Μεταγλωττιστές

PCL

Στραϊτούρη Ελένη
03115068

24 Απριλίου 2020

Περιεχόμενα

1 Εισαγωγή

Στόχος της παρούσας εργασίας αποτελεί η υλοποίηση ενός μεταγλωττιστή για τη γλώσσα PCL. Ως γλώσσα υλοποίησης επιλέχθηκε η OCaml (έκδοση 4.02.3), ενώ ακόμα χρησιμοποιήθηκαν τα εργαλεία ocamllex και menhir για τη λεκτική και συντακτική ανάλυση αντίστοιχα. Η ενδιάμεση αναπαράσταση του κώδικα έγινε σε LLVM (έκδοση 9.0.1, OCaml bindings 9.0.0) ενώ ο τελικός κώδικας που παράγεται έχει ελεγχθεί σε αρχιτεκτονική linux x86 64 bits. Σημειώνεται ακόμη ότι η υλοποίηση περιλαμβάνει πραγματικούς αριθμούς, ενώ χάρη στο llvm επιτυγχάνεται σχεδόν αυτόματα η βελτιστοποίηση ενδιάμεσου και τελικού κώδικα και η δέσμευση καταχωρητών. Το τελικό εκτελέσιμο του μεταγλωττιστή δέχεται παραμέτρους και παράγει είτε σε αρχεία είτε στο stdout ενδιάμεσο ή/και τελικό (βελτιστοποιημένο ή μη) κώδικα όπως έχει ορισθεί στην εκφώνηση ανάλογα με τις παραμέτρους που θα δοθούν. Παρακάτω για κάθε τμήμα του μεταγλωττιστή αναφερόμαστε στα αντίστοιχα αρχεία κώδικα και σχολιάζουμε τα βασικότερα σημεία.

2 Λεκτικός Αναλυτής

Το πρώτο στάδιο από το οποίο περνάει ο πηγαίος κώδικας που διαβάζεται είτε από το standard input είτε από αρχείο ανάλογα με τις παραμέτρους που έχουν δοθεί στο μεταγλωττιστή είναι η λεκτική ανάλυση. Οι κανόνες λεκτικής ανάλυσης περιέχονται στο αρχείο `Lexer.mll` και διακρίνονται στους **`lexer`** και **`comment`**. Ο πρώτος περιέχει patterns που αφορούν κανονική ροή εκτέλεσης του κώδικα και ο δεύτερος

καλείται όταν εντοπίζεται τμήμα σχολίων. Παράλληλα, ορίζονται βοηθητικές κανονικές εκφράσεις για την αναγνώριση ονομάτων μεταβλητών, ακεραίων ή πραγματικών αριθμών, κοινών και ειδικών ή κενών χαρακτήρων.

lexer Δεδομένου ότι εκτελείται η ενέργεια της οποίας το pattern είχε το μέγιστο ταίριασμα και σε περίπτωση "ισοπαλίας" το πρώτο που ορίστηκε φροντίζουμε να ορίσουμε πρώτα τις λεκτικές μονάδες που αντιστοιχούν στις λέξεις κλειδιά της PCL, προκειμένου αυτές να μην αναγνωρισθούν ως ονόματα μεταβλητών. Στη συνέχεια ακολουθούν patterns για τις ακέραιες και πραγματικές σταθερές ενώ στις ενέργειες αυτών γίνονται και οι αντίστοιχες μετατροπές τύπων (από συμβολοσειρά σε ακέραιο ή πραγματικό αριθμό αντίστοιχα). Ακολουθούν τα ονόματα , όλοι οι τελεστές και διαχωριστές, οι κενοί χαρακτήρες, σχόλια , οι σταθεροί χαρακτήρες και οι σταθερές συμβολοσειρές. Έχουμε διακρίνει την περίπτωση του χαρακτήρα αλλαγής γραμμής προκειμένου να μετράται σωστά το πλήθος των γραμμών που έχουν διαβαστεί και να είναι διαθέσιμη η θέση κάθε αναγνωρισθείσας λεκτικής μονάδας, έτσι ώστε να διευκολύνεται η εκτύπωση διαγνωστικού μηνύματος σε περίπτωση λεκτικού σφάλματος (π.χ. μη έγκυρου χαρακτήρα). Ακόμα και στην περίπτωση εσφαλμένα εκτεινόμενης σε πολλές γραμμές συμβολοσειράς φροντίζουμε τη σωστή ενημέρωση του πλήθους γραμμών.

comment Εφόσον αναγνωρισθεί η συμβολοσειρά εισαγωγής σχολίων και μέχρι να διαβαστεί η συμβολοσειρά τερματισμού αυτών ακολουθείται ο κανόνας comment, αγνοώντας πρακτικά οποιοδήποτε χαρακτήρα στο ενδιάμεσο. Αντίστοιχα και σε αυτή την περίπτωση σε περίπτωση χαρακτήρα αλλαγής γραμμής πραγματοποιείται ενημέρωση του πλήθους των γραμμών που έχουν διαβαστεί για το λόγο που αναφέραμε παραπάνω.

3 Συντακτικός Αναλυτής

Η αναγνώριση κάθε έγκυρης λεκτικής μονάδας έχει σαν αποτέλεσμα την αντιστοίχισή της στο κατάλληλο token προκειμένου αυτό να διαβασθεί στη συνέχεια από το συντακτικό αναλυτή. Οι οδηγίες για την παραγωγή του συντακτικού αναλυτή περιέχονται στο αρχείο `Parser.mly`,στο οποίο ορίζονται τα απαραίτητα tokens, η προτεραιότητα και προσεταιριστικότητα των τελεστών και η γραμματική της PCL. Οι τύποι των σημασιολογικών τιμών που επιστρέφει ο συντακτικός αναλυτής ορίζονται στα αρχεία `Ast.ml` και `Types.ml`, με βάση τις οποίες δομείται το αφηρημένο συντακτικό δέντρο.

3.1 Αφηρημένο Συντακτικό Δέντρο

Για τη δημιουργία του AST στο `Ast.ml` δημιουργήσαμε μια ιεραρχία τύπων, ικανή να περιγράψει κάθε συστατικό στοιχείο του προγράμματος. Στην κορυφή βρίσκονται οι τρεις βασικότεροι τύποι που είναι οι δηλώσεις `ast_decl`, οι εντολές `ast_stmt` και οι εκφράσεις `ast_expr` [1].

ast_decl Ο αναδρομικός τύπος δηλώσεων περιλαμβάνει δηλώσεις μεταβλητών, ετικετών, παραμέτρων, υποπρογραμμάτων καθώς και τον ορισμό υποπρογραμμάτων. Ο ορισμός ενός υποπρογράμματος αποτελείται από τη δήλωση της επικεφαλίδας του, τη λίστα δηλώσεων εντός της εμβέλειάς του και μία σύνθετη εντολή, την εντολή block η οποία αναδρομικά αποτελείται από τη λίστα των εντολών του υποπρογράμματος. Για τον ορισμό των κατασκευαστών των παραπάνω έγινε χρήση των βοηθητικών τύπων `name`

[1] Η χρήση τύπου Lexing.position γίνεται όπου κρίνεται απαραίτητο για τη βελτίωση των μηνυμάτων σε περίπτωση συντακτικού ή άλλου σφάλματος

όπου απαιτείται όνομα, `typ` για τον καθορισμό τύπων και `pass_mode` (ορισμός στο `Symbol.ml`), για τον τρόπο περάσματος παραμέτρων.

`ast_stmt` Ο επίσης αναδρομικός αυτός τύπος περιλαμβάνει κατασκευαστές για όλες τις εντολές του προγράμματος, όπως και για τη σύνθετη εντολή block, η οποία είναι στην ουσία η λίστα των εντολών. Για τον ορισμό των εντολών χρησιμοποιούνται πέραν του βασικού τύπου `ast_expr` για τις εκφράσεις και του βοηθητικού `name` για τα ονόματα, ο ειδικός βοηθητικός τύπος `l_value` που αναπαριστά l-values.

`ast_expr` Ομοίως ο τύπος των εκφράσεων είναι αναδρομικός, ενώ οι ορισμοί των l-values και των σταθερών εκφράσεων, ακολουθούν τη λογική της εκφώνησης. Όσον αφορά τις υπόλοιπες r-values διακρίναμε εκείνες που προκύπτουν από την εφαρμογή τελεστών κι εκείνες που προκύπτουν από την κλήση συναρτήσεων. Η αναπαράσταση r-values που περιέχουν τόσο τελεστές όσο και κλήσεις συναρτήσεων είναι φυσικά εφικτή χάρη στην αμοιβαία αναδρομική φύση των βασικών και βοηθητικών τύπων. Σχετικά με τους τελεστές η διάκριση γίνεται σε πρώτο επίπεδο ανάλογα με το πλήθος των τελουμένων, με εξαίρεση και ειδική περίπτωση να αποτελεί ο τελεστής διεύθυνσης. Στη συνέχεια, για τους τελεστές 2 τελουμένων γίνεται περαιτέρω διάκριση ανάλογα με το είδος τους (αριθμητικοί, συγκριτικοί, λογικοί, ισότητας).

3.2 Γραμματική

Σε γενικές γραμμές ακολουθήθηκε η λογική της δοθείσας γραμματικής, με πρόσθεση προφανώς κατάλληλων κανόνων για την ερμηνεία των ομάδων συμβόλων που μπορούν να επαναλαμβάνονται καμία, τουλάχιστον μία ή περισσότερες φορές. Για λόγους απόδοσης, έχει προτιμηθεί η αριστερή αναδρομή στους κανόνες όπου επιστρέφονται λίστες σημασιολογικών τιμών.

Συγκρούσεις Για την αποφυγή συγκρούσεων ορίστηκαν σε πρώτη φάση οι προτεραιότητες και προσεταιριστικότητες των τελεστών της γλώσσας (ο ορισμός προτεραιότητας των [] παραλήφθηκε ως περιττός όπως διεμήνυε το αντίστοιχο warning). Δεδομένης της διττής φύσης των τελεστών '+' και '-' , υποδηλώνοντας είτε πράξη, είτε πρόσημο, ορίσαμε δύο ακόμα εικονικά tokens `Un_minus`, `Un_plus` με την κατάλληλη προτεραιότητα και προσεταιριστικότητα, για να διακρίνουμε τις εκφράσεις όπου οι τελεστές αυτοί χρησιμοποιούνται ως πρόσημα. Παρόλο τον ορισμό προτεραιότητας και προσεταιριστικότητας, υπήρξε, πέραν της αναμενόμενης shift/reduce σύγκρουσης λόγω του dangling else, άλλη μία (reduce/reduce) στον κανόνα του τελεστή διεύθυνσης. Συγκεκριμένα σε περίπτωση που είχε διαβαστεί ο τελεστής @ και ακολουθούσε l-value θα μπορούσε να γίνει reduce είτε ο κανόνας @ l-value είτε ο κανόνας των απλών εκφράσεων l-value. Η σύγκρουσή αυτή λύθηκε μετατρέποντας τον κανόνα της διεύθυνσης από @ l-value σε @ expr, μεταφέροντας τον έλεγχο ότι ακολουθεί l-value να γίνεται σε επόμενη φάση. Τέλος το πρόβλημα του dangling else αντιμετωπίστηκε ορίζοντας υψηλότερη προτεραιότητα στο token else και χαμηλότερη στο then, έτσι ώστε το dangling else να αντιστοιχίζεται ρητά στο "πλησιέστερο" if και να αποφευχθεί η σύγκρουση.

4 Σημασιολογικός Αναλυτής

Η έξοδος του συντακτικού αναλυτή αποτελεί το αφηρημένο συντακτικό δέντρο του προγράμματος το οποίο είναι ένα ζεύγος τιμών, της λίστας των δηλώσεων και της σύνθετης εντολής block. Η λίστα δηλώσεων περιέχει και τον ορισμό όλων των υποπρογραμμάτων που καλούνται από το κύριο πρόγραμμα.

Το ζεύγος αυτό δίνεται ως όρισμα στο σημασιολογικό αναλυτή ο οποίος αφού πραγματοποιήσει τους κατάλληλους ελέγχους, καλεί τις απαραίτητες συναρτήσεις για την παραγωγή του ενδιάμεσου κώδικα [2]. Τα κύρια αρχεία υλοποίησης του σημασιολογικού αναλυτή είναι τα `Semantics.ml` και `Sem_expr.ml`. Στο πρώτο περιλαμβάνονται οι συναρτήσεις για τον έλεγχο των δηλώσεων και των ορισμών υποπρογραμμάτων, για τον έλεγχο των εντολών, και του κυρίου προγράμματος. Το δεύτερο περιέχει τις συναρτήσεις για τον έλεγχο των εκφράσεων. Για τον πίνακα συμβόλων έγινε χρήση του βοηθητικού κώδικα και των συναρτήσεων του Bonus_pack (από τη σελίδα του μαθήματος, αρχεία `Symbol.ml`, `Types.ml` `Hashcons.ml`, `Identifier.ml`. `Error.ml`, `extend.ml`) με κατάλληλες μετατροπές κι επεκτάσεις.

4.1 Κύριο Πρόγραμμα

Η βασική συνάρτηση του σημασιολογικού αναλυτή η οποία εκκινεί τη διαδικασία σημασιολογικού ελέγχου (και παραγωγής ενδιάμεσου κώδικα) για το κύριο πρόγραμμα είναι η `sem`, η οποία βρίσκεται στο αρχείο `Semantics.ml`. Ως ορίσματα δέχεται την έξοδο του συντακτικού αναλυτή και τη σημαία βελτιστοποίησης.

Σε πρώτη φάση αφού αρχικοποιήσει τον πίνακα συμβόλων, πραγματοποιεί σημασιολογικό έλεγχο των δηλώσεων επικεφαλίδων των συναρτήσεων και διαδικασιών βιβλιοθήκης, ο οποίος πραγματοποιείται προκειμένου να εισαχθούν οι συναρτήσεις αυτές στον πίνακα συμβόλων. Οι δηλώσεις αυτές είναι διαθέσιμες στο αρχείο `Utils.ml`, το οποίο περιέχει συναρτήσεις και τιμές βοηθητικού χαρακτήρα. Στη συνέχεια αφού δημιουργήσει νέα εμβέλεια (Scope) για το κύριο πρόγραμμα, καλεί τη συνάρτηση σημασιολογικού ελέγχου για όλες τις δηλώσεις και τους ορισμούς υποπρογραμμάτων του κυρίου προγράμματος, όπως επίσης και την αντίστοιχη για τον έλεγχο των εντολών. Αμφότερες οι συναρτήσεις αναλύονται στη συνέχεια. Τέλος, πριν το κλείσιμο της εμβέλειας ελέγχεται ο πίνακας συμβόλων στην τρέχουσα εμβέλεια για δηλωμένες ετικέτες για τις οποίες υπάρχει εντολή goto αλλά δεν έχουν ορισθεί ή για δηλωμένα υποπρογράμματα που επίσης δεν έχουν ορισθεί.

Σημειώνεται ότι τα υποπρογράμματα βιβλιοθήκης δεν δηλώνονται εσωτερικά της εμβέλειας του κυρίου προγράμματος έτσι ώστε να είναι δυνατό να 'επισκιαστούν' από μεταβλητές, παραμέτρους ή υποπρογράμματα του ιδίου ονόματος. Αυτό γιατί εντός της ίδιας εμβέλειας δεν έχουμε επιτρέψει δύο οντότητες ίδιου ή διαφορετικού είδους να έχουν το ίδιο αναγνωριστικό, όπως συμβαίνει και στην Pascal.

4.2 Δηλώσεις & Υποπρογράμματα

Η συνάρτηση `sem_decl` διαχειρίζεται τόσο τις δηλώσεις μεταβλητών , ετικετών και υποπρογραμμάτων όσο και τους ορισμούς των τελευταίων. Ως ορίσματα δέχεται τη σημασιολογική τιμή της δήλωσης ή του ορισμού και δύο βοηθητικά ορίσματα τα οποία έχουν νόημα στη δήλωση και ορισμό υποπρογραμμάτων και θα εξηγηθούν στη συνέχεια.

Μεταβλητές & Ετικέτες Για κάθε δήλωση μεταβλητής αφού ελεγχθεί ότι ο τύπος της είναι πλήρης κι έγκυρος, δημιουργείται και η αντίστοιχη εγγραφή στον πίνακα συμβόλων. Αντίστοιχα, για απλότητα καταχωρούνται και οι ετικέτες ως μεταβλητές στον πίνακα συμβόλων με τη διαφορά ότι δεν έχουν ορισμένο τύπο .

[2] Τα τμήματα κώδικα που βρίσκονται στο σημασιολογικό αναλυτή αλλά αφορούν τον ενδιάμεσο κώδικα εξηγούνται στο επόμενο κεφάλαιο.

Δήλωση Επικεφαλίδας Υποπρογραμμάτων Η δήλωση επικεφαλίδας υποπρογράμματος μπορεί να υποδεικνύει εμπρόσθια δήλωση αυτού, είτε να αποτελεί μέρος του ορισμού του. Στην πρώτη περίπτωση ωστόσο το υποπρόγραμμα πρέπει να δηλωθεί ως forward στον πίνακα συμβόλων. Η διάκριση των περιπτώσεων γίνεται χάρη στο τρίτο όρισμα της `sem_decl`, που αποτελεί flag για το αν πρόκειται για δήλωση ή ορισμό υποπρογράμματος. Σε κάθε περίπτωση, αφού η συνάρτηση προστεθεί στον πίνακα συμβόλων και δημιουργηθεί το Scope του υποπρογράμματος , ελέγχονται και τοποθετούνται στον πίνακα συμβόλων οι τυπικές παράμετροι της επικεφαλίδας. Για να γίνει αυτό είναι απαραίτητη η γνώση της συνάρτησης στην οποία ανήκουν και σε αυτή τη περίπτωση έχει νόημα η δεύτερη παράμετρος `sem_decl`, η οποία χρησιμοποιείται για να είναι διαθέσιμη η πληροφορία αυτή κατά την εισαγωγή των παραμέτρων στο πίνακα συμβόλων.

Ορισμός Υποπρογραμμάτων Κατά τον ορισμό ενός υποπρογράμματος σε πρώτη φάση γίνεται ο σημασιολογικός έλεγχος της επικεφαλίδας με τη μόνη διαφορά ότι εάν πρόκειται για συνάρτηση προστίθεται στον πίνακα συμβόλων η μεταβλητή της τιμής επιστροφής. Στη συνέχεια όπως και στο κύριο πρόγραμμα γίνεται αναδρομικά ο σημασιολογικός έλεγχος των τοπικών δηλώσεων και ορισμών συναρτήσεων ή διαδικασιών και στη συνέχεια του κυρίου σώματος των εντολών. Μετά τον έλεγχο αυτό (και την αντίστοιχη παραγωγή ενδιάμεσου κώδικα), πριν κλείσει η εμβέλεια του υποπρογράμματος γίνεται ο ίδιος έλεγχος για τις ετικέτες και τα υποπρογράμματα με αυτόν του κυρίου προγράμματος.

4.3 Εντολές

Οι εντολές ενός προγράμματος ή υποπρογράμματος αναπαρίστανται ως μία σύνθετη εντολή block που τις περιέχει ως λίστα. Κατά τη σημασιολογική ανάλυση αυτής της εντολής καλείται αναδρομικά ο σημασιολογικός αναλυτής για κάθε εντολή της λίστας. Σε γενικές γραμμές για κάθε εντολή που περιέχει εκφράσεις ελέγχεται αν ισχύουν οι προδιαγραφές που πρέπει να πληρούν αυτές σε κάθε περίπτωση με την ανάλυση να συνεχίζεται αναδρομικά στις περιπτώσεις που οι εντολές περιέχουν εμφωλευμένες άλλες εντολές. Η εντολή κλήσης διαδικασίας έχει την ίδια μεταχείριση με την κλήση συνάρτησης με τη διαφορά του κενού τύπου της τιμής επιστροφής. Για τις εντολές ορισμού ετικέτας και μετάβασης σε αυτή ελέγχεται αν αυτή έχει δηλωθεί προηγουμένως.

Ειδικά για τις εντολές δυναμικής παραχώρησης μνήμης προσθέτουμε στον πίνακα συμβόλων ψευδομεταβλητές με ειδικά αναγνωριστικά , προκειμένου να μην είναι δυνατή αποδέσμευση μη δεσμευμένης μνήμης (σφάλμα σε περίπτωση dispose χωρίς να έχει προηγηθεί new). Συγκεκριμένα στα αναγνωριστικά των ψευδομεταβλητών για νέα αντικείμενα πλήρους τύπου χρησιμοποιείται ο αριθμός 2 και η αναπαράσταση της l-value σε συμβολοσειρά , ενώ για τους πίνακες ο αριθμός 3 και η αναπαράσταση της dereferenced l-value σε συμβολοσειρά. Έτσι, όταν κληθούν οι αντίστοιχες εντολές αποδέσμευσης μνήμης αρκεί μια απλή αναζήτηση στον πίνακα συμβόλων για την ύπαρξη των αντίστοιχων ψευδομεταβλητών. Σημειώνεται ότι με το πρώτο new σε ορισμένη l-value που συναντάται προστίθεται η αντίστοιχη ψευδομεταβλητή στο πίνακα συμβόλων, για κάθε επόμενο new στην ίδια τιμή, εάν δεν έχει μεσολαβήσει dispose δεν επισυμβαίνει κάποια αλλαγή στον πίνακα συμβόλων, ενώ στο πρώτο dispose η εγγραφή της ψευδομεταβλητής αφαιρείται. Ειδικά, για την περίπτωση δυναμικής δέσμευσης πίνακα γίνεται κι ένας τετριμμένος έλεγχος για το εάν η διάσταση που ζητείται είναι θετική σε περίπτωση που αυτή είναι σταθερά.

4.4 Εκφράσεις

Οι συναρτήσεις σημασιολογικής ανάλυσης των εκφράσεων έχουν ως τιμή επιστροφής τον τύπο τους. Οι τύποι αυτοί ορίζονται στο αρχείο `Types.ml`, όπου επίσης ορίζονται και βοηθητικές συναρτήσεις σε

σχέση με αυτούς κατά τη σημασιολογική ανάλυση. Οι συναρτήσεις έχουν ακολουθήσει την ιεραρχία και τη λογική των τύπων εκφράσεων που ορίστηκαν στο `Ast.ml`.

R-values Στις σταθερές εκφράσεις απλά επιστρέφεται ο τύπος τους. Για τις r-values που προκύπτουν από την εφαρμογή τελεστών πλην του τελεστή διεύθυνσης, ελέγχονται αναδρομικά οι τύποι των τελουμένων κι εφόσον αυτοί είναι σωστοί επιστρέφεται ο τύπος της έκφρασης. Στην περίπτωση του τελεστή διεύθυνσης γίνεται έλεγχος αν η έκφραση είναι l-value, (περιορισμός που λόγω σύγκρουσης δε μπορούσε να επιβληθεί από το συντακτικό αναλυτή). Η περίπτωση r-value λόγω κλήσης συνάρτησης αναλύεται παρακάτω.

L-values Τα ονόματα μεταβλητών και παραμέτρων έχουν την ίδια αναπαράσταση. Για την εύρεση του τύπου τους γίνεται με αναζήτηση στον πίνακα συμβόλων κι επιστρέφεται ο τύπος που είχε δηλωθεί κατά τη δήλωσή τους, δηλαδή κατά την εισαγωγή τους σε αυτόν. Αντίστοιχα επιστρέφεται και ο τύπος της l-value result, όπως δηλώθηκε στον πίνακα συμβόλων κατά τον ορισμό του υποπρογράμματος. Ο τύπος σταθερών συμβολοσειρών επιστρέφεται τετριμμένα όπως ορίζει η εκφώνηση, αφού πριν την επιστροφή τους ως σημασιολογικές τιμές οι χαρακτήρες διαφυγής, οι οποίοι διαβάστηκαν ως strings μετατράπηκαν σε κανονικούς χαρακτήρες διαφυγής -το αντίστοιχο για τους σταθερούς χαρακτήρες γίνεται σε επόμενο στάδιο-.Ο τύπος έκφρασης αποδεικτοδότησης επιστρέφεται εφόσον ελεγχθεί ότι ο τελεστής εφαρμόζεται σε έκφραση τύπου δείκτη διάφορη του nil. Τέλος, για τα στοιχεία πινάκων πέραν του ελέγχου ότι η l-value είναι πράγματι πίνακας και η έκφραση θέσης ακέραια, πραγματοποιούμε έναν ακόμη τετριμμένο έλεγχο ότι δε βγαίνουμε εκτός εμβέλειας του πίνακα, εφόσον αυτός είναι στατικά ορισμένος κι εφόσον η έκφραση θέσης του στοιχείου είναι ακέραια σταθερά.

Κλήση Συναρτήσεων Κατά τη κλήση συνάρτησης ελέγχεται σε πρώτη φάση ότι υπάρχει η συνάρτηση που καλείται και σε δεύτερη ότι οι πραγματικές παράμετροι είναι σωστές σε σχέση με τις τυπικές που έχουν δηλωθεί στην επικεφαλίδα της συνάρτησης. Ο έλεγχος αυτός περιλαμβάνει πέραν του ελέγχου του πλήθους και τον έλεγχο της συμβατότητας των τύπων των πραγματικών παραμέτρων με τις τυπικές ανάλογα και με τον τρόπο περάσματός τους. Εφόσον, έχουν επιτύχει όλοι οι έλεγχοι, επιστρέφεται ο τύπος της συνάρτησης, όπως αυτός έχει δηλωθεί στον πίνακα συμβόλων.

5 Ενδιάμεσος Κώδικας

Η παραγωγή ενδιάμεσου κώδικα πραγματοποιείται σχεδόν παράλληλα με τους σημασιολογικούς ελέγχους, αφού μόλις ελεγχθεί επιτυχώς το εκάστοτε στοιχείο του προγράμματος καλείται η συνάρτηση παραγωγής ενδιάμεσου κώδικα για το στοιχείο αυτό. Οι κύριες συναρτήσεις παραγωγής ενδιάμεσου κώδικα Llvm βρίσκονται στα αρχεία `Compile.ml` και `Compile_expr.ml`. Βοηθητικές συναρτήσεις για την παραγωγή ενδιάμεσου κώδικα βρίσκονται ακόμη στα αχεία `Frames.ml`. Πέραν από τις συναρτήσεις παραγωγής ενδιάμεσου κώδικα για όλα τα συστατικά του προγράμματος, θα μας απασχολήσει και η δομή στοίβας `frame_pointers`, η οποία υλοποιεί τη στοίβα του προγράμματος.

5.1 Στοίβα Προγράμματος

Στην υλοποίησή μας, η στοίβα του προγράμματος είναι μία στοίβα από llvm δείκτες σε llvm structs, τα οποία αναπαριστούν τα frames του προγράμματος. Οι δομές αυτές δεσμεύουν μνήμη για όλες τις ονοματισμένες οντότητες που έχουν τιμές -μεταβλητές, παραμέτρους, τιμή επιστροφής συνάρτησης- εντός

μίας ορισμένης εμβέλειας, όπως κι επίσης χώρο για ένα δείκτη στη δομή της αμέσως εξωτερικότερης εμβέλειας. Έτσι, υποστηρίζεται η ύπαρξη εμφωλευμένων εμβελειών, καθώς οι τιμές ποσοτήτων που έχουν οριστεί σε εξωτερικότερο scope είναι διαθέσιμες σε εσωτερικότερα, εφόσον δεν έχουν επανορισθεί.

Frames Η στοίβα `frame_pointers` δημιουργείται στο αρχείο `Compile_expr.ml` και ως πρώτο στοιχείο εισάγεται ο μηδενικός δείκτης από τη συνάρτηση `sem`. Για κάθε νέα εμβέλεια που ορίζεται είτε αυτή αφορά το κύριο πρόγραμμα είτε κάποιο υποπρόγραμμα δημιουργείται ένα νέο frame του οποίου ο δείκτης εισάγεται στη στοίβα. Με το κλείσιμο της εμβέλειας αφαιρείται από τη στοίβα. Για τη δημιουργία ενός νέου frame εντοπίζονται αρχικά όλες οι ορισμένες μεταβλητές, παράμετροι αν πρόκειται για υποπρόγραμμα και η τιμή επιστροφής αν πρόκειται για συνάρτηση, εντός της τρέχουσας εμβέλειας κι επιστρέφεται μία λίστα με τους llvm τύπους τους (συναρτήσεις `find_all_vars` και `find_function_scope` αρχείο `Frames.ml`). Η συνάρτηση `new_frame` δημιουργεί μία llvm struct από τους παραπάνω τύπους και τον τρέχοντα frame_pointer, δηλαδή ένα νέο πλαίσιο και εισάγει το δείκτη του στη στοίβα. Στη περίπτωση που πρόκειται για το αρχικό frame του κυρίου προγράμματος η δομή δηλώνεται ως global ενώ σε περίπτωση υποπρογράμματος ως τοπική μεταβλητή. Εφόσον, πρόκειται για υποπρόγραμμα, είναι απαραίτητο τα πεδία της δομής που έχουν δεσμευθεί για τις παραμέτρους του να δείξουν σε αυτές. Για υποπρογράμματα που ορίζει ο χρήστης πέραν των παραμέτρων που ορίζει ο ίδιος, ορίζουμε ως επιπλέον παράμετρο, το τρέχοντα frame_pointer που θα αναλυθεί εν συνεχεία. Το binding των παραμέτρων στην εμβέλεια που ανήκουν παραγματοποιείται επίσης από την `new_frame`.

Nested Scopes Προκειμένου να ισχύουν οι κανόνες εμβέλειας της ISO Pascal, δεδομένου ότι το Llvm δεν υποστηρίζει εμφωλευμένες εμβέλειες, είναι απαραίτητο οποιαδήποτε ποσότητα που πρέπει να είναι ορατή σε ένα υποπρόγραμμα κι έχει ορισθεί εκτός αυτού, να περνιέται ως παράμετρος. Εξαίρεση αποτελούν τα υποπρογράμματα βιβλιοθήκης των οποίων η υλοποίηση πραγματοποιείται εξωτερικά και δεν χρειάζονται παρά μόνο τις παραμέτρους τους. Για όλα τα υποπρογράμματα που ορίζει ο χρήστης αρκεί να περαστεί μία ακόμα παράμετρος, ο τρέχων frame_pointer, αφού μέσα στη δομή που δείχνει πέραν από τις ορισμένες ποσότητες του αμέσως εξωτερικότερου scope, είναι αποθηκευμένος και ο δείκτης στο προηγούμενο scope και αναδρομικά είναι προσβάσιμες όλες οι ορατές ποσότητες για το υποπρόγραμμα. Επομένως, αρκεί στη δήλωση των παραμέτρων της συνάρτησης Llvm να δηλωθεί ως τελευταία παράμετρος ο frame_pointer που βρίσκεται στην κορυφή της στοίβας [3]. Αντίστοιχα, κατά την παραγωγή κώδικα Llvm για κλήση υποπρογράμματος χρειάζεται να περαστεί ως τελευταία παράμετρος ο frame_pointer που αντιστοιχεί στο scope μέσα στο οποίο ορίστηκε το υποπρόγραμμα. Η πληροφορία για το ζητούμενο scope υπάρχει στην εγγραφή του υποπρογράμματος στον πίνακα συμβόλων, ενώ γνωστό είναι και ποιο είναι το τρέχον scope. Έτσι, εύκολα βρίσκεται το ζητούμενο scope που πρέπει να δοθεί ως παράμετρος.[4]

Αναζήτηση Τιμών στη Στοίβα Σε οποιαδήποτε έκφραση χρειάζεται η τιμή ή η διεύθυνση μιας μεταβλητής, μιας παραμέτρου ή της τιμής επιστροφής μιας συνάρτησης κατά την παραγωγή ενδιάμεσου κώδικα, πρέπει να είναι γνωστός ο δείκτης σε αυτή. Η στοίβα του προγράμματος περιέχει ακριβώς αυτή την πληροφορία, αφού κάθε frame έχει δεσμεύσει τη μνήμη και περιέχει τις τιμές όλων των ονοματισμένων οντοτήτων της εμβέλειάς του. Κάθε τέτοια οντότητα εισάγεται στον πίνακα συμβόλων μαζί με τον αύξοντα αριθμό του scope στο οποίο ανήκει και το offset της στο αντίστοιχο frame. Οι πληροφορίες αυτές αρκούν για να εντοπιστεί στη στοίβα του προγράμματος και να επιστραφεί ο δείκτης της (συνάρτηση `get_val_ptr`, αρχείο `Compile_expr.ml`).

[3] βλ. συνάρτηση `compile_proto` αρχείο `Compile.ml`

[4] βλ. συνάρτηση `compile_expr` αρχείο `Compile_expr.ml`

5.2 Κύριο Πρόγραμμα

Για το Llvm τόσο το κύριο πρόγραμμα όσο και τα υποπρογράμματα αποτελούν συναρτήσεις. Το κύριο πρόγραμμα διακρίνεται αποτελώντας την κύρια συνάρτηση του προγράμματος τύπου void. Το πρώτο βήμα της μεταγλώττισης του κύριου προγράμματος σε ενδιάμεσο κώδικα πραγματοποιείται από τη συνάρτηση `compile_main` (κλήση από τη `sem`, ορισμός στο `Compile.ml`), στην οποία δηλώνεται η main συνάρτηση και δημιουργείται το πρώτο basic block του κώδικα από το οποίο θα αρχίσει να εκτελείται το πρόγραμμα. Ακόμα, εφόσον έχει επιλεγεί να γίνει βελτιστοποίηση αρχικοποιείται κατάλληλα και ο pass manager, με την πρόσθεση περασμάτων βελτιστοποίησης. Δεδομένου ότι όλες οι συναρτήσεις στο Llvm πρέπει να ολοκληρώνονται με την εντολή return η οποία δηλώνει και το τέλος ενός basic block, φροντίζουμε να ορίζουμε μία μοναδική εικονική ετικέτα η οποία δηλώνει το τέλος της συνάρτησης και στην οποία αντιστοιχίζεται το τελευταίο basic block με την εντολή επιστροφής . Εξασφαλίζουμε τη μοναδικότητα της θέτοντας ως αρχικό χαρακτήρα του αναγνωριστικού της το ψηφίο 4. Εκμεταλλευόμενοι την υλοποίηση για τον χειρισμό δηλώσεων και ορισμού ετικετών, δημιουργούμε και τοποθετούμε στη σωστή θέση το basic block με την εντολή επιστροφής. Αφού ολοκληρωθεί η παραγωγή ενδιάμεσου κώδικα ελέγχεται η εγκυρότητα του module, γίνονται οι βελτιστοποιήσεις και τελικά επιστρέφεται το module (συνάρτηση `sem`, αρχείο `Semantics.ml`).

5.3 Υποπρογράμματα

Σε ό,τι αφορά το τμήμα δηλώσεων τα υποπρογράμματα είναι τα μόνα για τα οποία πρέπει να παραχθεί ενδιάμεσος κώδικας. Ομοίως, απαιτούνται ενέργειες κατά τον ορισμό των υποπρογραμμάτων, πέραν από τη μεταγλώττιση των εντολών (συνάρτηση `sem_decl` αρχείο `Semantics.ml`).

Δήλωση επικεφαλίδας Είτε πρόκειται για εμπρόσθια δήλωση υποπρογράμματος, είτε για ορισμό του χρειάζεται η συνάρτηση να δηλωθεί στο Llvm μαζί με τις αντίστοιχες παραμέτρους. Για να μη γίνεται η δήλωση αυτή δύο φορές σε περίπτωση εμπρόσθιας δήλωσης η llvalue της συνάρτησης φυλάσσεται στον πίνακα συμβόλων μαζί με την εγγραφή του υποπρογράμματος. Η δήλωση αυτή πραγματοποιείται από τη συνάρτηση `compile_proto` (αρχείο `Compile.ml`).

Στη συνάρτηση αυτή καθορίζονται και οι τύποι των παραμέτρων που διαφέρουν ανάλογα με το αν πρόκειται για συνάρτηση βιβλιοθήκης ή όχι. Στην πρώτη περίπτωση οι τύποι των παραμέτρων που δηλώνονται είναι ίδιοι με αυτούς που δηλώθηκαν στον πίνακα συμβόλων -προφανώς και οι συμβολικοί τύποι που ορίστηκαν για την κατασκευή του αφηρημένου συντακτικού δέντρου αντιστοιχίζονται σε τύπους του Llvm-. Στη δεύτερη περίπτωση, επειδή για τη μεταβολή μίας τιμής στο llvm χρειάζεται να είναι διαθέσιμος ο δείκτης της, ο τύπος της παραμέτρου ορίζεται ως δείκτης στον τύπο που δηλώθηκε στον πίνακα συμβόλων. Επίσης, για τις ορισμένες από το χρήστη συναρτήσεις, όπως προαναφέραμε, ορίζεται και μία ακόμη τυπική παράμετρος που αντιστοιχεί στον εκάστοτε frame_pointer που βρίσκεται στην κορυφή της στοίβας.

Αφού καθοριστούν οι τύποι και τα ονόματα των τυπικών παραμέτρων δηλώνεται η συνάρτηση. Μένει να καθοριστεί και ο τρόπος περάσματός τους. Στον llvm εφόσον έχουμε περάσει δείκτες, η προκαθορισμένη συμπεριφορά είναι οι τιμές να περνώνται κατ' αναφορά. Για να δηλώσουμε τρόπο περάσματος κατ' αξία πρέπει για την αντίστοιχη παράμετρο να της προσθέσουμε ρητά το attribute "byval", το οποίο και κάνουμε.

Σημειώνεται ακόμη ότι στο πρόγραμμά μας επιτρέπεται να υπάρχουν υποπρογράμματα του ιδίου ονόματος σε διαφορετικά scopes, κάτι που δεν υφίσταται στο Llvm. Προκειμένου να ικανοποιήσουμε τη συγκεκριμένη προδιαγραφή κατά τον ορισμό των συναρτήσεων στο Llvm προσθέτουμε στο τέλος του ονόματος της συνάρτησης τον αύξοντα αριθμό του scope που βρισκόμαστε, έτσι ώστε να εξασφαλίσουμε

τη μοναδικότητα των ονομάτων των συναρτήσεων στο Llvm, επιτρέποντας την υλοποίηση υποπρογραμμάτων ίδίου ονόματος σε διαφορετικά scopes.

Ορισμός Υποπρογράμματος Κατά τον ορισμό ενός υποπρογράμματος, για την επικεφαλίδα γίνονται όλα τα παραπάνω και ακόμη δημιουργείται όπως και για το κύριο πρόγραμμα το basic block που θα φιλοξενήσει τον κώδικα του υποπρογράμματος. Με την llvalue της συνάρτησης, πλέον στον πίνακα συμβόλων, το basic block τοποθετείται στο τέλος του ορισμού της επικεφαλίδας και ο builder τοποθετείται κατάλληλα για τις εντολές που θα ακολουθήσουν. Πριν από τη μεταγλώττιση της επικεφαλίδας φροντίζουμε να κρατήσουμε το τρέχον basic block έτσι ώστε μετά το τέλος του ορισμού της συνάρτησης η μεταγλώττιση να συνεχίσει από το σημείο που έμεινε. Για το block για την εντολή επιστροφής ακολουθείται η ίδια διαδικασία, όπως έγινε και με το κύριο πρόγραμμα, με τη διαφορά ότι εάν πρόκειται για υποπρόγραμμα συναρτηση, φορτώνεται κι επιστρέφεται η τιμή της l-value result, ενώ ο builder τοποθετείται στο τέλος του block που ήταν το τρέχον αμέσως πριν από τον ορισμό της συνάρτησης.

5.4 Εντολές

Κάθε επιτυχής σημασιολογικός έλεγχος του κάθε συστατικού μιας εντολής ακολουθείται από την κλήση της αντίστοιχης συνάρτησης ενδιάμεσου κώδικα. . Η εντολή ανάθεσης αποτελεί απλά την αποθήκευση της μεταγλωττισμένης τιμής της r-value στη θέση που δείχνει η l-value αφού φυσικά η πρώτη μετατραπεί στον κατάλληλο τύπο εφόσον χρειάζεται. Η εντολή επιστροφής ισοδυναμεί με μία εντολή goto στην εικονική ετικέτα που προσθέσαμε η οποία αντιστοιχίζεται στο basic block με την εντολή επιστροφής, ενώ η κλήση διαδικασίας έχει την ίδια μεταχείριση με την κλήση συνάρτησης (βλ.επόμενη υποενότητα). Κατά τη δυναμική δέσμευση μνήμης για l-value που είναι δείκτης είτε σε πλήρη τύπο είτε σε πίνακα, δεσμεύουμε με malloc τη μνήμη του νέου αντικειμένου και θέτουμε το δείκτη να δείχνει σε αυτή. Στην περίπτωση πίνακα πραγματοποιούμε κι έναν τετριμμένο έλεγχο ότι η διάσταση είναι θετική εφόσον αυτή αποτελεί μια έκφραση σταθερών. Κατά την αποδέσμευση μετά το free θέτουμε το δείκτη σε null και στις δύο περιπτώσεις .Περισσότερο ενδιαφέρον παρουσιάζουν οι εντολές ελέγχου ροής που αναλύονται στη συνέχεια.

5.4.1 Έλεγχος ροής

`If` Για τη μεταγλώττιση της if then else [5] δημιουργούνται σε πρώτη φάση τέσσερα basic blocks, ένα για τη συνθήκη, ένα για τις εντολές που ακολουθούν το then, ένα για τις εντολές που ακολουθούν το else κι ένα για τον υπόλοιπο κώδικα που ακολουθεί την εντολή. Για την αποφυγή λαθών φροντίζουμε να τερματίζουμε το basic block πριν από την if με άλμα στο basic block για το άλμα υπό συνθήκη της if. Επίσης, δεδομένου ότι ενδέχεται οι εντολές που ακολουθούν τόσο το then όσο και το else, να περιέχουν εκ νέου if then else ή εντολές που θα έχουν αλλάξει το τρέχον basic block φροντίζουμε με τη μεταγλώττιση των εντολών κάθε κλάδου να τερματίζουμε το basic block με εντολή άλματος στο block που ορίσαμε για τις εντολές που ακολουθούν. Η μεταγλώττιση του απλού if είναι ίδια με τη διαφορά ότι το basic block που προορίζεται για το else παραμένει κενό.

`While` Σε αντίστοιχη λογική με την παραπάνω για τη μεταγλώττιση της εντολής while σε ενδιάμεσο κώδικα δημιουργούμε τρία basic blocks, ένα για τον έλεγχο της συνθήκης, ένα για το βρόχο επανάληψης κι ένα για τις εντολές που ακολουθούν [6]. Όπως κι πριν φροντίζουμε να τερματίσουμε το τρέχον basic block με άλμα στο basic block για τη συνθήκη. Σε αυτό εκτελείται άλμα στον κύριο βρόχο εάν ισχύει η

[5] συναρτήσεις `compile_if_then`, `compile_else`, `compile_merge`, αρχείο `Compile_expr.ml`

[6] συναρτήσεις `compile_while`, `compile_while_end`, αρχείο `Compile.ml`

συνθήκη αλλιώς στο basic block που έχει οριστεί για τις επόμενες εντολές. Στη συνέχεια μεταγλωττίζονται οι εντολές εντός του βρόχου επανάληψης και στο τέλος τους προστίθεται εντολή άλματος στο basic block ελέγχου της συνθήκης, προκειμένου αυτή να ελέγχεται μετά από κάθε επανάληψη.

Goto Η ετικέτα στην οποία πρέπει να μεταβεί η ροή του προγράμματος μπορεί να οριστεί πριν ή μετά την εντολή goto. Σε κάθε περίπτωση στον ορισμό της ετικέτας θα πρέπει να ορισθεί νέο basic block. Για τη διάκριση των δύο περιπτώσεων και τον κατάλληλο χειρισμό τους χρησιμοποιούμε τρεις διαφορετικούς κατασκευαστές του τύπου `llvm_val` που δημιουργήσαμε για τις llvalues που χρειάζεται να αποθηκευτούν βοηθητικά στον πίνακα συμβόλων. Αυτοί είναι οι LL_dummy, LL και LLB με τους δύο τελευταίους να χρειάζονται όρισμα llvalue. Με τη δήλωση της ετικέτας αποθηκεύεται στον πίνακα συμβόλων LL_dummy, ενώ σε κάθε περίπτωση δημιουργείται από τη συνάρτηση μεταγλώττισης της goto το block στο οποίο θα συνεχίσει η παραγωγή ενδιάμεσου κώδικα με τον builder να τοποθετείται κατάλληλα μετά τη πέρας της μεταγλώττισης της εντολής.

Σε περίπτωση που συναντάται πρώτα η goto, οπότε και στον πίνακα συμβόλων στην εγγραφή της ετικέτας θα υπάρχει LL_dummy, η ίδια φροντίζει να δημιουργήσει το basic block της ετικέτας και να το αποθηκεύσει ώστε και να είναι ορισμένο για να δημιουργηθεί έγκυρη για το llvm εντολή άλματος προς αυτό, αλλά και να είναι διαθέσιμο στη συνέχεια όταν πραγματικά οριστεί η ετικέτα. Το block τρέπεται σε llvalue και αποθηκεύεται ως όρισμα του κατασκευαστή LL. Όταν βρεθεί και ο ορισμός της ετικέτας κι εντοπιστεί κατασκευαστής LL στο πίνακα συμβόλων φροντίζουμε να μετακινηθεί το ήδη ορισθέν block στην κατάλληλη θέση κι από εκεί να συνεχίζει η μεταγλώττιση της εντολής της ετικέτας. Ακόμη με τον ορισμό της ετικέτας αποθηκεύεται στον πίνακα συμβόλων ο κατασκευαστής LLB (με το κατάλληλο όρισμα), που συμβολίζει ότι η ετικέτα ορίστηκε.

Στην απλή περίπτωση που πρώτα ορίζεται η ετικέτα αποθηκεύεται κατευθείαν στον πίνακα συμβόλων LLB με όρισμα την llvalue του block. Κατά τη μεταγλώττιση της εντολής goto που ακολουθεί λαμβάνεται η πληροφορία αυτή από τον πίνακα συμβόλων και προστίθεται η εντολή άλματος στο block αυτό. Βασικές συναρτήσεις για την υλοποίηση των παραπάνω αποτελούν οι `compile_goto`, `compile_label` και `compile_label_after` (αρχείο `Compile.ml`).

5.5 Εκφράσεις

Η μεταγλώττιση των εκφράσεων σε ενδιάμεσο κώδικα έχει ακολουθήσει κι αυτή λογική της ιεραρχίας των τύπων του `Ast.ml` και υλοποιείται από τις συναρτήσεις του αρχείου `Compile_expr.ml` . Η μεταγλώττιση των σταθερών γίνεται με τη δημιουργία της αντίστοιχης σταθεράς Llvm, εφόσον γίνουν οι κατάλληλες μετατροπές τύπων όπου χρειάζεται. Για την περίπτωση της σταθεράς χαρακτήρα οι χαρακτήρες διαφυγής που έχουν διαβαστεί ως συμβολοσειρές μετατρέπονται στους κανονικούς χαρακτήρες διαφυγής.

5.5.1 L-values

Η γενικότερη λογική της μεταγλώττισης l-values περιλαμβάνει load της τιμής που ζητάται αφού βρεθεί ο δείκτης της (συνάρτηση `get_val_ptr` αρχείο `Compile_expr.ml`). Η εύρεση του δείκτη για τις περιπτώσεις μεταβλητής, παραμέτρου ή τιμής επιστροφής συνάρτησης όπως έχει ήδη αναλυθεί παραπάνω, πραγματοποιείται χάρη στη στοίβα, γνώσει του αριθμού του scope ορισμού και του offset εσωτερικά του frame που αποθηκεύονται στον πίνακα τιμών. Όσον αφορά τα στοιχεία πίνακα, αφού βρεθεί αναδρομικά ο δείκτης του πίνακα, βρίσκεται η llvalue της έκφρασης του δείκτη του στοιχείου , τρέπεται στον κατάλληλο τύπο κι επιστρέφεται ο δείκτης του στοιχείου. Για την αποδεικτοδότηση η έκφραση αποτελεί ήδη δείκτη

στο στοιχείο οπότε απλά επιστρέφεται η llvalue τιμή της. Τέλος, για τις σταθερές συμβολοσειρές απαιτείται να δεσμευθεί χώρος πίνακα για να αποθηκευτούν, του οποίου πίνακα επιστρέφεται ο δείκτης.

5.5.2 R-values

Εκφράσεις με τελεστές Στην περίπτωση των εκφράσεων αριθμητικών και λογικών τελεστών ενός τελουμένου η μεταγλώττιση γίνεται τετριμμένα, ενώ για τον τελεστή διεύθυνσης επιστρέφεται απλά ο δείκτης της έκφρασης. Στην περίπτωση εκφράσεων με τελεστές δύο τελουμένων, στις περιπτώσεις αριθμητικών, συγκριτικών και τελεστών ισότητας, παράγονται οι αντίστοιχες Llvm εντολές αποτίμησης εκφράσεων, εφόσον γίνουν οι κατάλληλες μετατροπές τύπων, όπου χρειάζεται.

Αναφορικά με τις εκφράσεις με λογικούς τελεστές δύο τελουμένων, προκειμένου να υλοποιήσουμε αποτίμηση με βραχυκύκλωση, εκμεταλλευόμαστε τις συναρτήσεις για τη μεταγλώττιση της if then else, ενώ δημιουργούμε και μια τοπική μεταβλητή για την αποθήκευση του αποτελέσματος της έκφρασης. Συγκεκριμένα, για την αποτίμηση του and αν το πρώτο τελούμενο είναι αληθές (συνθήκη if) τότε στο δείκτη του αποτελέσματος της έκφρασης αποθηκεύεται η τιμή αποτίμησης και των δύο τελουμένων (εντολές εντός then block), ενώ αν είναι ψευδές η τιμή false (εντολές εντός else block). Τέλος στο merge block φορτώνεται η τιμή αποτελέσματος η οποία κι επιστρέφεται. Αντίστοιχα, για τον τελεστή or ακολουθείται ακριβώς η ίδια λογική, με την αποθήκευση της τιμής true στην τιμή αποτελέσματος εάν το πρώτο τελούμενο είναι αληθές (then block), και με την αποθήκευση της τιμής αποτίμησης όλης της έκφρασης εάν είναι ψευδές (else block).

Κλήση υποπρογραμμάτων Η μεταγλώττιση κλήσης υποπρογραμμάτων σε ενδιάμεσο κώδικα απαιτεί το σωστό πέρασμα παραμέτρων (συνάρτηση `compile_expr` αρχείο `Compile_expr.ml`). Για την υλοποίηση εμφωλευμένων εμβελειών, όπως ήδη αναφέρθηκε, ως παράμετρος οφείλει να περαστεί -εφόσον δεν πρόκειται για συνάρτηση βιβλιοθήκης- και ο δείκτης του frame που αντιστοιχεί στο scope που ήταν στην κορυφή της στοίβας όταν ορίστηκε το υποπρόγραμμα. Έτσι, σε πρώτη φάση, πραγματοποιείται αναζήτηση κι εύρεση αυτού. Στη συνέχεια, προστίθεται στη λίστα των παραμέτρων, οι οποίες έχουν μετατραπεί στο κατάλληλο llvalue το οποίο είτε είναι δείκτης σε αυτές (υποπρογράμματα χρήστη), είτε η τιμή τους (υποπρογράμματα βιβλιοθήκης). Ειδική περίπτωση αποτελούν τα ορίσματα τύπου πίνακα, τα οποία σε κάθε περίπτωση πρέπει να μετατραπούν σε τύπο πίνακα αγνώστου μεγέθους. Στη συνέχεια, διατρέχονται άλλη μία φορά οι πραγματικές παράμετροι και σε περίπτωση που κάποια είναι διαφορετικού τύπου από την τυπική μετατρέπεται κατάλληλα (ο έλεγχος για συμβατότητα έχει γίνει σε προηγούμενο στάδιο) και τέλος πραγματοποιείται η κλήση.

6 Τελικός Κώδικας

Εφόσον, ολοκληρωθεί επιτυχώς η παραγωγή ενδιάμεσου κώδικα, επιστρέφεται το παραχθέν module και σε περίπτωση που έχει επιλεχθεί να τυπωθεί μόνο αυτό στο standard output, τρέπεται σε string και τυπώνεται (συνάρτηση `print_intermediate` ,αρχείο `Main.ml`). Σε κάθε άλλη περίπτωση, προσδιορίζεται η αρχιτεκτονική (λειτουργικό, επεξεργαστής, μέγεθος λέξης) για την οποία προορίζεται ο τελικός κώδικας και τίθεται το επίπεδο των βελτιστοποιήσεων τελικού κώδικα αναλόγως (-O3 αν έχουν ζητηθεί -O0 αλλιώς). Τέλος, ο τελικός κώδικας παράγεται είτε απευθείας σε αρχείο, είτε σε memory buffer προκειμένου να τραπεί στη συνέχεια σε string και να τυπωθεί στο standard output, ανάλογα με το τι έχει ζητηθεί. Για την παραγωγή του τελικού εκτελέσιμου απαιτείται η σύνδεση του τελικού κώδικα με την υλοποίηση των συναρτήσεων βιβλιοθήκης (αρχείο `lib.a` παραχθέν από την εκτέλεση του `libs.sh` στο runtime library repo) μέσω clang (δοκιμασμένες έκδοσεις 9.0.1 και 8.0.1).