτελεί. Η αποτίμηση εκφράσεων γίνεται από αριστερά προς τα δεξιά.

2.4 Στατική σημασιολογία

Παρουσιάζουμε ένα σύστημα τύπων και εφέ που εγγυάται την απουσία σφαλμάτων μνήμης και συνθηκών ανταγωνισμού. Η σύνταξη τύπων, εφέ και περιβαλλόντων παρουσιάζεται στο Σχήμα 2.5. Η σχέση τύπων είναι της μορφής $R; M; \Delta; \Gamma \vdash e : \tau \,\&\, \gamma$ και σημαίνει ότι η έκφραση e έχει τύπο τ και παράγει το εφέ γ. Οι κανόνες τύπων είναι διαθέσιμοι στο Σχήμα 2.6. Οι κανόνες επιστρέφουν

$$\frac{\mathsf{live}(\theta) = \emptyset}{S; T, \langle \theta; () \rangle \rightsquigarrow S; T} \quad \textit{(E-T)}$$

$$\frac{\mathsf{merge}(\xi) \vdash \theta = \theta' \oplus \theta'' \qquad \mathsf{dom}(\theta'') \subseteq \mathsf{live}(\theta)}{S; T, \langle \theta; E[\mathtt{spawn}_\xi\ e] \rangle \rightsquigarrow S; T, \langle \theta'; E[()] \rangle, \langle \theta''; \Box[e] \rangle} \quad \textit{(E-SP)}$$

$$\frac{f \equiv \lambda x.\, e}{S; T, \langle \theta; E[f\ \ v] \rangle \rightsquigarrow S; T, \langle \theta; E[e[v/x]] \rangle} \quad \textit{(E-A)}$$

$$\frac{f \equiv \Lambda \rho.\, f'}{S; T, \langle \theta; E[f\ [\imath]] \rangle \rightsquigarrow S; T, \langle \theta; E[f'[\imath/\rho]] \rangle} \quad \textit{(E-RP)}$$

$$\frac{f \equiv \mathtt{fix}\ x.\, f'}{S; T, \langle \theta; E[f\ \ v] \rangle \rightsquigarrow S; T, \langle \theta; E[f'[f/x]\ \ v] \rangle} \quad \textit{(E-FX)}$$

$$\frac{}{S; T, \langle \theta; E[\mathtt{if\ true\ then}\ e_1\ \mathtt{else}\ e_2] \rangle \rightsquigarrow S; T, \langle \theta; E[e_1] \rangle} \quad \textit{(E-IT)}$$

$$\frac{}{S; T, \langle \theta; E[\mathtt{if\ false\ then}\ e_1\ \mathtt{else}\ e_2] \rangle \rightsquigarrow S; T, \langle \theta; E[e_2] \rangle} \quad \textit{(E-IF)}$$

$$\frac{\jmath \in \mathsf{live}(\theta) \cup \{\bot\} \qquad \text{fresh}\ \imath \qquad \theta' = \theta, \imath \mapsto ((1,1,0), \jmath)}{S; T, \langle \theta; E[\mathsf{newrgn}\ \rho, x\ @\ \mathtt{rgn}_\jmath\ \mathtt{in}\ e] \rangle \rightsquigarrow S, \imath \mapsto \emptyset; T, \langle \theta'; E[e[\imath/\rho][\mathtt{rgn}_\imath/x]] \rangle} \quad \textit{(E-NR)}$$

$$\frac{\imath \in \mathsf{live}(\theta) \qquad \text{fresh}\ \ell}{S; T, \langle \theta; E[\mathsf{new}\ v\ @\ \mathtt{rgn}_\imath] \rangle \rightsquigarrow S[\imath \mapsto S(\imath), \ell \mapsto v]; T, \langle \theta; E[\mathtt{loc}_\ell] \rangle} \quad \textit{(E-NL)}$$

$$\frac{\ell \mapsto v' \in S(\imath) \qquad \imath \in \mathsf{wlocked}(\theta) \qquad \imath \notin \mathsf{rwlocked}(T)}{S; T, \langle \theta; E[\mathtt{loc}_\ell := v] \rangle \rightsquigarrow S[\imath \mapsto S(\imath)[\ell \mapsto v]]; T, \langle \theta; E[()] \rangle} \quad \textit{(E-AS)}$$

$$\frac{\ell \mapsto v \in S(\imath) \qquad \imath \in \mathsf{rwlocked}(\theta) \qquad \imath \notin \mathsf{wlocked}(T)}{S; T, \langle \theta; E[\mathtt{deref}\ \mathtt{loc}_\ell] \rangle \rightsquigarrow S; T, \langle \theta; E[v] \rangle} \quad \textit{(E-D)}$$

$$\frac{\imath \in \mathsf{live}(\theta) \qquad \theta' = \theta, \imath \mapsto (\eta + \eta', \jmath) \qquad \mathsf{mutex}(\{\theta'\} \cup \{\theta'' \mid \langle \theta''; e' \rangle \in T\})}{S; T, \langle \theta, \imath \mapsto (\eta, \jmath); E[\mathtt{cap}_{\eta'}\ \mathtt{rgn}_\imath] \rangle \rightsquigarrow S; T, \langle \theta'; E[()] \rangle} \quad \textit{(E-CP)}$$

Σχήμα. 2.3: Σχέση αποτίμησης $C \rightsquigarrow C'$.

νέα εφέ όταν υπάρχουν παρενέργειες. Διαφορετικά, επιστρέφουν το κενό εφέ. Ο έλεγχος εφέ γίνεται στον κανόνα *T-NR* από την συνάρτηση translate, η οποία ορίζεται στο Σχήμα 2.7. Το εφέ αναδρομικών συναρτήσεων (κανόνας *T-FX*) υπολογίζεται από την συνάρτηση summary, η οποία ορίζεται στο Σχήμα 2.8.

2.5 Ασφάλεια τύπων

Σε αυτήν την ενότητα παρουσιάζουμε συνοπτικά το θεώρημα ασφάλειας τύπων το οποίο εγγυάται ότι τα έγκυρα προγράμματα δεν μπορούν να κολλήσουν. Διαισθητικά, ένα πρόγραμμα θεωρείται ότι δεν έχει κολλήσει όταν κάθε νήμα του μπορεί να εκτελέσει ένα βήμα ή περιμένει κάποια κλειδαριά. Ένα νήμα που περιέχει σφάλματα πρόσβασης μνήμης και συνθήκες ανταγωνισμού δεν μπορεί να εκτελέσει βήματα. Η ασφάλεια τύπων βασίζεται στην απόδειξη των λημμάτων *διατήρησης* και *προόδου*.

Λήμμα 2.1 (Πρόοδος) Έστω $R; M$ ένα καθολικό περιβάλλον τύπων και $S; T$ ένα έγκυρο πρόγραμμα τέτοιο ώστε $R; M \vdash S; T$. Τότε ισχύει ότι το $S; T$ δεν έχει κολλήσει ($\vdash S; T$).

$$
\begin{array}{lcll}
\mathsf{merge}(\emptyset) & = & \emptyset & \\
\mathsf{merge}(\xi, r \mapsto \eta) & = & \mathsf{merge}(\xi), r \mapsto \eta & \text{if } r \notin \{r' \mid r' \mapsto \eta' \in \xi\} \\
\mathsf{merge}(\xi, r \mapsto \eta, r \mapsto \eta') & = & \mathsf{merge}(\xi, r \mapsto (\eta + \eta')) & \\
\mathsf{ok}(n_1, n_2, n_3) & = & n_1 \geq 0 \wedge n_2 \geq 0 \wedge n_3 \geq 0 & \\
(c_1, w_1, z_1) \oplus (c_2, w_2, z_2) & = & (c_1 + c_2, w_1 + w_2, z_1 + z_2) & \text{if } \mathsf{ok}(c_1, w_1, z_1) \wedge \mathsf{ok}(c_2, w_2, z_2) \wedge \\
& & & \quad (w_1 = 0 \vee w_2 = 0) \wedge (c_2 > 0) \wedge \\
& & & \quad (c_1 = 0 \implies w_1 = z_1 = 0) \wedge \\
& & & \quad (w_1 > 0 \implies z_2 = 0) \wedge \\
& & & \quad (w_2 > 0 \implies z_1 = 0)
\end{array}
$$

$$
\begin{array}{lcll}
\mathsf{ancestors}(\theta, \bot) & = & \emptyset & \\
\mathsf{ancestors}(\theta, \imath) & = & \{\imath\} \cup \mathsf{ancestors}(\theta', \jmath) & \text{if } \theta = \theta', \imath \mapsto (\eta, \jmath) \\
\mathsf{ok}(\theta) & = & \forall \imath \mapsto (\eta, \jmath) \in \theta.\ \mathsf{ok}(\eta) \wedge \mathsf{ancestors}(\theta, \imath) \text{ defined} & \\
\mathsf{live}(\theta) & = & \{\imath \mid \forall \jmath \in \mathsf{ancestors}(\theta, \imath).\ \exists \jmath \mapsto (\eta, \jmath') \in \theta.\ \mathsf{ok}(\eta - (1, 0, 0))\} & \\
\mathsf{wlocked}(\theta) & = & \{\imath \mid \imath \in \mathsf{live}(\theta) \wedge \exists \jmath \mapsto (\eta, \jmath') \in \theta.\ \jmath \in \mathsf{ancestors}(\theta, \imath) \wedge \mathsf{ok}(\eta - (0, 1, 0))\} & \\
\mathsf{rlocked}(\theta) & = & \{\imath \mid \imath \in \mathsf{live}(\theta) \wedge \exists \jmath \mapsto (\eta, \jmath') \in \theta.\ \jmath \in \mathsf{ancestors}(\theta, \imath) \wedge \mathsf{ok}(\eta - (0, 0, 1))\} & \\
\mathsf{rwlocked}(\theta) & = & \mathsf{rlocked}(\theta) \cup \mathsf{wlocked}(\theta) & \\
\mathsf{wlocked}(T) & = & \{\imath \mid \exists \langle \theta, e \rangle \in T.\ \imath \in \mathsf{wlocked}(\theta)\} & \\
\mathsf{rwlocked}(T) & = & \{\imath \mid \exists \langle \theta, e \rangle \in T.\ \imath \in \mathsf{rwlocked}(\theta)\} & \\
\mathsf{mutex}(\{\theta_1, \ldots, \theta_n\}) & = & \forall \imath \neq \jmath.\ \mathsf{rwlocked}(\theta_\imath) \cap \mathsf{wlocked}(\theta_\jmath) = \mathsf{wlocked}(\theta_\imath) \cap \mathsf{rwlocked}(\theta_\jmath) = \emptyset & \\
\mathsf{hierarchy_ok}(\theta_1; \theta_2) & = & \forall \imath \mapsto (\eta, \jmath) \in \theta_1.\ \exists\, \imath \mapsto (\eta', \jmath') \in \theta_2.\ (\jmath = \jmath' \vee (\jmath = \bot \wedge \jmath' \notin \mathsf{dom}(\theta_1))) &
\end{array}
$$

$$
\frac{}{\emptyset \vdash \theta = \theta \oplus \emptyset}
\qquad
\frac{\begin{array}{cc} \eta = \eta_1 \oplus \eta_2 & \xi \vdash \theta = \theta_1 \oplus \theta_2 \\ \forall \imath' \in \mathsf{dom}(\xi).\ \imath \notin \mathsf{ancestors}(\theta, \imath') & \jmath' = \text{if } \jmath \in \mathsf{dom}(\xi) \text{ then } \jmath \text{ else } \bot \end{array}}{\xi, \imath \mapsto \eta_2 \vdash \theta, \imath \mapsto (\eta, \jmath) = \theta_1, \imath \mapsto (\eta_1, \jmath) \oplus \theta_2, \imath \mapsto (\eta_2, \jmath')}
$$

$$
\frac{}{\emptyset - \emptyset = \emptyset}
\qquad
\frac{\eta_1 \geq \eta_2 \qquad \theta_1 - \theta_2 = \theta'}{\theta_1, r \mapsto \eta_1 - \theta_2 \mapsto \eta_2 = \theta', r \mapsto \eta_1 - \eta_2}
$$

Σχήμα. 2.4: Βοηθητικές συναρτήσεις και κατηγορήματα.

$$
\begin{array}{lcl}
\textbf{Type} & \tau & ::= \texttt{unit} \mid \texttt{bool} \mid \tau \xrightarrow{\gamma} \tau \mid \forall \rho.\, \tau \mid \texttt{Ref}(\tau, r) \mid \texttt{Rgn}(r) \\
\textbf{Constraint} & \delta & ::= \texttt{R} \mid \texttt{W} \mid \neg\texttt{RW} \mid \neg\texttt{W} \mid \texttt{Live} \mid \neg\texttt{Live} \\
\textbf{Event} & \zeta & ::= \texttt{Cap}\, \xi \mid \delta\, r \mid \texttt{Spawn}\, \xi\, \gamma \mid \texttt{Join}\, \gamma\, \gamma \\
\textbf{Effect} & \gamma & ::= \emptyset \mid \zeta :: \gamma \\
\textbf{Type context} & \square & ::= \emptyset \mid \Gamma, x : \tau \\
\textbf{Region context} & \square & ::= \emptyset \mid \Delta, \rho \\
\textbf{Heap context} & M & ::= \emptyset \mid M, \ell \mapsto (\tau, \imath) \\
\textbf{Store context} & R & ::= \emptyset \mid R, \imath
\end{array}
$$

Σχήμα. 2.5: Σύνταξη τύπων, εφέ και περιβαλλόντων.

$$\frac{x : \tau \in \Gamma \qquad \vdash R; M; \Delta; \Gamma}{R; M; \Delta; \Gamma \vdash x : \tau \,\&\, \emptyset} \textit{(T-V)} \qquad \frac{\vdash R; M; \Delta; \Gamma}{R; M; \Delta; \Gamma \vdash () : \texttt{unit} \,\&\, \emptyset} \textit{(T-U)}$$

$$\frac{\vdash R; M; \Delta; \Gamma}{R; M; \Delta; \Gamma \vdash \texttt{true} : \texttt{bool} \,\&\, \emptyset} \textit{(T-TR)} \qquad \frac{\vdash R; M; \Delta; \Gamma}{R; M; \Delta; \Gamma \vdash \texttt{false} : \texttt{bool} \,\&\, \emptyset} \textit{(T-FL)}$$

$$\frac{\imath \in R \cup \{\bot\} \qquad \vdash R; M; \Delta; \Gamma}{R; M; \Delta; \Gamma \vdash \texttt{rgn}_\imath : \texttt{Rgn}(\imath) \,\&\, \emptyset} \textit{(T-R)} \qquad \frac{\ell \mapsto (\tau, \imath) \in M \qquad \vdash R; M; \Delta; \Gamma}{R; M; \Delta; \Gamma \vdash \texttt{loc}_\ell : \texttt{Ref}(\tau, \imath) \,\&\, \emptyset} \textit{(T-L)}$$

$$\frac{R; M; \Delta; \Gamma, x : \tau_1 \vdash e : \tau_2 \,\&\, \gamma}{R; M; \Delta; \Gamma \vdash \lambda x.\, e : \tau_1 \xrightarrow{\gamma} \tau_2 \,\&\, \emptyset} \textit{(T-F)}$$

$$\frac{R; M; \Delta; \Gamma \vdash e_1 : \tau_1 \xrightarrow{\gamma} \tau_2 \,\&\, \gamma_1 \qquad R; M; \Delta; \Gamma \vdash e_2 : \tau_1 \,\&\, \gamma_2}{R; M; \Delta; \Gamma \vdash e_1\ e_2 : \tau_2 \,\&\, \gamma_1 :: \gamma_2 :: \gamma} \textit{(T-A)}$$

$$\frac{R; \Delta \vdash \Gamma \qquad R; M; \Delta, \rho; \Gamma \vdash f : \tau \,\&\, \emptyset}{R; M; \Delta; \Gamma \vdash \Lambda\rho.\, f : \forall\rho.\, \tau \,\&\, \emptyset} \textit{(T-RF)}$$

$$\frac{R; M; \Delta; \Gamma \vdash e : \forall\rho.\, \tau \,\&\, \gamma \qquad R; \Delta \vdash r \qquad r \neq \bot}{R; M; \Delta; \Gamma \vdash e\,[r] : \tau[r/\rho] \,\&\, \gamma} \textit{(T-RP)}$$

$$\frac{R; M; \Delta; \Gamma \vdash e : \texttt{bool} \,\&\, \gamma \qquad R; M; \Delta; \Gamma \vdash e_1 : \tau \,\&\, \gamma_1 \qquad R; M; \Delta; \Gamma \vdash e_2 : \tau \,\&\, \gamma_2}{R; M; \Delta; \Gamma \vdash \texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 : \tau \,\&\, \gamma :: \texttt{Join}\,\gamma_1\,\gamma_2} \textit{(T-IF)}$$

$$\frac{R; \Delta \vdash \xi \qquad R; M; \Delta; \Gamma \vdash e : \texttt{unit} \,\&\, \gamma \qquad \mathsf{dom}(\xi) = \mathsf{dom}(\gamma)}{R; M; \Delta; \Gamma \vdash \texttt{spawn}_\xi\ e : \texttt{unit} \,\&\, \texttt{Spawn}\,\xi\,\gamma} \textit{(T-SP)}$$

$$\frac{\begin{array}{c}\gamma_L = \{\texttt{Live}\,r \mid r \in \mathsf{dom}(\phi(\emptyset))\} \qquad \gamma_s = \mathsf{summary}(\phi(\gamma_L)) \\ R; M; \Delta; \Gamma, x : \tau_1 \xrightarrow{\gamma_s} \tau_2 \vdash f : \tau_1 \xrightarrow{\phi(\gamma_s)} \tau_2 \,\&\, \emptyset\end{array}}{R; M; \Delta; \Gamma \vdash \texttt{fix}\ x.\, f : \tau_1 \xrightarrow{\gamma_s} \tau_2 \,\&\, \emptyset} \textit{(T-FX)}$$

$$\frac{R; M; \Delta; \Gamma \vdash e : \texttt{Rgn}(r) \,\&\, \gamma \qquad r \neq \bot}{R; M; \Delta; \Gamma \vdash \texttt{cap}_\eta\ e : \texttt{unit} \,\&\, \gamma :: \texttt{Cap}\,\{r \mapsto \eta\}} \textit{(T-CP)}$$

$$\frac{R; M; \Delta; \Gamma \vdash e_1 : \texttt{Ref}(\tau, r) \,\&\, \gamma_1 \qquad R; M; \Delta; \Gamma \vdash e_2 : \tau \,\&\, \gamma_2 \qquad r \neq \bot}{R; M; \Delta; \Gamma \vdash e_1 := e_2 : \texttt{unit} \,\&\, \gamma_1 :: \gamma_2 :: \texttt{W}\,r} \textit{(T-AS)}$$

$$\frac{R; M; \Delta; \Gamma \vdash e : \texttt{Ref}(\tau, r) \,\&\, \gamma \qquad r \neq \bot}{R; M; \Delta; \Gamma \vdash \texttt{deref}\ e : \tau \,\&\, \gamma :: \texttt{R}\,r} \textit{(T-D)}$$

$$\frac{R; M; \Delta; \Gamma \vdash e_1 : \tau \,\&\, \gamma_1 \qquad R; M; \Delta; \Gamma \vdash e_2 : \texttt{Rgn}(r) \,\&\, \gamma_2 \qquad r \neq \bot}{R; M; \Delta; \Gamma \vdash \texttt{new}\ e_1 \mathbin{@} e_2 : \texttt{Ref}(\tau, r) \,\&\, \gamma_1 :: \gamma_2 :: \texttt{Live}\,r} \textit{(T-NL)}$$

$$\frac{\begin{array}{ll} R; M; \Delta; \Gamma \vdash e_1 : \texttt{Rgn}(r) \,\&\, \gamma_1 & R; M; \Delta, \rho; \Gamma, x : \texttt{Rgn}(\rho) \vdash e_2 : \tau \,\&\, \gamma_2 \\ R; \Delta \vdash \tau & \mathsf{translate}(\gamma_2, \rho, (1, 1, 0), r) = \gamma_2' \end{array}}{R; M; \Delta; \Gamma \vdash \texttt{newrgn}\ \rho, x \mathbin{@} e_1\ \texttt{in}\ e_2 : \tau \,\&\, \gamma_1 :: \texttt{Live}\,r :: \gamma_2'} \textit{(T-NR)}$$

Σχήμα. 2.6: Κανόνες τύπων.

$$
\begin{array}{lcll}
\mathsf{rg}(n_1, n_2, n_3) & = & n_1 & \\
\mathsf{wr}(n_1, n_2, n_3) & = & n_2 & \\
\mathsf{rd}(n_1, n_2, n_3) & = & n_3 & \\
\mathsf{bot}(\delta, \bot) & = & \emptyset & \text{if } \delta \notin \{\mathtt{R}, \mathtt{W}\} \\
\mathsf{bot}(\delta, r) & = & \delta\, r & \text{if } r \neq \bot \\
\mathsf{solve}(\mathtt{R}, r, \eta) & = & \mathsf{bot}(\mathtt{Live}, r) & \text{if } \mathsf{ok}(\eta - (1,0,0)) \wedge \mathsf{wr}(\eta) + \mathsf{rd}(\eta) > 0 \\
\mathsf{solve}(\mathtt{R}, r, \eta) & = & \mathsf{bot}(\mathtt{R}, r) & \text{if } \mathsf{ok}(\eta - (1,0,0)) \wedge \mathsf{wr}(\eta) + \mathsf{rd}(\eta) = 0 \\
\mathsf{solve}(\mathtt{W}, r, \eta) & = & \mathsf{bot}(\mathtt{Live}, r) & \text{if } \mathsf{ok}(\eta - (1,0,0)) \wedge \mathsf{wr}(\eta) > 0 \\
\mathsf{solve}(\mathtt{W}, r, \eta) & = & \mathsf{bot}(\mathtt{W}, r) & \text{if } \mathsf{ok}(\eta - (1,0,0)) \wedge \mathsf{wr}(\eta) = 0 \\
\mathsf{solve}(\neg\mathtt{RW}, r, \eta) & = & \mathsf{bot}(\neg\mathtt{RW}, r) & \text{if } \mathsf{ok}(\eta - (1,0,0)) \wedge \mathsf{wr}(\eta) = \mathsf{rd}(\eta) = 0 \\
\mathsf{solve}(\neg\mathtt{W}, r, \eta) & = & \mathsf{bot}(\neg\mathtt{W}, r) & \text{if } \mathsf{ok}(\eta - (1,0,0)) \wedge \mathsf{wr}(\eta) = 0 \\
\mathsf{solve}(\mathtt{Live}, r, \eta) & = & \mathsf{bot}(\mathtt{Live}, r) & \text{if } \mathsf{ok}(\eta - (1,0,0)) \\
\mathsf{solve}(\neg\mathtt{Live}, r, \eta) & = & \emptyset & \text{if } \mathsf{ok}(\eta) \wedge \mathsf{rg}(\eta) = 0 \\
\mathsf{solve}(\neg\mathtt{Live}, r, \eta) & = & \neg\mathtt{Live}\, r & \text{if } \mathsf{ok}(\eta - (1,0,0)) \wedge r \neq \bot \\
\mathsf{p\text{-}constraint}(r, \eta) & = & \mathsf{bot}(\neg\mathtt{RW}, r) & \text{if } \mathsf{wr}(\eta) > 0 \\
\mathsf{p\text{-}constraint}(r, \eta) & = & \mathsf{bot}(\neg\mathtt{W}, r) & \text{if } \mathsf{wr}(\eta) = 0 \wedge \mathsf{rd}(\eta) > 0 \\
\mathsf{p\text{-}constraint}(r, \eta) & = & \emptyset & \text{if } \mathsf{wr}(\eta) = \mathsf{rd}(\eta) = 0
\end{array}
$$

$$
\frac{\mathsf{solve}(\neg\mathtt{Live}, r', \eta) = \gamma}{\mathsf{translate}(\emptyset, r, \eta, r') = \gamma}\ \textit{(TR-E)}
\qquad
\frac{r \notin \mathsf{dom}(\xi) \qquad \mathsf{translate}(\gamma, r, \eta, r') = \gamma'}{\mathsf{translate}(\mathtt{Cap}\,\xi :: \gamma, r, \eta, r') = \mathtt{Cap}\,\xi :: \gamma'}\ \textit{(TR-CN)}
$$

$$
\frac{\begin{array}{c}\mathsf{merge}(\xi) = \xi', r \mapsto \eta' \qquad \gamma_s = \mathsf{solve}(\mathtt{Live}, r', \eta) :: \mathtt{Cap}\,\xi' \\ \mathsf{translate}(\gamma, r, \eta + \eta', r') = \gamma' \qquad \mathsf{ok}(\eta + \eta')\end{array}}{\mathsf{translate}(\mathtt{Cap}\,\xi :: \gamma, r, \eta, r') = \gamma_s :: \gamma'}\ \textit{(TR-CT)}
$$

$$
\frac{r_1 \neq r_2 \qquad \mathsf{translate}(\gamma, r_2, \eta, r') = \gamma'}{\mathsf{translate}(\delta\, r_1 :: \gamma, r_2, \eta, r') = \delta\, r_1 :: \gamma'}\ \textit{(TR-DN)}
$$

$$
\frac{\mathsf{solve}(\delta, r', \eta) = \gamma_s \qquad \mathsf{translate}(\gamma, r, \eta, r') = \gamma'}{\mathsf{translate}(\delta\, r :: \gamma, r, \eta, r') = \gamma_s :: \gamma'}\ \textit{(TR-DT)}
$$

$$
\frac{\mathsf{translate}(\gamma_1 :: \gamma, r, \eta, r') = \gamma_1' \qquad \mathsf{translate}(\gamma_2 :: \gamma, r, \eta, r') = \gamma_2'}{\mathsf{translate}(\mathtt{Join}\,\gamma_1\,\gamma_2 :: \gamma, r, \eta, r') = \mathtt{Join}\,\gamma_1'\,\gamma_2'}\ \textit{(TR-J)}
$$

$$
\frac{r \notin \mathsf{dom}(\xi) \qquad \mathsf{translate}(\gamma, r, \eta, r') = \gamma'}{\mathsf{translate}(\mathtt{Spawn}\,\xi\,\gamma_s :: \gamma, r, \eta, r') = \mathtt{Spawn}\,\xi\,\gamma_s :: \gamma'}\ \textit{(TR-SN)}
$$

$$
\frac{\begin{array}{c}\mathsf{merge}(\xi) = \xi', r \mapsto \eta_s \qquad \eta = \eta_r \oplus \eta_s \qquad r_s = \text{if } r' \in \mathsf{dom}(\xi) \text{ then } r' \text{ else } \bot \\ \mathsf{p\text{-}constraint}(r_s, \eta_r) = \gamma_s' \qquad \mathsf{translate}(\gamma_s, r, \eta_s, r_s) = \gamma_s'' \\ \mathsf{p\text{-}constraint}(r', \eta_s) = \gamma_r' \qquad \mathsf{translate}(\gamma, r, \eta_r, r') = \gamma_r'' \qquad \gamma_0 = \mathsf{solve}(\mathtt{Live}, r', \eta)\end{array}}{\mathsf{translate}(\mathtt{Spawn}\,\xi\,\gamma_s :: \gamma, r, \eta, r') = \gamma_0 :: \mathtt{Spawn}\,\xi'\,(\gamma_s' :: \gamma_s'') :: \gamma_r' :: \gamma_r''}\ \textit{(TR-ST)}
$$

Σχήμα. 2.7: Κανόνες επικύρωσης και μετασχηματισμού εφέ.

$$\begin{array}{rcl} \xi_1 - \xi_2 & = & \{r \mapsto \xi_1(r) - \xi_2(r)\} \quad \text{where } \xi(r) = \eta \text{ if } r \mapsto \eta \in \xi \text{ and } \xi(r) = (0,0,0) \text{ otherwise} \\ \mathsf{ok}(\xi) & = & \forall \imath \mapsto \eta \in \xi.\, \mathsf{ok}(\eta) \end{array}$$

$$\frac{\mathsf{ok}(\xi)}{\mathsf{recursive}(\xi; \emptyset) = \xi} \textit{(R-E)} \qquad \frac{\delta \notin \{\neg\mathtt{RW}, \neg\mathtt{W}, \neg\mathtt{Live}\} \quad \mathsf{recursive}(\xi; \gamma) = \xi'}{\mathsf{recursive}(\xi; \delta\, r :: \gamma) = \xi'} \textit{(R-D)}$$

$$\frac{\mathsf{ok}(\xi) \quad \mathsf{recursive}(\xi - \xi'; \gamma) = \xi''}{\mathsf{recursive}(\xi; \mathtt{Cap}\, \xi' :: \gamma) = \xi''} \textit{(R-C)}$$

$$\frac{\begin{array}{cc} \mathsf{ok}(\xi) & \xi_r = \xi - \xi_s \\ \forall r \mapsto \eta \in \xi_s.\, \mathsf{rd}(\eta) = \mathsf{wr}(\eta) = 0 & \mathsf{recursive}(\xi_r; \gamma) = \xi_r' \end{array}}{\mathsf{recursive}(\xi; \mathtt{Spawn}\, \xi_s\, \gamma_s :: \gamma) = \xi_r'} \textit{(R-S)}$$

$$\frac{\mathsf{recursive}(\xi; \gamma_1) = \xi' \quad \mathsf{recursive}(\xi; \gamma_2) = \xi' \quad \mathsf{recursive}(\xi'; \gamma) = \xi''}{\mathsf{recursive}(\xi; \mathtt{Join}\, \gamma_1\, \gamma_2 :: \gamma) = \xi''} \textit{(R-J)}$$

$$\frac{\begin{array}{c} \mathsf{recursive}(\xi_1; \gamma) = \xi_1 \\ \xi_1 = \{r \mapsto (1,0,0) \mid r \in \mathsf{dom}(\gamma)\} \quad \xi_2 = \{r \mapsto (-1,0,0) \mid r \in \mathsf{dom}(\gamma)\} \end{array}}{\mathsf{summary}(\gamma) = \mathtt{Cap}\, \xi_1 :: \mathtt{Spawn}\, \xi_1\, (\gamma :: \mathtt{Cap}\, \xi_2)} \textit{(SUM)}$$

Σχήμα. 2.8: Συνοψισμένα εφέ αναδρομικών συναρτήσεων.

Λήμμα 2.2 (Διατήρηση) Έστω $R; M$ ένα καθολικό περιβάλλον τύπων και $S; T$ ένα έγκυρο πρόγραμμα τέτοιο ώστε $R; M \vdash S; T$. Εάν η λειτουργική σημασιολογία κάνει ένα βήμα $S; T \rightsquigarrow S'; T'$, τότε υπάρχουν $R' \supseteq R$ και $M' \supseteq M$ τέτοια ώστε το τελικό πρόγραμμα είναι έγκυρο ($R'; M' \vdash S'; T'$).

Θεώρημα 2.1 (Ασφάλεια τύπων) Έστω ότι υπάρχει κάποια έκφραση e τέτοια ώστε $\emptyset; \emptyset; \emptyset; \emptyset \vdash e : \mathtt{unit} \,\&\, \emptyset$. Εάν η λειτουργική σημασιολογία κάνει οποιονδήποτε αριθμό βημάτων $S_0; T_0 \rightsquigarrow^n S_n; T_n$, τότε η τελική κατάσταση του προγράμματος $S_n; T_n$ δεν έχει κολλήσει.

Απόδειξη. Έστω $R_0 = \emptyset$ και $M_0 = \emptyset$. Χρησιμοποιώντας τις υποθέσεις είναι εύκολο να αποδείξουμε ότι $R_0; M_0 \vdash S_0; T_0$. Με επαγωγή στον αριθμό των βημάτων n και χρησιμοποιώντας το Λήμμα 2.2 υπάρχει $R_n \supseteq R_0$ and $M_n \supseteq M_0$ τέτοιο ώστε $R_n; M_n \vdash S_n; T_n$. Το Λήμμα 2.1 δίνει ότι η τελική κατάσταση $S_n; T_n$ δεν έχει κολλήσει.

$$\frac{\vdash R; M; \Delta; \Gamma \qquad R; \Delta \vdash \tau}{R; M; \Delta; \Gamma \vdash \square : \tau \xrightarrow{\tau} \emptyset\gamma_2} \textit{(E0)} \qquad \frac{\begin{array}{c} R; M; \Delta; \Gamma \vdash E : \tau_2 \xrightarrow{\tau_3} \gamma_2 \\ R; M; \Delta; \Gamma \vdash F : \tau_1 \xrightarrow{\tau_2} \gamma_1 \end{array}}{R; M; \Delta; \Gamma \vdash E[F] : \tau_1 \xrightarrow{\tau_3} \gamma_1 :: \gamma_2} \textit{(E1)}$$

$$\frac{\begin{array}{c} \tau \equiv \tau_1 \xrightarrow{\gamma_a} \tau_2 \qquad R; \Delta \vdash \tau \\ R; M; \Delta; \Gamma \vdash e_2 : \tau_1 \,\&\, \gamma_1 \end{array}}{R; M; \Delta; \Gamma \vdash \square\ e_2 : \tau \xrightarrow{\tau_2} \gamma_1 :: \gamma_a} \textit{(F1)} \qquad \frac{\begin{array}{c} \tau \equiv \tau_1 \xrightarrow{\gamma_a} \tau_2 \\ R; M; \Delta; \Gamma \vdash v_1 : \tau \,\&\, \emptyset \end{array}}{R; M; \Delta; \Gamma \vdash v_1\ \square : \tau_1 \xrightarrow{\tau_2} \gamma_a\gamma_3} \textit{(F2)}$$

$$\frac{\begin{array}{c} R; M; \Delta, \rho; \Gamma, x : \mathtt{Rgn}(\rho) \vdash e_2 : \tau \,\&\, \gamma_1 \qquad R; \Delta \vdash r \\ R; \Delta \vdash \tau \qquad \mathtt{Live}\, r :: \mathsf{translate}(\gamma_1, \rho, (1,1,0), r) = \gamma_2 \end{array}}{R; M; \Delta; \Gamma \vdash \mathtt{newrgn}\ \rho, x\ @\ \square\ \mathtt{in}\ e_2 : \mathtt{Rgn}(r) \xrightarrow{\tau} \gamma_2\gamma'} \textit{(F3)}$$

$$\frac{\vdash R; M; \Delta; \Gamma \qquad R; \Delta \vdash \forall\rho.\,\tau \qquad R; \Delta \vdash r \qquad r \neq \bot}{R; M; \Delta; \Gamma \vdash \square\,[r] : \forall\rho.\,\tau \xrightarrow{\tau[r/\rho]} \emptyset} \textit{(F4)}$$

$$\frac{R; M; \Delta; \Gamma \vdash e_2 : \mathtt{Rgn}(r) \,\&\, \gamma_2 \qquad R; \Delta \vdash \tau \qquad r \neq \bot}{R; M; \Delta; \Gamma \vdash \mathtt{new}\ \square\ @\ e_2 : \tau \xrightarrow{\mathtt{Ref}(\tau,r)} \gamma_2 :: \mathtt{Live}\, r\gamma'} \textit{(F5)}$$

$$\frac{R; \Delta \vdash r \qquad R; M; \Delta; \Gamma \vdash v : \tau \,\&\, \emptyset \qquad r \neq \bot}{R; M; \Delta; \Gamma \vdash \mathtt{new}\ v\ @\ \square : \mathtt{Rgn}(r) \xrightarrow{\mathtt{Ref}(\tau,r)} \mathtt{Live}\, r\gamma'} \textit{(F6)}$$

$$\frac{R; \Delta \vdash r \qquad R; M; \Delta; \Gamma \vdash e_2 : \tau \,\&\, \gamma_1 \qquad r \neq \bot}{R; M; \Delta; \Gamma \vdash \square := e_2 : \mathtt{Ref}(\tau, r) \xrightarrow{\mathtt{unit}} \&\, \gamma_1 :: \mathtt{W}\, r} \textit{(F7)}$$

$$\frac{R; M; \Delta; \Gamma \vdash \mathtt{loc}_\ell : \mathtt{Ref}(\tau, \imath) \,\&\, \emptyset \qquad \imath \neq \bot}{R; M; \Delta; \Gamma \vdash \mathtt{loc}_\ell := \square : \tau \xrightarrow{\mathtt{unit}} \&\, \mathtt{W}\, \imath} \textit{(F8)}$$

$$\frac{\vdash R; M; \Delta; \Gamma \qquad R; \Delta \vdash \mathtt{Ref}(\tau, r) \qquad r \neq \bot}{R; M; \Delta; \Gamma \vdash \mathtt{deref}\ \square : \mathtt{Ref}(\tau, r) \xrightarrow{\tau} \&\, \mathtt{R}\, r} \textit{(F9)}$$

$$\frac{\vdash R; M; \Delta; \Gamma \qquad R; \Delta \vdash \mathtt{Rgn}(r) \qquad r \neq \bot}{R; M; \Delta; \Gamma \vdash \mathtt{cap}_\eta\ \square : \mathtt{Rgn}(r) \xrightarrow{\mathtt{unit}} \&\, \mathtt{Cap}\, \{r \mapsto \eta\}} \textit{(F10)}$$

$$\frac{R; M; \Delta; \Gamma \vdash e_1 : \tau \,\&\, \gamma_2 \qquad R; M; \Delta; \Gamma \vdash e_2 : \tau \,\&\, \gamma_3}{R; M; \Delta; \Gamma \vdash \mathtt{if}\ \square\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 : \mathtt{bool} \xrightarrow{\tau} \&\, \mathtt{Join}\, \gamma_2\, \gamma_3} \textit{(F11)}$$

Σχήμα. 2.9: Κανόνες τύπων για το πλαίσιο αποτίμησης.

Κεφάλαιο 3

Ταυτόχρονη Cyclone

3.1 Επισκόπηση

Η Cyclone είναι μια ασφαλής παραλλαγή της C που προσφέρει σύγχρονα χαρακτηριστικά της γλωσσών προγραμματισμού, όπως πολυμορφισμό πρώτης κατηγορίας, εξαιρέσεις, πλειάδες, αλγεβρικούς τύπους δεδομένων και διαχείριση μνήμης με περιοχές. Ωστόσο, η Cyclone δεν έχει ενσωματωμένη υποστήριξη για ταυτοχρονισμό. Σε αυτό το κεφάλαιο παρουσιάζουμε τα κύρια χαρακτηριστικά της Cyclone και δείχνουμε ότι οι εγγυήσεις ασφάλειας μνήμης διακυβεύονται με την εισαγωγή ταυτοχρονισμού στη γλώσσα μέσω εξωτερικών βιβλιοθηκών, όπως η pthreads. Στη συνέχεια συζητούμε την ενσωμάτωση του συστήματος τύπων και των γλωσσικών δομών του προηγούμενου κεφαλαίου στη Cyclone καθώς και τις αποφάσεις που πήραμε για να διασφαλιστεί απουσία σφαλμάτων μνήμης και συνθηκών ανταγωνισμού. Επίσης παρουσιάζουμε τα συγκριτικά αποτελέσματα μεταξύ προγραμμάτων Ταυτόχρονης Cyclone και προγραμμάτων C/pthreads.

3.2 Cyclone: Μια ασφαλής διάλεκτος της C

Σε αυτή την ενότητα παρουσιάζουμε την διαχείριση μνήμης με περιοχές στην γλώσσα Cyclone και εντοπίζουμε κάποιες πτυχές της που βελτιώνονται στην Ταυτόχρονη Cyclone. Όλα τα αποσπάσματα κώδικα σε αυτό το κεφάλαιο χρησιμοποιούν σύνταξη Cyclone.

3.2.1 Διαχείρηση μνήμης στη Cyclone

Η Cyclone μεταχειρίζεται ομοιόμορφα τα διάφορα τμήματα μνήμης, όπως την μνήμη σωρού, την μνήμη στοίβας και την μνήμη των επιμέρους περιοχών με την αντιστοίχηση τους σε *λογικές περιοχές μνήμης*, οι οποίες έχουν κοινές ιδιότητες. Για παράδειγμα, ένα πλαίσιο στοίβας αντιμετωπίζεται ως μια (λογική) περιοχή που κρατάει τις τιμές των μεταβλητών που δηλώνονται σε μια συνάρτηση. Ως άλλο παράδειγμα, όλες οι καθολικές μεταβλητές ενός προγράμματος περιέχονται σε μια αθάνατη περιοχή. Το σύστημα τύπων της Cyclone παρακολουθεί το σύνολο των *ζωντανών περιοχών* (δηλαδή, το εφέ) σε κάθε σημείο του προγράμματος και επαληθεύει ότι οι περιοχές που προσπελαύνονται είναι πράγματι υποσύνολο των ζωντανών περιοχών.

```
{ region<‘r> h ;                          // ζωντανές περιοχές: {‘r}
  int *‘r z = rnew(h) 42;                 //                    {‘r}
      ...                                 //                    {‘r}
}                                         //                    { }
```

Το παραπάνω παράδειγμα δείχνει πώς μια περιοχή λεξικής εμβέλειας μπορεί να δημιουργηθεί και να χρησιμοποιηθεί στην Cyclone: η πρώτη δήλωση δεσμεύει ένα νέο τμήμα μνήμης, και συσχετίζει το τμήμα αυτό με μια νέο όνομα λογικής περιοχής (μεταβλητή τύπου) ‘r. Χρησιμοποιούμε το σύμβολο ‘ πριν από ένα αναγνωριστικό περιοχής για να αναφερθούμε στην περιοχή σε επίπεδο τύπων. Διαφορετικά, το αναγνωριστικό περιοχής αναφέρεται αποκλειστικά σε επίπεδο προγράμματος. Τα

σχόλια στην δεξιά πλευρά του κώδικα δείχνουν το σύνολο των ζωντανών περιοχών σε κάθε σημείο του προγράμματος.

Η νέα περιοχή μπορεί να προσπελασθεί μέσω ενός *χειριστηρίου περιοχής* (h), η οποία αντιστοιχεί στον *μονήρη τύπο* `region`$<`r>$. Η δεύτερη δήλωση χρησιμοποιεί το χειριστήριο $h$ για να δεσμεύσει έναν ακέραιο και αποθηκεύει την τιμή 42. Ο τύπος της νέας αναφοράς επισημειώνεται με την περιοχή $`r$ (δηλαδή, `int` $*$ $`r$). Το σύστημα τύπων διασφαλίζει ότι η αναφορά είναι προσπελάσιμη μόνο όταν το $`r$ βρίσκεται στο τρέχον εφέ. Στη Cyclone, η ενιαία αντιμετώπιση των διάφορων περιοχών μνήμης επιτρέπει πολυμορφισμό σε διαφορετικά τμήματα μνήμης.

$$\texttt{void swap}\ (\texttt{int} * `r_1\ x\ ,\ \texttt{int} * `r_2\ y);$$

Για παράδειγμα, η παραπάνω γραμμή κώδικα δηλώνει μια συνάρτηση που αντιμεταθέτει το περιεχόμενο των μεταβλητών x και y, οι οποίες βρίσκονται στις περιοχές $`r_1$ και $`r_2$ αντίστοιχα. Τα $`r_1$ και $`r_2$ είναι πολυμορφικά και μπορεί να αρχικοποιηθούν με οποιαδήποτε περιοχή. Η ακόλουθη γραμμή κώδικα αρχικοποιεί τα $`r_1$ και $`r_2$ με την ίδια περιοχή $`r$.

$$\begin{array}{l}
\{\ \texttt{region}<`r>h\ ; \\
\quad \texttt{int} * `r\ z = \texttt{rnew}(h)\ 42; \\
\quad \texttt{int} * `r\ y = \texttt{rnew}(h)\ 54; \\
\quad \texttt{swap}(z, y); \qquad\qquad // \ \textit{το εφέ της swap:}\ \{`r, `r\} \\
\}
\end{array}$$

Όπως φαίνεται στο παράδειγμα, η δυνατότητα απεριόριστης αντιγραφής περιοχών μνήμης στο εφέ μια συνάρτησης (π.χ, $\{`r, `r\}$) έχει ως συνέπεια ότι οι περιοχές δεν μπορούν να αποδεσμευτούν πρόωρα.

Για να επιτραπεί η άμεση αποδέσμευση πόρων στα προγράμματα Cyclone, υπάρχουν *τύποι παρακολούθησης* (tracked types) για αναφορές και περιοχές τα οποίοι δεν επιτρέπουν την ύπαρξη αντίγραφων περιοχών και αναφορών μέσα στο πρόγραμμα.

Η δική μας προσέγγιση επιτρέπει την κατασκευή σύνθετων ιεραρχικών περιοχών, την ατομική αποδέσμευση πολλαπλών περιοχών και την πρόωρη αποδέσμευση περιοχών σε οποιοδήποτε σημείο του προγράμματος χωρίς σημαντικούς περιορισμούς στην δημιουργία αντιγράφων περιοχών στο πρόγραμμα.

3.2.2 Ταυτοχρονισμός στη Cyclone

Η Cyclone δεν έχει ρητή γλωσσική υποστήριξη για ταυτοχρονισμό. Ωστόσο, παρέχει μια διεπαφή για τη βιβλιοθήκη pthreads, η οποία επιτρέπει στους προγραμματιστές να δημιουργούν νέα νήματα και να χρησιμοποιούν πολλές πρωτόγονες λειτουργίες συγχρονισμού για τον έλεγχο της αλληλεπίδρασης μεταξύ των νημάτων. Η διεπαφή για τη βιβλιοθήκη pthreads εξασφαλίζει ότι οι δομές του χρόνου εκτέλεσης έχουν αρχικοποιηθεί σωστά πριν τρέξει ένα νέο νήμα.

Για να διατηρηθεί η ασφάλεια μνήμης (π.χ., απουσία ξεκρέμαστων δεικτών), η Cyclone προϋποθέτει ότι οι μοιραζόμενες περιοχές μνήμης μπορούν να αποδεσμευτούν μόνο από τον συλλέκτη σκουπιδιών (garbage collector). Ο περιορισμός αυτός περιορίζει σημαντικά τα οφέλη που προέρχονται από την ρητή διαχείριση της διάρκειας ζωής των περιοχών μνήμης.

Επιπλέον, οι εγγυήσεις ασφάλειας μνήμης της Cyclone παύουν να ισχύουν σε ταυτόχρονα προγράμματα. Συγκεκριμένα, η ανάλυση ροής δεδομένων εντοπίζει τα σημεία του προγράμματος που μπορούν να απαλειφθούν οι δυναμικοί έλεγχοι ασφάλειας πρόσβασης μνήμης. Τα κριτήρια απαλοιφής της ανάλυσης αυτής ισχύουν μόνο για ακολουθιακά προγράμματα και δεν μπορούν να επεκταθούν σε ταυτόχρονα προγράμματα. Ακόμη, κάποια στοιχεία της Cyclone όπως το ταίριασμα προτύπων (pattern matching), η λειτουργία εναλλαγής (swap operation) και οι παχιοί δείκτες (fat pointers) δεν εκτελούνται *ατομικά* και μπορεί να προκαλέσουν σφάλματα πρόσβασης μνήμης σε ταυτόχρονα προγράμματα. Τρίτον, το σύστημα τύπων της Cyclone δεν εμποδίζει τις συνθήκες ανταγωνισμού. Όπως θα δούμε στις επόμενες ενότητες, έχουμε λύσει τα παραπάνω θέματα με την υλοποίηση και ενσωμάτωση του συστήματος τύπων μας στην Cyclone και την προσαρμογή του συστήματος χρόνου εκτέλεσης της στα νέα δεδομένα.

3.3 Ταυτόχρονη Cyclone

Σε αυτή την ενότητα, παρέχουμε αναλυτική περιγραφή της αλληλεπίδρασης μεταξύ του συστήματος τύπων μας και της Cyclone.

3.3.1 Εκτεταμένες περιοχές και σύστημα Ειδών

Σε αντίθεση με τις περιοχές λεξικής εμβέλειας, οι οποίες δεσμεύονται σε μορφή στοίβας, οι *εκτεταμένες* περιοχές δεσμεύονται σε οποιοδήποτε πρόγονο, σε οποιοδήποτε επίπεδο της ιεραρχίας. Θεωρούμε τον κύριο σωρό ($`H$) ως τη ρίζα της ιεραρχίας. Η διαχείριση ιεραρχικών περιοχών μνήμης γενικεύει την διαχείριση περιοχών σε μορφή στοίβας και επιτρέπει ακριβή έλεγχο της διάρκειας ζωής τους. Οι εκτεταμένες περιοχές είναι δυνάμει *μοιραζόμενες* και προστατεύονται από έμμεσες κλειδαριές. Οι περιοχές λεξικής εμβέλειας δεν μπορούν να μοιράζονται με ασφάλεια. Για παράδειγμα, ένα *πλαίσιο στοίβας* είναι μια περιοχή λεξικής εμβέλειας που δεν μπορεί να μοιρασθεί χωρίς να επιβαρυνθεί σημαντικά ο ταυτοχρονισμός μεταξύ νημάτων. Συνεπώς, χρησιμοποιούμε το σύστημα ειδών της Cyclone για να διαχωρίσουμε μεταξύ των αρχικών και των εκτεταμένων περιοχών σε επίπεδο τύπων και επιβάλουμε τον περιορισμό ότι μόνο οι περιοχές που ανήκουν στο εκτεταμένο είδος μπορούν να μοιρασθούν.

3.3.2 Εξαιρέσεις

Οι εξαιρέσεις μπορούν να προκύψουν *άμεσα* από τον προγραμματιστή ή *έμμεσα* από κάποιο δυναμικό έλεγχο ασφάλειας του συστήματος χρόνου εκτέλεσης. Οι εξαιρέσεις μεταβάλλουν ριζικά την ροή εκτέλεσης του προγράμματος και η μοντελοποίηση των εξαιρέσεων είναι σημαντική για την συνέπεια της ανάλυσης μας. Η ανάλυση μας συμπεραίνει αυτόματα τις εξαιρέσεις που εγείρονται στα σώματα των συναρτήσεων και επαληθεύει ότι οι ρητές επισημειώσεις εξαιρέσεων στις επικεφαλίδες συναρτήσεων είναι συνεπείς με τις αντίστοιχες της ανάλυσης.

3.3.3 Επανεισερχόμενες συναρτήσεις

Οι συναρτήσεις που καλούνται από νήματα εκτός του κυρίως νήματος ονομάζονται επανεισερχόμενες συναρτήσεις και φέρουν την επισημείωση `@re_entrant`. Οι συναρτήσεις αυτού του τύπου δεν έχουν πρόσβαση σε καθολικά δεδομένα, σε δεσμευμένη μνήμη του κυρίως σωρού (την ρίζα της ιεραρχίας μας) και μνήμη μη εκτεταμένων περιοχών που έχουν δεσμευτεί σε άλλα νήματα. Συνεπώς, το σύστημα μας διασφαλίζει απουσία συνθηκών ανταγωνισμού επιτρέποντας μόνο στις εκτεταμένες περιοχές να μοιράζονται μεταξύ νημάτων.

3.3.4 Πολυμορφισμός τύπων

Το σύστημα τύπων της Cyclone επιτρέπει την συγγραφή γενικών συναρτήσεων με πλήρως πολυμορφικά ορίσματα, τα οποία δεν φέρουν επισημειώσεις περιοχών. Τέτοιου είδους συναρτήσεις απαιτούν οι περιοχές των ορισμάτων τους είναι ζωντανές και προσβάσιμες κατά την κλήση τους. Αυτό όμως μπορεί να μην ισχύει με τις εκτεταμένες περιοχές μνήμης, οι οποίες μπορούν να αποδεσμευτούν και να ξεκλειδωθούν σε οποιοδήποτε σημείο του προγράμματος. Συνεπώς, δεν επιτρέπουμε σε εκτεταμένες περιοχές/αναφορές μνήμης να περαστούν σε γενικές συναρτήσεις.

3.3.5 Λειτουργία `cap`.

Ο τελεστής `cap` της τυπικής σημασιολογίας έχει υλοποιηθεί ως έχει:

```
#define rfree(h) cap (−1, 0, 0) (h)
region<‘r> h @ H;
    let z = rnew(h) 42;
```

```
...
*z = *z + 17;
...
rfree(h);
...
```

3.3.6 Εκτεταμένες λειτουργίες περιοχών.

Για να επιταχύνουμε την διαδικασία μεταγλώττισης, απαιτούμε σε συναρτήσεις που μεταχειρίζονται άμεσα ή μεταβατικά εκτεταμένες περιοχές μνήμης να φέρουν την επισημείωση @xrgn.

3.3.7 Δημιουργία νημάτων.

Η δημιουργία νημάτων γίνεται με τον τελεστή spawn. Ο τελεστής δέχεται δύο ορίσματα. Το πρώτο είναι μια λίστα πλειάδων που αποτελούνται από χειριστήρια περιοχών και τους αντίστοιχους μετρητές τους που περνάμε στο νέο νήμα. Το δεύτερο όρισμα είναι κλήση συνάρτησης, οι οποία θα εκτελεστεί στο νέο νήμα. Η συνάρτηση πρέπει να είναι επανεισερχόμενη και να μην παράγει εξαιρέσεις. Αυτή η λειτουργία επιστρέφει άμεσα μετά την δημιουργία του νέου νήματος χωρίς να μπλοκάρει.

3.4 Υλοποίηση

3.4.1 Μεταγλωττιστής

Υλοποιήσαμε το σύστημα τύπων μας ως μια ανεξάρτητη φάση της μεταγλώττισης. Ο μεταγλωττιστής πραγματοποιεί τις αρχικές φάσεις όπως έλεγχο τύπων, ανάλυση ροής δεδομένων και ανάλυση ροής έλεγχου του προγράμματος. Έπειτα εκτελεί ανάλυση εξαιρέσεων, όπως περιγράψαμε στην Ενότητα 3.3.2, χρησιμοποιώντας τον γράφο ροής δεδομένων. Σε αυτή τη φάση γίνεται ο έλεγχος της ορθότητας των επισημειώσεων εξαιρέσεων που βρίσκονται στις επικεφαλίδες των συναρτήσεων. Στη συνέχεια εκτελείται η ανάλυση εφέ εκτεταμένων περιοχών, η οποία μεταφέρει τα εφέ στον γράφο έλεγχου και προσθέτει νέα εφέ στο συνολικό εφέ χωρίς επιπλέον ελέγχους. Η *μετάφραση* και *επικύρωση* εφέ γίνεται αφότου συγκεντρώσουμε το συνολικό εφέ μιας περιοχής. Οι εντολές ελέγχου ροής και επανάληψης αντιμετωπίζονται *συντηρητικά*. Για παράδειγμα, απαιτούμε οι τελικοί μετρητές στην έξοδο των διακλαδώσεων ροής ελέγχου να είναι οι ίδιοι. Συνεπώς, είναι δυνατόν να δημιουργήσουμε σε μια διακλάδωση ένα νέο νήμα, το οποίο καταναλώνει κάποιες κλειδαριές και περιοχές μνήμης αρκεί να καταναλωθεί ο ίδιος αριθμός κλειδαριών και περιοχών στις υπόλοιπες διακλαδώσεις.

3.4.2 Παραγωγή κώδικα και σύστημα χρόνου εκτέλεσης

Παραγωγή κώδικα. Σε αυτό το στάδιο μεταφράζουμε τις εντολές spawn και region σε χαμηλού επιπέδου λειτουργίες, οι οποίες αναλαμβάνουν την δέσμευση μνήμης, μεταφορά παραμέτρων σε νέα νήματα και δημιουργία στοίβας.

Σύστημα χρόνου εκτέλεσης. Το σύστημα χρόνου εκτέλεσης εκτελεί τις ακόλουθες λειτουργίες:

- καταγράφει νέες περιοχές στην τοπική ιεραρχία κάθε νήματος.
- ενημερώνει τους μετρητές περιοχών της τοπικής ιεραρχίας, όταν αποδεσμεύεται ένα υποδέντρο.
- δημιουργεί την τοπική ιεραρχία νέων νημάτων, χρησιμοποιώντας την επισημείωση του τελεστή spawn.
- κλειδώνει την ιεραρχία με κατεύθυνση από πάνω τα κάτω, αριστερά προς τα δεξιά για όλα τα νήματα, ώστε να αποφευχθούν τα αδιέξοδα.
- εκτελεί ταυτόχρονες αιτήσεις δέσμευσης μνήμης χωρίς την χρήση κλειδαριών.

3.5 Αποτίμηση απόδοσης

Σε αυτή την ενότητα παρουσιάζουμε την αποτίμηση της Ταυτόχρονης Cyclone, υλοποιώντας προγράμματα από το "The Computer Language Benchmarks Game"[1] σε Ταυτόχρονη Cyclone και C/pthreads και αναλύουμε τα συγκριτικά αποτελέσματα. Σε όλες τις περιπτώσεις, τα συγκρινόμενα προγράμματα υλοποιούν τους ίδιους αλγόριθμους. Τα επτά προγράμματα που υλοποιήθηκαν είναι:[2]

binary-trees: αυτό το πρόγραμμα δεσμεύει, διασχίζει και αποδεσμεύει δυαδικά δέντρα. Η C/pthreads έκδοση χρησιμοποιεί διαχείριση μνήμης με περιοχές μέσω εξωτερικής βιβλιοθήκης.

chameneos-redux: αυτό το πρόγραμμα εξομοιώνει την αλληλεπίδραση διαφόρων πλασμάτων, τα οποία συγχρονίζονται ανά δυο με συμμετρικό τρόπο και ανταλλάσσουν χρώματα.

fannkuch-redux: αυτό το πρόγραμμα εκτελεί ταυτόχρονες προσβάσεις πινάκων που περιέχουν ακολουθίες ακεραίων αριθμών.

mandelbrot: αυτό το πρόγραμμα σχεδιάζει εικόνες bitmap που απεικονίζουν σύνολα Mandelbrot.

regex-dna: αυτό το πρόγραμμα καταγράφει τις συχνότητες μοτίβων DNA, επιτρέποντας ταυτόχρονη πρόσβαση νημάτων-εργατών στο διάνυσμα εισόδου.

spectral-norm: αυτό το πρόγραμμα χρησιμοποιεί επαναληπτική παραλληλία για τον υπολογισμό της φασματικής νόρμας ενός άπειρου πίνακα.

thread-ring: αυτό το πρόγραμμα δημιουργεί ένα μεγάλο αριθμό νημάτων, τα οποία σχηματίζουν ένα κύκλο, και περνάει ένα διακριτικό από το ένα νήμα στο επόμενο του.

Το μηχάνημα δοκιμής που χρησιμοποιείται είναι ένα quad-core 2.5GHz Intel (Q8300), με 4GB μνήμη RAM και 2x2MB μνήμη L2 cache, που τρέχει τον Linux πυρήνα 2.6.26-2. Ο GCC 4.3.2 χρησιμοποιήθηκε ως οπίσθιο τμήμα του μεταγλωττιστή μας και ως μεταγλωττιστής των προγραμμάτων C/pthreads. Τα συγκριτικά αποτελέσματα συνοψίζονται στον Πίνακα 3.1.

Όπως φαίνεται στον πίνακα τα προγράμματα χωρίζονται σε δυο κατηγορίες. Στην πρώτη κατηγορία έχουμε προγράμματα με περίπου την ίδια απόδοση με αυτά σε C/pthreads, όπως το spectral-norm (2% πιο αργό), mandelbrot (1% πιο γρήγορο), regex-dna (3% πιο αργό) και binary-trees (8% πιο γρήγορο). Στην περίπτωση των regex-dna και spectral-norm πετύχαμε παρόμοια απόδοση με αυτή των προγραμμάτων C/pthreads προσπελάζοντας ταυτόχρονα πίνακες που προστατεύονται από κλειδαριές ανάγνωσης. Στην περίπτωση του binary-trees, το αποτέλεσμα είναι ιδιαίτερα ενθαρρυντικό καθώς τα δύο προγράμματα ακολουθούν τον ίδιο αλγόριθμο και έχουν διαχείριση μνήμης με περιοχές.

Στην δεύτερη κατηγορία έχουμε προγράμματα που τρέχουν πιο αργά σε ταυτόχρονη Cyclone σε σχέση με τα προγράμματα C/pthreads, όπως τα fannkuch-redux (21% πιο αργό), thread-ring (56% πιο αργό) και chameneos-redux (492% πιο αργό). Στην περίπτωση των fannkuch-redux και thread-ring η επιβάρυνση οφείλεται στην υλοποίηση των λειτουργιών κλειδώματος μας που είναι λιγότερο βελτιστοποιημένη από την υλοποίηση της βιβλιοθήκης pthreads. Στην περίπτωση του chameneos-redux, χρησιμοποιούμε μόνο *μια* κλειδαριά εγγραφής για τον πίνακα που κρατάει τα δεδομένα των πλασμάτων. Το σύστημα τύπων μας αδυνατεί να χωρίσει τον πίνακα πλασμάτων σε διακριτά τμήματα που προσπελαύνονται ταυτόχρονα από πολλαπλά νήματα. Συνεπώς, αυτό δημιουργεί συμφόρηση και ελαττώνει δραματικά την παραλληλία μεταξύ νημάτων.

[1] Το URL είναι `http://shootout.alioth.debian.org/u32q/`.

[2] Η υλοποίηση των προγραμμάτων είναι διαθέσιμη από το URL: `http://www.softlab.ntua.gr/~pgerakios/cycinfer.tgz`.

benchmark	lang	CPU	memory	load per core (%)				elapsed	factor
binary-trees	c	13.281	100648	0	87	97	81	5.383	1.00
	cyc	15.725	121880	80	78	85	85	4.966	0.92
chameneos-redux	c	28.774	560	76	73	52	89	12.946	1.00
	cyc	241.679	720	88	87	78	85	73.199	5.65
fannkuch-redux	c	139.989	552	99	100	99	98	35.275	1.00
	cyc	171.499	716	100	99	100	100	42.947	1.22
mandelbrot	c	37.914	31868	81	100	85	97	10.485	1.00
	cyc	37.694	31924	99	83	84	99	10.354	0.99
regex-dna	c	6.500	832540	76	95	65	77	2.043	1.00
	cyc	6.568	832576	69	75	93	72	2.112	1.03
spectral-norm	c	10.609	616	100	99	98	99	2.686	1.00
	cyc	9.761	680	99	99	99	99	2.745	1.02
thread-ring	c	179.479	4520	8	42	5	40	121.565	1.00
	cyc	350.778	4748	44	45	1	5	188.443	1.55

Πίνακας. 3.1: Συγκριτικά αποτελέσματα προγραμμάτων Ταυτόχρονης Cyclone και C/pthreads προερχόμενα από το “The Computer Language Benchmarks Game.”

Κεφάλαιο 4

Εξαγωγή εφέ για αποφυγή αδιεξόδων

4.1 Επισκόπηση

Η εμφάνιση αδιεξόδων σε πολυνηματικά προγράμματα είναι ένα συχνά εμφανιζόμενο σφάλμα ταυτόχρονων προγραμμάτων που συγχρονίζονται με κλειδαριές. Σε αυτό το κεφάλαιο παρουσιάζουμε μια τεχνική που αποφεύγει πιθανά αδιέξοδα δυναμικά σε προγράμματα με αδόμητες λειτουργίες κλειδώματος, βασιζόμενη σε στατικά παρεχόμενες πληροφορίες χρήσης των κλειδαριών. Συγκεκριμένα, παρουσιάζουμε έναν λάμβδα λογισμό με λειτουργίες κλειδώματος, την λειτουργική σημασιολογία του και ένα σύστημα τύπων και εφέ, το οποίο εξάγει αυτόματα τις χρήσεις κλειδαριών κάθε νήματος. Το προτεινόμενο σύστημα χρόνου εκτέλεσης αποφεύγει τα αδιέξοδα χρησιμοποιώντας πληροφορίες σχετικά με την χρήση κλειδαριών. Παρουσιάζουμε συνοπτικά το θεώρημα ασφάλειας τύπων που εγγυάται την απουσία αδιεξόδων από έγκυρα προγράμματα.

4.2 Τυπική γλώσσα

Η σύνταξη της γλώσσας μας παρουσιάζεται στο Σχήμα 4.1, όπου τα x και ρ είναι μεταμεταβλητές όρων και κλειδαριών αντίστοιχα ενώ το $\imath$ είναι μεταβλητή σταθεράς κλειδαριάς. Η γλώσσα αποτελείται από σταθερές (`true`, `false` και $()$), συναρτήσεις (f) και εφαρμογές συναρτήσεων. Οι συναρτήσεις είναι μονομορφικές ($\lambda x.\, e$), πολυμορφικές ως προς τις κλειδαριές ($\Lambda\rho.\, f$), και αναδρομικές ($\texttt{fix}\ x.\, f$). Η εφαρμογή πολυμορφικών συναρτήσεων είναι ρητή ($e[r]$, όπου r μπορεί να είναι μεταμεταβλητή σταθεράς ή μεταβλητής κλειδαριάς). Η εφαρμογή μονομορφικών συναρτήσεων επισημειώνεται με το ξ, το οποίο είναι της μορφής seq και par για τοπικές και παράλληλες[1] εφαρμογές συναρτήσεων αντίστοιχα. Η εκφράσεις ελέγχου είναι της μορφής $\texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2$. Το κατασκεύασμα $\texttt{newlock}\ \rho, x\ \texttt{in}\ e$ δεσμεύει μια νέα ξεκλείδωτη επανεισερχόμενη κλειδαριά, και την αντιστοιχεί στις μεταβλητές ρ[2] και x αντίστοιχα για την εμβέλεια της έκφρασης e. Οι κλειδαριές μπορούν να περαστούν στις λειτουργίες $\texttt{lock}_\gamma\ e$ και $\texttt{unlock}\ e$ αντίστοιχα. Η σύνταξη τύπων είναι καθιερωμένη. Ο τύπος συναρτήσεων επισημειώνεται με ένα εφέ. Τα εφέ (γ) είναι ακολουθία από στοιχεία της μορφής r^+, r^- και $\gamma_1\ ?\ \gamma_2$ τα οποία είναι οι αναπαραστάσεις λειτουργιών κλειδώματος, ξεκλειδώματος και εκφράσεων ελέγχου αντίστοιχα.

4.3 Λειτουργική σημασιολογία

Η λειτουργική σημασιολογία (*μικρών βημάτων*) της γλώσσας μας παρουσιάζεται στα Σχήματα 4.2 και 4.3. Η σχέση αποτίμησης μετασχηματίζει δυαδικές πλειάδες αποτελούμενες από έναν αφηρημένο σωρό κλειδαριών S και μια λίστα νημάτων . Ο σωρός κλειδαριών διατηρεί κάποια μεταδεδομένα για κάθε κλειδαριά, όπως το αναγνωριστικό του νήματος που την κλείδωσε, ένα μετρητή κλειδωμάτων και δύο σύνολα κλειδαριών τα οποία είναι μόνο χρήσιμα για την απόδειξη ασφάλειας τύπων. Κάθε νήμα αποτελείται από ένα μοναδικό αναγνωριστικό n και την έκφραση e που εκτελεί. Η αποτίμηση των εκφράσεων γίνεται από αριστερά προς τα δεξιά. Ο πιο ενδιαφέρον κανόνας αποτίμησης είναι

[1] Δημιουργία νέων νημάτων.

[2] Η μεταβλητή τύπων ρ αναπαριστά μοναδικά την νέα κλειδαριά σε επίπεδο τύπων.

Expression	$e ::= x \mid v \mid (e\ e)^{\xi} \mid (e)\,[r] \mid \mathtt{pop}_{\gamma}\ e$	**Lock**	$r ::= \rho \mid \imath$
	$\mid\ \mathtt{newlock}\ \rho, x\ \mathtt{in}\ e$	**Calling mode**	$\xi ::= \mathsf{seq}(\gamma) \mid \mathsf{par}$
	$\mid\ \mathtt{lock}_{\gamma}\ e \mid \mathtt{unlock}\ e$	**Operation**	$\kappa ::= + \mid -$
	$\mid\ \mathtt{if}\ e\ \mathtt{then}\ e\ \mathtt{else}\ e$	**Effect**	$\gamma ::= \emptyset \mid r^{\kappa}, \gamma \mid \gamma\,?\,\gamma, \gamma$
Value	$v ::= () \mid \mathtt{true} \mid \mathtt{false} \mid f \mid \mathtt{lk}_{\imath}$		
Function	$f ::= \lambda x.\, e \mid \Lambda\rho.\, f \mid \mathtt{fix}\ x.\, f$		
Type	$\tau ::= \langle\rangle \mid \mathtt{Bool} \mid \mathtt{Lk}(r) \mid \tau \xrightarrow{\gamma} \tau \mid \forall\rho.\, \tau$		

Σχήμα. 4.1: Σύνταξη γλώσσας και τύπων.

Lock Store	$S ::= \emptyset \mid S, \imath \mapsto n; n; \epsilon; \epsilon$	**Context**	$E ::= \square \mid E[F]$
Threads	$T ::= \emptyset \mid T, n : e$	**Frame**	$F ::= (\square\ e)^{\xi} \mid (v\ \square)^{\xi} \mid (\square)\,[r] \mid \mathtt{pop}_{\gamma}\ \square$
Configuration	$C ::= S; T$		$\mid\ \mathtt{lock}_{\gamma_1}\ \square \mid \mathtt{unlock}\ \square$
Lockset	$\epsilon ::= \emptyset \mid \epsilon, \imath$		$\mid\ \mathtt{if}\ \square\ \mathtt{then}\ e\ \mathtt{else}\ e$

Σχήμα. 4.2: Λειτουργική σημασιολογία και περιβάλλον αποτίμησης.

ο *E-LK0*, ο οποίος υπολογίζει δυναμικά το "μελλοντικό" σύνολο κλειδαριών (ϵ) της κλειδαριάς $\imath$. Η συνάρτηση stack (ορίζεται στο Σχήμα 4.4) συνενώνει τα εφέ που βρίσκονται στην στοίβα κλήσεων[3] Ο υπολογισμός του "μελλοντικού" συνόλου κλειδαριών πραγματοποιείται από την συνάρτηση $\mathrm{run}(\gamma, \imath, k)$ (ορίζεται στο Σχήμα 4.4), η οποία δέχεται ως ορίσματα το συνενωμένο εφέ του νήματος γ, την κλειδαριά $\imath$ και τον αριθμό k των λειτουργιών ξεκλειδώματος στο $\imath$ που πρέπει να βρεθούν στο γ για να ολοκληρωθεί ο υπολογισμός. Η συνάρτηση available (ορίζεται στο Σχήμα 4.4) δέχεται ως ορίσματα τον σωρό κλειδαριών S και το αναγνωριστικό νήματος n και επιστρέφει το σύνολο των κλειδαριών που είναι διαθέσιμα για κλείδωμα από το νήμα n. Σε περίπτωση που κάποια κλειδαριά δεν είναι διαθέσιμη ο κανόνας δεν μπορεί να εφαρμοστεί και το νήμα θεωρείται μπλοκαρισμένο.

4.4 Στατική σημασιολογία

Παρουσιάζουμε ένα σύστημα τύπων και εφέ που εγγυάται την απουσία σφαλμάτων μνήμης και συνθηκών ανταγωνισμού. Η σύνταξη τύπων, εφέ και περιβαλλόντων τύπων παρουσιάζεται στο Σχήμα 4.1 (στη σελίδα 36). Η σχέση τύπων είναι της μορφής $R; M; \Delta; \Gamma \vdash e : \tau \,\&\, (\gamma;\gamma')$ και σημαίνει ότι η έκφραση e έχει τύπο τ και παράγει το εφέ γ έχοντας ως εφέ εισόδου το γ'. Τα εφέ στο σύστημα μας έχουν αντίστροφη ροή: το εφέ γ αναπαριστά μόνο τις παρενέργειες που ακολουθούν την εκτέλεση της έκφρασης e, ενώ το εφέ γ' αναπαριστά τις ίδιες παρενέργειες με το γ συν τις παρενέργειες της έκφρασης e. Οι κανόνες τύπων είναι διαθέσιμοι στο Σχήμα 4.5. Οι περισσότεροι κανόνες είναι απλοί με εξαίρεση τους κανόνες *T-FX*, *T-NG* και *T-PA*. Ο κανόνας *T-FX* συνοψίζει το εφέ του σώματος της συνάρτησης γ_b σε γ_a μέσω της συνάρτησης summary. Διαισθητικά, μπορεί να είναι αδύνατο να αναθέσουμε το ίδιο εφέ στην αναδρομική συνάρτηση και στο σώμα της αφού υπάρχει μια κυκλική σχέση μεταξύ τους. Η συνάρτηση summary υπολογίζει μια συντηρητική προσέγγιση του "αναδρομικού" εφέ. Οι κανόνες τύπων *T-NG* και *T-PA* επικυρώνουν εκφράσεις που δημιουργούν νέες κλειδαριές και εκφράσεις παράλληλης εφαρμογής συναρτήσεων αντίστοιχα. Αμφότεροι κανόνες απαιτούν ότι οι κλειδαριές είναι ξεκλείδωτες πριν και μετά την αποτίμηση των εκφράσεων που επικυρώνουν $(\rho; 0 \vdash_{ok} \gamma')$.

[3] Η στοίβα κλήσεων αναπαρίσταται από εκφράσεις pop, οι οποίες φέρουν επισημειώσεις εφέ και βρίσκονται στο περιβάλλον αποτίμησης .

$$\frac{\text{fresh } n'}{S; T, n : E[(v'\ \ v)^{\mathsf{par}}] \rightsquigarrow S; T, n : E[()], n' : \Box[(v'\ \ v)^{\mathsf{seq}(\emptyset)}]} \quad \textit{(E-SN)}$$

$$\frac{}{S; T, n : \Box[()] \rightsquigarrow S; T} \quad \textit{(E-T)}$$

$$\frac{}{S; T, n : E[((\lambda x.\, e_1)\ \ v)^{\mathsf{seq}(\gamma)}] \rightsquigarrow S; T, n : E[\mathtt{pop}_\gamma\ e_1[v/x]]} \quad \textit{(E-A)}$$

$$\frac{}{S; T, n : E[\mathtt{pop}_\gamma\ v] \rightsquigarrow S; T, n : E[v]} \quad \textit{(E-PP)}$$

$$\frac{}{S; T, n : E[(\Lambda \rho.\, f)[\imath]] \rightsquigarrow S; T, n : E[f[\imath/\rho]]} \quad \textit{(E-RP)}$$

$$\frac{v' = \mathtt{fix}\ x.\, f}{S; T, n : E[(v'\ \ v)^{\mathsf{seq}(\gamma)}] \rightsquigarrow S; T, n : E[(f[v'/x]\ \ v)^{\mathsf{seq}(\gamma)}]} \quad \textit{(E-FX)}$$

$$\frac{}{S; T, n : E[\mathtt{if\ true\ then}\ e_1\ \mathtt{else}\ e_2] \rightsquigarrow S; T, n : E[e_1]} \quad \textit{(E-IT)}$$

$$\frac{}{S; T, n : E[\mathtt{if\ false\ then}\ e_1\ \mathtt{else}\ e_2] \rightsquigarrow S; T, n : E[e_2]} \quad \textit{(E-IF)}$$

$$\frac{\text{fresh } \imath \qquad S' = S, \imath \mapsto n; 0; \emptyset; \emptyset}{S; T, n : E[\mathsf{newlock}\ \rho, x\ \mathtt{in}\ e_1] \rightsquigarrow S'; T, n : E[e_1[\imath/\rho][\mathtt{lk}_\imath/x]]} \quad \textit{(E-NG)}$$

$$\frac{\begin{array}{cc} S(\imath) = n_1; 0; \epsilon_1; \epsilon_2 & S' = S[\imath \mapsto n; 1; \mathsf{dom}(S); \epsilon] \\ \epsilon = \mathsf{run}(\mathsf{stack}(E[\mathtt{pop}_{\gamma_1}\ \Box]), \imath, 1) & \epsilon \cup \{\imath\} \subseteq \mathsf{available}(S, n) \end{array}}{S; T, n : E[\mathtt{lock}_{\gamma_1}\ \mathtt{lk}_\imath] \rightsquigarrow S'; T, n : E[()]} \quad \textit{(E-LK0)}$$

$$\frac{S(\imath) = n; n_2; \epsilon_1; \epsilon_2 \qquad n_2 > 0 \qquad S' = S[\imath \mapsto n; n_2 + 1; \epsilon_1; \epsilon_2]}{S; T, n : E[\mathtt{lock}_{\gamma_1}\ \mathtt{lk}_\imath] \rightsquigarrow S'; T, n : E[()]} \quad \textit{(E-LK1)}$$

$$\frac{S(\imath) = n; n_2; \epsilon_1; \epsilon_2 \qquad n_2 > 0 \qquad S' = S[\imath \mapsto n; n_2 - 1; \epsilon_1; \epsilon_2]}{S; T, n : E[\mathtt{unlock}\ \mathtt{lk}_\imath] \rightsquigarrow S'; T, n : E[()]} \quad \textit{(E-UL)}$$

Σχήμα. 4.3: Λειτουργική σημασιολογία.

4.5 Ασφάλεια τύπων και απουσία αδιεξόδων

Στην ενότητα αυτή παρουσιάζονται τα βασικά θεωρήματα που αποδεικνύουν την ασφάλεια τύπων για την γλώσσα μας, μαζί με κάποιες συνοπτικές αποδείξεις και ορισμούς. Το θεώρημα ασφάλειας τύπων συνεπάγεται απουσία αδιεξόδων από έγκυρα προγράμματα. Η απόδειξη βασίζεται στα λήμματα *διατήρησης ελευθερία αδιέξοδων* και *προόδου*. Άτυπα, ένα πρόγραμμα γραμμένο στη γλώσσα μας είναι ασφαλές όταν κάθε νήμα μπορεί να εκτελέσει ένα βήμα ή έχει μπλοκαριστεί περιμένοντας κάποια μη διαθέσιμη κλειδαριά. Επιπλέον, πρέπει να μην υπάρχει κανένα νήμα που να έχει φτάσει σε αδιέξοδο.

Ορισμός 4.1 (Συνέπεια εφέ νημάτων) Οι ακόλουθοι κανόνες ορίζουν νήματα με *συνεπή* νήματα.

$$\frac{\begin{array}{cc} \imath; n_1 \vdash_{ok} \gamma & \epsilon_3 = \mathsf{run}(\gamma, \imath, n_1) \\ n; \gamma \vdash S & \epsilon_1 \cap \epsilon_3 \subseteq \epsilon_2 \end{array}}{n; \gamma \vdash S, \imath \mapsto n; n_1; \epsilon_1; \epsilon_2} \qquad \frac{\imath; 0 \vdash_{ok} \gamma \qquad n \neq n_1 \qquad n; \gamma \vdash S}{n; \gamma \vdash S, \imath \mapsto n_1; n_2; \epsilon_1; \epsilon_2} \qquad \frac{}{n; \gamma \vdash \emptyset}$$

$$
\mathsf{run}(\gamma, \imath, n) = \begin{cases} \emptyset & \text{if } n = 0 \\ \mathsf{run}(\gamma', \imath, n+1) & \text{if } \gamma = \imath^+, \gamma' \text{ and } n > 0 \\ \mathsf{run}(\gamma', \imath, n-1) & \text{if } \gamma = \imath^-, \gamma' \text{ and } n > 0 \\ \mathsf{run}(\gamma', \imath, n) \cup \{\mathtt{j}\} & \text{if } \gamma = \mathtt{j}^+, \gamma' \text{ and } n > 0 \\ \mathsf{run}(\gamma', \imath, n) & \text{if } \gamma = \mathtt{j}^-, \gamma' \text{ and } n > 0 \\ \mathsf{run}((\gamma_1 :: \gamma'), \imath, n) \cup \mathsf{run}((\gamma_2 :: \gamma'), \imath, n) & \text{if } \gamma = \gamma_1 \,?\, \gamma_2, \gamma' \text{ and } n > 0 \end{cases}
$$

$$
\mathsf{stack}(E) = \begin{cases} \emptyset & \text{if } E = \square \\ \mathsf{stack}(E') & \text{if } E = E'[F] \text{ and } F \neq \mathsf{pop}_{\gamma'}\ \square \\ \gamma' :: \mathsf{stack}(E') & \text{if } E = E'[\mathsf{pop}_{\gamma'}\ \square] \end{cases}
$$

$$
\mathsf{available}(S, n) = \begin{cases} \emptyset & \text{if } S = \emptyset \\ \mathsf{available}(S', n) \cup \{\imath\} & \text{if } S = S', \imath \mapsto n_1; n_2; \epsilon_1; \epsilon_2 \text{ and } n_1 = n \text{ or } n_2 = 0 \\ \mathsf{available}(S', n) & \text{if } S = S', \imath \mapsto n_1, n_2; \epsilon_1; \epsilon_2 \text{ and } n_1 \neq n \text{ and } n_2 > 0 \end{cases}
$$

$$
\mathsf{dom}(S) = \{\imath \mid \imath \mapsto n_1; n_2; \epsilon_a; \epsilon_b \in S\}
$$

Σχήμα. 4.4: Βοηθητικές συναρτήσεις λειτουργικής σημασιολογίας.

Ορισμός 4.2 (Εγκυρότητα νημάτων) Οι ακόλουθοι κανόνες ορίζουν πότε μια συλλογή νημάτων είναι *έγκυρη*.

$$
\frac{}{S; M \vdash \emptyset} \qquad \frac{M; \emptyset; \emptyset \vdash e : \langle\rangle \,\&\, (\emptyset;\gamma) \quad S; M \vdash T \quad n \notin \mathsf{dom}(T) \quad n; \gamma \vdash S}{S; M \vdash T, n : e}
$$

Ορισμός 4.3 (Εγκυρότητα προγράμματος) Ένα πρόγραμμα $S; T$ είναι *έγκυρο* ως προς το M ($M \vdash S; T$), όταν $S; M \vdash T$ και $M = \mathrm{dom}(S)$.

$$
\frac{S; M \vdash T \quad M = \mathsf{dom}(S)}{M \vdash S; T}
$$

Ορισμός 4.4 (Κατάσταση αδιεξόδου) Ένα πρόγραμμα φτάνει σε *αδιέξοδο*, όταν υπάρχει ένα σύνολο νημάτων $n_0, \ldots, n_k$, για $k > 0$, και ένα σύνολο κλειδαριών $\ell_0, \ldots, \ell_k$, τέτοια ώστε κάθε νήμα $n_\imath$ έχει κλειδώσει το $\ell_{(\imath+1) \,\mathtt{mod}\, (k+1)}$ και περιμένει το $\ell_\imath$.

$$
\mathsf{locks}(S, \imath, n) = \begin{cases} n_2 & \text{if } S(\imath) = (n; n_2; \epsilon_1; \epsilon_2) \\ 0 & \text{if } S(\imath) = (n_1; n_2; \epsilon_1; \epsilon_2) \wedge n_1 \neq n \end{cases}
$$

$$
\begin{aligned} \mathsf{deadlocked}(T) \equiv\ & T \supseteq T_1, n_0 : E[\mathtt{lock}_{\gamma_0}\ \mathtt{lk}_{\imath_0}], \ldots, n_k : E_k[\mathtt{lock}_{\gamma_k}\ \mathtt{lk}_{\imath_k}] \wedge k > 0 \wedge \\ & \forall m_1 \in [0, k].\, m_2 = (m_1 + 1) \,\mathrm{mod}\, (k+1) \wedge \mathsf{locks}(S, \imath_{m_2}, m_1) > 0 \end{aligned}
$$

Ορισμός 4.5 (Μη κολλημένη κατάσταση) Ένα πρόγραμμα $S; T$ *δεν είναι κολλημένο* όταν κάθε νήμα στο T μπορεί να αποτιμηθεί από κάποιο κανόνα του Σχήματος 4.3 ή περιμένει για κάποια κλειδαριά που δεν είναι διαθέσιμη για κλείδωμα.

Παρουσιάζουμε τα βασικά αποτελέσματα αυτού το κεφαλαίου. Τα λήμματα *προόδου*, *απουσίας αδιεξόδων* και *διατήρησης* τυποποιούνται στο επίπεδο του προγράμματος.

Λήμμα 4.1 (Απουσία αδιεξόδων) Έστω ότι το αρχικό πρόγραμμα κάνει n βήματα, και κάθε πρόγραμμα από το αρχικό έως το $n-$ οστό είναι έγκυρο, τότε κανένα από τα νήματα του $n-$ οστό προγράμματος δεν είναι σε αδιέξοδο.

$$\frac{M;\Delta \vdash \Gamma \qquad M;\Delta \vdash \gamma}{M;\Delta;\Gamma \vdash () : \langle\rangle \,\&\, (\gamma;\gamma)} \textit{(T-U)}$$

$$\frac{\begin{array}{c} x : \tau \in \Gamma \\ M;\Delta \vdash \Gamma \qquad M;\Delta \vdash \gamma \end{array}}{M;\Delta;\Gamma \vdash x : \tau \,\&\, (\gamma;\gamma)} \textit{(T-V)}$$

$$\frac{M;\Delta \vdash \Gamma \qquad M;\Delta \vdash \gamma}{M;\Delta;\Gamma \vdash \texttt{true} : \texttt{Bool} \,\&\, (\gamma;\gamma)} \textit{(T-T)}$$

$$\frac{M;\Delta \vdash \Gamma \qquad M;\Delta \vdash \gamma}{M;\Delta;\Gamma \vdash \texttt{false} : \texttt{Bool} \,\&\, (\gamma;\gamma)} \textit{(T-F)}$$

$$\frac{\begin{array}{c} M;\Delta \vdash \Gamma \qquad M;\Delta \vdash \gamma \qquad \tau \equiv \tau_1 \xrightarrow{\gamma_b} \tau_2 \\ M;\Delta \vdash \tau \qquad M;\Delta;\Gamma, x : \tau_1 \vdash e_1 : \tau_2 \,\&\, (\emptyset;\gamma_b) \end{array}}{M;\Delta;\Gamma \vdash \lambda x.\, e_1 : \tau \,\&\, (\gamma;\gamma)} \textit{(T-FN)}$$

$$\frac{M;\Delta,\rho;\Gamma \vdash f : \tau \,\&\, (\gamma;\gamma)}{M;\Delta;\Gamma \vdash \Lambda\rho.\, f : \forall\rho.\, \tau \,\&\, (\gamma;\gamma)} \textit{(T-RF)}$$

$$\frac{r \in M \cup \Delta \qquad M;\Delta;\Gamma \vdash e_1 : \forall\rho.\, \tau \,\&\, (\gamma;\gamma')}{M;\Delta;\Gamma \vdash (e_1)\,[r] : \tau[r/\rho] \,\&\, (\gamma;\gamma')} \textit{(T-RP)}$$

$$\frac{M;\Delta \vdash \gamma \qquad M;\Delta;\Gamma \vdash e : \tau \,\&\, (\emptyset;\gamma')}{M;\Delta;\Gamma \vdash \texttt{pop}_\gamma\ e : \tau \,\&\, (\gamma;\gamma' :: \gamma)} \textit{(T-PP)}$$

$$\frac{M;\Delta \vdash \Gamma \qquad M;\Delta \vdash \gamma \qquad \imath \in M}{M;\Delta;\Gamma \vdash \texttt{lk}_\imath : \texttt{Lk}(\imath) \,\&\, (\gamma;\gamma)} \textit{(T-L)}$$

$$\frac{M;\Delta;\Gamma \vdash e : \texttt{Lk}(r) \,\&\, (r^+, \gamma;\gamma')}{M;\Delta;\Gamma \vdash \texttt{lock}_\gamma\, e : \langle\rangle \,\&\, (\gamma;\gamma')} \textit{(T-LK)}$$

$$\frac{M;\Delta;\Gamma \vdash e : \texttt{Lk}(r) \,\&\, (r^-, \gamma;\gamma')}{M;\Delta;\Gamma \vdash \texttt{unlock}\, e : \langle\rangle \,\&\, (\gamma;\gamma')} \textit{(T-UL)}$$

$$\frac{\begin{array}{c} M;\Delta;\Gamma \vdash e_1 : \tau_1 \xrightarrow{\gamma_a} \tau_2 \,\&\, (\gamma_1;\gamma') \\ M;\Delta;\Gamma \vdash e_2 : \tau_1 \,\&\, (\gamma_a :: \gamma;\gamma_1) \end{array}}{M;\Delta;\Gamma \vdash (e_1\ \ e_2)^{\mathsf{seq}(\gamma)} : \tau_2 \,\&\, (\gamma;\gamma')} \textit{(T-SA)}$$

$$\frac{\begin{array}{c} \forall r \in \mathsf{dom}(\gamma_a).\ r;0 \vdash_{ok}\ \gamma_a \\ M;\Delta;\Gamma \vdash e_1 : \tau_1 \xrightarrow{\gamma_a} \langle\rangle \,\&\, (\gamma_1;\gamma') \qquad M;\Delta;\Gamma \vdash e_2 : \tau_1 \,\&\, (\gamma;\gamma_1) \end{array}}{M;\Delta;\Gamma \vdash (e_1\ \ e_2)^{\mathsf{par}} : \langle\rangle \,\&\, (\gamma;\gamma')} \textit{(T-PA)}$$

$$\frac{\begin{array}{c} M;\Delta \vdash \tau \qquad \rho \notin \mathsf{dom}(\gamma) \qquad \rho;0 \vdash_{ok}\ \gamma' \\ M;\Delta,\rho;\Gamma, x : \texttt{Lk}(\rho) \vdash e_1 : \tau \,\&\, (\gamma;\gamma') \end{array}}{M;\Delta;\Gamma \vdash \texttt{newlock}\ \rho, x\ \texttt{in}\ e_1 : \tau \,\&\, (\gamma;\gamma' \setminus \rho)} \textit{(T-NG)}$$

$$\frac{\begin{array}{c} R;M;\Delta;\Gamma, x : \tau_a \vdash f : \tau_b \,\&\, (\gamma;\gamma) \\ \tau_a \equiv \tau_1 \xrightarrow{\gamma_a} \tau_2 \qquad \tau_b \equiv \tau_1 \xrightarrow{\gamma_b} \tau_2 \qquad \gamma_a = \mathsf{summary}(\gamma_b) \end{array}}{R;M;\Delta;\Gamma \vdash \texttt{fix}\ x.\, f : \tau_a \,\&\, (\gamma;\gamma)} \textit{(T-FX)}$$

$$\frac{\begin{array}{c} M;\Delta;\Gamma \vdash e_1 : \texttt{Bool} \,\&\, (\gamma_1 \,?\, \gamma_2, \gamma;\gamma') \\ M;\Delta;\Gamma \vdash e_2 : \tau \,\&\, (\gamma;\gamma_1 :: \gamma) \qquad M;\Delta;\Gamma \vdash e_3 : \tau \,\&\, (\gamma;\gamma_2 :: \gamma) \end{array}}{M;\Delta;\Gamma \vdash \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 : \tau \,\&\, (\gamma;\gamma')} \textit{(T-IF)}$$

Σχήμα. 4.5: Κανόνες τύπων.

$$\frac{}{M;\Delta \vdash \emptyset} \qquad \frac{r \in M \cup \Delta \qquad M;\Delta \vdash \gamma_1}{M;\Delta \vdash \gamma_1, r^{\kappa}} \qquad \frac{M;\Delta \vdash \gamma_1 \qquad M;\Delta \vdash \gamma_2 \qquad M;\Delta \vdash \gamma_3}{M;\Delta \vdash \gamma_1, \gamma_2 \,?\, \gamma_3}$$

$$\frac{}{M;\Delta \vdash \texttt{Bool}} \qquad \frac{M;\Delta, \rho \vdash \tau}{M;\Delta \vdash \forall \rho.\, \tau} \qquad \frac{r \in M \cup \Delta}{M;\Delta \vdash \texttt{Lk}(r)}$$

$$\frac{M;\Delta \vdash \tau_1 \qquad M;\Delta \vdash \gamma_1 \qquad M;\Delta \vdash \tau_2}{M;\Delta \vdash \tau_1 \xrightarrow{\gamma_1} \tau_2} \qquad \frac{}{M;\Delta \vdash \langle\rangle}$$

$$\frac{}{M;\Delta \vdash \emptyset} \qquad \frac{M;\Delta \vdash \tau_1 \qquad x \notin \mathsf{dom}(\Gamma_1) \qquad M;\Delta \vdash \Gamma_1}{M;\Delta \vdash \Gamma_1, x : \tau_1}$$

Σχήμα. 4.6: Κανόνες καλοσχηματισμένων τύπων.

Λήμμα 4.2 (Πρόοδος) dlslemma:thread-progress Εάν $S;T$ είμαι ένα κλειστό και έγκυρο πρόγραμμα τέτοιο ώστε $M \vdash S;T$, τότε το $S;T$ δεν έχει κολλήσει.

Λήμμα 4.3 (Διατήρηση) Έστω ότι το πρόγραμμα $S;T$ είναι τέτοιο ώστε $M \vdash S;T$. Εάν η λειτουργική σημασιολογία εκτελέσει ένα βήμα $S;T \rightsquigarrow S';T'$, τότε ισχύει $M' \vdash S';T'$ και $M' \supseteq M$ για κάποιο $'$.

Λήμμα 4.4 (Διατήρηση με πολλαπλά βήματα) Έστω το πρόγραμμα $S_0;T_0$ είναι κλειστό και έγκυρο για κάποιο M_0 και αποτιμάται σε n βήματα σε ένα άλλο πρόγραμμα $S_n;T_n$. Τότε, για κάθε $\imath \in [0,n]$ ισχύει ότι $M_\imath \vdash S_\imath;T_\imath$.

Απόδειξη. Επαγωγική απόδειξη στον αριθμό των βημάτων n, χρησιμοποιώντας το Λήμμα 4.3.

Θεώρημα 4.1 (Ασφάλεια τύπων) Εάν το αρχικό πρόγραμμα $S_0;T_0$ είναι κλειστό και έγκυρο ($\emptyset \vdash S_0;T_0$) και η λειτουργική σημασιολογία του κάνει οσαδήποτε βήματα ($S_0;T_0 \rightsquigarrow^n S_n;T_n$), τότε το τελικό πρόγραμμα $S_n;T_n$ δεν έχει κολλήσει και κανένα από τα νήματα στο T_n δεν βρίσκεται σε αδιέξοδο.

Απόδειξη. Η εφαρμογή του Λήμματος 4.4 στην παραγωγή τύπων του $S_0;T_0$ συνεπάγεται ότι $M_n \vdash S_n;T_n$. Συνεπώς το Λήμμα 4.1 δίνει ότι T_n δεν βρίσκεται σε κατάσταση αδιεξόδου και το Λήμμα 4.2 δίνει ότι το πρόγραμμα $S_n;T_n$ δεν έχει κολλήσει.

4.6 Καταληκτικές παρατηρήσεις

Σε αυτό το κεφάλαιο παρουσιάσαμε μια τεχνική που χρησιμοποιεί στατικά υπολογισμένες πληροφορίες κατά τον χρόνο εκτέλεσης για αποφευχθούν καταστάσεις αδιεξόδων. Προτείναμε μια γλώσσα που δεν απαιτεί ρητές επισημειώσεις εφέ από τον προγραμματιστή και επιτρέπει την αντιγραφή επανεισερχόμενων κλειδαριών χωρίς περιορισμούς. Η τυπική σημασιολογία που προτείναμε εγγυάται την απουσία αδιεξόδων από έγκυρα προγράμματα. Το στατικό σύστημα τύπων μας εξάγει αυτόματα τα εφέ των κλειδωμάτων κάθε νήματος και ως εκ τούτου είναι εφαρμόσιμο σε προγράμματα γλωσσών με αδόμητες λειτουργίες κλειδώματος και αδύναμα συστήματα τύπων, όπως η C/pthreads.

Sub-effect $\gamma \lhd \gamma' \equiv \exists \gamma''.\gamma' = \gamma'' :: \gamma$

$$\frac{M;\Delta \vdash \Gamma \qquad M;\Delta \vdash \gamma_1 \qquad M;\Delta \vdash \gamma_2 \qquad M;\Delta \vdash \tau}{M;\Delta;\Gamma \vdash \square : \tau \xrightarrow{\gamma_1;\gamma_2} \tau \,\&\, (\gamma_1;\gamma_2)} \quad \textit{(E0)}$$

$$\frac{\begin{array}{c} M;\Delta;\Gamma \vdash E : \tau_2 \xrightarrow{\gamma_5;\gamma_6} \tau_3 \,\&\, (\gamma_1;\gamma_2) \\ M;\Delta;\Gamma \vdash F : \tau_1 \xrightarrow{\gamma_3;\gamma_4} \tau_2 \,\&\, (\gamma_5;\gamma_6) \end{array}}{M;\Delta;\Gamma \vdash E[F] : \tau_1 \xrightarrow{\gamma_3;\gamma_4} \tau_3 \,\&\, (\gamma_1;\gamma_2)} \quad \textit{(E1)}$$

$$\frac{\begin{array}{c} M;\Delta \vdash \gamma_2 \qquad \gamma_1 \lhd \gamma_2 \qquad M;\Delta \vdash \tau_2 \\ M;\Delta;\Gamma \vdash e_2 : \tau_1 \,\&\, (\gamma_a :: \gamma;\gamma_1) \end{array}}{M;\Delta;\Gamma \vdash (\square\ e_2)^{\mathsf{seq}(\gamma)} : (\tau_1 \xrightarrow{\gamma_a} \tau_2) \xrightarrow{\gamma_1;\gamma_2} \tau_2 \,\&\, (\gamma;\gamma_2)} \quad \textit{(F1)}$$

$$\frac{\begin{array}{c} \gamma_2 \lhd \gamma_3 \qquad M;\Delta \vdash \tau_1 \xrightarrow{\gamma_a} \langle\rangle \\ M;\Delta;\Gamma \vdash e_2 : \tau_1 \,\&\, (\gamma_1;\gamma_2) \end{array}}{M;\Delta;\Gamma \vdash (\square\ e_2)^{\mathsf{par}} : (\tau_1 \xrightarrow{\gamma_a} \langle\rangle) \xrightarrow{\gamma_2;\gamma_3} \langle\rangle \,\&\, (\gamma_1;\gamma_3)} \quad \textit{(F2)}$$

$$\frac{\begin{array}{c} \gamma_2 = \gamma_1 :: \gamma_a \qquad M;\Delta \vdash \gamma_3 \qquad \gamma_2 \lhd \gamma_3 \\ M;\Delta;\Gamma \vdash v_1 : \tau_1 \xrightarrow{\gamma_a} \tau_2 \,\&\, (\gamma_3;\gamma_3) \end{array}}{M;\Delta;\Gamma \vdash (v_1\ \square)^{\mathsf{seq}(\gamma_1)} : \tau_1 \xrightarrow{\gamma_2;\gamma_3} \tau_2 \,\&\, (\gamma_1;\gamma_3)} \quad \textit{(F3)}$$

$$\frac{\begin{array}{c} M;\Delta \vdash \gamma_2 \qquad \gamma_1 \lhd \gamma_2 \\ M;\Delta;\Gamma \vdash v_1 : \tau_1 \xrightarrow{\gamma_a} \langle\rangle \,\&\, (\gamma_2;\gamma_2) \end{array}}{M;\Delta;\Gamma \vdash (v_1\ \square)^{\mathsf{par}} : \tau_1 \xrightarrow{\gamma_1;\gamma_2} \langle\rangle \,\&\, (\gamma_1;\gamma_2)} \quad \textit{(F4)}$$

$$\frac{M;\Delta \vdash \tau \qquad M;\Delta \vdash \gamma' \qquad \gamma' = \gamma_2 :: \gamma}{M;\Delta;\Gamma \vdash \mathtt{pop}_\gamma\ \square : \tau \xrightarrow{\emptyset;\gamma_2} \tau \,\&\, (\gamma;\gamma')} \quad \textit{(F5)}$$

$$\frac{\begin{array}{c} M;\Delta \vdash \Gamma \qquad M;\Delta \vdash \gamma' \\ M;\Delta \vdash \mathtt{Lk}(r) \qquad \gamma_1 = r^-, \gamma \qquad \gamma_1 \lhd \gamma' \end{array}}{M;\Delta;\Gamma \vdash \mathtt{unlock}\,\square : \mathtt{Lk}(r) \xrightarrow{\gamma_1;\gamma'} \langle\rangle \,\&\, (\gamma;\gamma')} \quad \textit{(F6)}$$

$$\frac{\begin{array}{c} M;\Delta \vdash \Gamma \qquad M;\Delta \vdash \gamma' \\ M;\Delta \vdash \mathtt{Lk}(r) \qquad \gamma_1 = r^+, \gamma \qquad \gamma_1 \lhd \gamma' \end{array}}{M;\Delta;\Gamma \vdash \mathtt{lock}_\gamma\ \square : \mathtt{Lk}(r) \xrightarrow{\gamma_1;\gamma'} \langle\rangle \,\&\, (\gamma;\gamma')} \quad \textit{(F7)}$$

$$\frac{\begin{array}{c} \gamma_3 = \gamma_1 \,?\, \gamma_2, \gamma \qquad M;\Delta \vdash \gamma' \qquad \gamma_3 \lhd \gamma' \\ M;\Delta;\Gamma \vdash e_2 : \tau \,\&\, (\gamma;\gamma_1 :: \gamma) \qquad M;\Delta;\Gamma \vdash e_3 : \tau \,\&\, (\gamma;\gamma_2 :: \gamma) \end{array}}{M;\Delta;\Gamma \vdash \mathtt{if}\ \square\ \mathtt{then}\ e_2\ \mathtt{else}\ e_3 : \mathtt{Bool} \xrightarrow{\gamma_3;\gamma'} \tau \,\&\, (\gamma;\gamma')} \quad \textit{(F8)}$$

$$\frac{M;\Delta \vdash \gamma' \qquad M;\Delta \vdash \forall \rho.\,\tau \qquad r \in M \cup \Delta \qquad \gamma \lhd \gamma'}{M;\Delta;\Gamma \vdash (\square)\,[r] : \forall \rho.\,\tau \xrightarrow{\gamma;\gamma'} \tau[r/\rho] \,\&\, (\gamma;\gamma')} \quad \textit{(F9)}$$

Σχήμα. 4.7: Κανόνες τύπων για το περιβάλλον αποτίμησης.

Κεφάλαιο 5

Εργαλείο αποφυγής αδιεξόδων

5.1 Επισκόπηση

Τα αδιέξοδα μπορεί να έχουν καταστροφικές συνέπειες στον κώδικα συστημάτων. Στο προηγούμενο κεφάλαιο παρουσιάσαμε σύστημα τύπων και εφέ, το οποίο αποδεδειγμένα αποφεύγει αδιέξοδα. Σε αυτό το κεφάλαιο παρουσιάζουμε ένα εργαλείο που χρησιμοποιεί μια συνεπή στατική ανάλυση βασισμένη στο σύστημα τύπων του προηγούμενου κεφαλαίου, η οποία ενσωματώνει τα στατικά υπολογισμένα εφέ στον αρχικό κώδικα C/pthreads και στη συνέχεια συνδέει τον μετασχηματισμένο κώδικα με ένα σύστημα χρόνου εκτέλεσης, το οποίο χρησιμοποιεί τα ενσωματωμένα εφέ για να αποφύγει τα αδιέξοδα δυναμικά. Σε αντίθεση με τα περισσότερα εργαλεία στατικής ανάλυσης που εγγυώνται απουσία αδιεξόδων από έγκυρα προγράμματα, το εργαλείο μας δεν απαιτεί την ύπαρξη μερικής διάταξης απόκτησης των κλειδαριών και υποστηρίζει τη χρήση αδόμητων λειτουργιών κλειδώματος, καθιστώντας το κατάλληλο για εφαρμογές συστημάτων. Αναφέρουμε επίσης κάποια πολύ ελπιδοφόρα συγκριτικά αποτελέσματα που δείχνουν ότι όλα τα πιθανά αδιέξοδα μπορεί να αποφευχθούν αυτόματα μόνο με μια μικρή επιβάρυνση στον χρόνο εκτέλεσης. Αυτό γίνεται χωρίς να απαιτείται η τροποποίηση του αρχικού πηγαίου κώδικα, αλλάζοντας τη σειρά απόκτησης των κλειδαριών ή προσθέτοντας ρητές επισημειώσεις.

5.2 Ανάλυση αποφυγής αδιεξόδων

Η ανάλυσή μας γίνεται σε δύο φάσεις. Η πρώτη φάση λαμβάνει χώρα κατά το χρόνο μεταγλώττισης και εκτελεί μια ευαίσθητη ανάλυση δεικτών, ως προς τα πεδία και τα περικλείοντα πλαίσια, η οποία ακολουθείται από ανάλυση εφέ συνεχειών (όπως ορίζεται στο Κεφάλαιο 4) και την ενσωμάτωση των υπολογισμένων εφέ στον πηγαίο κώδικα. Τέλος, το τελικό πρόγραμμα συνδέεται με σύστημα χρόνου εκτέλεσης. Η δεύτερη φάση είναι καθαρά δυναμική και λαμβάνει χώρα όταν το αρχικό πρόγραμμα ζητά μια κλειδαριά. Το "μελλοντικό" σύνολο κλειδαριών (όπως ορίζεται στο Κεφάλαιο 4) της αιτούμενης κλειδαριάς υπολογίζεται αξιοποιώντας τα ενσωματωμένα εφέ (όπως ορίζεται στο Κεφάλαιο 4). Το κλείδωμα μια κλειδαριάς είναι επιτυχές μόνο όταν η κλειδαριά και το "μελλοντικό" σύνολο κλειδαριών της είναι διαθέσιμα την ίδια στιγμή. Σε αυτό το κεφάλαιο, επεκτείνουμε την ανάλυση αποφυγής αδιεξόδων που παρουσιάστηκε στο προηγούμενο κεφάλαιο έτσι ώστε να χειρίζεται κλειδαριές που δεσμεύονται σε δομές δεδομένων στο σωρό ή τη στοίβα και βελτιστοποιεί το μέγεθος των ενσωματωμένων εφέ, ελαττώνοντας έτσι την επιβάρυνση των λειτουργιών κλειδώματος κατά τον χρόνο εκτέλεσης.

5.2.1 Στατική ανάλυση

Η ανάλυση μας δέχεται ως είσοδο ένα αφηρημένο συντακτικό δέντρο και κατασκευάζει ένα γράφο κλήσεων, τον οποίο επισκέπτεται από κάτω προς τα πάνω. Υπάρχουν τέσσερα βασικά στάδια στη ανάλυση μας που πραγματοποιούνται για κάθε κόμβο-συνάρτηση του γράφου κλήσεων.

Ανάλυση δεικτών. Κάνουμε χρήση μιας έτοιμης ανάλυσης δεικτών, που είναι ευαίσθητη ως προς τα πεδία των εγγραφών, και βασίζεται σε τεχνικές συμβολικής εκτέλεσης [Voun07], ώστε να μο-

ντελοποιήσει τον σωρό και τη στοίβα σε κάθε σημείο του προγράμματος. Στο τέλος της ανάλυσης αποκτούμε μια απεικόνιση μεταξύ εκφράσεων και συνόλων αφηρημένων θέσεων. Κάθε αφηρημένη θέση r μπορεί να είναι τυπική παράμετρος, καθολική μεταβλητή ή θέση μνήμης δεσμευμένη στον σωρό.

Ανάλυση εφέ. Το εφέ κάθε συνάρτησης υπολογίζεται με την εκτέλεση ανάλυσης ροής δεδομένων στον γράφο ροής ελέγχου της εκάστοτε συνάρτησης.

Κάθε κόμβος στον γράφο ροής ελέγχου συσχετίζεται με τα εφέ εισόδου, το υπάρχον εφέ (το εφέ που παράγεται από τον κόμβο) και το εφέ εξόδου.

Το εφέ εισόδου ενός κόμβου είναι η *ένωση* των εφέ που βρίσκονται στις εισερχόμενες ακμές του. Για παράδειγμα, εάν σε ένα κόμβο βρίσκονται τα εφέ $\gamma_1 \ldots \gamma_k$, τότε το εφέ εισόδου του είναι η συνένωση τους $\gamma_1 \ ? \ \ldots \ ? \ \gamma_k$ και υποδηλώνει επιλογή μεταξύ πιθανών εναλλακτικών εφέ $\gamma_1 \ldots \gamma_k$. Η αντιμετώπιση των κυκλικών ακμών (backedges) ενός που κόμβου περιγράφεται στην επόμενη παράγραφο.

Το υπάρχον εφέ ενός κόμβου μπορεί να είναι r^+ ή r^-, όταν ο κόμβος είναι λειτουργία κλειδώματος ή ξεκλειδώματος αντίστοιχα. Όταν ο κόμβος είναι λειτουργία δέσμευσης σε κάποια αφηρημένη θέση μνήμης ρ, τότε αναθέτουμε ως υπάρχον εφέ το `malloc` ρ. Όταν ο κόμβος είναι κλήση συνάρτησης r με αφηρημένα ορίσματα $r_1 \ldots r_n$ και αποτέλεσμα r', τότε αναθέτουμε ως υπάρχον εφέ το `call` $r\ (\ r_1, \ldots, r_n\ ) : r'$. Το εφέ εξόδου υπολογίζεται με την συνένωση του υπάρχοντος εφέ με το εφέ εισόδου και μεταφέρεται ως εφέ εισόδου στους διαδόχους ενός κόμβου. Το εφέ μιας συνάρτησης υπολογίζεται με την συνένωση εφέ εξόδου κόμβων που δεν έχουν διαδόχους.

Βρόχοι. Τα εφέ που προέρχονται από κυκλικές ακμές (εξερχόμενες ακμές του ίδιου κόμβου) πρέπει να είναι ισοδύναμες (ως προς τους μετρητές κλειδαριών) με το εφέ εισόδου του ίδιου κόμβου. Ένας βρόχος μπορεί να έχει οσεσδήποτε λειτουργίες (ξε-)κλειδώματος αρκεί στην έξοδο του οι μετρητές κλειδαριών να είναι ίσοι με τους μετρητές κλειδαριών στην είσοδο του.

Βελτιστοποιήσεις εφέ. Ο σκοπός των βελτιστοποιήσεων εφέ είναι να ελαχιστοποιήσουμε το μέγεθος τους, ώστε να ελαχιστοποιήσουμε την επιβάρυνση των λειτουργιών κλειδώματος κατά τον χρόνο εκτέλεσης. Οι βελτιστοποιήσεις για σύμπτυξη εφέ είναι οι εξής: ο υπολογισμός και εξαγωγή κοινών προθεμάτων από συνενώσεις εφέ, η επιπεδοποίηση εμφωλευμένων εφέ (π.χ., εφέ που προκύπτουν από συνενώσεις) και η απλοποίηση κενών εφέ. Τα εφέ κλήσεων αντικαθιστώνται από *περιλήψεις* εφέ συναρτήσεων. Ο αλγόριθμος μας εναλλάσσει τις βελτιστοποιήσεις εφέ μέχρι την εύρεση ενός σταθερού σημείου ως προς το μέγεθος του τελικού εφέ.

5.2.2 Μετασχηματισμός κώδικα

Η φάση παραγωγής κώδικα δημιουργεί στατικό κώδικα αρχικοποίησης εφέ κάθε συνάρτησης και εισάγει δείκτες στο εφέ αυτό σε διάφορα σημεία του προγράμματος, όπως κλήσεις συναρτήσεων και λειτουργίες ξεκλειδώματος. Σε αυτή τη φάση ενσωματώνουμε κώδικα διαχείρισης εφέ στην στοίβα χρόνου εκτέλεσης και κώδικα που αντιστοιχεί πραγματικές διευθύνσεις του σωρού σε αφηρημένες θέσεις μνήμης των εφέ.

5.2.3 Περιορισμοί

Κώδικας που δεν είναι γραμμένος σε C. Η ανάλυση μας μπορεί να χειριστεί μόνο τη γλώσσα C. Συνεπώς, δεν μπορούμε να χειριστούμε κώδικα βιβλιοθηκών, κώδικα γλώσσας μηχανής και μη τοπικά άλματα.

Ανάλυση δεικτών. Η ανάλυση δεικτών αποτυγχάνει όταν αντιμετωπίζει προγράμματα με αριθμητική δεικτών, πίνακες και αναδρομικές δομές δεδομένων. Επίσης, δεν μπορεί να παρακολουθήσει

benchmark	run in	user	system	elapsed	ratio
curlftpfs	C	0.002	0.758	33.450	0.982
	C+da	0.000	0.680	32.862	
flam3	C	63.660	3.910	49.050	1.003
	C+da	67.860	3.640	49.200	
migrate-n	C	5545.311	4631.341	4138.070	1.118
	C+da	5334.921	5020.346	4625.670	
ngorca	C	124.846	0.126	8.270	0.996
	C+da	124.467	0.126	8.240	
sshfs-fuse	C	0.000	0.890	20.880	1.000
	C+da	0.000	0.950	20.880	
tgrep	C	13.238	11.639	5.190	1.191
	C+da	14.801	11.655	6.180	

Πίνακας. 5.1: Απόδοση αρχικής C με μετασχηματισμένη C (C+da).

δεσμεύσεις μνήμης μη λεξικής εμβέλειας σε αναδρομικές συναρτήσεις και βρόχους και δεν επιτρέπει την ύπαρξη ζευγαριών κλειδώματος/ξεκλειδώματος σε διακριτές εντολές ελέγχου που έχουν ισοδύναμες συνθήκες. Τέλος, η ανάλυση απαιτεί ότι οι δείκτες κλειδαριών μεταβάλλονται πριν μοιρασθούν μεταξύ πολλαπλών νημάτων και ότι δεν υπάρχουν διπλοί δείκτες που αναφέρονται στην ίδια κλειδαριά.

5.2.4 Σύστημα χρόνου εκτέλεσης

Το σύστημα χρόνου εκτέλεσης παρακάμπτει τις τυπικές λειτουργίες κλειδώματος των pthreads, όπως `pthread_mutex_lock` and `pthread_cond_wait`.[1] Όταν κληθεί μια λειτουργία κλειδώματος από ένα νήμα και η κλειδαριά είναι κλειδωμένη από το νήμα αυτό, τότε ο μετρητής κλειδώματος αυξάνει κατά ένα. Διαφορετικά, το σύστημα χρόνου εκτέλεσης υπολογίζει το "μελλοντικό" σύνολο κλειδαριών και επαληθεύει ότι η ζητούμενη κλειδαριά και οι κλειδαριές του "μελλοντικού" συνόλου κλειδαριών της είναι διαθέσιμες την ίδια στιγμή. Ο υπολογισμός του "μελλοντικού" συνόλου κλειδαριών ξεκινά από τον δείκτη-μετατόπιση στο εφέ της συνάρτησης που έχει ενσωματωθεί κατά τη φάση της στατικής ανάλυσης και συνεχίζει μέχρι να βρεθεί η αντίστοιχη λειτουργία ξεκλειδώματος είτε στο εφέ της συνάρτησης είτε στην δυναμική στοίβα κλήσεων-εφέ. Έπειτα ο αλγόριθμος μας ελέγχει αρχικά αν όλες οι ζητούμενες κλειδαριές είναι διαθέσιμες. Αν κάποιες κλειδαριές δεν είναι διαθέσιμες, εκτελούμε μια λειτουργία αναμονής και δοκιμάζουμε ξανά. Εάν όλες οι κλειδαριές στο "μελλοντικό" σύνολο κλειδαριών είναι διαθέσιμες, τότε αποκτούμε δοκιμαστικά την ζητούμενη κλειδαριά. Στη συνέχεια, ελέγχουμε και πάλι το "μελλοντικό" σύνολο κλειδαριών. Εάν κάποια κλειδαριά δεν είναι διαθέσιμη τότε ξεκλειδώνουμε την κλειδαριά που κλειδώσαμε σε προηγούμενο βήμα και επαναλαμβάνουμε όλα τα στάδια από την αρχή.

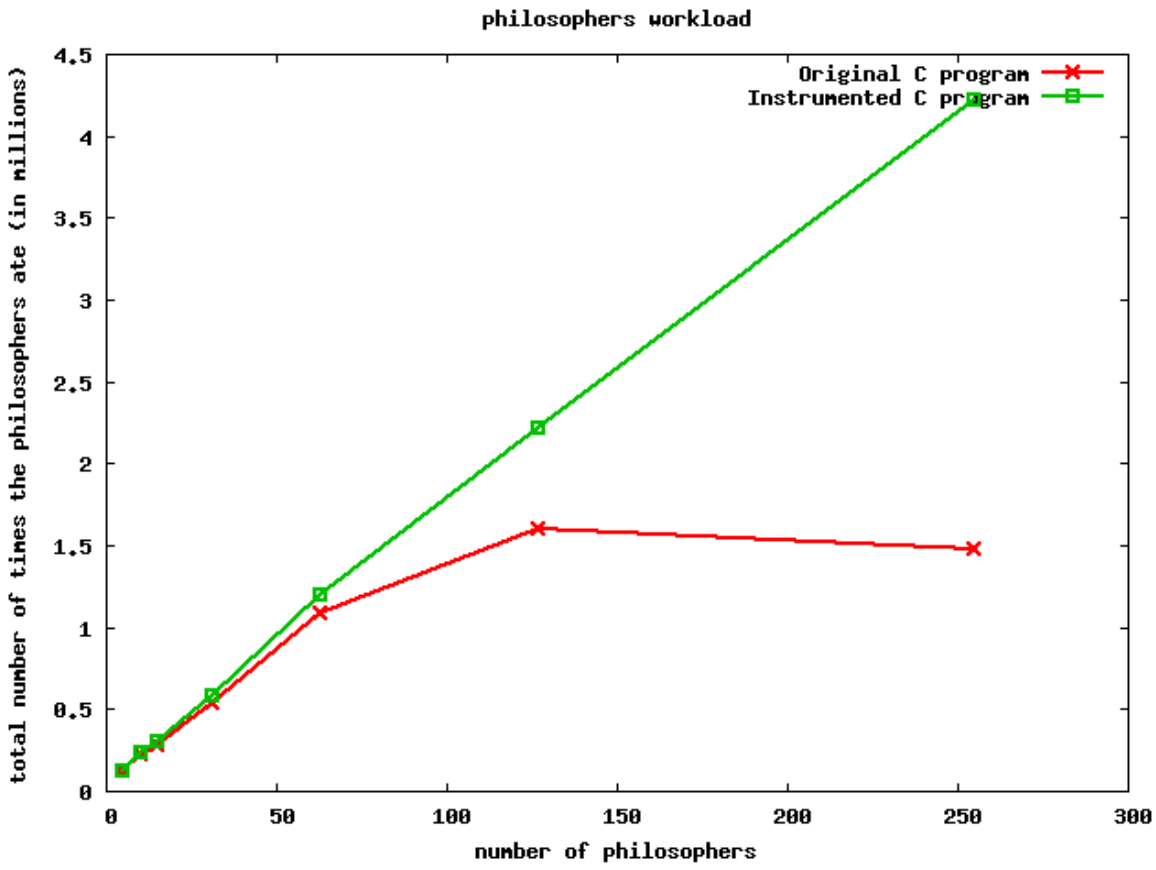
5.3 Αποτίμηση επιδόσεων

Σε αυτή την ενότητα περιγράφουμε κάποια πειραματικά αποτελέσματα, με στόχο να αποδείξουμε ότι η προσέγγιση μας επιβαρύνει ελάχιστα τον χρόνο εκτέλεσης. Τα πειράματα έγιναν σε ένα μηχάνημα με τέσσερις επεξεργαστές Intel Xeon E7340 (2,40 GHz), έχοντας συνολικά 16 πυρήνες και 16 GB μνήμης RAM, λειτουργικό σύστημα Linux 6.2.26-2-amd64 και GCC 4.3.2.

Χρησιμοποιήσαμε συνολικά επτά προγράμματα αξιολόγησης. Το πρώτο είναι γραμμένο από εμάς, ενώ το υπόλοιπο έξι είναι πραγματικές εφαρμογές (ο πηγαίος κώδικας τους είναι διαθέσιμος στο κοινό). Τα αποτελέσματα των συγκριτικών επιδόσεων παρουσιάζονται στον Πίνακα 5.1.

[1] Η υλοποίηση του εργαλείου μας και τα προγράμματα συγκριτικής αξιολόγησης είναι διαθέσιμα στο `http://www.softlab.ntua.gr/~pgerakios/deadlocks/`.

n	original	instrumented	improvement
5	126,536	126,961	0.34%
10	224,536	230,981	2.87%
15	284,150	298,563	5.07%
31	536,889	587,051	9.34%
63	1,080,322	1,193,509	10.48%
127	1,603,880	2,219,022	38.35%
255	1,480,183	4,220,603	185.14%



Σχήμα. 5.1: Συγκριτικά αποτελέσματα για το πρόβλημα των *συνδαιτυμόνων φιλοσόφων*. Μετράμε τον συνολικό αριθμό (n) που έφαγαν οι φιλόσοφοι.

συνδαιτυμόνες φιλόσοφοι: Σε αυτό το πρόγραμμα κάθε φιλόσοφος-νήμα προσπαθεί να σηκώσει το μολύβι-κλειδαριά από τον αριστερό και έπειτα από τον δεξιό του φιλόσοφο. Επειδή το αρχικό πρόγραμμα C μπορεί να εμφανίσει αδιέξοδα, συγκρίνουμε μόνο τις εκτελέσεις που δεν εμφάνισαν αδιέξοδα. Τα συγκριτικά αποτελέσματα μεταξύ του αρχικού και του μετασχηματισμένου προγράμματος C εμφανίζονται στο Σχήμα 5.1. Σε αντίθεση με το αρχικό πρόγραμμα, το μετασχηματισμένο πρόγραμμα ο αριθμός των φορών που τρώει κάθε φιλόσοφος αυξάνει γραμμικά καθώς αυξάνει ο αριθμός των φιλοσόφων. Συνεπώς, το μετασχηματισμένο πρόγραμμα κλιμακώνει καλύτερα.

curlftpfs [CURL] and sshfs-fuse [SSHF]: Αμφότερες οι εφαρμογές φτιάχνουν νήματα ώστε να εξυπηρετούν ταυτόχρονες αιτήσεις αναγνώσεων και εγγραφών σε εικονικά συστήματα αρχείων [FUSE], κάνοντας χρήση πολλαπλών διαφορετικών κλειδαριών για να συγχρονίσουν τις δομές δεδομένων, την καταγραφή συμβάντων και άλλες μοιραζόμενες δομές δεδομένων. Στα πειράματα μας, στέλνουμε παράλληλες αιτήσεις αντιγραφής στα εικόνα συστήματα αρχείων και παρατηρούμε ότι το ενσωματωμένο πρόγραμμα έχει την ίδια απόδοση με το αρχικό πρόγραμμα.

flam3: Δημιουργεί "κοσμικές αναδρομικές φλόγες fractal", δηλαδή εικόνες που δημιουργούνται αλγοριθμικά με βάση τα fractals [flam]. Μια κλειδαριά συγχρονίζει τις μοιραζόμενες δομές δεδομένων. Μετρήσαμε τον χρόνο που απαιτείται για την δημιουργία μεγάλων ακολουθιών εικόνων fractal. Τα αποτελέσματα της μέτρησης ήταν παρόμοια με τα αποτελέσματα του αρχικού προγράμματος C.

migrate-n: Υπολογίζει τις πληθυσμιακές παραμέτρους και τους ρυθμούς μετανάστευσης χρησιμοποιώντας γενετικά δεδομένα [MIGR]. Το πρόγραμμα διατηρεί μια μοιραζόμενη λίστα εργασιών σε εκκρεμότητα και ένα σύνολο νημάτων που εκτελούν τις εργασίες αυτές ώσπου να ολοκληρωθούν όλες οι εργασίες. Οι δυναμικά δεσμευμένες κλειδαριές του προγράμματος συγχρονίζουν την πρόσβαση στη μοιραζόμενη λίστα. Το μετασχηματισμένο πρόγραμμα C είναι 11% πιο αργό από το αρχικό.

ngorca: Ανακτά κωδικούς (κρυπτογραφημένους με DES) πρόσβασης χρησιμοποιώντας εξαντλητική αναζήτηση κλειδιών [NGOR]. Ο χώρος αναζήτησης μοιράζεται μεταξύ πολλαπλών νημάτων, τα οποία χρησιμοποιούν κλειδαριές για να συγχρονίσουν τις προσβάσεις μνήμης σε μοιραζόμενες δομές δεδομένων. Τα συγκριτικά αποτελέσματα μεταξύ του αρχικού και του μετασχηματισμένου προγράμματος είναι πανομοιότυπα.

tgrep: Είναι μια πολυνηματική έκδοση του προγράμματος grep, το οποίο μοιράζει τον χώρο αναζήτησης μεταξύ νημάτων, χρησιμοποιώντας επτά καθολικές κλειδαριές για συγχρονισμό σε μοιραζόμενες δομές δεδομένων. Το μετασχηματισμένο πρόγραμμα είναι 19% πιο αργό από το αρχικό πρόγραμμα. Η επιβάρυνση οφείλεται τόσο στο μέγεθος των εφέ όσο και στην εντατική χρήση κλειδαριών.

5.4 Καταληκτικές παρατηρήσεις

Τα αδιέξοδα είναι ένα σημαντικό πρόβλημα, ειδικά για τα κώδικα συστημάτων, ο οποίος είναι συχνά γραμμένος σε γλώσσες με αδόμητες λειτουργίες κλειδώματος. Σε αυτό το κεφάλαιο, παρουσιάζονται πτυχές της υλοποίησης και εφαρμογής ενός νέου εργαλείου, το οποίο αποφεύγει δυναμικά τα αδιέξοδα σε πολυνηματικά προγράμματα C. Η βασική ιδέα είναι η χρήση των ενσωματωμένων εφέ, που υπολογίζονται στατικά, για να αποφευχθούν τα αδιέξοδα κατά τον χρόνο εκτέλεσης. Περιγράφονται οι κύριες πτυχές της στατικής ανάλυσης μας και οι περιορισμοί της: Είναι ανακριβής και δεν μπορεί να υποστηρίξει αναδρομικές ή μη φραγμένες δομές δεδομένων που περιέχουν κλειδαριές. Ωστόσο, δείξαμε ότι η προσέγγισή μας είναι εφαρμόσιμη σε πολλά πολυνηματικά προγράμματα C που περιέχουν κώδικα συστημάτων και επιβαρύνει ελάχιστα τον χρόνο εκτέλεσης των προγραμμάτων.

Κεφάλαιο 6

Σχετικές εργασίες

Το έργο που παρουσιάζεται στην εργασία αυτή συνδέεται με πολλούς ερευνητικούς τομείς όπως ασφαλείς γλώσσες προγραμματισμού συστημάτων, ασφαλής διαχείριση μνήμης με περιοχές, απουσία συνθηκών ανταγωνισμού και αδιεξόδων από ταυτόχρονα προγράμματα. Στις ενότητες που ακολουθούν, συζητούμε τις σχετικές εργασίες σε κάθε μία από αυτές τις περιοχές.

6.1 Ασφαλείς γλώσσες συστημάτων

Γλώσσες προγραμματισμού, όπως η C, μπορούν να διαφοροποιηθούν από γλώσσες υψηλού επιπέδου όσον αφορά την υψηλή απόδοση τους κατά το χρόνο εκτέλεσης, τη ρητή διαχείριση μνήμης και το χαμηλό επίπεδο αναπαράστασης των δεδομένων και ελέγχου μνήμη. Η έλλειψη ισχυρών εγγυήσεων ασφάλειας όσον αφορά την συμπεριφορά των προγραμμάτων ευθύνεται για ένα μεγάλο αριθμό των τρωτών σημείων και σοβαρών παραβιάσεων ασφάλειας. Ο Austin και λοιποί εντοπίζουν δύο είδη σφαλμάτων πρόσβασης μνήμης σε διαδοχικά προγράμματα [Aust94]: χωρικά λάθη πρόσβασης (π.χ., πρόσβαση εκτός των ορίων ενός πίνακα) και χρονικά λάθη πρόσβασης (π.χ., πρόσβαση σε ένα αποδεσμευμένο τμήμα μνήμης). Γλώσσες με αδύναμα συστήματα τύπων, οι οποίες επιτρέπουν αυθαίρετες μετατροπές τύπων και μη ελεγχόμενη χρήση μνήμης εμφανίζουν λάθη πρόσβασης μνήμης. Τα πολυνηματικά προγράμματα συστημάτων, τα οποία παράγουν τυχαίες διεμπλοκές εκτέλεσης, εισαγάγουν πρόσθετα σφάλματα προσπελάσεων μνήμης και ταυτοχρονισμού.

Η υπάρχουσα έρευνα επικεντρώνεται στην παροχή στατικών εγγυήσεων ασφάλειας σε γλώσσες προγραμματισμού συστημάτων. Δύο πολλά υποσχόμενες προσεγγίσεις για την επίτευξη αυτού του στόχου είναι η χρήση συστημάτων τύπων και λογικών συστημάτων, όπως η λογική διαχωρισμού [Reyn02].

Στην βιβλιογραφία παρουσιάζονται προηγμένα συστήματα τύπων, τα οποία παρέχουν εγγυήσεις ασφάλειας για γλώσσες στο επίπεδο αφαίρεσης της C. Για παράδειγμα, η Cyclone παρέχει διαχείριση μνήμης με περιοχές με στατικές εγγυήσεις απουσίας σφαλμάτων χρονικής πρόσβασης [Jim02, Gros02]. Η CQual επιτυγχάνει παρόμοιους στόχους [Fost02] με την χρήση προσδιοριστικών τύπων, οι οποίοι επεκτείνουν το αδύναμο σύστημα τύπων της C. Το Vault υποστηρίζει γραμμικούς τύπους που αλλάζουν καταστάσεις για να παρακολουθεί την χρήση αντικειμένων κατά τον χρόνο μεταγλώττισης [DeLi01]. Με αυτό τον τρόπο επιβάλει ασφαλή πρωτόκολλα διαχείρισης μνήμης στατικά.

Η CCured εκτελεί στατική ανάλυση ολόκληρου προγράμματος ώστε να εξάγει διάφορα είδη δεικτών τα οποία χρησιμοποιούνται ως μεταδεδομένα κατά τον χρόνο εκτέλεσης για την διασφάλιση ασφάλειας μνήμης. Το σύστημα τους απομακρύνει κάποιους περιττούς δυναμικούς ελέγχους [Necu05].

Το Deputy, διάδοχος της CCured, χρησιμοποιεί ένα σύστημα εξαρτώμενων τύπων, το οποίο επιτρέπει έλεγχο τύπων ανά ενότητες (modular type checking) και αποφεύγει τη προσθήκη μεταδεδομένων σε αντικείμενα μνήμης [Cond07].

Καμία από τις παραπάνω ασφαλείς γλώσσες συστημάτων δεν παρέχουν εγγυήσεις ασφάλειας μνήμης και ταυτοχρονισμού για πολυνηματικά προγράμματα.

Υπάρχουν κάποιες εργασίες [Frig98, Gay03, Ande08a] που επεκτείνουν την γλώσσα C με ταυτοχρονισμό. Ωστόσο καμία από αυτές δεν συνδυάζει εγγυήσεις ασφάλειας μνήμης και ταυτοχρονισμού.

Σε αντίθεση με τα συστήματα τύπων, η εφαρμογή της λογικής διαχωρισμού στη C [Hobo08] επιτρέπει την διατύπωση πιο λεπτομερών προδιαγραφών ασφάλειας, αλλά δεν υπάρχει πάντα αυτόματος τρόπος επίλυσης τους.

6.2 Διαχείριση μνήμης με περιοχές

Οι Tofte και Talpin [Toft94] ανέπτυξαν το την πρώτη στατική ανάλυση για γλώσσες με ασφαλείς περιοχές μνήμης.

Έκτοτε, έχουν προταθεί αρκετές επεκτάσεις που επιτρέπουν την πρόωρη και ασφαλή αποδέσμευση περιοχών [Heng01, Walk01, Flue06, DeLi01]. Αυτές οι εργασίες χρησιμοποιούν γραμμικά συστήματα τύπων, τα οποία είναι περιοριστικά (απαγορεύονται τα αντίγραφα δεικτών) και χρειάζεται προσθήκη ρητών επισημειώσεων τύπων. Οι εγγυήσεις ασφάλειας μνήμης που προσφέρουν ισχύουν μόνο για διαδοχικά προγράμματα.

Σε αντίθεση με τις παραπάνω εργασίες, επιτρέπουμε την ύπαρξη πολλαπλών αντιγράφων δεικτών και επεκτείνουμε τις εγγυήσεις ασφάλειας μνήμης σε ταυτόχρονα προγράμματα. Επίσης, παρέχουμε εγγυήσεις για ασφαλή ταυτοχρονισμό.

Οι RC [Gay01] και Cyclone [Gros02] ήταν οι πρώτες προστακτικές γλώσσες που επέτρεψαν ρητή και διαχείριση μνήμης με περιοχές και πρόωρη αποδέσμευση περιοχών. Η RC εισήγαγε την έννοια των ιεραρχικών περιοχών μνήμης.

Σε αντίθεση με την Cyclone, η RC δεν παρέχει καμία στατική εγγύηση ασφάλειας. Αμφότερες δεν παρέχουν καμία εγγύηση για ασφαλή ταυτοχρονισμό με ιεραρχικές περιοχές μνήμης.

Η προδιαγραφή της Real-Time Java [Boll02] (RTJ) προσθέτει στην Java βιβλιοθήκες που υποστηρίζουν διαχείριση μνήμης με περιοχές. Οι εγγυήσεις ασφάλειας μνήμης στην RTJ υλοποιείται με δυναμικούς ελέγχους πρόσβασης μνήμης των περιοχών. Στατικά συστήματα περιοχών έχουν επίσης προταθεί για την RTJ για να εξαλειφθούν οι δυναμικοί έλεγχοι. Οι Boyapati και λοιποί πρότειναν ασφαλή διαχείριση μνήμης με ιεραρχικές περιοχές χρησιμοποιώντας τύπους ιδιοκτησίας [Boya02, Boya03]. Το σύστημα τύπων τους δεν υποστηρίζει πρόωρη αποδέσμευση περιοχών/κλειδαριών και δεν επιτρέπει την μετανάστευση δεδομένων ή την μεταφορά κλειδωμένων δεδομένων μεταξύ νημάτων. Επίσης, δεν επιτρέπουν στις περιοχές να εναλλάσσονται από "τοπικές" σε "μοιραζόμενες" και συνεπώς δεν επιτρέπουν σε όλες τις περιοχές να μοιρασθούν μεταξύ νημάτων. Επίσης δεν επιτρέπουν σε όλες τις περιοχές να κλειδωθούν.

Οι Zhao και λοιποί [Zhao04, Zhao08] προτείνουν τους τύπους εμβέλειας, όπου οι περιοχές λεξικής εμβέλειας περικλείονται σε αντικείμενα που ονομάζονται *εμβέλειες*. Οι στατικές ιεραρχίες περιοχών μπορούν να κωδικοποιηθούν με την εμφώλευση αντικειμένων-εμβελειών.

Το κύριο πλεονέκτημα της προσέγγισής τους είναι ότι τα προγράμματα απαιτούν λιγότερες επισημειώσεις τύπων σε σύγκριση με τα προγράμματα με ρητές περιοχές και οι περιοχές τους μπορούν να μοιράζονται σε οποιοδήποτε σημείο του προγράμματος. Το προτεινόμενο σύστημα τύπων στο Κεφάλαιο 2 επιτρέπει πρόωρη αποδέσμευση περιοχών και παρέχει επιπλέον εγγυήσεις ασφάλειας, όπως απουσία συνθηκών ανταγωνισμού.

6.3 Ασφαλής Ταυτοχρονισμός

6.3.1 Απουσία συνθηκών ανταγωνισμού

Πολλά εργαλεία και τεχνικές που έχουν αναπτυχθεί για την ανίχνευση συνθηκών ανταγωνισμού χρησιμοποιώντας στατική ανάλυση, δυναμική ανάλυση ή τον συνδυασμό τους. Το κύριο πλεονέκτημα των δυναμικών αναλύσεων ανίχνευσης συνθηκών ανταγωνισμού [Sava97, Agar05, OCal03, Choi02] είναι ότι δεν απαιτούν χειροκίνητη παρέμβαση από τον χρήστη. Ωστόσο, οι δυναμικές αναλύσεις επιβαρύνουν τον χρόνο εκτέλεσης και δεν βρίσκουν όλες τις πιθανές συνθήκες ανταγωνισμού.

Πολλά συστήματα τύπων έχουν αναπτυχθεί για τον στατικό έλεγχο ύπαρξης συνθηκών ανταγωνισμού [Flan00, Flan99b, Boya01, Boya02, Flan02] για προγράμματα Java. Τέτοια συστήματα συ-

σχετίζουν χειριστήρια κλειδαριών και θέσεων μνήμης με μονήρεις τύπους και παρακολουθούν την χρήση των μονήρων τύπων στο πρόγραμμα. Τα περισσότερα συστήματα τύπων απαιτούν την εισαγωγή ρητών επισημειώσεων στα προγράμματα. Ο Grossman πρότεινε ένα παρόμοιο σύστημα τύπων για την Cyclone [Gros03]. Το σύστημα του δεν επιτρέπει την πρόωρη αποδέσμευση περιοχών μνήμης και δεν υποστηρίζει την μετανάστευση δεδομένων ή κλειδωμάτων μεταξύ νημάτων. Οι περιοχές στο σύστημα του δεν μπορούν να εναλλάσσονται από "τοπικές" σε "μοιραζόμενες".

Στα πλαίσια της Java, οι Cunningham και λοιποί προτείνουν ένα σύστημα τύπων συμπάντων, το οποίο εγγυάται ελευθερία από συνθήκες ανταγωνισμού σε ένα λογισμό αντικειμένων [Cunn07]. Παρόμοια με το σύστημα μας, οι ιεραρχίες αντικειμένων μπορούν να κλειδωθούν ατομικά σε οποιοδήποτε επίπεδο. Το σύστημα των Cunningham και λοιπών δεν υποστηρίζει πρόωρη αποδέσμευση κλειδαριών και μεταφορά κλειδαριών μεταξύ νημάτων.

Στην βιβλιογραφία παρουσιάζονται εργαλεία αυτόματης ανάλυσης προγραμμάτων Java για εύρεση πιθανών συνθηκών ανταγωνισμού [Flan01] και C [Prat06, Voun07]. Οι τελευταίες δύο αναλύσεις επαληθεύουν ότι οι προσβάσεις σε μοιραζόμενες θέσεις μνήμης προστατεύονται από την ίδια κλειδαριά σε όλα τα νήματα. Ο Terauchi [Tera08] πρότεινε μια τεχνική αναγωγής του προβλήματος συνθηκών ανταγωνισμού σε πρόβλημα γραμμικού προγραμματισμού. Έχουν προταθεί και άλλες τεχνικές εύρεσης συνθηκών ανταγωνισμού, όπως αφηρημένη διερμηνεία [Gots07] και έλεγχος μοντέλων [Henz04, Qade04]. Σε αντίθεση με την ανάλυση μας, οι αναλύσεις αυτές δεν ελέγχουν εάν υπάρχουν ξεκρέμαστες λειτουργίες κλειδώματος, οι οποίες μπορούν να οδηγήσουν σε συνθήκες ανταγωνισμού και μη ντετερμινιστική συμπεριφορά των προγραμμάτων. Πρόσφατες εργασίες στατικής ανάλυσης για C προγράμματα [Ande08b, Ande09] αναγνωρίζουν αυτόματα τις μοιραζόμενες θέσεις μνήμης και απορρίπτουν προγράμματα που μοιράζονται δεδομένα ακούσια.

Πιο πρόσφατα, οι Bocchino και λοιποί πρότειναν ένα σύστημα τύπων και εφέ για την DPJ (ντετερμινιστική παράλληλη Java) [Bocc09] το οποίο διαχωρίζει τον σωρό σε διακριτά ιεραρχικά τμήματα για να αποσαφηνίσει προσβάσεις σε διαφορετικές θέσεις μνήμης. Το σύστημα τύπων της DPJ, εξασφαλίζει απουσία συνθηκών ανταγωνισμού με την επιβολή της αναλλοίωτης συνθήκης ότι οι μη συγχρονισμένες και ταυτόχρονες προσβάσεις σε μοιραζόμενες θέσεις μνήμης είναι μόνο για ανάγνωση. Οι ταυτόχρονες εγγραφές μοιραζόμενων τμημάτων μνήμης είναι αποδεκτές μόνο όταν τα τμήματα μνήμης είναι ξένα μεταξύ τους. Η προσέγγιση τους αποτρέπει την ύπαρξη πολλαπλών αντιγράφων μνήμης σε επίπεδο τύπων και συνεπώς περιορίζει σημαντικά την εκφραστικότητα της γλώσσας. Αντίθετα, στο σύστημα τύπων μας οι περιοχές μπορούν να αντιγραφούν σε οποιοδήποτε σημείο του προγράμματος. Το σύστημα τύπων της DPJ επεκτάθηκε πρόσφατα με λειτουργίες συγχρονισμού χωρίς αναμονή, οι οποίες εγγυώνται ισχυρή απομόνωση δεδομένων [Bocc11].

Το σύστημα μας μπορεί να κωδικοποιήσει τις παραπάνω δυνατότητες με την χρήση κλειδαριών ανάγνωσης/εγγραφής. (Ωστόσο, ο συγχρονισμός στην δική μας περίπτωση απαιτεί αναμονή.) Επιπλέον, οι χαμηλού επιπέδου γλώσσες, όπως αυτή που προτείνουμε, συχνά αποτελούν τη γλώσσα μεταγλώττισης υψηλού επιπέδου γλωσσών, όπως η Java και η DPJ. Στο πλαίσιο αυτό, το σύστημα τύπων μας δεν δεσμεύεται σε κάποια συγκεκριμένη γλώσσα προγραμματισμού, αλλά είναι εφαρμόσιμο σε οποιαδήποτε γλώσσα.

Οι Matsakis και Gross προτείνουν μια άλλη παραλλαγή της Java με στατικές εγγυήσεις απουσίας συνθηκών ανταγωνισμού εισάγοντας την έννοια των *διαστημάτων* [Bocc11]. Τα διαστήματα είναι αντικείμενα πρώτης τάξης που αναπαριστούν χρονικά διαστήματα στα οποία εκτελείται ένα ορισμένο κομμάτι κώδικα. Τα διαστήματα μπορούν να διαταχθούν μερικώς και να δομηθούν με ιεραρχικό τρόπο. Ομοίως με τις περιοχές μας, τα ιεραρχικά διαστήματα κληρονομούν δικαιώματα πρόσβασης από διαστήματα-προγόνους. Ταυτόχρονες αναγνώσεις επιτρέπονται μόνο σε διαστήματα που ακολουθούν διαστήματα, στα οποία πραγματοποιούνται εγγραφές από ένα νήμα μόνο. Οι ταυτόχρονες εγγραφές είναι αποδεκτές αρκεί να υπάρχει συγχρονισμός με κλειδαριές, οι οποίες κλειδώνονται έμμεσα από το σύστημα χρόνου εκτέλεσης. Σε αντίθεση με το σύστημα μας, το σύστημα τύπων τους απαιτεί ρητές επισημειώσεις διάταξης των κλειδαριών και των διαστημάτων κάθε συνάρτησης.

6.3.2 Απουσία αδιεξόδων

Η πλειοψηφία της βιβλιογραφίας για αναλύσεις που εγγυώνται απουσία αδιεξόδων από έγκυρα προγράμματα εμπίπτει στην πρόληψη αδιεξόδων και τον δυναμικό εντοπισμό και ανάνηψη από αυτά. Στην κατηγορία πρόληψης αδιεξόδων, έχουν αναπτυχθεί συστήματα τύπων και εφέ [Flan99b, Boya02, Koba06, Suen08, Vasc10] τα οποία εγγυώνται απουσία αδιεξόδων με την επιβολή μιας καθολικής διάταξης απόκτησης των κλειδαριών από όλα τα νήματα. Οι Flanagan και Abadi [Flan99b], ήταν οι πρώτοι που πρότειναν σύστημα μονήρων τύπων όπου τα χειριστήρια κλειδαριών συσχετίζονται με ονόματα κλειδαριών σε επίπεδο τύπων. Για παράδειγμα το χειριστήριο $lk_\imath$ είναι τύπου $\mathtt{lk}(\imath)$. Το ίδιο ισχύει για μεταβλητές χειριστηρίων κλειδαριών. Το σύστημα τύπων και εφέ που προτείνουν παρακολουθεί την διάταξη των λειτουργιών κλειδώματος μέσω μονήρων τύπων και προσδιορίζει εάν όλα τα νήματα αποκτούν κλειδαριές με τον ίδια διάταξη. Αυτό μπορεί να είναι υπερβολικά περιοριστικό για ορισμένα προγράμματα. Για παράδειγμα, στο παρακάτω πρόγραμμα έχουμε δύο παράλληλα νήματα, τα οποία κλειδώνουν δυο διαφορετικές κλειδαριές, x and y με αντίστροφη σειρά.

$$(\mathtt{lock}\ x\ \mathtt{in}\ \ldots\ \mathtt{lock}\ y\ \mathtt{in}\ \ldots) \mid\mid (\mathtt{lock}\ y\ \mathtt{in}\ \ldots\ \mathtt{lock}\ x\ \mathtt{in}\ \ldots)$$

Το πρόγραμμα αυτό απορρίπτεται από όλα τα συστήματα τύπων [Flan99b, Koba06, Suen08, Vasc10] που απαιτούν ύπαρξη καθολικής διάταξης απόκτησης κλειδαριών. Μια αξιοσημείωτη εξαίρεση στην έρευνα πρόληψης αδιεξόδων, είναι το σύστημα τύπων των Boyapati και λοιποί [Boya02] το οποίο είναι λιγότερο συντηρητικό. Συγκεκριμένα, επιτρέπει στους προγραμματιστές να κλειδώνουν σε ένα καθορισμένο αριθμό κλάσεων ισοδυναμίας (επίπεδα κλειδαριών), να χρησιμοποιούν αναδρομικές δεντρικές δομές δεδομένων, οι οποίες περιγράφουν την μερική τάξη κλειδαριών, καθώς επίσης να εκτελούν ένα περιορισμένο εύρος μεταβολών σε αυτές τις δομές δεδομένων επιτρέποντας έτσι να μεταβληθεί η διάταξη των επιμέρους κλειδαριών. Ακόμα και σε αυτό το σύστημα όμως, η διάταξη μεταξύ επιπέδων κλειδαριών καθορίζεται στατικά. Σε αντίθεση, το σύστημα μας δεν επιβάλλει μερική διάταξη κλειδαριών κατά τον χρόνο εκτέλεσης, αλλά επιτρέπει την ασφαλή απόκτηση τους κατά τον χρόνο εκτέλεσης με βάση τις πραγματικές ανάγκες του προγράμματος.

Στατικά εργαλεία πρόληψης αδιεξόδων χρησιμοποιούν ευαίσθητες αναλύσεις [Engl03] και αποδείκτες θεωρημάτων [Flan02] για να εντοπίσουν προγράμματα που δεν τηρούν καθολική διάταξη απόκτησης κλειδαριών από όλα τα νήματα. Στη θεωρία, οι στατικές προσεγγίσεις είναι ελκυστικές επειδή δεν επιβαρύνουν τον χρόνο εκτέλεσης των προγραμμάτων. Στην πράξη, η υιοθέτηση αυστηρής διάταξης κλειδώματος είναι ακατάλληλη για προγραμματισμό συστημάτων και είναι δύσκολο να επιβληθεί σε λογισμικό γραμμένο από μεγάλες ομάδες προγραμματιστών. Οι καθαρά στατικές προσεγγίσεις είναι εξ ορισμού συντηρητικές, απορρίπτουν συχνά προγράμματα χωρίς λόγο ή οδηγούν σε μεγάλο αριθμό λανθασμένων συναγερμών.

Πρόσφατα ο Boudol ανέπτυξε ένα σύστημα τύπων και εφέ αποφυγής αδιεξόδων [Boud09] για προγράμματα με δομημένες λειτουργίες κλειδώματος. Το σύστημα εφέ υπολογίζει το σύνολο αποκτώμενων κλειδαριών κάθε έκφρασης και επισημειώνει κάθε λειτουργία κλειδώματος με το "μελλοντικό" σύνολο αποκτώμενων κλειδαριών της. Το σύστημα χρόνου εκτέλεσης επιτρέπει να κλειδωθεί μια κλειδαριά μόνο όταν όλες οι κλειδαριές του "μελλοντικού" συνόλου κλειδώματος είναι διαθέσιμες. Το βασικό όφελος της προσέγγισης του Boudol είναι ότι επιτρέπει περισσότερα έγκυρα προγράμματα σε σύγκριση με τον αριθμό έγκυρων προγραμμάτων καθαρά στατικών αναλύσεων.

Το παράδειγμα που εξετάσαμε προηγουμένως μπορεί να διατυπωθεί στην γλώσσα του Boudol ως εξής:

$$(\mathtt{lock}_{\{y\}}\ x\ \mathtt{in}\ \ldots\ \mathtt{lock}_{\emptyset}\ y\ \mathtt{in}\ \ldots) \mid\mid (\mathtt{lock}_{\{x\}}\ y\ \mathtt{in}\ \ldots\ \mathtt{lock}_{\emptyset}\ x\ \mathtt{in}\ \ldots)$$

Το παραπάνω πρόγραμμα είναι έγκυρο στο σύστημα τύπων του Boudol, το οποίο δεν επιβάλει κανένα περιορισμό στην διάταξη απόκτησης των κλειδαριών. Κατά τον χρόνο εκτέλεσης, εάν το πρώτο νήμα κλειδώσει το x, τότε η πρώτη λειτουργία κλειδώματος του δεύτερου νήματος θα μπλοκάρει, ώσπου αμφότερες οι κλειδαριές x και y γίνουν διαθέσιμες. Ωστόσο, το σύστημα του Boudol δεν χειρίζεται προγράμματα με αδόμητο κλείδωμα.

Ένα εργαλείο που είναι αρκετά παρόμοιο με το δικό μας είναι το Gadara [Wang08]. Το Gadara πραγματοποιεί στατική ανάλυση ολόκληρου προγράμματος για να μοντελοποιήσει την ροή των κλειδαριών σε πολυνηματικά προγράμματα και συνθέτει μια παράλληλη λογική που αποφεύγει αδιέξοδα κατά το χρόνο εκτέλεσης χρησιμοποιώντας διακριτή θεωρία συστημάτων ελέγχου [Wang09]. Το Gadara στοχεύει προγράμματα C/pthreads και εκτελεί το μεγαλύτερο μέρος της αποφυγής αδιεξόδων στατικά. (Το Gadara δεν είναι διαθέσιμο στο κοινό.) Η ως επί το πλείστον στατική προσέγγιση που ακολουθείται από το Gadara, καθώς και η έλλειψη ανάλυσης αντιγράφων, οδηγεί σε ανακριβείς προσεγγίσεις των "μελλοντικών" συνόλων κλειδαριών και συνεπώς σε ελαττωμένη παραλληλία.

Καθαρά δυναμικές προσεγγίσεις που βασίζονται στην ανίχνευση αδιεξόδων [Qin07, Jula08] δεν εμφανίζουν ψευδώς θετικά αποτελέσματα. Ωστόσο η ανάνηψη και επαναφορά προγραμμάτων που βρίσκονται σε αδιέξοδο δεν είναι πάντα δυνατή (π.χ., όταν έχουν πραγματοποιηθεί λειτουργίες I/O).

6.3.3 Διαφορετικές προσεγγίσεις για ασφαλή ταυτοχρονισμό

Μνήμη συναλλαγών. Η υιοθέτηση μνήμης συναλλαγών [Harr03, Harr05, Herl03, Ring05, Welc04] εξαλείφει αυτόματα σφάλματα ταυτοχρονισμού, όπως συνθήκες ανταγωνισμού και αδιέξοδα. Η βασική ιδέα της μνήμης συναλλαγών είναι ότι οι προσβάσεις μνήμης καταγράφονται στα πλαίσια συναλλαγών σε μια βοηθητική μνήμη. Οι λειτουργίες εγγραφής καταγράφονται στην κυρίως μνήμη στο πέρας των συναλλαγών, όταν οι καταγεγραμμένες προσβάσεις της βοηθητικής μνήμης είναι συνεπείς με την τρέχουσα κατάσταση της κυρίως μνήμης. Διαφορετικά, η συναλλαγή αποσύρεται και επανακινείται. Οι συναλλαγές δεν αλληλεπιδρούν καλά με τις λειτουργίες I/O και σε ορισμένες περιπτώσεις η απόδοση τους μπορεί να είναι χειρότερη από το κλείδωμα εξαιτίας της επιβάρυνσης των λειτουργιών καταγραφής και επανεκκίνησης.

Επικοινωνία με μηνύματα. Το πέρασμα μηνυμάτων είναι μια εναλλακτική μορφή επικοινωνίας πολυνηματικών προγραμμάτων. Η βασική ιδέα είναι ότι ο απαραίτητος συγχρονισμός γίνεται με ανταλλαγή μηνυμάτων και ανακουφίζει τον προγραμματιστή από τη χρήση ρητών κλειδαριών και μοιραζόμενων θέσεων μνήμης. Λάθη ταυτοχρονισμού, όπως οι συνθήκες ανταγωνισμού, εξαλείφονται δεν υφίσταται μοιραζόμενη μνήμη στο μοντέλο επικοινωνίας μέσω ανταλλαγής μηνυμάτων. Δυστυχώς, δεν ισχύει το ίδιο για άλλα σφάλματα ταυτοχρονισμού όπως τα αδιέξοδα.

Μια μινιμαλιστική επίσημη περιγραφή της ανταλλαγής μηνυμάτων είναι ο π-λογισμός [Miln99]. Σε γλώσσες χαμηλού επιπέδου, όπως η C, δεν υπάρχει ρητή υποστήριξη του μοντέλου ανταλλαγής μηνυμάτων και γίνεται χρήση εξωτερικών βιβλιοθηκών, όπως το MPI [Grop96]. Η Erlang [Arms07] είναι από τις λίγες γλώσσες προγραμματισμού που έχουν ενσωματωμένη υποστήριξη για επικοινωνία με ανταλλαγή μηνυμάτων. Η προδιαγραφή της γλώσσας αποτρέπει τη χρήση μοιραζόμενης μνήμης. Ωστόσο, έχουν αναφερθεί [Chri10] συνθήκες ανταγωνισμού που προέρχονται από την υλοποίηση της γλώσσας και εξωτερικές βιβλιοθήκες.

Κεφάλαιο 7

Συμπεράσματα

Στην παρούσα εργασία, προτείναμε μια σειρά από συνεπείς στατικές αναλύσεις που εγγυώνται την απουσία ορισμένων σφαλμάτων μνήμης και ταυτοχρονισμού σε έγκυρα προγράμματα. Επίσης προτείναμε μια τεχνική που συνδυάζει στατική ανάλυση με δυναμικούς έλεγχους για την αποφυγή αδιεξόδων από έγκυρα προγράμματα. Ας συνοψίσουμε τις κύριες πτυχές και συνεισφορές της παρούσας εργασίας:

- Παρουσιάσαμε μια πολυνηματική γλώσσα που δεν απαιτεί ρητές σημειώσεις τύπων και χρησιμοποιεί προηγμένη διαχείριση μνήμης με ιεραρχικές περιοχές και ιεραρχικό συγχρονισμό με κλειδαριές ανάγνωσης/εγγραφής. Η προσέγγιση μας δίνει ρητό έλεγχο στην διάρκεια ζωής και κλειδώματος των περιοχών. Επίσης, υποστηρίζει κοινά ιδιώματα πολυνηματικού προγραμματισμού, όπως η μετανάστευση και μεταφορά δεδομένων, το πέρασμα κλειδωμένων δεδομένων μεταξύ νημάτων. Επιπλέον, τα δεδομένα εναλλάσσονται μεταξύ " τοπικής" και "μοιραζόμενης" κατάστασης. Ορίσαμε την λειτουργική σημασιολογία και τα σύστημα τύπων για την γλώσσα αυτή και αποδείξαμε τα θεωρήματα που εγγυώνται ότι τα έγκυρα προγράμματα δεν εμφανίζουν παραβιάσεις πρόσβασης μνήμης και συνθήκες ανταγωνισμού. Η προσέγγιση μας είναι η πρώτη που συνδυάζει τις προαναφερθείσες λειτουργίες και εγγυήσεις ασφάλειας.

- Υλοποιήσαμε και ενσωματώσαμε τους φορμαλισμούς μας στην γλώσσα Cyclone και συγκρίναμε την απόδοση προγραμμάτων Cyclone με C/pthreads προγράμματα. Στις περισσότερες περιπτώσεις, η επιβάρυνση κατά τον χρόνο εκτέλεσης ήταν αμελητέα.

- Παρουσιάσαμε μια γλώσσα με αδόμητες λειτουργίες κλειδώματος. Ορίσαμε την λειτουργική σημασιολογία και το σύστημα τύπων για τη γλώσσα αυτή και αποδείξαμε ότι δεν εμφανίζονται αδιέξοδα σε έγκυρα προγράμματα. Αυτή είναι η πρώτη τεχνική αποφυγής αδιεξόδων, η οποία βασίζεται σε συστήματα τύπων και εγγυάται απουσία αδιέξοδων για με γλώσσες αδόμητες λειτουργίες κλειδώματος.

- Υλοποιήσαμε το σύστημα τύπων για την αποφυγή αδιεξόδων ως μια στατική ανάλυση για C/pthreads και περιγράψαμε τις αποφάσεις για το σχεδιασμό και τις βελτιστοποιήσεις της ανάλυσης που κρίθηκαν αναγκαίες. Η συγκριτική αξιολόγηση της ανάλυσης μας δείχνουν μικρές επιβαρύνσεις στον χρόνο εκτέλεσης προγραμμάτων C/pthreads.

Συνολικά, στηρίξαμε την θέση ότι η αξιοπιστία λογισμικού μπορεί να βελτιωθεί αποτελεσματικά και αποδοτικά με την κατάργηση σφαλμάτων μνήμης και ταυτοχρονισμού, όπως εσφαλμένες προσβάσεις μνήμης, συνθήκες ανταγωνισμού και αδιέξοδα με την χρήση στατικών αναλύσεων. Υπάρχουν πιθανές επεκτάσεις και βελτιώσεις στις θεωρίες και οι υλοποιήσεις που παρουσιάζονται στην παρούσα διατριβή, οι οποίες αποτελούν αντικείμενο μελλοντικών εργασιών:

Συμπερασμός αναδρομικών εφέ: σχετικά με το σύστημα περιοχών, υπάρχουν αρκετοί περιορισμοί που εφαρμόζονται στις "εξωτερικές" περιοχές (δηλαδή σε περιοχές που υπάρχουν πριν την κλήση αναδρομικών συναρτήσεων σε αντίθεση με τις περιοχές που δεσμεύονται στο σώμα αναδρομικών συναρτήσεων) κατά τον υπολογισμό εφέ αναδρομικών συναρτήσεων. Για παράδειγμα, δεν επιτρέπουμε σε αναδρομικές συναρτήσεις να αποδεσμεύουν εξωτερικές περιοχές

και απαιτούμε οι μετρητές περιοχών να είναι ίσοι πριν και μετά την κλήση αναδρομικών συναρτήσεων. Οι αναδρομικές συναρτήσεις δεν μπορούν να περάσουν κλειδωμένες κλειδαριές εξωτερικών περιοχών σε νέα νήματα και θεωρούν τις κλειδαριές εξωτερικών περιοχών ως ξεκλείδωτες. Ομοίως με το σύστημα περιοχών, το σύστημα αποφυγής αδιεξόδων απαιτεί οι μετρητές κλειδαριών να είναι ίσοι πριν και μετά την κλήση αναδρομικών συναρτήσεων. Κατά τη διάρκεια της διαδικασίας συγκριτικής αξιολόγησης κατά την οποία μεταφέραμε αρκετά προγράμματα από C σε Cyclone, δεν εμφανίσθηκαν προγράμματα που δεν ήταν δυνατόν να αναλυθούν από τον αλγόριθμο μας. Θα ήταν ενδιαφέρον να διερευνήσουμε περαιτέρω την ανάπτυξη ενός νέου αλγόριθμου συμπερασμού εφέ που δεν έχει τους προαναφερθέντες περιορισμούς.

Ταυτόχρονη Cyclone Η υλοποίηση μας επιδέχεται πολλές βελτιστοποιήσεις και βελτιώσεις. Έχουμε εντοπίσει τις τρεις πιο σημαντικές, σύμφωνα με τις εμπειρίες που αποκτήσαμε από την διαδικασία συγκριτικής αξιολόγησης. Πρώτον, χρειαζόμαστε κατασκευή λεξικής εμβέλειας, η οποία θα επιτρέπει σε νήματα-γονείς να περιμένουν τα νήματα-παιδιά τους να τερματίσουν όπως γίνεται στη γλώσσα Cilk [Frig98].

```
join {
        for (int i = 0 ; i < size ; i++)
                spawn worker(a[i]);
      }
```

Δεύτερον, για ορισμένες εφαρμογές, θα ήταν προτιμότερο να κλειδώνουμε αναφορές αντί να κλειδώνουμε ολόκληρες περιοχές. Το σύστημα τύπων μας είναι αρκετά γενικό ώστε να υποστηρίζει κάτι τέτοιο εάν αντιμετωπίσουμε τις αναφορές ως περιοχές μνήμης. Δηλαδή, μία νέα περιοχή θα μπορούσε να ανατίθεται σε αναφορές που έχουν την δυνατότητα κλειδώματος. Τρίτον, σε ορισμένα από τα προγράμματα C που ξεπέρασαν τα αντίστοιχα προγράμματα Cyclone, όπως το chameneos-redux, μη επικαλυπτόμενα τμήματα πινάκων προσπελάζονταν ταυτόχρονα. Η επέκταση του προτεινόμενου συστήματος τύπων και του συστήματος χρόνου εκτέλεσης έτσι ώστε μόνο να επιτρέπουμε ταυτόχρονη πρόσβαση εγγραφής σε μη επικαλυπτόμενα μέρη ενός πίνακα ή μιας αναδρομικής δομής δεδομένων είναι μια ενδιαφέρουσα επέκταση της εργασίας μας.

Συστήματα τύπων για αποφυγή αδιεξόδων σε C/pthreads: Η διαδικασία αξιολόγησης του εργαλείου αποφυγής αδιεξόδων αποκάλυψε ότι οι κλειδαριές χρησιμοποιούνται σε πίνακες και αναδρομικές δομές δεδομένων σε ένα μεγάλο αριθμό πραγματικών C προγραμμάτων. Το σύστημα τύπων για αποφυγή αδιεξόδων που προτείνουμε είναι ανεπαρκές για προγράμματα με αναδρομικές δομές δεδομένων και πίνακες που περιέχουν ή αναφέρονται σε κλειδαριές. Μια συντηρητική λύση στο πρόβλημα αυτό θα απαιτούσε τα "μελλοντικά" σύνολα κλειδαριών να περιέχουν όλες τις πιθανές κλειδαριές στο πρόγραμμα. Λιγότερο συντηρητικές λύσεις απαιτούν περαιτέρω έρευνα. Δεύτερον, η επιβάρυνση του χρόνου εκτέλεσης που προκαλείται από την τεχνική μας μπορεί να αποδοθεί στον χρόνο υπολογισμού του συνόλου κλειδαριών και στον χρόνο αναμονής σε λειτουργίες κλειδώματος όταν το σύνολο κλειδαριών δεν είναι διαθέσιμο. Ο χρόνος υπολογισμού του συνόλου κλειδαριών αποτελεί το μισό του συνολικού χρόνου επιβάρυνσης σε προγράμματα με μεγάλο αριθμό εμφωλευμένων κλειδαριών. Η χειρότερη περίπτωση χρόνου εκτέλεσης του υπολογισμού των "μελλοντικών" συνόλων κλειδαριών εξαρτάται από το μέγεθος της στοίβας εφέ όταν καλείται μια λειτουργία κλειδώματος. Ένας τρόπος για την εξάλειψη αυτής της καθυστέρησης είναι η προσωρινή αποθήκευση συνόλων κλειδαριών που έχουν ήδη έχουν υπολογιστεί. Μια άλλη δυνατότητα είναι να υπολογιστούν στατικά τα σύνολα "μελλοντικών" κλειδαριών λαμβάνοντας υπόψιν τα περικλείοντα πλαίσια (context-sensitive analysis). Η βασική ιδέα είναι η διατήρηση απεικόνισης από πλαίσια σε σύνολα κλειδαριών: δεδομένης της υπογραφής ενός πλαισίου, το στατικά υπολογισμένο σύνολο κλειδαριών μπορεί να ανακτηθεί χωρίς επιπλέον επιβάρυνση. Φυσικά, οι αναδρομικές συναρτήσεις χρειάζονται ειδική μεταχείριση και αυτό είναι θέμα μελλοντικών εργασιών. Τρίτον, η ανάλυση δεικτών θα μπορούσε να

βελτιωθεί έτσι ώστε οι μεταλλάξεις τους να εντοπίζονται από άλλα νήματα. Επί του παρόντος, η ανάλυσή μας απαιτεί ότι οι δείκτες μεταβάλλονται πριν μοιρασθούν μεταξύ πολλαπλών νημάτων. Τέλος, ο κώδικας συστημάτων περιλαμβάνει χαρακτηριστικά όπως ενσωματωμένη γλώσσα μηχανής, μη-τοπικά άλματα και ειδικές οδηγίες για την πραγματοποίηση κλειδώματος. Επεκτείνοντας το εργαλείο μας για να υποστηρίζει πλήρως κώδικα συστημάτων είναι στόχος μελλοντικών εργασιών.

Βιβλιογραφία

[Agar05] Rahul Agarwal, Amit Sasturkar, Liqiang Wang and Scott D. Stoller, "Optimized run-time race detection and atomicity checking using partial discovered types", in *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, pp. 233–242, New York, NY, USA, 2005, ACM.

[Ande08a] Todd Anderson, Neal Glew, Peng Guo, Brian T. Lewis, Wei Liu, Zhanglin Liu, Leaf Petersen, Mohan Rajagopalan, James M. Stichnoth, Gansha Wu and Dan Zhang, "Pillar: A Parallel Implementation Language", in *Proceedings of the International Workshop on Languages and Compilers for Parallel Computing*, vol. 5234 of *LNCS*, pp. 141–155, Springer, 2008.

[Ande08b] Zachary R. Anderson, David Gay, Robert Ennals and Eric Brewer, "SharC: Checking data sharing strategies for multithreaded C", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 149–158, New York, NY, USA, 2008, ACM.

[Ande09] Zachary R. Anderson, David Gay and Mayur Naik, "Lightweight Annotations for Controlling Sharing in Concurrent Data Structures", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 98–109, New York, NY, USA, 2009, ACM.

[Arms07] Joe Armstrong, "Erlang - Software for a Concurrent World", in *European Conference on Object-Oriented Programming*, pp. 1–1, Springer, 2007.

[Aust94] Todd M. Austin, Scott E. Breach and Gurindar S. Sohi, "Efficient detection of all pointer and array access errors", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 290–301, ACM, 1994.

[Bocc09] Robert L. Bocchino, Jr., Vikram S. Adve, Danny Dig, Sarita V. Adve, Stephen Heumann, Rakesh Komuravelli, Jeffrey Overbey, Patrick Simmons, Hyojin Sung and Mohsen Vakilian, "A Type and Effect System for Deterministic Parallel Java", in *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pp. 97–116, New York, NY, USA, 2009, ACM.

[Bocc11] Robert L. Bocchino, Jr., Stephen Heumann, Nima Honarmand, Sarita V. Adve, Vikram S. Adve, Adam Welc and Tatiana Shpeisman, "Safe Nondeterminism in a Deterministic-by-default Parallel Language", in *Conference Record of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 535–548, New York, NY, USA, 2011, ACM.

[Boll02] Gregory Bollella and James Gosling, "The real-time specification for Java", *Computer*, vol. 33, no. 6, pp. 47–54, 2002.

[Boud09] Gérard Boudol, "A Deadlock-Free Semantics for Shared Memory Concurrency", in Martin Leucker and Carroll Morgan, editors, *Proceedings of the International Colloquium on Theoretical Aspects of Computing*, vol. 5684 of *LNCS*, pp. 140–154, Springer, 2009.

[Boya01] Chandrasekhar Boyapati and Martin Rinard, "A Parameterized Type System for Race-free Java Programs", in *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pp. 56–69, New York, NY, USA, 2001, ACM Press.

[Boya02] Chandrasekhar Boyapati, Robert Lee and Martin Rinard, "Ownership Types for Safe Programming: Preventing Data Races and Deadlocks", in *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pp. 211–230, New York, NY, USA, November 2002, ACM Press.

[Boya03] Chandrasekhar Boyapati, Alexandru Salcianu, William S. Beebee and Martin Rinard, "Ownership Types for Safe Region-based Memory Management in Real-Time Java", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 324–337, New York, NY, USA, June 2003, ACM Press.

[Choi02] Jong-Deok Choi, Keunwoo Lee, Alexey Loginov, Robert O'Callahan, Vivek Sarkar and Manu Sridharan, "Efficient and precise datarace detection for multithreaded object-oriented programs", in *Proceedings of the ACM SIGPLAN 2002 Conference on Programming Language Design and Implementation*, pp. 258–269, ACM, 2002.

[Chri10] Maria Christakis and Konstantinos Sagonas, "Static Detection of Race Conditions in Erlang", in *Practical Aspects of Declarative Languages*, vol. 5937 of *LNCS*, pp. 119–133, Springer, 2010.

[Cond07] Jeremy Condit, Matthew Harren, Zachary R. Anderson, David Gay and George C. Necula, "Dependent Types for Low-level Programming", in Rocco De Nicola, editor, *Programming Language and Systems: Proceedings of the European Symposium on Programming*, vol. 4421 of *LNCS*, pp. 520–535, Springer, 2007.

[Cunn07] David Cunningham, Sophia Drossopoulou and Susan Eisenbach, "Universes for Race Safety", in *Proceedings of the Workshop on Verification and Analysis of Multi-threaded Java-like Programs*, pp. 20–51, 2007.

[CURL] "A FTP filesystem based on cURL and FUSE", `http://curlftpfs.sourceforge.net/`.

[DeLi01] Robert DeLine and Manuel Fähndrich, "Enforcing High-level Protocols in Low-level Software", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 59–69, New York, NY, USA, 2001, ACM Press.

[Engl03] Dawson Engler and Ken Ashcraft, "RacerX: effective, static detection of race conditions and deadlocks", in *Proceedings of ACM Symposium on Operating Systems Principles*, pp. 237–252, New York, NY, USA, 2003, ACM.

[flam] flam3.com, "Cosmic Recursive Fractal Flames", `http://flam3.com/`.

[Flan99a] Cormac Flanagan and Martín Abadi, "Object Types Against Races", in Jos C. M. Baeten and Sjouke Mauw, editors, *International Conference on Concurrency Theory*, vol. 1664 of *LNCS*, pp. 288–303, Springer, 1999.

[Flan99b] Cormac Flanagan and Martín Abadi, "Types for Safe Locking", in *Programming Language and Systems: Proceedings of the European Symposium on Programming*, vol. 1576 of *LNCS*, pp. 91–108, Springer, 1999.

[Flan00] Cormac Flanagan and Stephen N. Freund, "Type-based race detection for Java", in *Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation*, pp. 219–232, ACM, 2000.

[Flan01] Cormac Flanagan and K. Rustan M. Leino, "Houdini, an Annotation Assistant for ESC/Java", in *Formal Methods Europe*, vol. 2021 of *LNCS*, pp. 500–517, Springer, 2001.

[Flan02] Cormac Flanagan, K. Rustan M. Leino, Mark Lillibridge, Greg Nelson, James B. Saxe and Raymie Stata, "Extended Static Checking for Java", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 234–245, New York, NY, USA, 2002, ACM.

[Flue06] Matthew Fluet, Gregory Morrisett and Amal Ahmed, "Linear Regions Are All You Need", in Peter Sestoft, editor, *Programming Language and Systems: Proceedings of the European Symposium on Programming*, vol. 3924 of *LNCS*, pp. 7–21, Springer, 2006.

[Fost02] Jeffrey S. Foster, *Type qualifiers: lightweight specifications to improve software quality*, Ph.D. thesis, Berkeley, 2002.

[Frig98] Matteo Frigo, Charles E. Leiserson and Keith H. Randall, "The Implementation of the Cilk-5 Multithreaded Language", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 212–223, New York, NY, USA, 1998, ACM Press.

[FUSE] "A filesystem in userspace", `http://fuse.sourceforge.net/`.

[Gay01] David Gay and Alexander Aiken, "Language Support for Regions", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 70–80, New York, NY, USA, 2001, ACM Press.

[Gay03] David Gay, Philip Levis, J. Robert von Behren, Matt Welsh, Eric A. Brewer and David E. Culler, "The nesC Language: A Holistic Approach to Networked Embedded Systems", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 1–11, New York, NY, USA, 2003, ACM.

[Gots07] Alexey Gotsman, Josh Berdine, Byron Cook and Mooly Sagiv, "Thread-modular shape analysis", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 266–277, ACM, 2007.

[Grop96] William Gropp, Ewing L. Lusk, Nathan E. Doss and Anthony Skjellum, "A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard", *Parallel Computing*, vol. 22, no. 6, pp. 789–828, 1996.

[Gros02] Dan Grossman, Gregory Morrisett, Trevor Jim, Michael Hicks, Yanling Wang and James Cheney, "Region-based Memory Management in Cyclone", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 282–293, New York, NY, USA, 2002, ACM Press.

[Gros03] Dan Grossman, "Type-safe Multithreading in Cyclone", in *Proceedings of the ACM SIGPLAN International Workshop on Types in Languages Design and Implementation*, pp. 13–25, New York, NY, USA, 2003, ACM Press.

[Harr03] Timothy L. Harris and Keir Fraser, "Language Support for Lightweight Transactions", in *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pp. 388–402, New York, NY, USA, 2003, ACM Press.

[Harr05] Tim Harris, Simon Marlow, Simon L. Peyton Jones and Maurice Herlihy, "Composable Memory Transactions", in *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 48–60, ACM, 2005.

[Heng01] Fritz Henglein, Henning Makholm and Henning Niss, "A Direct Approach to Control-flow Sensitive Region-based Memory Management", in *Proceedings of the International Conference on Principles and Practice of Declarative Programming*, pp. 175–186, New York, NY, USA, 2001, ACM.

[Henz04] Thomas A. Henzinger, Ranjit Jhala and Rupak Majumdar, "Race Checking by Context Inference", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 1–13, ACM, 2004.

[Herl03] Maurice Herlihy, Victor Luchangco, Mark Moir and William N. Scherer III, "Software transactional memory for dynamic-sized data structures", in *Proceedings of the Symposium on Principles of Distributed Computing*, pp. 92–101, ACM, 2003.

[Hobo08] Aquinas Hobor, Andrew W. Appel and Francesco Zappa Nardelli, "Oracle Semantics for Concurrent Separation Logic", in *Programming Language and Systems: Proceedings of the European Symposium on Programming*, vol. 4960 of *LNCS*, pp. 353–367, Springer, 2008.

[Jim02] Trevor Jim, Greg Morrisett, Dan Grossman, Michael Hicks, James Cheney and Yanling Wan, "Cyclone: A safe dialect of C", in *Usenix Annual Technical Conference*, USENIX Association, 2002.

[Jula08] Horatiu Jula, Daniel Tralamazza, Cristian Zamfir and George Candea, "Deadlock Immunity: Enabling Systems to Defend Against Deadlocks", in Richard Draves and Robbert van Renesse, editors, *Proceedings of the Symposium on Operating Systems Design and Implementation*, pp. 295–308, USENIX Association, 2008.

[Koba06] Naoki Kobayashi, "A New Type System for Deadlock-Free Processes", in C. Baier and H. Hermanns, editors, *International Conference on Concurrency Theory*, vol. 4137 of *LNCS*, pp. 233–247, Springer, 2006.

[MIGR] "A tool that estimates population size and migration rate", `http://popgen.sc.fsu.edu/Migrate/Migrate-n.html`.

[Miln99] Robin Milner, *Communicating and mobile systems - the Pi-calculus*, Cambridge University Press, 1999.

[Necu05] George C. Necula, Jeremy Condit, Matthew Harren, Scott McPeak and Westley Weimer, "CCured: Type-safe Retrofitting of Legacy Software", *ACM Transactions on Programming Languages and Systems*, vol. 27, no. 3, pp. 477–526, 2005.

[NGOR] "A password recovery tool for Oracle Database", `http://code.google.com/p/ngorca/`.

[OCal03] Robert O'Callahan and Jong-Deok Choi, "Hybrid Dynamic Data Race Detection", in *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 167–178, New York, NY, USA, 2003, ACM Press.

[Prat06] Polyvios Pratikakis, Jeffrey S. Foster. and Michael Hicks, "Locksmith: context-sensitive correlation analysis for race detection", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 320–331, New York, NY, USA, 2006, ACM.

[Qade04] Shaz Qadeer and Dinghao Wu, "KISS: Keep It Simple and Sequential", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 14–24, ACM, 2004.

[Qin07] Feng Qin, Joseph Tucek, Yuanyuan Zhou and Jagadeesan Sundaresan, "Rx: Treating bugs as allergies — a safe method to survive software failures", *ACM Transactions on Computer Systems*, vol. 25, no. 3, p. 7/2, 2007.

[Reyn02] John C. Reynolds, "Separation logic: A logic for shared mutable data structures", in *Proceedings of the IEEE Symposium on Logic in Computer Science*, pp. 55–74, IEEE Computer Society Press, 2002.

[Ring05] Michael F. Ringenburg and Dan Grossman, "AtomCaml: first-class atomicity via rollback", in *Proceedings of the ACM SIGPLAN International Conference on Functional Programming*, pp. 92–104, ACM, 2005.

[Sava97] Stefan Savage, Michael Burrows, Greg Nelson, Patrick Sobalvarro and Thomas E. Anderson, "Eraser: A Dynamic Data Race Detector for Multi-Threaded Programs", in *Proceedings of ACM Symposium on Operating Systems Principles*, pp. 27–37, ACM, 1997.

[SSHF] "SSH FileSystem", `http://fuse.sourceforge.net/sshfs.html`.

[Suen08] Kohei Suenaga, "Type-Based Deadlock-Freedom Verification for Non-Block-Structured Lock Primitives and Mutable References", in G. Ramalingam, editor, *Asian Symposium on Programming Languages and Systems*, vol. 5356 of *LNCS*, pp. 155–170, Springer, 2008.

[Tera08] Tachio Terauchi, "Checking race freedom via linear programming", in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 1–10, ACM, 2008.

[Toft94] Mads Tofte and Jean-Pierre Talpin, "Implementation of the Typed Call-by-value λ-calculus using a Stack of Regions", in *Conference Record of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 188–201, New York, NY, USA, 1994, ACM Press.

[Vasc10] Vasco Vasconcelos, Francisco Martin and Tiago Cogumbreiro, "Type Inference for Deadlock Detection in a Multithreaded Polymorphic Typed Assembly Language", in Alastair R. Beresford and Simon Gay, editors, *Proceedings of the Workshop on Programming Language Approaches to Concurrency and Communication-cEntric Software*, vol. 17 of *EPTCS*, pp. 95–109, 2010.

[Voun07] Jan Wen Voung, Ranjit Jhala and Sorin Lerner, "RELAY: static race detection on millions of lines of code", in *Proceedings of the joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pp. 205–214, New York, NY, USA, 2007, ACM.

[Walk01] David Walker and Kevin Watkins, "On Regions and Linear Types", in *Proceedings of the ACM SIGPLAN International Conference on Functional Programming*, pp. 181–192, New York, NY, USA, October 2001, ACM Press.

[Wang08] Yin Wang, Terence Kelly, Manjunath Kudlur, Stéphane Lafortune and Scott Mahlke, "Gadara: Dynamic Deadlock Avoidance for Multithreaded Programs", in Richard Draves and Robbert van Renesse, editors, *Proceedings of the Symposium on Operating Systems Design and Implementation*, pp. 281–294, USENIX Association, 2008.

[Wang09] Yin Wang, Stéphane Lafortune, Terence Kelly, Manjunath Kudlur and Scott Mahlke, "The theory of deadlock avoidance via discrete control", in *Conference Record of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 252–263, New York, NY, USA, 2009, ACM.

[Welc04] Adam Welc, Suresh Jagannathan and Antony L. Hosking, "Transactional Monitors for Concurrent Objects", in *European Conference on Object-Oriented Programming*, pp. 519–542, Springer, 2004.

[Zhao04] Tian Zhao, James Noble and Jan Vitek, "Scoped Types for Real-Time Java", in *Proceedings of the 25th IEEE International Real-Time Systems Symposium*, pp. 241–251, IEEE Computer Society, 2004.

[Zhao08] Tian Zhao, Jason Baker, James Hunt, James Noble and Jan Vitek, "Implicit ownership types for memory management", *Science of Computer Programming*, vol. 71, no. 3, pp. 213–241, 2008.