Type Systems

Iasonas Nikolaou

October 2021

Εισαγωγή

Στην εργασία αυτή θα ορίσουμε ένα type system για μια γλώσσα χαμηλού επιπέδου που περιγράφει την λειτουργία μιας μηχανής στοίβας (stack machine). Τα προγράμματα p τής γλώσσας ορίζονται από την ακόλουθη γραμματική:

$$p = n \mid true \mid false \mid + \mid - \mid * \mid / \mid < \mid = \mid and \mid not$$
$$\mid nop \mid dup \mid pop \mid swap \mid swap2 \mid p_1 p_2 \mid cond\ [p1|p2] \mid loop\ [p]$$

Η λειτουργική σημασιολογία τής γλώσσας παρατίθεται στην εκφώνηση τής άσκησης. Η εκτέλεση ενός προγράμματος αρχίζει με κενή στοίβα ($\emptyset$) και τερματίζεται όταν το πρόγραμμα γίνει ίσο με nop. Η εκτέλεση "κολλάει" αν τα τελούμενα στην κορυφή τής στοίβας είναι διαφορετικού είδους από αυτά που απαιτεί μία εντολή, ή αν η εντολή απαιτεί τελούμενα αλλά η στοίβα είναι άδεια από αυτά που απαιτεί μία εντολή, ή αν η εντολή απαιτεί τελούμενα αλλά η στοίβα είναι άδεια.
Στόχος μας είναι να ορίσουμε ένα σύστημα τύπων για την παραπάνω γλώσσα, τέτοιο ώστε τα προγράμματα που δεν έχουν σφάλματα τύπων να μην είναι δυνατό να κολλήσουν κατά την εκτέλεση (με μοναδική εξαίρεση τη διαίρεση με το μηδέν).
Η μορφή τής σχέσης τύπων που θα χρησιμοποιήσουμε είναι η ακόλουθη:

$$\frac{p; \sigma : \tau_1}{p'; \sigma' : \tau_2}.$$

Δηλαδή αν έχουμε το πρόγραμμα p με στοίβα σ τύπου τ_1, τότε για το πρόγραμμα p' η στοίβα σ' έχει τύπο τ_2. Όταν το πρόγραμμα p' είναι κενό, τότε παραλείπεται και γράφουμε:

$$\frac{p; \sigma : \tau_1}{\sigma' : \tau_2}.$$

Κανόνες

Ορίζουμε τύπους Int και $Bool$ για τις σταθερές. Επιπλέον, ορίζουμε τον τύπο $Unit$ που θα μας χρησιμεύσει στον ακόλουθο ορισμό της στοίβας. Ο (αναδρομικός) ορισμός της στοίβας είναι:

$$Stack = \mu\alpha.\, Unit + Int \times Stack + Bool \times Stack$$

Ακολουθούν οι κανόνες:
Εντολές προσθήκης στην στοίβα

- $n : Int \quad true : Bool \quad false : Bool$
- $n; \sigma \to \sigma \cdot n$:

$$\frac{n; \sigma : \tau}{\sigma \cdot n : Int \times \tau}$$

- $true; \sigma \rightarrow \sigma \cdot true$:

$$\frac{true; \sigma : \tau}{\sigma \cdot true : Bool \times \tau}$$

- $false; \sigma \rightarrow \sigma \cdot false$:

$$\frac{false; \sigma : \tau}{\sigma \cdot false : Bool \times \tau}$$

Αριθμητικές εντολές

- $+; \sigma \cdot n_1 \cdot n_2 \rightarrow \sigma \cdot (n_1 + n_2)$:

$$\frac{+; \sigma \cdot n_1 \cdot n_2 : Int \times Int \times \tau}{\sigma \cdot (n_1 + n_2) : Int \times \tau}$$

- $*; \sigma \cdot n_1 \cdot n_2 \rightarrow \sigma \cdot (n_1 * n_2)$:

$$\frac{*; \sigma \cdot n_1 \cdot n_2 : Int \times Int \times \tau}{\sigma \cdot (n_1 * n_2) : Int \times \tau}$$

- $-; \sigma \cdot n \rightarrow \sigma \cdot (-n)$:

$$\frac{-; \sigma \cdot n : Int \times \tau}{\sigma \cdot (-n) : Int \times \tau}$$

- $\backslash; \sigma \cdot n_1 \cdot n_2 \rightarrow \sigma \cdot q \cdot r$:

$$\frac{+; \sigma \cdot n_1 \cdot n_2 : Int \times Int \times \tau}{\sigma \cdot q \cdot r : Int \times Int \times \tau}$$

Λογικές εντολές

- $and; \sigma \cdot b_1 \cdot b_2 \rightarrow \sigma \cdot (b_1 \wedge b_2)$:

$$\frac{and; \sigma \cdot b_1 \cdot b_2 : Bool \times Bool \times \tau}{\sigma \cdot (b1 \wedge b_2) : Bool \times \tau}$$

- $not; \sigma \cdot b \rightarrow \sigma \cdot (\neg b)$:

$$\frac{not; \sigma \cdot b : Bool \times \tau}{\sigma \cdot (\neg b) : Bool \times \tau}$$

Εντολές σύγκρισης

- $<; \sigma \cdot n_1 \cdot n_2 \rightarrow \sigma \cdot (n_1 < n_2)$:

$$\frac{<; \sigma \cdot n_1 \cdot n_2 : Int \times Int \times \tau}{\sigma \cdot (n_1 < n_2) : Bool \times \tau}$$

- $=; \sigma \cdot n_1 \cdot n_2 \rightarrow \sigma \cdot (n_1 = n_2)$:

$$\frac{=; \sigma \cdot n_1 \cdot n_2 : Int \times Int \times \tau}{\sigma \cdot (n_1 = n_2) : Bool \times \tau}$$

Εντολές nop, dup, pop, swap, swap2

- $nop; \sigma \to \sigma$:

$$\frac{nop; \sigma : \tau}{\sigma : \tau}$$

- $dup; \sigma \cdot v \to \sigma \cdot v \cdot v$:

$$\frac{dup; \sigma \cdot v : \tau_1 \times \tau}{\sigma \cdot v \cdot v : \tau_1 \times \tau_1 \times \tau}$$

- $swap; \sigma \cdot v_1 \cdot v_2 \to \sigma \cdot v_2 \cdot v_1$:

$$\frac{swap; \sigma \cdot v_1 \cdot v_2 : \tau_1 \times \tau_2 \times \tau}{\sigma \cdot v_2 \cdot v_1 : \tau_2 \times \tau_1 \times \tau}$$

- $swap2; \sigma \cdot v_1 \cdot v_2 \cdot v_3 \to \sigma \cdot v_3 \cdot v_1 \cdot v_2$:

$$\frac{swap2; \sigma \cdot v_1 \cdot v_2 \cdot v_3 : \tau_1 \times \tau_2 \times \tau_3 \times \tau}{\sigma \cdot v_3 \cdot v_1 \cdot v_2 : \tau_3 \times \tau_1 \times \tau_3 \times \tau}$$

Επιπλέον έχουμε τον παρακάτω κανόνα, ο οποίος καθορίζει την φορά εκτέλεσης τού προγράμματος - από αριστερά προς τα δεξιά.

- $\frac{p_1; \sigma \to p_1'; \sigma'}{p_1 p_2; \sigma \to p_1' p_2; \sigma'}$:

$$\frac{p_1; \sigma : \tau \to p_1'; \sigma' : \tau'}{p_1 p_2; \sigma : \tau \to p_1' p_2; \sigma' : \tau'}$$

Εντολές cond, loop
Για τις εντολές αυτές θα είμαστε συντηρητικοί.
Στην *cond* $[p_1|p_2]$ (if condition) θα θεωρήσουμε ότι πρέπει και οι δύο κλάδοι p_1 και p_2 να τρέχουν χωρίς πρόβλημα ξεκινώντας από τον ίδιο τύπο στοίβας και να τερματίζουν σε κοινό τύπο στοίβας. Θα μπορούσαμε να μην εφαρμόσουμε αυτήν την πολιτική, αλλά τότε θα ήταν δυσκολότερο να εγγυηθούμε την ασφάλεια τού προγράμματος.
Στην *loop* $[p]$ θα θεωρήσουμε ότι πρέπει το σώμα τής επανάληψηςp, όταν εκτελεστεί με αρχική στοίβα $\sigma : \tau$ να δίνει στοίβα $\sigma' : Bool \times \tau$. Δηλαδή κάθε επανάληψη θα διατηρεί σταθερό τον τύπο τής στοίβας και στην κορυφή θα έχουμε boolean value, ωστέ να μπορεί να εκτελεστεί και η επόμενη (πιθανή) επανάληψη. Βεβαίως, και πάλι, αποκλείουμε προγράμματα που θα εκτελούνταν χωρίς σφάλμα, για να εγγυηθούμε μέσω τού συστήματος τύπων την ασφάλεια τού προγράμματος.

- $cond\ [p_1|p_2]; \sigma \cdot b$:

$$\frac{cond\ [p_1|p_2]; \sigma \cdot b : Bool \times \tau, \;\; p_1; \sigma : \tau \to^* \sigma_1 : \tau', \;\; p_2; \sigma : \tau \to^* \sigma_2 : \tau'}{\sigma' : \tau'}$$

- $loop\ [p]; \sigma \cdot b$:

$$\frac{loop\ [p]; \sigma \cdot b : Bool \times \tau, \quad p; \sigma : \tau \to^* \sigma' : Bool \times \tau}{\sigma' : \tau}$$

Με $\to^*$ συμβολίζουμε το reflexive transitive closure τού $\to$.

Εφαρμογή

Θα εφαρμόσουμε τους προηγούμενους κανόνες για να κάνουμε type check το πρόγραμμα που δίνεται στην εκφώνηση τής άσκησης ως παράδειγμα:
p = 1 3 true loop [dup 2 $<$ cond [false | dup 1 $-$ $+$ swap2 $*$ swap true]] pop dup 1 $+$ $*$
Αρχικά, από το 1 3 true υπολογίζουμε τύπο στοίβας $Bool \times Int^2 = Bool \times Int \times Int$.
Τώρα πρέπει να ελέγξουμε ότι το σώμα τού loop δίνει τύπο: $Bool \times Int^2$ ξεκινώντας με στοίβα τύπου Int^2:
dup 2 $<$ cond [false | dup 1 $-$ $+$ swap2 $*$ swap true
Από το dup 2 $<$ υπολογίζουμε τύπο: $Bool \times Int^2$
Τώρα για να βρούμε τον τύπο τού $cond[p_1|p_2]$ πρέπει να ελέγξουμε ότι οι τύποι των p_1 και p_2 ξεκινώντας από στοίβα τύπου Int^2 δίνουν στοίβα με κοινό τύπο.
Το p_1 = false δίνει στοίβα τύπου $Bool \times Int^2$
Το p_2 = dup 1 $-$ $+$ swap2 $*$ swap true δίνει επίσης $Bool \times Int^2$.
Πράγματι, λοιπόν, έχουμε ταύτιση των τύπων, αφού και οι δύο ροές δίνουν $Bool \times Int^2$. Επομένως, το $cond[p_1|p_2]$ δίνει τύπο $Bool \times Int^2$.
Άρα, το σώμα τού loop δίνει τύπο $Bool \times Int^2$.
Το loop δίνει τύπο στοίβας Int^2.
Τέλος, το pop dup 1 $+$ $*$ ξεκινώντας με τύπο Int^2 δίνει στοίβα με τύπο Int, όπως αναμέναμε αφού το αποτέλεσμα τής εκτέλεσης τού προγράμματος είναι ο αριθμός 42 στο stack.
Εφόσον, βρήκαμε τον τύπο τής στοίβας, έχουμε αποδείξει ότι το πρόγραμμα θα τερματίσει!