Γλώσσες Προγραμματισμού 2
'Ασκηση 7

'Αγγελος Πλεύρης
3115038

August 29, 2020

Συστήματα Τύπων

Μορφή της σχέσης τύπων

Μας δίνεται μια μηχανή στοίβας καθώς και η λειτουργική σημασιολογία μίας γλώσσας για αυτή τη μηχανή. Ζητείται να δημιουργήσουμε ένα σύστημα τύπων για τη γλώσσα αυτή. Για το ζητούμενο σύστημα τύπων έχουμε δύο σύνολα τύπων:

τ := Int | Bool , βασικοί τύποι που δίνονται από την εκφώνηση
Σ:= $\tau_0, \tau_1, ..., \tau_n$, οι τύποι των περιεχομένων της στοίβας. Προφανώς, οι τύποι αυτοί δεν είναι γνωστοί από πριν αλλά προστίθενται κατά τη διάρκεια της στατικής ανάλυσης με βάση τους κανόνες τύπων. Ουσιαστικά, το σύνολο Σ είναι και αυτό μια στοίβα με τύπους. Στους κανόνες τύπων όταν γράφουμε α : τ, σημαίνει ότι η έκφραση α επιστρέφει τύπο τ, ο οποίος προστίθεται στη στοίβα Σ. Σε κάποιες περιπτώσεις χρησιμοποιούμε το συμβολισμό α : $\tau_0 \cdot \tau_1$ που σημαίνει ότι η αποτίμηση της έκφρασης α προσθέτει στην κορυφή της στοίβας Σ τους τύπους τ_0, τ_1 (το τ_0 θα είναι στην κορυφή), όμοια με τον συμβολισμό που χρησιμοποιείται στην εκφώνηση. Ακόμη θεωρούμε ότι όταν γράφουμε $\Sigma \vdash n : \tau$, ότι το στοιχείο n είναι στην κορυφή της στοίβας τύπων Σ και έχει τύπο τ καθώς και ότι γίνεται pop από τη στοίβα. Τέλος, θεωρούμε ότι σε έναν κανόνα της μορφής $\dfrac{x, y}{z}$ αποτιμάται πρώτα ο κανόνας x και μετά ο y, γιατί παίζει ρόλο η σειρά που βρίσκονται οι τύποι στη στοίβα τύπων Σ.

Όλες οι εκφράσεις της γλώσσας λαμβάνουν τύπο στο περιβάλλον Σ των τύπων της στοίβας. Έτσι για τους δοσμένους κανόνες λειτουργικής σημασιολογίας διατυπώνουμε τους επόμενους κανόνες τύπων.

Κανόνες Τύπων

$$n : Int, \qquad true : Bool, \qquad false : Bool, \qquad \frac{\Sigma \vdash n_1 : Int \quad \Sigma \vdash n_2 : Int}{\Sigma \vdash +; \sigma \cdot n_1 \cdot n_2 : Int},$$

$$\frac{\Sigma \vdash n : Int}{\Sigma \vdash -; \sigma \cdot n : Int}, \qquad \frac{\Sigma \vdash n_1 : Int \quad \Sigma \vdash n_2 : Int}{\Sigma \vdash *; \sigma \cdot n_1 \cdot n_2 : Int}, \qquad \frac{\Sigma \vdash n_1 : Int \quad \Sigma \vdash n_2 : Int}{\Sigma \vdash /; \sigma \cdot n_1 \cdot n_2 : Int \cdot Int},$$

$$\frac{\Sigma \vdash n_1 : Int \quad \Sigma \vdash n_2 : Int}{\Sigma \vdash <; \sigma \cdot n_1 \cdot n_2 : Bool}, \qquad \frac{\Sigma \vdash n_1 : Int \quad \Sigma \vdash n_2 : Int}{\Sigma \vdash =; \sigma \cdot n_1 \cdot n_2 : Bool}, \qquad \frac{\Sigma \vdash n_1 : Bool \quad \Sigma \vdash n_2 : Bool}{\Sigma \vdash and; \sigma \cdot n_1 \cdot n_2 : Bool},$$

$$\frac{\Sigma \vdash n : Bool}{\Sigma \vdash not; \sigma \cdot n : Bool}, \qquad \frac{\Sigma \vdash n : \tau}{\Sigma \vdash dup; \sigma \cdot n : \tau \cdot \tau}, \qquad \frac{\Sigma \vdash n : \tau}{\Sigma \vdash pop; \sigma \cdot n : -}(1),$$

$$\frac{\Sigma \vdash n_1 : \tau_1 \quad \Sigma \vdash n_2 : \tau_2}{\Sigma \vdash swap; \sigma \cdot n_1 \cdot n_2 : \tau_2 \cdot \tau_1}, \qquad \frac{\Sigma \vdash n_1 : \tau_1 \quad \Sigma \vdash n_2 : \tau_2 \quad \Sigma \vdash n_3 : \tau_3}{\Sigma \vdash swap2; \sigma \cdot n_1 \cdot n_2 \cdot n_3 : \tau_3 \cdot \tau_2 \cdot \tau_1},$$

$nop : -(2)$, $\quad \dfrac{\Sigma \vdash n : Bool \quad \Sigma \vdash p_1 : \tau \quad \Sigma \vdash p_2 : \tau}{\Sigma \vdash cond[p1|p2]; \sigma \cdot n : \tau}$, $\quad \dfrac{\Sigma \vdash p_1 : \tau_1 \quad \Sigma \vdash p_2 : \tau_2}{\Sigma \vdash p_1 p_2; \sigma : \tau_2}$,
$\dfrac{\Sigma \vdash n : Bool \quad \Sigma \vdash p : \tau}{\Sigma \vdash loop[p]; \sigma \cdot n : \tau}$

(1): Έχοντας θεωρήσει ότι στην αποτίμηση $\Sigma \vdash n : \tau$ γίνεται pop ο τύπος από τη στοίβα Σ, η εντολή pop δεν προσθέτει κάποιο τύπο στη στοίβα τύπων Σ. Αυτό το συμβολίζουμε με -.
(2): Η εντολή nop δεν αλλάζει τα περιεχόμενα της στοίβας τύπων Σ και επομένως δεν προσθέτει(ούτε αφαιρεί προφανώς) κάποιον τύπο στη στοίβα. Αυτό το συμβολίζουμε με -.

Με αυτό το σύστημα τύπων θα υπάρξουν προγράμματα τα οποία δεν θα περάσουν τον έλεγχο τύπων αλλά θα έτρεχαν κανονικά (π.χ. προγράμματα που στην εντολή cond τα p_1 και p_2 έχουν διαφορετικό τύπο), αλλά μας εγγυάται ότι όποια προγράμματα περάσουν τον έλεγχο τύπων δεν θα κολλήσουν.