

Γλώσσες Προγραμματισμού II

<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 μεταβλητή εφαρμογή αφαίρεση
 - x
 - $F A$
 - $\lambda x. E[x]$
- 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: $\int_{-\pi}^{\pi} \frac{\sin x + \cos y}{\cos x - \sin y} dx$

- 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 $\int_{-\pi}^{\pi} \frac{\sin x + \cos y}{\cos x - \sin y} dx$
- 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

$$(\lambda x. x) \quad \lambda x. x$$

- Η εφαρμογή είναι αριστερά προσηταιριστική

$$(\dots ((F M_1) M_2) \dots M_n) \quad F M_1 M_2 \dots M_n$$

- Η αφαίρεση εκτείνεται όσο είναι δυνατόν

$$\lambda x. (M_1 M_2 \dots M_n) \quad \lambda x. M_1 M_2 \dots M_n$$

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: $M \equiv N$

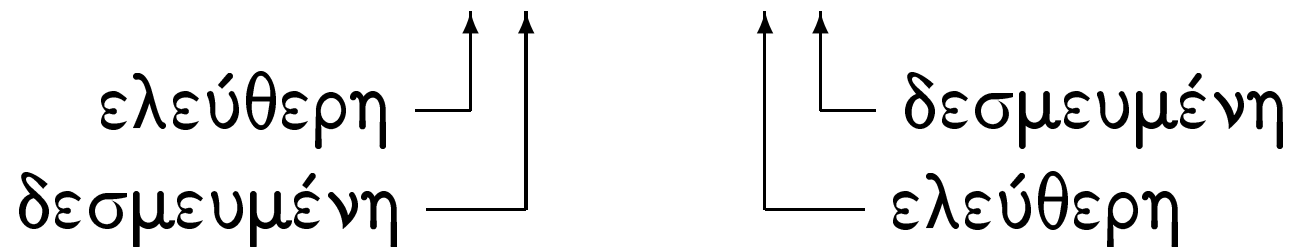
$$x \equiv y \quad \text{αν } x = y$$

$$(M \ N) \equiv (P \ Q) \quad \text{αν } M \equiv P \text{ και } N \equiv Q$$

$$(\lambda x. M) \equiv (\lambda y. N) \quad \text{αν } x = y \text{ και } M \equiv N$$

- Ελεύθερες και δεσμευμένες μεταβλητές

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



λ-λογισμός χωρίς τύπους

(iv)

- Ελεύθερες μεταβλητές

$$\text{FV}(x) = \{x\}$$

$$\text{FV}(M N) = \text{FV}(M) \cup \text{FV}(N)$$

$$\text{FV}(\lambda x. M) = \text{FV}(M) - \{x\}$$

- Κλειστοί όροι (closed terms ή combinators)

$$I \equiv \lambda x. x$$

$$K \equiv \lambda x. \lambda y. x$$

$$K_* \equiv \lambda x. \lambda y. y$$

$$S \equiv \lambda x. \lambda y. \lambda z. (x z) (y z)$$

λ-λογισμός χωρίς τύπους

(iii)

■ Αντικατάσταση

$$x[x := N] \equiv N$$

$$y[x := N] \equiv y \quad \text{αν } y \neq x$$

$$(P Q)[x := N] \equiv P[x := N] Q[x := N]$$

$$(\lambda x. P)[x := N] \equiv \lambda x. P \quad \text{⏏}$$

$$(\lambda y. P)[x := N] \equiv \lambda y. P[x := N] \quad \text{⏏}$$

αν $y \neq x$, και $(y \notin \text{FV}(N) \text{ ή } x \notin \text{FV}(P))$

$$(\lambda y. P)[x := N] \equiv \lambda z. P[y := z][x := N]$$

αν $y \neq x$, και $y \in \text{FV}(N)$ και $x \in \text{FV}(P)$,

όπου $z \notin \text{FV}(P) \cup \text{FV}(N)$

Μετατροπές στο λ-λογισμό (i)

- Συμβατές σχέσεις

$$M \sim N \Rightarrow (M P) \sim (N P)$$

$$M \sim N \Rightarrow (P M) \sim (P N)$$

$$M \sim N \Rightarrow (\lambda x. M) \sim (\lambda x. N)$$

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

$$\lambda x. M \rightarrow_\alpha \lambda y. M[x := y] \quad y \notin \text{FV}(M)$$

$$(\lambda x. M) N \rightarrow_\beta M[x := N]$$

$$\lambda x. M x \rightarrow_\eta M \quad x \notin \text{FV}(M)$$

Μετατροπές στο λ-λογισμό (ii)

■ α-μετατροπή

$$\boxed{\lambda x. M} \rightarrow_{\alpha} \lambda y. M[x := y] \quad y \notin \text{FV}(M)$$

■ Παραδείγματα

$$\lambda x. x \rightarrow_{\alpha} \lambda y. y$$

$$\lambda x. z x \rightarrow_{\alpha} \lambda y. z y$$

$$\lambda x. \lambda y. z x y \rightarrow_{\alpha} \lambda y. \lambda w. z y w$$

$$\lambda x. \lambda z. z x y \not\rightarrow_{\alpha} \lambda y. \lambda z. z y y \quad \text{γιατί;}$$

Μετατροπές στο λ-λογισμό (iii)

■ β-μετατροπή

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

■ Παραδείγματα

$$(\lambda x. z x) w \rightarrow_{\beta} z w$$

$$(\lambda x. \lambda y. z x y) w \rightarrow_{\beta} \lambda y. z w y$$

$$(\lambda y. z y (\lambda x. x y)) w \rightarrow_{\beta} z w (\lambda x. x w)$$

$$(\lambda x. \lambda y. z x y) (w y) \not\rightarrow_{\beta} \lambda y. z (w y) y \quad \text{γιατί;}$$

$$(\lambda x. \lambda y. z x y) (w y) \rightarrow_{\beta} \lambda t. z (w y) t \quad t \text{ νέα}$$

Μετατροπές στο λ-λογισμό (iv)

■ η-μετατροπή

$$\boxed{\lambda x. M x} \rightarrow_{\eta} M \quad x \notin \text{FV}(M)$$

■ Παραδείγματα

$$\lambda x. z x \rightarrow_{\eta} z$$

$$\lambda y. z x y \rightarrow_{\eta} z x$$

$$\lambda x. z x x \not\rightarrow_{\eta} z x \quad \text{γιατί;}$$

Μετατροπές στο λ-λογισμό (v)

- Αναγωγές και ισότητες (κλείσιμο της \rightarrow)

\rightarrow ανακλαστικό + μεταβατικό

$=$ ανακλαστικό + μεταβατικό + συμμετρικό

- $M \rightarrow^* N$

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

- $M = N$

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

Κανονικές μορφές

(i)

- Κανονική μορφή
όρος N χωρίς β και η -redex
- Παραδείγματα

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

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

- 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;

$$M_1 \equiv \lambda z. \boxed{(\lambda f. \lambda x. f \ z \ x) (\lambda y. y)}$$

$$\rightarrow_{\beta} \lambda z. \lambda x. \boxed{(\lambda y. y) \ z} \ x$$

$$\rightarrow_{\beta} \lambda z. \boxed{\lambda x. z \ x} \rightarrow_{\eta} \lambda z. z$$

$$M_1 \equiv \lambda z. \boxed{(\lambda f. \lambda x. f \ z \ x) (\lambda y. y)}$$

$$\rightarrow_{\beta} \lambda z. \boxed{\lambda x. (\lambda y. y) \ z} \ x$$

$$\rightarrow_{\eta} \lambda z. \boxed{(\lambda y. y) \ z} \rightarrow_{\beta} \lambda z. z$$

Κανονικές μορφές

(iii)

- Κανονικοποιήσιμος όρος
όρος M με $M \rightarrow_\beta N$ και N κανονική μορφή
- Υπάρχουν όροι που δεν είναι
κανονικοποιήσιμοι;

$$\begin{aligned}\Omega &\equiv (\lambda x. x x) (\lambda x. x x) \\ &\rightarrow_\beta \Omega \rightarrow_\beta \Omega \rightarrow_\beta \dots \\ M_2 &\equiv (\lambda x. x x y) (\lambda x. x x y) \\ &\rightarrow_\beta M_2 y \rightarrow_\beta M_2 y y \rightarrow_\beta \dots\end{aligned}$$

Κανονικές μορφές

(iv)

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

$$M_3 \equiv (\lambda z. y) \boxed{((\lambda x. x x) (\lambda x. x x))}$$

$$\rightarrow_{\beta} M_3 \rightarrow_{\beta} M_3 \rightarrow_{\beta} \dots$$

$$M_3 \equiv \boxed{(\lambda z. y) ((\lambda x. x x) (\lambda x. x x))}$$

$$\rightarrow_{\beta} y$$

Ιδιότητες του λ-λογισμού (i)

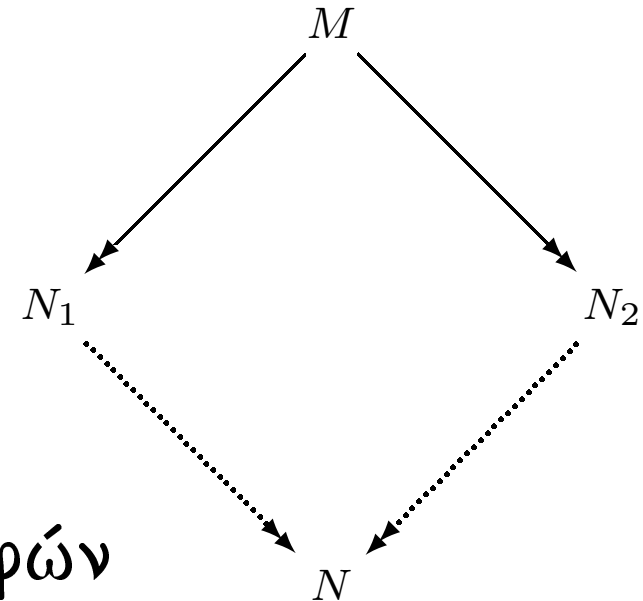
- Θεώρημα Church-Rosser

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

$$F \equiv \lambda f. \lambda n. \text{if } n = 0 \text{ then } 1 \text{ else } n \cdot f(n - 1)$$

$$M \equiv Y F \quad \text{παραγοντικό}$$

$$\begin{aligned} M 2 &\equiv Y F 2 = F (Y F) 2 \\ &= \text{if } 2 = 0 \text{ then } 1 \text{ else } 2 \cdot (Y F)(2 - 1) \\ &= 2 \cdot Y F 1 = 2 \cdot F (Y F) 1 = \dots \end{aligned}$$

Εκφραστική δύναμη

(i)

■ Λογικές τιμές

`true` $\equiv \lambda x. \lambda y. x$

`false` $\equiv \lambda x. \lambda y. y$

`not` $\equiv \lambda z. z \text{ false true}$

`if B then N else M` $\equiv B N M$

■ Θεώρημα: `not true = false`

`not true` $\equiv (\lambda z. z \text{ false true}) \text{ true}$

$\rightarrow_{\beta} \text{true false true}$

$\equiv (\lambda x. \lambda y. x) \text{ false true} \twoheadrightarrow_{\beta} \text{false}$

Εκφραστική δύναμη

(ii)

- Διατεταγμένα ζεύγη

$$\langle N, M \rangle \equiv \lambda z. z \ N \ M$$

$$\text{fst} \equiv \lambda z. z \ \text{true}$$

$$\text{snd} \equiv \lambda z. z \ \text{false}$$

- Θεωρήματα

$$\text{fst} \ \langle N, M \rangle = N$$

$$\text{snd} \ \langle N, M \rangle = M$$

Εκφραστική δύναμη

(iii)

- Φυσικοί αριθμοί (αριθμοειδή του Church)

$$\mathbf{c}_n \equiv \lambda f. \lambda x. f^n(x)$$

$$F^0(A) \equiv A$$

$$F^{n+1}(A) \equiv F(F^n(A))$$

$$\mathbf{succ} \equiv \lambda n. \lambda f. \lambda x. n f (f x)$$

$$\mathbf{A}_+ \equiv \lambda n. \lambda m. \lambda f. \lambda x. n f (m f x)$$

$$\mathbf{A}_* \equiv \lambda n. \lambda m. \lambda f. n (m f)$$

$$\mathbf{A}_{\text{exp}} \equiv \lambda n. \lambda m. m n$$

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 ελλειπείς** 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.

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

- **Κανόνες τύπων**

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

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) $\text{erase} : L_T \rightarrow L_U$
- Το πρόβλημα της εξαγωγής τύπων:
δεδομένης μίας έκφρασης $e_U \in L_U$,
βρες μία έκφραση $e_T \in L_T$ τέτοια ώστε
 - $\text{erase}(e_T) = e_U$ 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.

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

- 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 $inc : \alpha \rightarrow \beta$, δηλαδή $x : \alpha$ και $x + 1 : \beta$
2. πρέπει $x : int$ άρα $\alpha = int$
3. τότε $x + 1 : int$ άρα και $\beta = int$
4. επομένως $inc : int \rightarrow int$

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 $\{ @1, @2, \dots \}$ 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

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

- Εφαρμογή αντικατάστασης τύπων: ταυτόχρονα και μία φορά

$$\sigma(\alpha) = int \qquad \sigma(\beta) = bool \rightarrow \alpha$$

$$\sigma(\beta \rightarrow \gamma) = (bool \rightarrow \alpha) \rightarrow \gamma$$

- 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`:
 $\Gamma_b = \{ f : @1 \rightarrow @2 \rightarrow @3, m : @4 \rightarrow @5 \rightarrow @6 \}$

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)

$$\sigma = [\begin{array}{lll} @1 \mapsto int \rightarrow int \rightarrow int, & @2 \mapsto int, & @3 \mapsto int, \\ @4 \mapsto int, & @5 \mapsto int, & @6 \mapsto int \end{array}]$$
$$\tau = int$$

■ 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:

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

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

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

Το \rightarrow είναι δεξιά προσεταιριστικό, π.χ.

$$(\alpha \rightarrow (\beta \rightarrow \gamma)) \quad \alpha \rightarrow \beta \rightarrow \gamma$$

■ Κανόνες τύπων à-la Curry

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

$$\frac{\Gamma \vdash M : (\sigma \rightarrow \tau) \quad \Gamma \vdash N : \sigma}{\Gamma \vdash (M N) : \tau}$$

Εφαρμογή στο λ-λογισμό (ii)

- **Ενοποίηση**: επίλυση συνόλου περιορισμών

$\text{unify}(\emptyset) = \sigma_0$ {η κενή αντικατάσταση}

$\text{unify}(\{ \tau_1 = \tau_2 \} \cup C) =$

if $\tau_1 \equiv \tau_2$ **then**

$\text{unify}(C)$

else if $\tau_1 \equiv \alpha$ και δεν εμφανίζεται στο τ_2 **then**

$\text{unify}([\alpha \mapsto \tau_2]C) \circ [\alpha \mapsto \tau_2]$

else if $\tau_2 \equiv \alpha$ και δεν εμφανίζεται στο τ_1 **then**

$\text{unify}([\alpha \mapsto \tau_1]C) \circ [\alpha \mapsto \tau_1]$

else if $\tau_1 \equiv \tau_{11} \rightarrow \tau_{12}$ και $\tau_2 \equiv \tau_{21} \rightarrow \tau_{22}$ **then**

$\text{unify}(C \cup \{ \tau_{11} = \tau_{21}, \tau_{12} = \tau_{22} \})$

else

η ενοποίηση αποτυγχάνει