

Chapter 10

λ - Calculus

SLIDES TAKEN FROM National University of Athens

Ιστορική εξέλιξη λ-λογισμού - 1

- Αναπτύχθηκε αρχικά από τον Alonzo Church στις αρχές της δεκαετίας του 1930, πολύ πριν αρχίσουν να χρησιμοποιούνται οι ηλεκτρονικοί υπολογιστές.
- Ήταν μέρος μιας γενικότερης θεωρίας με στόχο τη θεμελίωση μαθηματικών και λογικής [Chur32, Chur33]

Ιστορική εξέλιξη λ-λογισμού - 2

- Η γενική θεωρία ήταν ασυνεπής, όπως αποδείχθηκε αργότερα [Klee35]
- Το τμήμα της, που ασχολήθηκε με συναρτήσεις είχε σημαντικές εφαρμογές στην πληροφορική, κυρίως μετά το 1960
- Το τμήμα αυτό είναι ο λάμβδα λογισμός ή λ-λογισμός

λ-Λογισμός

- Πλήρες υπολογιστικό μοντέλο
- Δύο πολύ σημαντικά αποτελέσματα
 - Όλες οι αναδρομικές συναρτήσεις παριστάνονται στο λ-λογισμό [Klee35]
 - Ως υπολογιστικό μοντέλο, ο λ-λογισμός είναι ισοδύναμος με τη μηχανή Turing [Turi37].
 - Η μηχανή Turing αποτέλεσε τη βάση των υπολογιστών von Neumann στους οποίους ανήκουν οι σημερινοί υπολογιστές και οδήγησε στη δημιουργία των πρώτων γλωσσών προγραμματισμού

λ-Λογισμός

- Σχεδιασμός νέων αρχιτεκτονικών υπολογιστών
 - Μηχανές αναγωγής (reduction machines) και
 - Υπολογιστές ροής δεδομένων (data-flow computers)
 - Όταν πρωτο-δημιουργήθηκαν εκτελούσαν αποκλειστικά προγράμματα γραμμένα σε κάποια διάλεκτο του λ-λογισμού.
- Δημιουργία συναρτησιακού προγραμματισμού
 - John McCarthy σχεδίασε τη γλώσσα προγραμματισμού LISP στα τέλη της δεκαετίας του 1950. Αργότερα δημιουργήθηκαν οι Scheme, ML, Miranda, Haskell, κλπ.

λ- Λογισμός

- Υπολογιστές και συναρτησιακός προγραμματισμός δεν έτυχαν ευρείας αποδοχής όπως οι υπολογιστές von Neumann και ο προστακτικός προγραμματισμός
- Οι επιδόσεις των πρώτων μηχανών αναγωγής και οι υλοποιήσεις των συναρτησιακών γλωσσών ήταν χειρότερες από τα παραδοσιακά συστήματα.

λ-Λογισμός

- Ιδιαίτερα πρόσφορος ως συμβολισμός για την περιγραφή σημασιολογικών ιδιοτήτων των γλωσσών προγραμματισμού.
- Διευκόλυνε τη μελέτη και απομόνωση προβλημάτων σχεδίασης και υλοποίησης των γλωσσών προγραμματισμού
 - Μηχανισμό κλήσης υπο-προγραμμάτων και δομή συστήματος τύπων

λ-Λογισμός

- Διατύπωση θεωρίας πεδίων
- Θεμελίωση ερευνητικού πεδίου της σημασιολογίας γλωσσών προγραμματισμού

Διαισθητική εισαγωγή

- Ο λ-λογισμός είναι θεωρία συναρτήσεων
- Δύο κύριες λειτουργίες
 - *Εφαρμογή* συνάρτησης F πάνω σε ένα όρισμα A , που συμβολίζεται με $F A$
 - *Αφαίρεση*. Έστω ότι x μεταβλητή και $E[x]$ έκφραση, που εξαρτάται από τη x . Η έκφραση $\lambda x.E[x]$ συμβολίζει τη συνάρτηση

$$x \mapsto E[x]$$

Διαισθητική εισαγωγή

- Η συνάρτηση δέχεται ως όρισμα μία τιμή u και επιστρέφει ως αποτέλεσμα την τιμή $E[u]$. Η x δεν εμφανίζεται απαραίτητα στην έκφραση $E[x]$. Αν αυτό δεν συμβαίνει, τότε η $\lambda x.E[x]$ είναι μία σταθερή συνάρτηση.

Παράδειγμα

Η λχ. $x^2 - 3x + 2$ συμβολίζει μία συνάρτηση, που σε κάθε τιμή x απεικονίζει την τιμή $x^2 - 3x + 2$.

Αν η συνάρτηση εφαρμοσθεί στο όρισμα 8, προκύπτει

$$(\text{λχ. } x^2 - 3x + 2)8 = 8^2 - 3 \cdot 8 + 2 = 42$$

Η τιμή του ορίσματος αντικαθιστά την παράμετρο x

Ελεύθερη και Δεσμευμένη Μεταβλητή

- Η αφαίρεση λχ. $E[x]$ δεσμεύει τη μεταβλητή x μέσα στην έκφραση $E[x]$.
- Μεταβλητή μη δεσμευμένη ονομάζεται ελεύθερη.

Παράδειγμα: λχ. $x^2 - 3y + 2$

x δεσμευμένη και y ελεύθερη

Παράδειγμα: (λχ. $x^2 - 3y + 2$)($4x + 1$)

Η πρώτη εμφάνιση του x (στο x^2) είναι δεσμευμένη.

Παράδειγμα Δέσμμευσης

$$\int \frac{\sin x + \cos y}{\cos x - \sin y} dx$$

Η μεταβλητή x στο εσωτερικό του ολοκληρώματος είναι δεσμευμένη. Η μεταβλητή y είναι ελεύθερη.

Λάμβδα όροι

Ο λ-λογισμός είναι μία τυπική γλώσσα Λ , η σύνταξη της οποίας δίνεται από τον ακόλουθο επαγωγικό ορισμό:

Ορισμός 10.1

Έστω V ένα αριθμήσιμο σύνολο μεταβλητών. Το σύνολο Λ των όρων του λ-λογισμού είναι το μικρότερο σύνολο, που ικανοποιεί τις παρακάτω ιδιότητες:

$$x \in V \Rightarrow x \in \Lambda$$

$$M, N \in \Lambda \Rightarrow (M \ N) \in \Lambda$$

$$x \in V, M \in \Lambda \Rightarrow (\lambda x.M) \in \Lambda$$

λ-όροι

Τα στοιχεία του συνόλου Λ ονομάζονται επίσης λ-όροι (λ-terms). Υπάρχουν τριών ειδών:

Μεταβλητές (Variables), δηλαδή στοιχεία του συνόλου V

Εφαρμογές (Applications), με μορφή $(M N)$, όπου M και N είναι λ-όροι

Αφαιρέσεις (Abstractions), με μορφή $(\lambda x.M)$, όπου x μεταβλητή και M λ-όρος

λ-όροι

- Κατά σύμβαση χρησιμοποιούνται μικρά γράμματα του λατινικού αλφαβήτου (x,y,z κλπ) για συμβολισμό μεταβλητών και κεφαλαία (M,N,F,G, P κλπ) για λ-όρους.

λ-όροι

- Χρησιμοποιώντας αφηρημένη σύνταξη BNF και θεωρώντας ότι η συντακτική κλάση των μεταβλητών παριστάνεται με το μη-τερματικό σύμβολο (var), η γλώσσα Λ των λ-όρων περιγράφεται ισοδύναμα

$\langle \text{term} \rangle ::= \langle \text{var} \rangle$

| $(\langle \text{term} \rangle \langle \text{term} \rangle)$

| $(\lambda \langle \text{var} \rangle . \langle \text{term} \rangle)$

Παράδειγμα 10.1

Οι παρακάτω είναι λ-όροι:

$(x\ y)$

$(\lambda x.x)$

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

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

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

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

Συμβάσεις

Για απλοποίηση των λ-όρων και αποφυγή μεγάλου αριθμού παρενθέσεων (με αυστηρή τήρηση ορισμού 10.1) χρησιμοποιούνται οι συμβάσεις

- Εξωτερικές παρενθέσεις δεν γράφονται
 - λx.x συντομογραφία του (λx.x)
- Η εφαρμογή είναι αριστερά προσεταιριστική
 - $F M_1 M_2 \dots M_n$ συντομογραφία του $(\dots((F M_1)M_2)\dots M_n)$

Συμβάσεις

- Η αφαίρεση εκτείνεται όσο περισσότερο είναι δυνατό, δηλαδή ως το επόμενο κλείσιμο παρένθεσης ή το τέλος του όρου:
 - λχ. $M_1 M_2 \dots M_n$ συντομογραφία του
λχ. $(M_1 M_2 \dots M_n)$

Παράδειγμα 10.2

Ακολουθώντας τις συμβάσεις, οι λ-όροι του παραδείγματος 10.1 γράφονται

$x \ y$

$\lambda x.x$

$\lambda x.\lambda y.x \ y$

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

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

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

Ορισμός 10.2

Η σχέση ταυτότητας \equiv στο σύνολο των λ-όρων ορίζεται επαγωγικά ως εξής:

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

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

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

Όπου $x = y$ συμβολίζεται η σχέση ισότητας στο σύνολο V (δηλαδή x και y είναι η ίδια μεταβλητή). Αν $M \equiv N$, οι όροι $M, N \in \Lambda$ ονομάζονται ταυτόσημοι (identical)

Μεταβλητές

- Η μεταβλητή x στην αφαίρεση $\lambda x.M$ ονομάζεται δεσμεύουσα μεταβλητή (binding variable)
- Η εμβέλεια (scope) της αφαίρεσης λx είναι ο λ-όρος M , εκτός από τυχόν αφαιρέσεις, που αυτός περιέχει και στις οποίες δεσμεύουσα μεταβλητή είναι πάλι η x .

Μεταβλητές

- Εμφανίσεις της μεταβλητής x που βρίσκονται στην εμβέλεια κάποιου λx ονομάζονται δεσμευμένες (bound).
- Εμφανίσεις, που δεν βρίσκονται στην εμβέλεια κανενός λx ονομάζονται ελεύθερες.

Ορισμός 10.3

Το σύνολο των ελεύθερων μεταβλητών (free variables) ενός λ-όρου $M \in \Lambda$ συμβολίζεται με $FV(M)$ και ορίζεται επαγωγικά ως εξής:

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

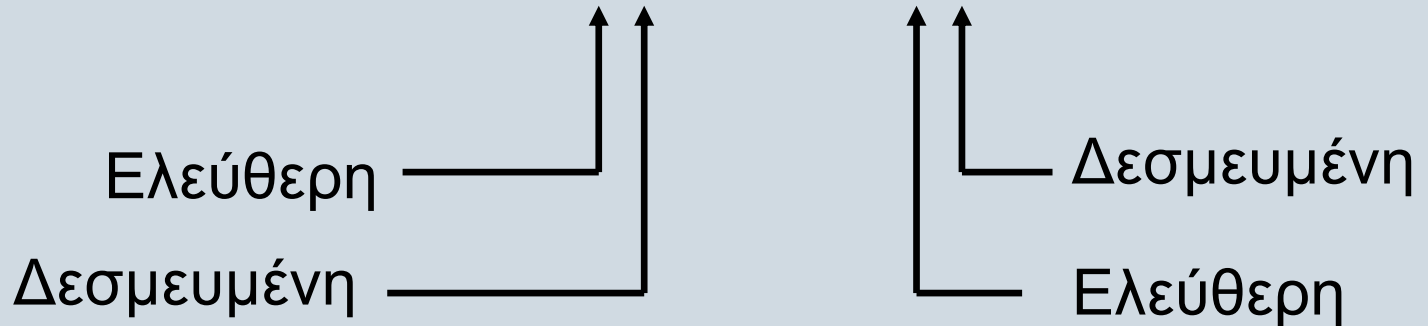
$$FV(M N) = FV(M) \cup FV(N)$$

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

Παράδειγμα 10.3

Έστω ο παρακάτω όρος:

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



Παράδειγμα 10.3

Ακολουθώντας τον ορισμό 10.3

$$\begin{aligned}FV(M) &= FV(\lambda x.y x)(\lambda y.x y) \\ &= FV(\lambda x.y x) \cup FV(\lambda y.x y) \\ &= (FV(y x) - \{x\}) \cup (FV(x y) - \{y\}) \\ &= ((FV(y) \cup FV(x)) - \{x\}) \cup ((FV(x) \cup FV(y)) - \{y\}) \\ &= ((\{y\} \cup \{x\}) - \{x\}) \cup ((\{x\} \cup \{y\}) - \{y\}) \\ &= (\{x,y\} - \{x\}) \cup (\{x,y\} - \{y\}) \\ &= \{y\} \cup \{x\} \\ &= \{x,y\}\end{aligned}$$

Ορισμός 10.4

Ένας λ-όρος $M \in \Lambda$ ονομάζεται κλειστός λ-όρος (closed λ-term ή combinator) αν $FV(M) = \emptyset$. Το σύνολο των κλειστών λ-όρων συμβολίζεται με Λ^0 .

Παράδειγμα 10.4

Οι ακόλουθοι λ-όροι είναι κλειστοί και στη βιβλιογραφία ονομάζονται *πρότυποι κλειστοί όροι* (standard combinators).

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

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

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

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

Αντικατάσταση

- Βασική πράξη συμβολικής επεξεργασίας των λ-όρων
- Επηρεάζει ελεύθερες μεταβλητές
- Ο συμβολισμός $M[x := N]$ παριστάνει το αποτέλεσμα αντικατάστασης στον όρο M όλων των **ελεύθερων εμφανίσεων της μεταβλητής x** με τον όρο N .

Μετατροπές

- Τρία είδη μετατροπής (α,β και η)
- Κάθε είδος μετατροπής περιγράφεται από τον κανόνα $M \rightarrow_{\chi} N$ όπου $\chi \in \{\alpha, \beta, \eta\}$ και $M, N \in \Lambda$. Ο όρος M ονομάζεται χ -redex ή απλά redex και ο N όρος contractum.

Ορισμός 10.5

Μία σχέση \sim πάνω στο Λ ονομάζεται συμβατή (compatible) αν για κάθε $x \in V$ και για κάθε $M, N, P \in \Lambda$ ισχύουν οι παρακάτω ιδιότητες:

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

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

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

Η σχέση ταυτότητας \equiv μεταξύ των λ -όρων είναι συμβατή.

Ορισμός 10.6

Μία σχέση αναγωγής (reduction relation) είναι μία συμβατή, ανακλαστική (αντιμεταθετική) και μεταβατική σχέση πάνω στο Λ .

Ορισμός 10.7

Η σχέση \rightarrow_a ορίζεται ως η μικρότερη συμβατή σχέση πάνω στο Λ για την οποία για κάθε $M \in \Lambda$ και για κάθε $x, y \in V$ τέτοια ώστε $y \notin FV(M)$ ισχύει

$$\lambda x.M \rightarrow_a \lambda y.M[x:=y]$$

Ο περιορισμός $y \notin FV(M)$ εξασφαλίζει ότι η μετονομασία της δεσμεύουσας μεταβλητής δεν προκαλεί δέσμευση των μεταβλητών του όρου M , που ήταν αρχικά ελεύθερες.

Ορισμός 10.8

Η σχέση \rightarrow_β ορίζεται ως η μικρότερη συμβατή σχέση πάνω στο Λ για την οποία για κάθε $M, N \in \Lambda$ και για κάθε $x, y \in V$ ισχύει

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

Ορισμός 10.9

Η σχέση \rightarrow_n ορίζεται ως η μικρότερη συμβατή σχέση πάνω στο Λ για την οποία για κάθε $M \in \Lambda$ και για κάθε $x \in V$ τέτοια ώστε $x \notin FV(M)$ ισχύει

$$\lambda x.M \ x \rightarrow_n M$$

Μετατροπή, αναγωγή και ισότητα

Οι τρεις κανόνες μετατροπής (α), (β) και (η) ορίζουν τρεις βασικές σχέσεις μετατροπής \rightarrow_α , \rightarrow_β , \rightarrow_η για το λ -λογισμό.

Η συνολική σχέση μετατροπής $M \rightarrow N$ θα υποδηλώνει ότι ο όρος M μετατρέπεται σε ένα βήμα στον όρο N με έναν από τους τρεις κανόνες.

Ορισμός 10.10

Με \rightarrow συμβολίζεται η ένωση των σχέσεων \rightarrow_a , \rightarrow_β , \rightarrow_n , δηλαδή η σχέση \rightarrow ορίζεται ως η μικρότερη σχέση για την οποία ισχύουν:

$$M \rightarrow_a N \Rightarrow M \rightarrow N$$

$$M \rightarrow_\beta N \Rightarrow M \rightarrow N$$

$$M \rightarrow_n N \Rightarrow M \rightarrow N$$

Συμβολισμός

$$M \equiv N_0 \rightarrow N_1 \rightarrow N_2 \rightarrow \dots \rightarrow N_{n-1} \rightarrow N_n \equiv N$$

Ο όρος M μετατρέπεται στον όρο N με ακολουθία βημάτων, πιθανώς κενή. Η σχέση συμβολίζεται

$$M \rightarrow N$$

Αν $n = 1$ τότε η μετατροπή γίνεται σε ένα βήμα

$M \rightarrow N$ και άρα η σχέση \rightarrow εμπεριέχει την \rightarrow .

Αν $n=0$, η ακολουθία είναι κενή και δε συμβαίνει μετατροπή, δηλαδή $M \equiv N$ και άρα η σχέση \rightarrow εμπεριέχει την \equiv .

Κανονικές μορφές

- λ-λογισμός θεωρείται ως υπολογιστικό μοντέλο επεξεργασίας συναρτήσεων
- Σχέσεις μετατροπής
 - παριστάνουν την πραγματοποίηση βημάτων επεξεργασίας πάνω στους λ-όρους
 - Αποσκοπούν στον υπολογισμό κάποιου αποτελέσματος

β - Μετατροπή

- Κατ' εξοχήν υπολογιστική πράξη, αφού διαισθητικά παριστάνει εφαρμογή μίας συνάρτησης σε κάποιο όρισμα

α - Μετατροπή

- Δεν προάγει τη διαδικασία υπολογισμού.
- Τα ονόματα των δεσμευμένων μεταβλητών δεν έχουν ουσιαστική σημασία.
- Η μετονομασία των μεταβλητών, που επιτελείται μέσω της α-μετατροπής δεν αποτελεί ουσιαστικό υπολογιστικό βήμα

η – μετατροπή

- Δεν συμβάλλει άμεσα στην υπολογιστική διαδικασία.
- Υλοποιεί έναν μηχανισμό για την υλοποίηση λ-όρων, που παριστάνουν συναρτήσεις

Παρατηρήσεις

- Η σχέση μετατροπής $M \rightarrow N$ παριστάνει ένα βήμα στη διαδικασία υπολογισμού
- Η σχέση $M \rightarrow\!\!\rightarrow N$ παριστάνει (πιθανώς κενή) ακολουθία από τέτοια βήματα.
- Όταν $M \rightarrow\!\!\rightarrow N$ μπορεί να θεωρηθεί ότι ο όρος N προέκυψε κατά την αποτίμηση του όρου M .
 - Αποτίμηση ενός όρου είναι η διαδοχική εφαρμογή κανόνων μετατροπής σε αυτόν.

Κανονικές μορφές

- Όταν σε έναν όρο M δεν εφαρμόζεται κανένα αμιγώς υπολογιστικό βήμα, δηλαδή καμία β - ή η -μετατροπή, τότε η αποτίμηση του θεωρείται ολοκληρωμένη και ο όρος είναι τελικό αποτέλεσμα.
- Τέτοιοι πλήρως αποτιμημένοι όροι ονομάζονται *κανονικές μορφές*.

Ορισμός 10.11

Ένας όρος $M \in \Lambda$ είναι σε κανονική μορφή (normal form) όταν δεν περιέχει κανένα β -redex ή n -redex.

Παράδειγμα 10.11

Οι όροι $\lambda x.x$ και $\lambda f.f$ ($\lambda x.x f$) είναι σε κανονική μορφή. Αντίθετα ο όρος $\lambda z.(\lambda f.\lambda x.f z x) (\lambda y.y)$ δεν είναι σε κανονική μορφή, γιατί περιέχει το β -redex:

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

ΠΑΡΑΤΗΡΗΣΗ

Υπάρχουν λ-όροι η αποτίμηση των οποίων οδηγεί, μετά από διαδοχικές μετατροπές σε κάποια κανονική μορφή. Υπάρχουν όροι για τους οποίους αυτό δεν ισχύει και η αποτίμηση τους ενδέχεται να συνεχίζεται επ' άπειρον χωρίς να καταλήγει σε κανονική μορφή.

Ορισμός 10.12

Ένας όρος $M \in \Lambda$ λέμε ότι έχει κανονική μορφή αν για κάποιον όρο $N \in \Lambda$, ισχύει $M \rightarrow N$ και ο N είναι σε κανονική μορφή. Στην περίπτωση αυτή, ο όρος M ονομάζεται κανονικοποιήσιμος (normalizing).

Ορισμός 10.13

Ένας όρος M ονομάζεται ισχυρά κανονικοποιήσιμος (strongly normalizing) αν όλες οι ακολουθίες μετατροπής, που ξεκινούν με τον M καταλήγουν σε κανονική μορφή.

Ορισμός 10.14

Η στρατηγική κατά την οποία επιλέγεται για μετατροπή κάθε φορά το αριστερότερο $redex$ ενός όρου, δηλαδή αυτό του οποίου το σύμβολο λ βρίσκεται όσο το δυνατόν πιο αριστερά, ονομάζεται στρατηγική της αναγωγής κανονικής σειράς (normal order reduction strategy)

Πρόταση 10.14

Αν $M_1 = M_2$, τότε υπάρχει ένας όρος N τέτοιος ώστε $M_1 \twoheadrightarrow N$ και $M_2 \twoheadrightarrow N$.

Θεώρημα

Σταθερό σημείο – Fixed point

- i) Για κάθε όρο $F \in \Lambda$ υπάρχει όρος $X \in \Lambda$ τέτοιος ώστε να ισχύει $F X = X$
- ii) Υπάρχει ένας τελεστής σταθερού σημείου, δηλαδή ένας όρος $Y \in \Lambda$ τέτοιος ώστε για κάθε $F \in \Lambda$ να ισχύει $F(Y F) = Y F$.

Ορισμός 10.15

Πως μπορούμε να αναπαραστήσουμε τις λογικές τιμές στο λάμβδα-λογισμό;

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

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

Ορισμός 10.16

Πως αναπαριστούμε τους λογικούς τελεστές;

not \equiv λz. z **false true**

Θεώρημα

i) **not true = false**

ii) **not false = true**

Απόδειξη

(μόνο του πρώτου, παρόμοια γίνεται και του δεύτερου)

not true \equiv **($\lambda z.z$ false true) true**

\rightarrow_{β} **true false true**

\equiv **($\lambda x. \lambda y. x$) false true**

\rightarrow_{β} **false**

Ορισμός 10.17

cond $\equiv \lambda z. \lambda x. \lambda y. z \ x \ y$

if B then N else M \equiv **cond** B N M

Αποδεικνύεται εύκολα ότι η παραπάνω δομή πληροί τις απαιτούμενες ιδιότητες.

Θεώρημα

Για κάθε N, M ισχύουν τα παρακάτω

i) **if true then N else M** = N

ii) **if false then M else N** = M

Απόδειξη (του πρώτου)

if true then N else M \equiv **cond** true N M

$\equiv (\lambda z. \lambda x. \lambda y. z x y)$ **true** N M

\rightarrow_{β} **true** N M

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

\rightarrow_{β} N

Διατεταγμένα ζεύγη

Στον λ-λογισμό κωδικοποιούνται διατεταγμένα ζεύγη όρων (που με τη σειρά τους κωδικοποιούν άλλα μαθηματικά αντικείμενα). Μία τέτοια κωδικοποίηση γίνεται μέσω του **pair**. Ο συμβολισμός $\langle N, M \rangle$ χρησιμοποιείται για διευκόλυνση στη γραφή όρων, που κωδικοποιούν διατεταγμένα ζεύγη.

Ορισμοί 10.18 και 10.19

pair $\equiv \lambda x. \lambda y. \lambda z. z \ x \ y$

$\langle N, M \rangle \equiv \mathbf{pair} \ N \ M$

Οι πράξεις **fst** και **snd**, που επιστρέφουν το πρώτο και το δεύτερο στοιχείο ενός διατεταγμένου ζεύγους κωδικοποιούνται ως:

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

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

Αποδεικνύεται ότι οι πράξεις πληρούν απαιτούμενες ιδιότητες και άρα η δομή $\langle N, M \rangle$ χρησιμεύει ως διατεταγμένο ζεύγος.

Θεώρημα

Για κάθε $N, M \in \Lambda$ ισχύουν τα εξής:

i) **fst** $\langle N, M \rangle = N$

ii) **snd** $\langle N, M \rangle = M$

Θεώρημα

Απόδειξη (μόνο του πρώτου)

$$\text{fst } \langle N, M \rangle \equiv (\lambda z. z \text{ true}) \langle N, M \rangle$$
$$\rightarrow_{\beta} \langle N, M \rangle \text{ true}$$
$$\equiv \text{pair } N \ M \ \text{true}$$
$$\equiv (\lambda z. \lambda x. \lambda y. z \ x \ y) \ N \ M \ \text{true}$$
$$\rightarrow_{\beta} \text{true } N \ M$$
$$\equiv (\lambda x. \lambda y. x) \ N \ M$$
$$\rightarrow_{\beta} N$$

Φυσικοί αριθμοί

Έχουν προταθεί πολλές διαφορετικές κωδικοποιήσεις των φυσικών αριθμών στον λ-λογισμό. Η πρώτη και η πιο γνωστή από αυτές είναι τα αριθμοειδή του Church.

Ορισμός 10.20

Έστω φυσικός αριθμός $n \in \mathbb{N}$ και όροι $F, A \in \Lambda$.

Ο όρος $F^n(A) \in \Lambda$ ορίζεται επαγωγικά ως:

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

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

Ο ορισμός του $F^n(A)$ θα μπορούσε ισοδύναμα να δοθεί με τη μορφή $F^{n+1}(A) \equiv F^n(F A)$.

Με επαγωγή αποδεικνύεται ότι

$$F^n(FA) \equiv F(F^n(A)) \quad \text{και} \quad F^n(F^m(A)) \equiv F^{n+m}(A).$$

Ορισμός 10.21

(Αριθμοειδή του Church – Church numerals)

Για κάθε φυσικό αριθμό $n \in \mathbb{N}$ ορίζεται

ένας όρος $c_n \in \Lambda$ ως:

$$c_n \equiv \lambda f. \lambda x. f^n(x)$$

Ορισμός 10.21

Το αριθμοειδές, που αντιστοιχεί στον αριθμό 0 είναι το $c_0 \equiv \lambda f. \lambda x. x$, στον αριθμό 1 το $c_1 \equiv \lambda f. \lambda x. f x$, στον αριθμό 2 το $c_2 \equiv \lambda f. \lambda x. f (f x)$ κοκ. Όλα τα αριθμοειδή είναι σε β -κανονική μορφή (μάλιστα όλα εκτός του c_1 είναι και σε η -κανονική μορφή). Ούτε όλοι οι λ -όροι είναι αριθμοειδή, ούτε όλοι ανάγονται σε αριθμοειδή.

Ορισμός 10.22

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

$$A_+ \equiv \lambda n. \lambda m. \lambda f. \lambda x. n f (m f x)$$

$$A_* \equiv \lambda n. \lambda m. \lambda f. n (m f)$$

$$A_{\text{exp}} \equiv \lambda n. \lambda m. m n$$

Το ακόλουθο θεώρημα αποδεικνύει την ορθότητα της κωδικοποίησης πράξεων του παραπάνω ορισμού. Για την απόδειξη του είναι χρήσιμο το ακόλουθο λήμμα.

Θεώρημα

Για κάθε $n, m \in \mathbb{N}$ ισχύουν τα εξής:

i) $\text{succ } C_n = C_{n+1}$

ii) $A_+ C_n C_m = C_{n+m}$

iii) $A_* C_n C_m = C_{nm}$

iv) Αν $m > 0$ τότε $A_{\text{exp}} C_n C_m = C_n^m$

Θεώρημα

Απόδειξη

$$\begin{aligned} \text{i) succ } c_n &\equiv (\lambda n. \lambda f. \lambda x. n \text{ f } (f \text{ x})) (\lambda f. \lambda x. f^n(x)) \\ &\rightarrow_{\beta} \lambda f. \lambda x. (\lambda f. \lambda x. f^n(x)) f (f \text{ x}) \\ &\twoheadrightarrow_{\beta} \lambda f. \lambda x. f^n(f \text{ x}) \\ &\equiv c_{n+1} \end{aligned}$$

Θεώρημα

Απόδειξη

$$\begin{aligned} \text{ii) } A_+ c_n c_m &\equiv (\lambda n. \lambda m. \lambda f. \lambda x. n f (m f x)) c_n c_m \\ &\rightarrow_{\beta} \lambda f. \lambda x. c_n f (c_m f x) \\ &= \lambda f. \lambda x. c_{n+m} f x \quad (\text{από γνωστό λήμμα...}) \\ &\rightarrow_{\eta} c_{n+m} \end{aligned}$$

Θεώρημα

Απόδειξη

$$\begin{aligned} \text{iii) } A_* C_n C_m &\equiv (\lambda n. \lambda m. \lambda f. \lambda x. n (m f)) C_n C_m \\ &\rightarrow_{\beta} \lambda f. C_n f (C_m f) \\ &\equiv \lambda f. (\lambda f. \lambda x. f^n(x)) (C_m f) \\ &\rightarrow_{\beta} \lambda f. \lambda x. (C_m f)^n(x) \\ &= \lambda f. \lambda x. f^{mn}(x) \text{ (από γνωστό λήμμα...)} \\ &\equiv C_{nm} \end{aligned}$$

Θεώρημα

Απόδειξη

$$\text{iv) } A_{\text{exp}} C_n C_m \equiv (\lambda n. \lambda m. m n) C_n C_m$$

$$\rightarrow_{\beta} C_m C_n$$

$$\equiv (\lambda f. \lambda x. f^m(x)) (C_n)$$

$$\rightarrow_{\beta} \lambda x. (C_n)^m(x)$$

$$= \lambda x. C_n^m x \quad (\text{από γνωστό λήμμα...})$$

$$\rightarrow_n C_n^m$$

Λάμβδα λογισμός και πέρασμα παραμέτρων

- Η σχέση ομοιότητας του λ-λογισμού με τις γλώσσες προγραμματισμού δεν περιορίζεται στην εκφραστική του ικανότητα ως προγραμματιστικού μοντέλου.
- Οι στρατηγικές αναγωγής λ-όρων είναι πολύ στενά συνδεδεμένες με μεθόδους πέρασματος παραμέτρων των γλωσσών προγραμματισμού.

Αναλογία λ-λογισμού και γλωσσών προγραμματισμού

- Οι λ-όροι αντιστοιχούν σε εκφράσεις ή εντολές
- Η αφαίρεση και η εφαρμογή αντιστοιχούν στον ορισμό και την κλήση συναρτήσεων ή διαδικασιών και
- Η διαδικασία της αναγωγής αντιστοιχεί στην αποτίμηση εκφράσεων ή την εκτέλεση εντολών.

Οκνηρή αποτίμηση (lazy evaluation)

- Τερματισμός της αποτίμησης μιας έκφρασης πριν προκύψει πλήρως αποτιμημένο αποτέλεσμα.
- Υποστήριξη από γλώσσες συναρτησιακού προγραμματισμού
 - Haskell και Miranda

Οκνηρή αποτίμηση

- Τερματισμός όταν το αποτέλεσμα είναι **τιμή** (value).
- Η ακριβής μορφή των τιμών διαφέρει από γλώσσα σε γλώσσα
 - Αριθμητικές και λογικές σταθερές, καθώς και συναρτησιακές αφαιρέσεις είναι τιμές.
- Στον λ-λογισμό μόνο οι **αφαιρέσεις** θεωρούνται **τιμές**

Αναγωγή Κανονικής Μορφής

- Η στρατηγική της αναγωγής κανονικής σειράς σχετίζεται στενά με την οκνηρή αποτίμηση.
- Η μετατροπή κάθε φορά του αριστερότερου β -redex σημαίνει ότι δεν έχει προηγηθεί μετατροπή μέσα στον όρο, πάνω στον οποίο εφαρμόζεται η αφαίρεση, που αντιστοιχεί σε αυτό το β – redex.

Αναλογία με Οκνηρή Αποτίμηση

- Κατ' αναλογία στην οκνηρή αποτίμηση μία συνάρτηση καλείται **χωρίς να προηγηθεί αποτίμηση των παραμέτρων της**.
- Στην γλώσσα προγραμματισμού Algol 60, η συμπεριφορά αυτή επιτυγχάνεται με τη μέθοδο του περάσματος παραμέτρων κατ' όνομα (**call by name**).

Πρόθυμη Αποτίμηση

- Στις γλώσσες αυτές το πέρασμα παραμέτρων γίνεται κατ' αξία (**by value**) και η τιμή της παραμέτρου **g** αποτιμάται πριν κληθεί η συνάρτηση **f**.
- Η κλήση της **g** οδηγεί σε μη τερματισμό.
- Πρόθυμη αποτίμηση (**eager evaluation**) είναι η αποτίμηση κατά την οποία οι παράμετροι αποτιμώνται πριν γίνει η κλήση στη συνάρτηση.

Πρόθυμη αποτίμηση στο λ-λογισμό

- Στον λ-λογισμό , το ανάλογο της πρόθυμης αποτίμησης και του περάσματος παραμέτρων κατ' αξία είναι μία στρατηγική αποτίμησης που κατά την αποτίμηση του όρου $(\lambda x. \lambda y. y)\Omega$ **θα αποτιμούσε πρώτα τον όρο Ω .**

Πρόθυμη αποτίμηση στο λ-λογισμό

- Τέτοιου είδους στρατηγικές προκύπτουν υποχρεωτικά αν στον κανόνα της β-μετατροπής

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

Προσθέσουμε τον περιορισμό ότι ο όρος N πρέπει να είναι τιμή.