Συναρτησιακός Προγραμματισμός 2008
Τρίτο Φύλλο Ασκήσεων - Project

Το project αυτό μπορεί να γίνει από ομάδες 1-3 ατόμων και αντιστοιχεί στο 15% του βαθμού στο μάθημα.

Συνολικό Άθροισμα Βαθμών: 150

Προθεσμία Παράδοσης: Δευτέρα 30 Ιουνίου

Σε αυτό το project θα υλοποιήσετε τον αλγόριθμο εξαγωγής τύπου Hindley-Milner για ένα απλό λ-λογισμό χωρίς τύπους και ένα απλό σύστημα τύπων.

Ο λ-λογισμός που θα χρησιμοποιήσετε περιλαμβάνει φυσικούς αριθμούς, αληθοτιμές, λίστες καθώς και μερικές ενσωματωμένες συναρτήσεις. Περιγράφεται από την εξής γραμματική:

$t ::=$	x	μεταβλητή
	n	φυσική σταθερά (`0,1,...` κτλ.)
	`true`|`false`	
	`if` t `then` t `else` t	
	$u\ t$	μοναδιαία συνάρτηση
	$t\ b\ t$	δυαδικός τελεστής
	`emptylist`	κενή λίστα
	`cons` $t\ t$	κατασκευή μη κενής λίστας
	$t\ t$	
	$\lambda x.t$	
$u ::=$	`head`|`tail`|`isempty`	
$b ::=$	`+`|`-`|`*`|`/`	τελεστές `Nat`→`Nat`→`Nat`
	`>`|`<`|`=`	τελεστές `Nat`→`Nat`→`Bool`

Δεν είναι στους σκοπούς της άσκησης να υλοποιήσετε parser για το λογισμό αυτό. Θεωρήστε ότι ο parser σας έχει ήδη δωθεί και προχωρήστε χρησιμοποιώντας ένα αλγεβρικό τύπο στον οποίο να αναπαρίσταται το *δέντρο αφηρημένης σύνταξης (abstract syntax tree)*.

Το σύστημα τύπων που θα χρησιμοποιήσετε είναι το παρακάτω:

$$\frac{(x:T)\in\Gamma}{\Gamma\vdash x:T}\qquad \frac{}{\Gamma\vdash n:\mathtt{Nat}}\qquad \frac{c\in\{\mathtt{true},\mathtt{false}\}}{\Gamma\vdash c:\mathtt{Bool}}$$

$$\frac{\Gamma\vdash t_1:\mathtt{Bool}\quad \Gamma\vdash t_2:T\quad \Gamma\vdash t_3:T}{\Gamma\vdash \mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3:T}$$

$$\frac{\Gamma \vdash t : [T]}{\Gamma \vdash \texttt{head}\ \ t : T} \qquad \frac{\Gamma \vdash t : [T]}{\Gamma \vdash \texttt{tail}\ \ t : [T]} \qquad \frac{\Gamma \vdash t : [T]}{\Gamma \vdash \texttt{isempty}\ \ t : \texttt{Bool}}$$

$$\frac{\Gamma \vdash t_1 : \texttt{Nat} \quad \Gamma \vdash t_2 : \texttt{Nat} \quad b \in \{\texttt{+}, \texttt{-}, \texttt{*}, \texttt{/}\}}{\Gamma \vdash t_1\ b\ t_2 : \texttt{Nat}}$$

$$\frac{\Gamma \vdash t_1 : \texttt{Nat} \quad \Gamma \vdash t_2 : \texttt{Nat} \quad b \in \{\texttt{>}, \texttt{<}, \texttt{=}\}}{\Gamma \vdash t_1\ b\ t_2 : \texttt{Bool}}$$

$$\frac{}{\Gamma \vdash \texttt{emptylist} : [T]} \qquad \frac{\Gamma \vdash t_1 : T \qquad \Gamma \vdash t_2 : [T]}{\Gamma \vdash \texttt{cons}\ \ t_1\ t_2 : [T]}$$

$$\frac{\Gamma \vdash t_1 : T_2 \to T_1 \quad \Gamma \vdash t_2 : T_2}{\Gamma \vdash t_1\ t_2 : T_1} \qquad \frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \lambda x : T_1 . t : T_1 \to T_2}$$

Ο αλγόριθμος θα παίρνει μία έκφραση του παραπάνω λ-λογισμού χωρίς τύπους και θα επιστρέφει τον πιο γενικό τύπο αυτής της έκφρασης ή θα αναφέρει σφάλμα σε περίπτωση που (α) τέτοιος τύπος δεν υπάρχει ή (β) η έκφραση δεν είναι κλειστή (περιέχει αδέσμευτες μεταβλητές).

Ο αλγόριθμος που θα φτιάξετε χωρίζεται σε τρια βήμα. Τα δύο πρώτα βήματα μπορούν να γίνουν ταυτόχρονα. Το τρίτο βήμα ακολουθεί.

Βήμα I - Ανάθεση Μεταβλητών Τύπων

Στο πρώτο βήμα, κάθε υποέκφραση (συμπεριλαμβανομένης και ολόκληρης της έκφρασης) καθώς και κάθε δεσμεύουσα μεταβλητή αντιστοιχίζεται με μία ξεχωριστή μεταβλητή τύπων. Για τις μεταβλητές τύπων σε αυτό το φυλλάδιο χρησιμοποιούμε κεφαλαία λατινικά γράμματα.

Παράδειγμα.

$$(\lambda \texttt{x.x})\ \texttt{emptylist}$$

Η ανάθεση τύπων γίνεται ως εξής:

- $(\lambda$`x.x) emptylist :  A`
- $(\lambda$`x.x) :  B`
- δεσμεύουσα `x :  C`
- `x :  D`
- `emptylist :  E`

Βήμα II - Εξαγωγή Περιορισμών

Στο δεύτερο βήμα εξάγεται ένα σύνολο περιορισμών C μεταξύ των μεταβλητών τύπων που εισήγαγε το Βήμα I. Οι περιορισμοί είναι όλοι της μορφής $T_1 = T_2$ όπου T_i είναι έκφραση τύπων με μεταβλητές τύπων.

Για την εξαγωγή περιορισμών εξετάζουμε κάθε υποέκφραση ξεχωριστά, ξεκινώντας από ολόκληρη την έκφραση. Για κάθε υποέκφραση ακολουθούμε τους κανόνες εξαγωγής περιορισμών που περιγράφουμε παρακάτω.

Έστω E μία από τις υποεκφράσεις και έστω X η μεταβλητή τύπων που αντιστοίχησε το Βήμα I στην E. Έχουμε τις εξής περιπτώσεις:

- $E = x$, όπου x μια μεταβλητή. Έστω Y η μεταβλητή τύπου που εισήγαγε το Βήμα I κατά τη δέσμευση[1] της μεταβλητής x. Εισάγεται ο περιορισμός

  $$X = Y$$

- $E = c$, όπου μία σταθερά τύπου T με $T = \texttt{Nat}$ ή $T = \texttt{Bool}$. Εισάγεται ο περιορισμός

  $$X = T$$

- $E = \texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3$, όπου οι τύποι που εισήγαγε το Βήμα I είναι

  $$t_1 : Y, t_2 : Z, t_3 : W$$

  Εισάγονται οι περιορισμοί

  $$Y = \texttt{Bool}, Z = W, X = Z$$

- $E = \texttt{head}\ \ t$, όπου (από Βήμα I)

  $$t : Y$$

  Εισάγεται ο περιορισμός

  $$Y = \texttt{[}X\texttt{]}$$

- $E = \texttt{tail}\ \ t$, όπου (από Βήμα I)

  $$t : Y$$

  Εισάγονται οι περιορισμοί

  $$Y = X, X = [Z]$$

  όπου Z είναι μια καινούρια μεταβλητή τύπων.

[1] Αν η x δεν έχει δεσμευθεί, τότε η έκφραση δεν είναι κλειστή και έχουμε σφάλμα.

- $E =$ `isempty` t, όπου (από Βήμα I)

  $$t : Y$$

  Εισάγονται οι περιορισμοί

  $$Y = [Z], X = \texttt{Bool}$$

  όπου Z είναι μια καινούρια μεταβλητή τύπων.

- $E = t_1\ b\ t_2$, όπου ο δυαδικός τελεστής b είναι τύπου $T \to T \to U$ και (από Βήμα I)

  $$t_1 : Y, t_2 : Z$$

  Εισάγονται οι περιορισμοί

  $$Y = T, Z = T, X = U$$

- $E =$ `emptylist`. Εισάγεται ο περιορισμός

  $$X = [Y]$$

  όπου Y είναι μια καινούρια μεταβλητή τύπων

- $E =$ `cons` $t_1\ t_2$, όπου (Βήμα I)

  $$t_1 : Y, t_2 : Z$$

  Εισάγονται οι περιορισμοί

  $$Z = [Y], X = Z$$

- $E = t_1\ t_2$, όπου (Βήμα I)

  $$t_1 : Y, t_2 : Z$$

  Εισάγεται ο περιορισμός

  $$Y = Z \to X$$

- $E = \lambda x.t$, όπου (Βήμα I)

  $$x : Y, t : Z$$

  Εισάγεται ο περιορισμός

  $$X = Y \to Z$$

Παράδειγμα. Εξάγονται οι περιορισμοί: `B=E→A, B=C→D, D=C, E=[F]`

Βήμα III - Ενοποίηση

Έστω C το σύνολο των περιορισμών από το Βήμα II. Προσθέτουμε στο C έναν ακόμη περιορισμό `A'=A`. Έστω a ο νέος αυτός περιορισμός. Η μεταβλητή τύπων `A'` είναι καινούρια. Ακολουθούμε την εξής διαδικασία:

- Για κάθε περιορισμό $c \in C - \{a\}$
    - Αν ο c είναι της μορφής
        * $X = T$, όπου X μεταβλητή τύπων και $X \notin FV(T)$, τότε εφαρμόζουμε την μετατροπή $[X := T]$ σε κάθε κανόνα του C.
        * $T = X$, όπου X μεταβλητή τύπων και $X \notin FV(T)$, τότε χειριζόμαστε όπως και στην πάνω περίπτωση, για $X = T$
        * `[`T_1`]` $=$ `[`T_2`]`, τότε προσθέτουμε τον περιορισμό $T_1 = T_2$ στο C
        * $S_1 \to T_1 = S_2 \to T_2$, τότε προσθέτουμε τους περιορισμούς $S_1 = S_2$ και $T_1 = T_2$ στο C
        * $T = T$ για σταθερό τύπο T ή $X = X$ για μεταβλητή X, τότε δεν κάνουμε τίποτα
    - Σε κάθε άλλη περίπτωση, τερματίζουμε με αποτυχία
    - Αφαιρούμε τον c από το C

Στο τέλος, έχουμε μείνει με ένα μόνο περιορισμό, της μορφής `A'=`T. Ο T είναι ο πιο γενικός τύπος που αναζητάμε.

Παράδειγμα. Μετά την εφαρμογή του πρώτου περιορισμού έχουμε:

`E→A=C→D, D=C, E=[F], A'=A`

Με την εφαρμογή του επόμενου περιορισμού έχουμε

`E=C, A=D, D=C, E=[F], A'=A`

Με την εφαρμογή του επόμενου περιορισμού έχουμε

`A=D, D=C, C=[F], A'=A`

Με την εφαρμογή του επόμενου περιορισμού έχουμε

`D=C, C=[F], A'=D`

Με την εφαρμογή του επόμενου περιορισμού έχουμε

`C=[F], A'=C`

Με την εφαρμογή του επόμενου περιορισμού έχουμε

`A'=[F]`

Ο πιο γενικός πολυμορφικός τύπος είναι `[F]`, όπου `F` μία μεταβλητή τύπων. Στη Haskell θα γράφαμε `[a]`.