Γλώσσες Προγραμματισμού ΙΙ

http://courses.softlab.ntua.gr/pl2/

Κωστής Σαγώνας — kostis@cs.ntua.gr
Νίκος Παπασπύρου — nickie@softlab.ntua.gr

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχ. και Μηχ. Υπολογιστών
Εργαστήριο Τεχνολογίας Λογισμικού
Πολυτεχνειούπολη, 15780 Ζωγράφου.

Εισαγωγή στο λ-λογισμό (i)

- Alonso Church, αρχές δεκαετίας 1930
- Θεωρία για τη θεμελίωση των μαθηματικών
- Απλό αλλά πλήρες υπολογιστικό μοντέλο
- Βάση για τη δημιουργία του συναρτησιακού μοντέλου προγραμματισμού
- Πρόσφορος συμβολισμός για την περιγραφή της σημασιολογίας γλωσσών προγραμματισμού
- Αρχική μορφή: χωρίς τύπους
- Παραλλαγές με τύπους προτάθηκαν αργότερα (Curry, Church, κ.λπ.)

Εισαγωγή στο λ-λογισμό (ii)

- Διαισθητική περιγραφή: μια θεωρία συναρτήσεων
  - x — μεταβλητή
  - $F\,A$ — εφαρμογή
  - $\lambda x.\,E[x]$ — αφαίρεση
- Αν x είναι μια μεταβλητή και $E[x]$ μια έκφραση που περιέχει το x, τότε $\lambda x.\,E[x]$ είναι η συνάρτηση f, όπου $f(x) = E[x]$
- Παράδειγμα: $\lambda x.\,x^2 - 3x + 2$

$$(\lambda x.\,x^2 - 3x + 2)\,8 \;=\; 8^2 - 3\cdot 8 + 2 \;=\; 42$$

Εισαγωγή στο λ-λογισμό (iii)

- Μεταβλητές ελεύθερες και δεσμευμένες

$$\lambda x.\,x^2 - 3y + 2$$
$$(\lambda x.\,x^2 - 3y + 2)\,(4x + 1)$$

- Ανάλογο στα μαθηματικά: $\displaystyle\int_{-\pi}^{\pi} \frac{\sin x + \cos y}{\cos x - \sin y}\,dx$
- Αντικατάσταση μεταβλητής με έκφραση

$$(\lambda x.\,x^2 - 3y + 2)[y := z + 1]$$
$$\equiv\; \lambda x.\,x^2 - 3(z+1) + 2$$

Εισαγωγή στο λ-λογισμό (iv)

- Αντικατάσταση (συνέχεια)
  - Αποφυγή προβλημάτων $\displaystyle\int_{-\pi}^{\pi} \frac{\sin x + \cos y}{\cos x - \sin y}\,dx$
  - Όχι σε δεσμευμένες μεταβλητές!

$$\int_{-\pi}^{\pi} \frac{\sin 42 + \cos y}{\cos 42 - \sin y}\,d42$$

  - Να μην προκαλεί δέσμευση μεταβλητών!

$$\int_{-\pi}^{\pi} \frac{\sin x + \cos(3x+1)}{\cos x - \sin(3x+1)}\,dx$$

λ-λογισμός χωρίς τύπους (i)

- Σύνταξη: $x \in V$ και $M, N \in \Lambda$

$$M, N \;::=\; x \mid (\lambda x.\,M) \mid (M\,N)$$

- Συντακτικές συμβάσεις
  - Οι εξωτερικές παρενθέσεις δε γράφονται

$$(\lambda x.\,x) \qquad \lambda x.\,x$$

  - Η εφαρμογή είναι αριστερά προσεταιριστική

$$(\ldots((F\,M_1)\,M_2)\,\ldots\,M_n) \quad F\,M_1\,M_2\,\ldots\,M_n$$

  - Η αφαίρεση εκτείνεται όσο είναι δυνατόν

$$\lambda x.\,(M_1\,M_2\,\ldots\,M_n) \qquad \lambda x.\;M_1\,M_2\,\ldots\,M_n$$

λ-λογισμός χωρίς τύπους (ii)

- Παραδείγματα

$(x\,y)$	$x\,y$
$(\lambda x.\,x)$	$\lambda x.\,x$
$(\lambda x.\,(\lambda y.\,(x\,y)))$	$\lambda x.\,\lambda y.\,x\,y$
$(((\lambda x.\,x)\,y)\,(\lambda x.\,z))$	$((\lambda x.\,x)\,y)\,(\lambda x.\,z)$
$((\lambda x.\,(\lambda y.\,z))\,(\lambda x.\,x))$	$(\lambda x.\,\lambda y.\,z)\,(\lambda x.\,x)$
$(\lambda x.\,((\lambda y.\,y)\,(\lambda z.\,x)))$	$\lambda x.\,(\lambda y.\,y)\,(\lambda z.\,x)$

λ-λογισμός χωρίς τύπους (iii)

- Σχέση ταυτότητας: $M \equiv N$

$$\begin{array}{lcll} x & \equiv & y & \text{αν } x = y \\ (M\,N) & \equiv & (P\,Q) & \text{αν } M \equiv P \text{ και } N \equiv Q \\ (\lambda x.\,M) & \equiv & (\lambda y.\,N) & \text{αν } x = y \text{ και } M \equiv N \end{array}$$

- Ελεύθερες και δεσμευμένες μεταβλητές

$$M \equiv (\lambda x.\,y\,x)\,(\lambda y.\,x\,y)$$

ελεύθερη — δεσμευμένη; δεσμευμένη — ελεύθερη

λ-λογισμός χωρίς τύπους (iv)

- Ελεύθερες μεταβλητές

$$\begin{array}{lcl} \mathrm{FV}(x) & = & \{\,x\,\} \\ \mathrm{FV}(M\,N) & = & \mathrm{FV}(M) \cup \mathrm{FV}(N) \\ \mathrm{FV}(\lambda x.\,M) & = & \mathrm{FV}(M) - \{\,x\,\} \end{array}$$

- Κλειστοί όροι (closed terms ή combinators)

$$\begin{array}{lcl} \mathsf{I} & \equiv & \lambda x.\,x \\ \mathsf{K} & \equiv & \lambda x.\,\lambda y.\,x \\ \mathsf{K}_* & \equiv & \lambda x.\,\lambda y.\,y \\ \mathsf{S} & \equiv & \lambda x.\,\lambda y.\,\lambda z.\,(x\,z)\,(y\,z) \end{array}$$

λ-λογισμός χωρίς τύπους (iii)

- Αντικατάσταση

$$\begin{array}{lcll} x[x := N] & \equiv & N & \\ y[x := N] & \equiv & y & \text{αν } y \neq x \\ (P\,Q)[x := N] & \equiv & P[x := N]\,Q[x := N] & \\ (\lambda x.\,P)[x := N] & \equiv & \lambda x.\,P & \\ (\lambda y.\,P)[x := N] & \equiv & \lambda y.\,P[x := N] & \\ & & \text{αν } y \neq x, \text{ και } (y \notin \mathrm{FV}(N) \text{ ή } x \notin \mathrm{FV}(P)) & \\ (\lambda y.\,P)[x := N] & \equiv & \lambda z.\,P[y := z][x := N] & \\ & & \text{αν } y \neq x, \text{ και } y \in \mathrm{FV}(N) \text{ και } x \in \mathrm{FV}(P), & \\ & & \text{όπου } z \notin \mathrm{FV}(P) \cup \mathrm{FV}(N) & \end{array}$$

Μετατροπές στο λ-λογισμό (i)

- Συμβατές σχέσεις

$$\begin{array}{l} M \sim N \;\Rightarrow\; (M\,P) \sim (N\,P) \\ M \sim N \;\Rightarrow\; (P\,M) \sim (P\,N) \\ M \sim N \;\Rightarrow\; (\lambda x.\,M) \sim (\lambda x.\,N) \end{array}$$

- Μετατροπές $\to_\alpha, \to_\beta, \to_\eta$: οι ελάχιστες συμβατές σχέσεις που πληρούν τα παρακάτω

$$\begin{array}{lcll} \lambda x.\,M & \to_\alpha & \lambda y.\,M[x := y] & y \notin \mathrm{FV}(M) \\ (\lambda x.\,M)\,N & \to_\beta & M[x := N] & \\ \lambda x.\,M\,x & \to_\eta & M & x \notin \mathrm{FV}(M) \end{array}$$

Μετατροπές στο λ-λογισμό (ii)

- α-μετατροπή

$$\boxed{\lambda x.\,M} \;\to_\alpha\; \lambda y.\,M[x := y] \qquad y \notin \mathrm{FV}(M)$$

- Παραδείγματα

$$\begin{array}{lcll} \lambda x.\,x & \to_\alpha & \lambda y.\,y & \\ \lambda x.\,z\,x & \to_\alpha & \lambda y.\,z\,y & \\ \lambda x.\,\lambda y.\,z\,x\,y & \to_\alpha & \lambda y.\,\lambda w.\,z\,y\,w & \\ \lambda x.\,\lambda z.\,z\,x\,y & \not\to_\alpha & \lambda y.\,\lambda z.\,z\,y\,y & \text{γιατί;} \end{array}$$

Μετατροπές στο λ-λογισμό (iii)

- β-μετατροπή

  $\boxed{(\lambda x.\, M)\, N} \;\to_\beta\; M[x := N]$

- Παραδείγματα

$$
\begin{array}{lll}
(\lambda x.\, z\, x)\, w & \to_\beta & z\, w \\
(\lambda x.\, \lambda y.\, z\, x\, y)\, w & \to_\beta & \lambda y.\, z\, w\, y \\
(\lambda y.\, z\, y\, (\lambda x.\, x\, y))\, w & \to_\beta & z\, w\, (\lambda x.\, x\, w) \\
(\lambda x.\, \lambda y.\, z\, x\, y)\, (w\, y) & \not\to_\beta & \lambda y.\, z\, (w\, y)\, y \qquad \text{γιατί;} \\
(\lambda x.\, \lambda y.\, z\, x\, y)\, (w\, y) & \to_\beta & \lambda t.\, z\, (w\, y)\, t \qquad t \text{ νέα}
\end{array}
$$

Μετατροπές στο λ-λογισμό (iv)

- η-μετατροπή

  $\boxed{\lambda x.\, M\, x} \;\to_\eta\; M \qquad x \notin \mathrm{FV}(M)$

- Παραδείγματα

$$
\begin{array}{lll}
\lambda x.\, z\, x & \to_\eta & z \\
\lambda y.\, z\, x\, y & \to_\eta & z\, x \\
\lambda x.\, z\, x\, x & \not\to_\eta & z\, x \qquad \text{γιατί;}
\end{array}
$$

Μετατροπές στο λ-λογισμό (v)

- Αναγωγές και ισότητες (κλείσιμο της $\to$)

  $\twoheadrightarrow$ ανακλαστικό + μεταβατικό

  $=$ ανακλαστικό + μεταβατικό + συμμετρικό

- $M \twoheadrightarrow N$

$$M \equiv N_0 \to N_1 \to N_2 \to N_3 \to N_4 \to N_5 \equiv N$$

- $M = N$

$$M \equiv N_0 \to N_1 \leftarrow N_2 \to N_3 \leftarrow N_4 \leftarrow N_5 \equiv N$$

Κανονικές μορφές (i)

- Κανονική μορφή

  όρος N χωρίς β και η-redex

- Παραδείγματα

$$x \qquad \lambda x.\, x \qquad \lambda f.\, f\, (\lambda x.\, x\, f)$$

$$\lambda z.\, \boxed{(\lambda f.\, \boxed{\lambda x.\, f\, z\, x})\, (\lambda y.\, y)}$$

- Ισοδυναμία κανονικών μορφών

  Αν ο όρος $M \in \Lambda$ είναι σε κανονική μορφή και ισχύει $M \twoheadrightarrow N$ για κάποιον όρο $N \in \Lambda$, τότε $M =_\alpha N$.

Κανονικές μορφές (ii)

- Στρατηγική αναγωγής: ποιο redex να επιλεγεί για μετατροπή;

$$
\begin{array}{lll}
M_1 & \equiv & \lambda z.\, \boxed{(\lambda f.\, \lambda x.\, f\, z\, x)\, (\lambda y.\, y)} \\
& \to_\beta & \lambda z.\, \lambda x.\, \boxed{(\lambda y.\, y)\, z}\, x \\
& \to_\beta & \lambda z.\, \boxed{\lambda x.\, z\, x} \;\to_\eta\; \lambda z.\, z \\
M_1 & \equiv & \lambda z.\, \boxed{(\lambda f.\, \lambda x.\, f\, z\, x)\, (\lambda y.\, y)} \\
& \to_\beta & \lambda z.\, \boxed{\lambda x.\, (\lambda y.\, y)\, z}\, x \\
& \to_\eta & \lambda z.\, \boxed{(\lambda y.\, y)\, z} \;\to_\beta\; \lambda z.\, z
\end{array}
$$

Κανονικές μορφές (iii)

- Κανονικοποιήσιμος όρος

  όρος M με $M \twoheadrightarrow N$ και N κανονική μορφή

- Υπάρχουν όροι που δεν είναι κανονικοποιήσιμοι;

$$
\begin{array}{lll}
\Omega & \equiv & (\lambda x.\, x\, x)\, (\lambda x.\, x\, x) \\
& \to_\beta & \Omega \;\to_\beta\; \Omega \;\to_\beta\; \ldots \\
M_2 & \equiv & (\lambda x.\, x\, x\, y)\, (\lambda x.\, x\, x\, y) \\
& \to_\beta & M_2\, y \;\to_\beta\; M_2\, y\, y \;\to_\beta\; \ldots
\end{array}
$$

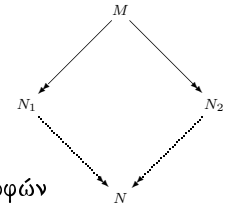
Κανονικές μορφές (iv)

- Ισχυρά κανονικοποιήσιμος όρος
  κάθε ακολουθία μετατροπών καταλήγει σε κανονική μορφή
- Υπάρχουν κανονικοποιήσιμοι όροι που δεν είναι ισχυρά κανονικοποιήσιμοι;

$$\begin{aligned} M_3 &\equiv (\lambda z.\, y)\boxed{((\lambda x.\, x\, x)\,(\lambda x.\, x\, x))} \\ &\to_\beta M_3 \to_\beta M_3 \to_\beta \ldots \\ M_3 &\equiv \boxed{(\lambda z.\, y)\,((\lambda x.\, x\, x)\,(\lambda x.\, x\, x))} \\ &\to_\beta y \end{aligned}$$

Ιδιότητες του λ-λογισμού (i)

- Θεώρημα Church-Rosser
  Έστω $M, N_1, N_2 \in \Lambda$
  τ.ω. $M \twoheadrightarrow N_1$ και $M \twoheadrightarrow N_2$.
  Τότε υπάρχει $N \in \Lambda$
  τ.ω. $N_1 \twoheadrightarrow N$ και $N_2 \twoheadrightarrow N$.



- Μοναδικότητα κανονικών μορφών
  αν υπάρχουν και modulo $=_\alpha$
- Θεώρημα κανονικοποίησης
  Αν ο όρος M έχει κανονική μορφή, τότε η μετατροπή του αριστερότερου redex οδηγεί σε αυτήν

Ιδιότητες του λ-λογισμού (ii)

- Τελεστής σταθερού σημείου: $F\,(Y\,F) = Y\,F$

$$Y \equiv \lambda f.\,(\lambda x.\, f\,(x\,x))\,(\lambda x.\, f\,(x\,x))$$

- Χρησιμεύει για την αναπαράσταση αναδρομικών συναρτήσεων

$$\begin{aligned} F &\equiv \lambda f.\,\lambda n.\,\textbf{if}\ n = 0\ \textbf{then}\ 1\ \textbf{else}\ n \cdot f(n-1) \\ M &\equiv Y\,F \qquad \text{παραγοντικό} \\ M\,2 &\equiv Y\,F\,2 = F\,(Y\,F)\,2 \\ &= \textbf{if}\ 2 = 0\ \textbf{then}\ 1\ \textbf{else}\ 2 \cdot (Y\,F)(2-1) \\ &= 2 \cdot Y\,F\,1 = 2 \cdot F\,(Y\,F)\,1 = \ldots \end{aligned}$$

Εκφραστική δύναμη (i)

- Λογικές τιμές

$$\begin{aligned} \textbf{true} &\equiv \lambda x.\,\lambda y.\, x \\ \textbf{false} &\equiv \lambda x.\,\lambda y.\, y \\ \textbf{not} &\equiv \lambda z.\, z\,\textbf{false}\,\textbf{true} \\ \textbf{if}\ B\ \textbf{then}\ N\ \textbf{else}\ M &\equiv B\,N\,M \end{aligned}$$

- Θεώρημα: $\textbf{not}\,\textbf{true} = \textbf{false}$

$$\begin{aligned} \textbf{not}\,\textbf{true} &\equiv (\lambda z.\, z\,\textbf{false}\,\textbf{true})\,\textbf{true} \\ &\to_\beta \textbf{true}\,\textbf{false}\,\textbf{true} \\ &\equiv (\lambda x.\,\lambda y.\, x)\,\textbf{false}\,\textbf{true} \twoheadrightarrow_\beta \textbf{false} \end{aligned}$$

Εκφραστική δύναμη (ii)

- Διατεταγμένα ζεύγη

$$\begin{aligned} \langle N, M\rangle &\equiv \lambda z.\, z\,N\,M \\ \textbf{fst} &\equiv \lambda z.\, z\,\textbf{true} \\ \textbf{snd} &\equiv \lambda z.\, z\,\textbf{false} \end{aligned}$$

- Θεωρήματα

$$\begin{aligned} \textbf{fst}\,\langle N, M\rangle &= N \\ \textbf{snd}\,\langle N, M\rangle &= M \end{aligned}$$

Εκφραστική δύναμη (iii)

- Φυσικοί αριθμοί (αριθμοειδή του Church)

$$\begin{aligned} \mathbf{c}_n &\equiv \lambda f.\,\lambda x.\, f^n(x) \\ F^0(A) &\equiv A \\ F^{n+1}(A) &\equiv F\,(F^n(A)) \\ \textbf{succ} &\equiv \lambda n.\,\lambda f.\,\lambda x.\, n\,f\,(f\,x) \\ \mathbf{A}_+ &\equiv \lambda n.\,\lambda m.\,\lambda f.\,\lambda x.\, n\,f\,(m\,f\,x) \\ \mathbf{A}_* &\equiv \lambda n.\,\lambda m.\,\lambda f.\, n\,(m\,f) \\ \mathbf{A}_{\exp} &\equiv \lambda n.\,\lambda m.\, m\,n \end{aligned}$$

Εκφραστική δύναμη (iv)

- Πέρασμα παραμέτρων στις γλώσσες προγραμματισμού.
  - οι λ-όροι αντιστοιχούν σε εκφράσεις ή εντολές
  - η αφαίρεση και η εφαρμογή αντιστοιχούν στον ορισμό και την κλήση συναρτήσεων ή διαδικασιών
  - η διαδικασία της αναγωγής αντιστοιχεί στην αποτίμηση εκφράσεων ή την εκτέλεση εντολών.

Εκφραστική δύναμη (v)

- Πέρασμα παραμέτρων (συνέχεια)
  - το πέρασμα κατ' αξία / πρόθυμη αποτίμηση (call-by-value / eager evaluation) αντιστοιχεί στη στρατηγική αποτίμησης που ανάγει ένα β-redex μόνο αν το όρισμα είναι κανονική τιμή
  - το πέρασμα κατ' όνομα / οκνηρή αποτίμηση (call-by-name / lazy evaluation) αντιστοιχεί στην στρατηγική αποτίμησης που ανάγει το αριστερότερο β-redex

Εξαγωγή τύπων (i)

(Type inference)

- Μετασχηματίζει εκφράσεις χωρίς τύπους ή με ελλιπείς τύπους σε εκφράσεις με σωστούς τύπους, συμπληρώνοντας τις πληροφορίες τύπων που λείπουν
- Ενδιαφέροντα θεωρητικά ζητήματα αλλά και πολύ σημαντικές πρακτικές εφαρμογές
- Ιδιαίτερα χρήσιμη σε γλώσσες που υποστηρίζουν (παραμετρικό) πολυμορφισμό

Εξαγωγή τύπων (ii)

Μία βιαστική εισαγωγή στα συστήματα τύπων

- Γλώσσα εκφράσεων e και γλώσσα τύπων τ
- Σχέση αντιστοίχισης τύπων $\Gamma \vdash e : \tau$
- Περιβάλλον τύπων Γ:
  απεικόνιση μεταβλητών x σε τύπους τ, π.χ.

$$\Gamma = \{ i : int,\ z : real,\ f : int \to int \}$$

- Κανόνες τύπων

$$\frac{\Gamma \vdash e_1 : int \qquad \Gamma \vdash e_2 : int}{\Gamma \vdash e_1 + e_2 : int}$$

Εξαγωγή τύπων (iii)

- Έστω L_T μια γλώσσα με δηλώσεις τύπων
- Έστω L_U μια παραλλαγή της ίδιας γλώσσας στην οποία οι δηλώσεις τύπων παραλείπονται
- Έστω μια συνάρτηση σβησίματος τύπων (type erasure function) $\mathsf{erase} : L_T \to L_U$
- Το πρόβλημα της εξαγωγής τύπων:
  δεδομένης μίας έκφρασης $e_U \in L_U$,
  βρες μία έκφραση $e_T \in L_T$ τέτοια ώστε
  - $\mathsf{erase}(e_T) = e_U$ και
  - $\Gamma \vdash e_T : \tau$ (για κάποια τ και Γ)

Εξαγωγή τύπων (iv)

- Ιδέα: μετασχηματισμός του προβλήματος $? \vdash e : ?$ σε ένα σύνολο E που περιέχει εξισώσεις τύπων, π.χ.

$$E = \{\ \alpha = \beta,\ \gamma \to \alpha = (\beta \to \alpha) \to \gamma\ \}$$

- Αν το E έχει λύση, βρίσκουμε Γ και τ τέτοια ώστε $\Gamma \vdash e : \tau$
- Διαφορετικά δεν υπάρχουν Γ και τ τέτοια ώστε $\Gamma \vdash e : \tau$

Μερικά παραδείγματα (i)

Με τον interpreter της OCaml www.ocaml.org

```
# let inc (x : int) : int = x + 1;;
val f : int -> int = <fun>
# let inc x = x + 1;;
val f : int -> int = <fun>
```

- Πώς βρήκε τον τύπο;
  1. έστω inc : $\alpha \to \beta$, δηλαδή x : α και x + 1 : β
  2. πρέπει x : int άρα $\alpha = int$
  3. τότε x + 1 : int άρα και $\beta = int$
  4. επομένως inc : $int \to int$

Μερικά παραδείγματα (ii)

- Πολυμορφισμός

```
# let id x = x;;
val id : 'a -> 'a = <fun>
# let fst (x, y) = x;;
val fst : 'a * 'b -> 'a = <fun>
# let rec map f l =
    match l with
    | []     -> []
    | h :: t -> f h :: map f t;;
val map : ('a -> 'b) -> 'a list -> 'b list = <fun>
```


Μερικά παραδείγματα (iii)

- Περιορισμός: let polymorphism

```
# let strange f = (f 5, f "hello");;
Characters 24-31:
  let strange f = (f 5, f "hello");;
                          ^^^^^^^
This expression has type string
but is here used with type int
```

- ο τύπος του f είναι μονομορφικός!

Εξαγωγή τύπων στην πράξη (i)

- Προσθέτουμε μεταβλητές στη γλώσσα των τύπων
- Έστω $\{$ @1, @2, ... $\}$ ένα αριθμήσιμο υποσύνολο των μεταβλητών τύπων (ακόμα άγνωστοι τύποι)
- Συμπληρώνουμε τους τύπους που λείπουν στην αρχική έκφραση βάζοντας φρέσκες μεταβλητές

```
let f g x = g x (x + 1)
and m a b = a * b
```

γίνεται:

```
let f (g : @1) (x : @2) : @3 = g x (x + 1)
and m (a : @4) (b : @5) : @6 = a * b
```


Εξαγωγή τύπων στην πράξη (ii)

- Αντικατάσταση τύπων (type substitution): απεικόνιση μεταβλητών τύπων σε τύπους

$$\sigma \;=\; [\alpha \mapsto int,\ \beta \mapsto bool \to \alpha]$$

- Εφαρμογή αντικατάστασης τύπων: ταυτόχρονα και μία φορά

$$\sigma(\alpha) \;=\; int \qquad \sigma(\beta) \;=\; bool \to \alpha$$
$$\sigma(\beta \to \gamma) \;=\; (bool \to \alpha) \to \gamma$$

- Σύνθεση αντικαταστάσεων $\sigma_1 \circ \sigma_2$ έτσι ώστε $(\sigma_1 \circ \sigma_2)(\tau) = \sigma_1(\sigma_2(\tau))$

Εξαγωγή τύπων στην πράξη (iii)

- Το πρόβλημα $? \vdash e : ?$ ανάγεται στην εύρεση μίας αντικατάστασης σ και ενός τύπου τ ώστε $\sigma(\Gamma) \vdash \sigma(e) : \tau$ για το αρχικό περιβάλλον Γ
- Παράδειγμα

```
let f (g : @1) (x : @2) : @3 = g x (x + 1)
and m (a : @4) (b : @5) : @6 = a * b
in  f m 6
```

- Αρχικό περιβάλλον: $\Gamma = \emptyset$
- Αρχικό περιβάλλον για το σώμα f m 6: $\Gamma_b = \{$ f : @1 $\to$ @2 $\to$ @3, m : @4 $\to$ @5 $\to$ @6 $\}$

Εξαγωγή τύπων στην πράξη (iv)

- Παράδειγμα (συνέχεια)

```
let f (g : @1) (x : @2) : @3 = g x (x + 1)
and m (a : @4) (b : @5) : @6 = a * b
in  f m 6
```

- Μια λύση (η μοναδική)

$$\sigma = [\ \texttt{@1} \mapsto int \to int \to int,\ \texttt{@2} \mapsto int,\ \texttt{@3} \mapsto int,\ \texttt{@4} \mapsto int,\ \texttt{@5} \mapsto int,\ \texttt{@6} \mapsto int\ ]$$
$$\tau = int$$

- Γενικά οι λύσεις δεν είναι μοναδικές!

```
let id (x : @1) : @2 = x
```


Περιορισμοί και επίλυση (i)

- Πώς βρίσκεται η λύση;
- Περιορισμός (constraint): εξίσωση τύπων $\tau_1 = \tau_2$
- Πρόβλημα 1: εύρεση συνόλου περιορισμών C

```
let f (g : @1) (x : @2) : @3 = g x (x + 1)
and m (a : @4) (b : @5) : @6 = a * b
in  f m 6
```

οδηγεί στο σύνολο περιορισμών:

$$C = \{\ \texttt{@1} = \texttt{@2} \to int \to \texttt{@3},\ \texttt{@2} = int,\ \texttt{@4} = int,\ \texttt{@5} = int,\ \texttt{@6} = int,\ \texttt{@1} = \texttt{@4} \to \texttt{@5} \to \texttt{@6},\ \texttt{@2} = int\ \}$$

Περιορισμοί και επίλυση (ii)

- Παραγωγή τύπων με περιορισμούς

$$\Gamma \vdash E : \tau \mid C$$

- Πρόβλημα 2: επίλυση συνόλου περιορισμών
- Μια αντικατάσταση σ λέγεται ενοποιητής (unifier) για τον περιορισμό $\tau_1 = \tau_2$ αν οι τύποι $\sigma(\tau_1)$ και $\sigma(\tau_2)$ ταυτίζονται $\sigma(\tau_1) \equiv \sigma(\tau_2)$
- Λύση για το πρόβλημα της εξαγωγής τύπων:
  - ένας ενοποιητής σ για κάθε περιορισμό του C
  - ο τύπος $\tau = \sigma(\tau')$

Περιορισμοί και επίλυση (iii)

- Ο ενοποιητής σ είναι πιο γενικός από τον σ' αν υπάρχει αντικατάσταση σ_δ τέτοια ώστε $\sigma' = \sigma_\delta \circ \sigma$
- Πιο γενικός ενοποιητής (most general unifier): ενοποιητής σ τέτοιος ώστε να είναι πιο γενικός από κάθε άλλον ενοποιητή σ'
- Αν υπάρχει πιο γενικός ενοποιητής, αυτός δίνει τον πρωτεύοντα τύπο (principal type)
- Ο αλγόριθμος W για το λ-λογισμό (διαφ. 42) υπολογίζει τον πιο γενικό ενοποιητή

Εφαρμογή στο λ-λογισμό (i)

- Τύποι

$$\sigma, \tau ::= \alpha \mid (\sigma \to \tau)$$

Το $\to$ είναι δεξιά προσεταιριστικό, π.χ.

$$(\alpha \to (\beta \to \gamma)) \qquad \alpha \to \beta \to \gamma$$

- Κανόνες τύπων à-la Curry

$$\frac{(x : \sigma) \in \Gamma}{\Gamma \vdash x : \sigma} \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x.\, M) : (\sigma \to \tau)}$$

$$\frac{\Gamma \vdash M : (\sigma \to \tau) \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash (M\, N) : \tau}$$

Εφαρμογή στο λ-λογισμό (ii)

- Ενοποίηση: επίλυση συνόλου περιορισμών

unify$(\emptyset) = \sigma_0$ {η κενή αντικατάσταση}
unify$(\{\ \tau_1 = \tau_2\ \} \cup C) =$
  if $\tau_1 \equiv \tau_2$ **then**
    unify(C)
  else if $\tau_1 \equiv \alpha$ και δεν εμφανίζεται στο τ_2 **then**
    unify$([\alpha \mapsto \tau_2]C) \circ [\alpha \mapsto \tau_2]$
  else if $\tau_2 \equiv \alpha$ και δεν εμφανίζεται στο τ_1 **then**
    unify$([\alpha \mapsto \tau_1]C) \circ [\alpha \mapsto \tau_1]$
  else if $\tau_1 \equiv \tau_{11} \to \tau_{12}$ και $\tau_2 \equiv \tau_{21} \to \tau_{22}$ **then**
    unify$(C \cup \{\ \tau_{11} = \tau_{21}, \tau_{12} = \tau_{22}\ \})$
  else
    η ενοποίηση αποτυγχάνει