# ”Grace” Language Compiler Project for Compilers Course in DIT at UOA 2017

Μάνος Πιτσικάλης
sdi1300143@di.uoa.gr

## Implemantation

Η υλοποιήση έγινε σε γλώσσα java σε σύστημα gnu/linux.

## Execution

- Για την μεταγλώττιση του μεταγλωττιστή χρησιμοποιείται η εντολή:
  *mvn clean package*

- Για την μεταγλωττίση ενός αρχείου grace χρησιμοποιουνται οι ακόλουθες εντολες:
  *$java -jar compiler-1.0-SNAPSHOT.jar agraceprogr.grc/grace [-O]*
  *$gcc -m32 agraceprogr.s lib.s -o agraceprogr*
  *$./agraceprogr*

  Το αρχείο ”lib.s” περιέχει τις συναρτήσεις της βιβλιοθήκης grace.

  Η επιλογή -O επιλέγεται για βελτιστοποίηση (beta).

  Σημείωση: Υπάρχει και το αρχείο grccomp.sh το οποίο κάνει τα παραπάνω βήματα και τρέχει με τον εξης τρόπο:
  *$./grcccomp.sh agraceprogr.grc lib.s*
  *$./agraceprogr*

## Lexical analysis, Parser, AST

Για την λεκτική ανάλυση και τον συντακτικό αναλυτή χρησιμοποιήθηκε το SableCC. Η γραμματική προκυπτεί με τις καταλληλες μετατροπές για να μην ειναι διφορούμενη και βρίσκεται μάζι με τα tokens στο αρχείο parser.grammar μέσα στον φάκελο ”src/main/sablecc”. Επιπλέον πάλι με την βοήθεια του SableCC μετασχηματιστηκε το CST σε AST για μεγαλύτερη ευκολία στην σημασιολογική ανάλυση.

# Semantic Analysis

- **SymbolTable**
  Για την σημασιολογική ανάλυση υλοποιήθηκε το SymbolTable συμφωνά με την τρίτη υλοποιήση των διαφανειών (hashtable με ids ως key και value την αρχή της λίστας, που βρισκέται σε στοίβα με παρομοιες λίστες για αλλα id, με την τελευταία εμφάνιση του συγκεκριμένου id καθως επίσης και τη στοίβα με τους δείκτες βαθών )

- **Πρόσβαση σε στοιχεία του δέντρου**
  Για την πρόσβαση σε στοιχεία του δέντρου χρησιμοποιείται μια στοίβα τύπων η οποία γεμίζει κατά την επίσκεψη ενός κατάλληλου κόμβου ( πχ id ) και αδειάζει όταν σε ένα κόμβο με μικρότερο βάθος (πχ addition) χρειαστεί πληροφορία για τον κατώτερο κόμβο, και αντίστοιχα θα προσθέσει τύπο στην στοίβα (πχ σε addition θα προσθέσει ένα κόμβο τύπου int-const).

# Intermediate Code

Ο ενδιάμεσος κώδικας έχει την μορφή των τετράδων οπώς αυτή υπάρχει στις διαφανείες. Παράγεται κάτα την διάρκεια της σημασιολογικής ανάλυσης και απόθηκευται στην μνήμη, σε περίπτωση κάποιου σημασιολογικού λάθους σταματάει η παραγωγή ενδιαμεσού κώδικα, εκτυπώνεται μήνυμα λάθους και σταματάει η μεταγλώττιση.

# Assembly

Η assembly στην οποία παράγονται οι εντολές είναι 8086, και παράγεται στο τέλος του σημασιολογικού ελεγχού μίας ολοκληρωμένης συνάρτησης (στο out του function definition). Αν όλος ο κώδικας είναι σημασιολογικά ορθός τότε παράγεται ένα αρχείο με ίδιο όνομα χώρις την κατάληξη "grace"/"grc" άλλα με κατάληξη ".s". Από αυτό το αρχείο σε συνδιασμό με το αρχείο lib.s αν υπαρχούν στο αρχείο συναρτήσεις της βιβλιοθήκης grace μπορεί να παραχθεί ένα εκτελέσιμο.

# Optimization

Για την βελτιστοποιήση ο συγκεκριμένος μεταγλωττιστής δεν κάνει πολλά. Αυτό που κάνει είναι τοπική βελτιστοποιήση σε βασικά μπλοκ εφαρμόζοντας τις τεχνικές copy propagation, common subexpression elimination και algebraic simplification.

# Execution examples

Για ευκολία υπάρχει το script grccomp.sh το οποίο παράγει το εκτελέσιμο με το ίδιο όνομα χωρίς κατάληξη.

- bsort.grace
  *$ ./grccomp.sh bsort.grace lib.s*
  *$ ./bsort*
  *Initial array: 35, 67, 8, 6, 36, 6, 38, 80, 78, 7, 78, 9, 51, 49, 79, 49*
  *Sorted array: 6, 6, 7, 8, 9, 35, 36, 38, 49, 49, 51, 67, 78, 78, 79, 80*

- hanoi.grace
  *$ ./grccomp.sh hanoi.grace lib.s*
  *$ ./hanoi*
  *Rings: 3*
  *Moving from left to right.*
  *Moving from left to middle.*
  *Moving from right to middle.*
  *Moving from left to right.*
  *Moving from middle to left.*
  *Moving from middle to right.*
  *Moving from left to right.*

- hello.grace
  *$ ./grccomp.sh hello.grace lib.s*
  *$ ./hello*
  *Hello world!*

- primes.grace
  *$ ./grccomp.sh primes.grace lib.s*
  *$ ./primes*
  *Limit: 30*
  *Primes:*
  *2*
  *3*
  *5*
  *7*
  *11*
  *13*
  *17*
  *19*
  *23*
  *29*

*Total: 10*

- reverse.grace
  *$ ./grccomp.sh reverse.grace lib.s*
  *$ ./reverse*
  *Hello world!*