Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
Εργαστήριο Τεχνολογίας Λογισμικού

# Γλώσσες Προγραμματισμού II

## Εξέταση στη Θεωρία Γλωσσών Προγραμματισμού

Νίκος Παπασπύρου

Φεβρουάριος 2005 — Διάρκεια: 2 ώρες

## Θέμα 1 (25%)

Έστω μια επέκταση της γλώσσας εκφράσεων με βασικούς τύπους που περιγράφεται στην ενότητα 4.2 (σελ. 48) των σημειώσεων με έναν επιπλέον τελεστή, ο οποίος υλοποιεί τη μη ντετερμινιστική επιλογή μεταξύ των δύο τελουμένων.

$$e \ ::= \ \ldots \mid e_1 \oplus e_2$$

Για τον τελεστή αυτόν, ζητείται:

1. (5%) Να ορίσετε κατάλληλους κανόνες τύπων.
2. (5%) Να ορίσετε μια κατάλληλη λειτουργική σημασιολογία, έτσι ώστε το τελούμενο που δεν επιλέγεται να μην αρχίζει να αποτιμάται.
3. (5%) Να ορίσετε μια κατάλληλη λειτουργική σημασιολογία, έτσι ώστε και τα δύο τελούμενα να αποτιμώνται πλήρως, αλλά το αποτέλεσμα να είναι η τιμή του τελούμενου που επιλέγεται.
4. (5%) Να ορίσετε μια κατάλληλη λειτουργική σημασιολογία, έτσι ώστε και τα δύο τελούμενα να αρχίζουν να αποτιμώνται και τουλάχιστον αυτό που επιλέγεται να αποτιμάται πλήρως.
5. (5%) Να σχολιάσετε κατά πόσον η σημασιολογία που δώσατε στο προηγούμενο υποερώτημα είναι "δίκαιη", δηλαδή αν τυχόν ευνοεί κάποιο από τα τελούμενα, π.χ. ανάλογα με τη θέση ή τη δομή του.

### Λύση:

1. Κανόνας τύπων

$$\frac{e_1 : \tau \qquad e_2 : \tau}{e_1 \oplus e_2 : \tau}$$

2. Λειτουργική σημασιολογία: επιλογή στην αρχή.

$$e_1 \oplus e_2 \longrightarrow e_1 \qquad e_1 \oplus e_2 \longrightarrow e_2$$

3. Λειτουργική σημασιολογία: επιλογή στο τέλος.

$$\frac{e_1 \longrightarrow e_1'}{e_1 \oplus e_2 \longrightarrow e_1' \oplus e_2} \qquad \frac{e_2 \longrightarrow e_2'}{e_1 \oplus e_2 \longrightarrow e_1 \oplus e_2'}$$

$$v_1 \oplus v_2 \longrightarrow v_1 \qquad v_1 \oplus v_2 \longrightarrow v_2$$

4. Λειτουργική σημασιολογία: επιλογή ενδιάμεσα.

$$\frac{e_1 \longrightarrow e_1'}{e_1 \oplus e_2 \longrightarrow e_1' \oplus e_2} \qquad \frac{e_2 \longrightarrow e_2'}{e_1 \oplus e_2 \longrightarrow e_1 \oplus e_2'}$$

$$e_1 \oplus e_2 \longrightarrow e_1 \qquad e_1 \oplus e_2 \longrightarrow e_2$$

5. Η σημασιολογία είναι δίκαιη. Οι δύο πρώτοι κανόνες δίνουν ίσες ευκαιρίες στις $e_1$ και $e_2$ να αποτιμηθούν (υποθέτοντας ότι δεν είναι ήδη αποτιμημένες) προτού γίνει η επιλογή. Οι δύο επόμενοι κανόνες δίνουν ίσες ευκαιρίες στις $e_1$ και $e_2$ να επιλεγούν για το αποτέλεσμα της συνολικής έκφρασης.

## Θέμα 2 (25%)

Έστω ότι στην απλή γλώσσα προστακτικού προγραμματισμού, που περιγράφεται στο κεφάλαιο 6 των σημειώσεων, προσθέτουμε συναρτήσεις μιας ακέραιας παραμέτρου και ακέραιου αποτελέσματος. Οι ορισμοί τους, που προηγούνται των εντολών του προγράμματος, είναι της μορφής:

$$\texttt{function}\ f\ (x)\ \texttt{do}\ \mathbf{C}\ \texttt{return}\ \mathbf{N}$$

και οι κλήσεις τους γίνονται σε ακέραιες εκφράσεις ως εξής:

$$\mathbf{N} \ ::= \ \ldots \ \mid \ f(\mathbf{N})$$

Κατά την κλήση μιας συνάρτησης, αποτιμάται η πραγματική παράμετρος, το αποτέλεσμα τοποθετείται στην τυπική παράμετρο, εκτελείται το σώμα που ακολουθεί το `do` και στη συνέχεια αποτιμάται η έκφραση που ακολουθεί το `return`. Η τιμή της τελευταίας είναι το αποτέλεσμα της συνάρτησης.

Ζητείται:

1. (15%) Αγνοώντας τη σημασιολογία του ορισμού και των κλήσεων των συναρτήσεων, να εξηγήσετε τις επιπτώσεις που θα έχει η προσθήκη τους στη σημασιολογία των εκφράσεων και των εντολών. Να περιγράψετε εν συντομία τυχόν αλλαγές που θα πρέπει να γίνουν στον ορισμό των σημασιολογικών συναρτήσεων $\mathcal{N}[\![\mathbf{N}]\!]$, $\mathcal{B}[\![\mathbf{B}]\!]$ και $\mathcal{C}[\![\mathbf{C}]\!]$.

2. (10%) Να περιγράψετε περιληπτικά (και όχι κατ' ανάγκη με αυστηρό τρόπο) πώς μπορεί να οριστεί η σημασιολογία του ορισμού και της κλήσης συναρτήσεων.

### Λύση:

1. Η προσθήκη συναρτήσεων επιτρέπει στον υπολογισμό αριθμητικών εκφράσεων να μεταβάλλει την κατάσταση του προγράμματος. Κατά συνέπεια πρέπει να τροποποιηθεί ο τύπος της σημασιολογικής συνάρτησης $\mathcal{N}[\![N]\!]$ ώστε να επιστρέφει την κατάσταση που θα έχει προκύψει μετά τον υπολογισμό:

$$\mathcal{N}[\![N]\!] \ : \ \boldsymbol{S} \to \boldsymbol{N} \times \boldsymbol{S}$$

Οι υπάρχουσες εξισώσεις χρειάζονται τροποποίηση κατάλληλα, π.χ.

$$\begin{array}{lcl} \mathcal{N}[\![n]\!](s) & = & \langle n, s\rangle \\ \mathcal{N}[\![x]\!](s) & = & \langle s(x), s\rangle \end{array}$$

αλλά το κυριότερο είναι ότι η κατάσταση που προκύπτει από την αποτίμηση υποεκφράσεων πρέπει να μεταφέρεται, κάτι που σημαίνει ότι η σειρά αποτίμησης πρέπει να οριστεί ρητά με τις εξισώσεις για τους τελεστές με δύο τελούμενα, π.χ.

$$\begin{array}{lcl} \mathcal{N}[\![-N]\!](s) & = & \langle -v, s'\rangle \qquad \text{όπου } \langle v, s'\rangle = \mathcal{N}[\![N]\!](s) \\ \mathcal{N}[\![N_1 + N_2]\!](s) & = & \langle v_1 + v_2, s''\rangle \text{ όπου } \langle v_1, s'\rangle = \mathcal{N}[\![N_1]\!](s) \text{ και } \langle v_2, s''\rangle = \mathcal{N}[\![N_2]\!](s') \end{array}$$

Επίσης πρέπει να αλλάξει η σημασιολογική συνάρτηση για τις λογικές εκφράσεις:

$$\mathcal{B}[\![B]\!] \;:\; \boldsymbol{S} \to \boldsymbol{T} \times \boldsymbol{S}$$

με παρόμοιο τρόπο, καθώς και οι εξισώσεις της συνάρτησης $\mathcal{C}[\![C]\!]$ όπου χρησιμοποιούνται οι $\mathcal{N}[\![N]\!]$ και $\mathcal{B}[\![B]\!]$, π.χ.

$$\begin{array}{lcl} \mathcal{C}[\![x := N]\!](s) & = & (s' \mid x \mapsto v) \text{ όπου } \langle v, s'\rangle = \mathcal{N}[\![N]\!](s) \\ \mathcal{C}[\![\textit{if } B \textit{ then } C_1 \textit{ else } C_2]\!](s) & = & \begin{cases} \mathcal{C}[\![C_1]\!](s') \text{ αν } v = \mathit{true} \\ \mathcal{C}[\![C_2]\!](s') \text{ αν } v = \mathit{false} \end{cases} \quad \text{όπου } \langle v, s'\rangle = \mathcal{B}[\![B]\!](s) \end{array}$$

2. Για τη σημασιολογία της κλήσης συναρτήσεων χρειάζεται ένα "περιβάλλον" $\rho$ που θα δίνει τη σημασία κάθε συνάρτησης: για κάθε όνομα συνάρτησης $f$ θα είναι

   $$\rho(f) \;:\; \boldsymbol{N} \times \boldsymbol{S} \to \boldsymbol{N} \times \boldsymbol{S}$$

   Το περιβάλλον $\rho$ μπορεί να δίνεται ως παράμετρος των σημασιολογικών συναρτήσεων. Η κλήση συναρτήσεων τότε γίνεται

   $$\mathcal{N}[\![f(N)]\!](\rho)(s) \;=\; \rho(f)(\mathcal{N}[\![N]\!](s))$$

   Για κάθε ορισμό συνάρτησης

   `function` $f\ (x)$ `do` $\mathbf{C}$ `return` $\mathbf{N}$

   θα πρέπει

   $$\rho(f)\langle v_f, s\rangle \;=\; \langle v_a, s''\rangle \text{ όπου } s' = \mathcal{C}[\![C]\!](\rho)(s \mid x \mapsto v_f) \text{ και } \langle v_a, s''\rangle = \mathcal{N}[\![N]\!](\rho)(s')$$

   Προκειμένου να επιτραπεί ο αναδρομικός ορισμός συναρτήσεων, απαιτείται η χρήση σταθερών σημείων, όπως στο κεφάλαιο 7 των σημειώσεων. Το περιβάλλον $\rho$ πρέπει να είναι το ελάχιστο που ικανοποιεί την παραπάνω εξίσωση για κάθε ορισμό συνάρτησης.

## Θέμα 3 (20%)

Έστω η άπειρη ακολουθία λ-όρων $(Y_n)_{n\in\omega}$ που ορίζεται παρακάτω:

$$\begin{array}{lcllcl} Y_0 & \equiv & Y & \text{όπου:} \quad Y & \equiv & \lambda f.\, (\lambda x.\, f\,(x\,x))\,(\lambda x.\, f\,(x\,x)) \\ Y_{n+1} & \equiv & Y_n\, G & \quad G & \equiv & \lambda y.\, \lambda f.\, f\,(y\,f) \end{array}$$

Να δείξετε ότι:

1. (10%) Κάθε όρος $Y_n$ είναι τελεστής σταθερού σημείου.
2. (10%) Κάθε όρος $Y_n$ είναι σταθερό σημείο της $G$.

**Λύση:**

1. Για κάθε $F$ και $n$, πρέπει να δείξω ότι $Y_n\,F = F\,(Y_n\,F)$. Με επαγωγή στο $n$. Για $n = 0$ ο τελεστής $Y$ πληροί το ζητούμενο (θεώρημα 3.3, σελ. 40 των σημειώσεων). Έστω ότι ισχύει για τυχόν $n$. Τότε

$$\begin{array}{lcll} Y_{n+1}\,F & \equiv & Y_n\,G\,F & \text{ορισμός } Y_{n+1} \\ & = & G\,(Y_n\,G)\,F & \text{υπόθεση της επαγωγής} \\ & = & F\,(Y_n\,G)\,F & \text{ορισμός } G \text{ και } \beta\text{-αναγωγή} \\ & \equiv & F\,(Y_{n+1}\,F & \text{ορισμός } Y_{n+1} \end{array}$$

2. Για κάθε $n$, πρέπει να δείξω ότι $G\,Y_n = Y_n$.

$$\begin{array}{lcll} G\,Y_n & \equiv & (\lambda\,y.\ \lambda\,f.\ f\,(y\,f))\,Y_n & \text{ορισμός } G \\ & = & \lambda\,f.\ f\,(Y_n\,f) & \beta\text{-αναγωγή} \\ & = & \lambda\,f.\ Y_n\,f & \text{απόδειξη πρώτου μέρους της άσκησης} \\ & = & Y_n & \eta\text{-αναγωγή} \end{array}$$

## Θέμα 4 (30%)

Δίνεται το παρακάτω τμήμα προγράμματος σε μια απλή γλώσσα προστακτικού προγραμματισμού:

```
i:=0; while n>1 do (n:=n div 2; i:=i+1)
```

Ζητούνται:

1. (5%) Να περιγραφεί με λόγια το αποτέλεσμα της εκτέλεσης αυτού του τμήματος προγράμματος.
2. (10%) Να βρεθούν η κατάλληλη προσυνθήκη $P$ και η κατάλληλη μετασυνθήκη $Q$ ώστε να δοθεί μια ακριβής προδιαγραφή αυτού του τμήματος προγράμματος με τη μορφή τριάδας Floyd-Hoare:

$$\{P\}\ \texttt{i:=0; while n>1 do (n:=n div 2; i:=i+1)}\ \{Q\}$$

3. (15%) Να αποδειχθεί ότι η παραπάνω προδιαγραφή είναι αληθής, με χρήση των κανόνων αξιωματικής σημασιολογίας της απλής προστακτικής γλώσσας.

**Λύση:**

1. Αν αρχικά $n \geq 1$, το πρόγραμμα αυτό υπολογίζει τον (ακέραιο) λογάριθμο με βάση 2 του $n$ με διαδοχικές διαιρέσεις. Δηλαδή βρίσκει έναν ακέραιο $i$ τέτοιο ώστε $2^i \leq n < 2^{i+1}$.
2. Η προσυνθήκη $P$ και η μετασυνθήκη $Q$ σύμφωνα με το παραπάνω:

$$\begin{array}{lcl} P & \equiv & n = X\ \wedge\ n \geq 1 \\ Q & \equiv & 2^i \leq X < 2^{i+1} \end{array}$$

3. Από το αξίωμα ανάθεσης, παραλείποντας την ταυτολογία $0 = 0$ στην προσυνθήκη, προκύπτει

$$\{n = X\ \wedge\ n \geq 1\}\ \texttt{i:=0}\ \{n = X\ \wedge\ n \geq 1\ \wedge\ i = 0\} \tag{1}$$

Επιλέγουμε ως αναλλοίωτη για το βρόχο τη συνθήκη

$$I \quad \equiv \quad X/2^i = n \ \wedge \ n \geq 1$$

Εύκολα προκύπτει

$$n = X \ \wedge \ n \geq 1 \ \wedge \ i = 0 \quad \Rightarrow \quad I \tag{2}$$

και από τις (1), (2) και τον κανόνα αποδυνάμωσης της μετασυνθήκης προκύπτει

$$\{P\} \texttt{ i:=0 } \{I\} \tag{3}$$

Από το αξίωμα ανάθεσης

$$\{X/2^{i+1} = n \ \wedge \ n \geq 1\} \texttt{ i:=i+1 } \{X/2^i = n \ \wedge \ n \geq 1\} \tag{4}$$

και ομοίως

$$\{X/2^{i+1} = n/2 \ \wedge \ n/2 \geq 1\} \texttt{ n:=n/2 } \{X/2^{i+1} = n \ \wedge \ n \geq 1\} \tag{5}$$

Από τις (4), (5) και τον κανόνα παράθεσης προκύπτει

$$\{X/2^{i+1} = n/2 \ \wedge \ n/2 \geq 1\} \texttt{ n:=n/2; i:=i+1 } \{I\} \tag{6}$$

Εύκολα διαπιστώνει κανείς ότι

$$I \ \wedge \ n > 1 \quad \Rightarrow \quad X/2^{i+1} = n/2 \ \wedge \ n/2 \geq 1 \tag{7}$$

Το αριστερό μέλος είναι ισοδύναμο με

$$X/2^i = n \ \wedge \ n > 1$$

Διαιρώντας τον πρώτο όρο με 2 προκύπτει ο πρώτος όρος του δεξιού μέλους και από το δεύτερο όρο προφανώς προκύπτει ότι $n \geq 2$ και άρα $n/2 \geq 1$. Επομένως, από τις (6), (7) και τον κανόνα ενδυνάμωσης της προσυνθήκης προκύπτει

$$\{I \ \wedge \ n > 1\} \texttt{ n:=n/2; i:=i+1 } \{I\} \tag{8}$$

Από την (8) και τον κανόνα του `while` προκύπτει

$$\{I\} \texttt{ while n>1 do (n:=n/2; i:=i+1) } \{I \ \wedge \ n \leq 1\} \tag{9}$$

και από τις (3), (9) και τον κανόνα παράθεσης προκύπτει

$$\{P\} \texttt{ i:=0; while n>1 do (n:=n/2; i:=i+1) } \{I \ \wedge \ n \leq 1\} \tag{10}$$

Εύκολα επίσης διαπιστώνει κανείς ότι

$$I \ \wedge \ n \leq 1 \quad \Rightarrow \quad 2^i \leq X < 2^{i+1} \tag{11}$$

Το αριστερό μέλος είναι ισοδύναμο με

$$X/2^i = n \ \wedge \ n = 1$$

που σημαίνει $X/2^i = 1$. Από αυτό συμπεραίνουμε αφενός ότι $2^i \leq X$, αφετέρου ότι $X/2^{i+1} = 1/2 = 0$ και άρα $X < 2^{i+1}$. Επομένως, από τις (10), (11) και τον κανόνα αποδυνάμωσης της μετασυνθήκης προκύπτει τελικά το ζητούμενο

$$\{P\} \texttt{ i:=0; while n>1 do (n:=n/2; i:=i+1) } \{Q\} \tag{12}$$