Chapter 10_1
λ – Calculus
(Θεωρητική Προσέγγιση)

Γ.Γαροφαλάκης, Σ.Σιούτας
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 λx.E[x] συμβολίζει τη συνάρτηση

$$x \mapsto E[x]$$

Διαισθητική εισαγωγή

- Η συνάρτηση δέχεται ως όρισμα μία τιμή υ και επιστρέφει ως αποτέλεσμα την τιμή Ε[υ]. Η x δεν εμφανίζεται απαραίτητα στην έκφραση Ε[x]. Αν αυτό δεν συμβαίνει, τότε η λx.Ε[x] είναι μία σταθερή συνάρτηση.

Παράδειγμα

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

Αν η συνάρτηση εφαρμοσθεί στο όρισμα 8, προκύπτει

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

Η τιμή του ορίσματος αντικαθιστά την παράμετρο x

Ελεύθερη και Δεσμευμένη Μεταβλητή

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

Παράδειγμα: $\lambda x.\ x^2 - 3y + 2$

x δεσμευμένη και y ελεύθερη

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

Η πρώτη εμφάνιση του x (στο x^2) είναι δεσμευμένη, γιατί είναι στο εσωτερικό της αφαίρεσης, ενώ η δεύτερη (στο 4x + 1) είναι ελεύθερη)

Παράδειγμα Δέσμευσης

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

Η μεταβλητή x στο εσωτερικό του ολοκληρώματος είναι δεσμευμένη και κατά συνέπεια η τιμή του ολοκληρώματος δεν εξαρτάται από την τιμή του x έξω από αυτό. Η μεταβλητή y είναι ελεύθερη και η τιμή του ολοκληρώματος εξαρτάται από την τιμή του y έξω από αυτό.

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 (λx.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

⟨term⟩ ::= ⟨var⟩

| (⟨term⟩ ⟨term⟩)

| (λ ⟨var⟩·⟨term⟩)

Παράδειγμα 10.1

Οι παρακάτω είναι λ-όροι:

(x y)

(λx.x)

(λx.((λy.(x y))))

(((λx.x)y)(λx.z))

((λx.(λy.z)) (λx.x))

(λx.((λy.y)(λz.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 ... M_n συντομογραφία του (...((F M_1)M_2)... M_n)

Συμβάσεις

- Η αφαίρεση εκτείνεται όσο περισσότερο είναι δυνατό, δηλαδή ως το επόμενο κλείσιμο παρένθεσης ή το τέλος του όρου:
  - λx. M_1 M_2 … M_n συντομογραφία του

    λx. (M_1 M_2 … 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:

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

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

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

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 λx.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:

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

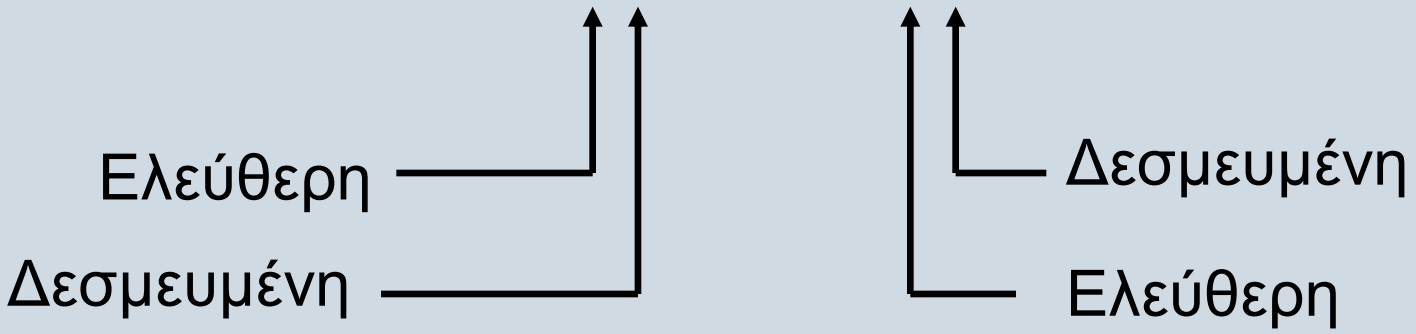
$$FV(M\ N) = FV(M) \cup FV(N)$$

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

Παράδειγμα 10.3

Έστω ο παρακάτω όρος:

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



Παράδειγμα 10.3

Ακολουθώντας τον ορισμό 10.3

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

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 ε**λεύθερων 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.

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 $M \twoheadrightarrow N$ παριστάνει (πιθανώς κενή) ακολουθία από τέτοια βήματα.
- Όταν $M \twoheadrightarrow N$ 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 Μ 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 η-redex.

Παράδειγμα 10.11

Οι όροι λx.x και λf.f (λx.x f) είναι σε κανονική μορφή. Αντίθετα ο όρος

λz.(λf.λx.f z x) (λy.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

$M \twoheadrightarrow N$ 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.

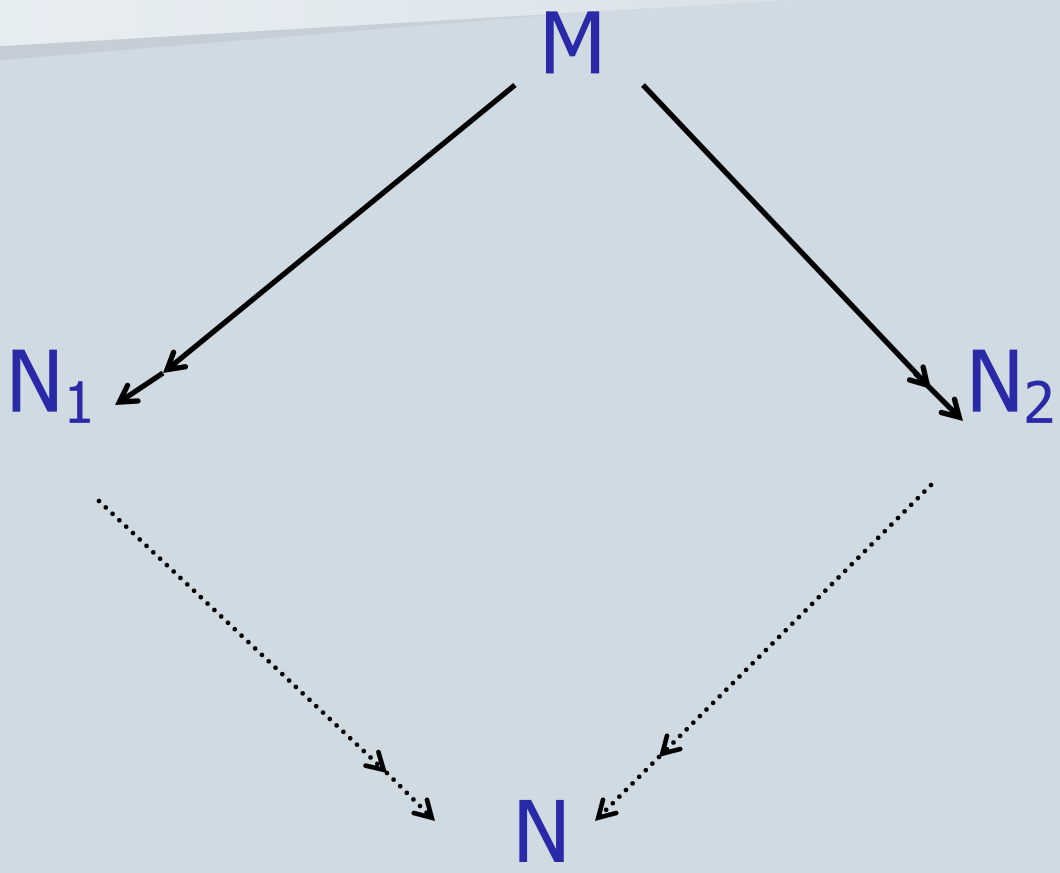
Ορισμός 10.14

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)

Θεώρημα (Church-Rosser)

Έστω όροι $M, N_1, N_2 \in \Lambda$ τέτοιοι ώστε $M \twoheadrightarrow N_1$ και $M \twoheadrightarrow N_2$.Τότε υπάρχει όρος $N \in \Lambda$ τέτοιος ώστε $N_1 \twoheadrightarrow N$ και $N_2 \twoheadrightarrow N$.

Διάγραμμα



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 ≡ λx. λy. x

false ≡ λx. λy. y

Ορισμός 10.16

Πως αναπαριστούμε τους λογικούς τελεστές;

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

Θεώρημα

i) **not true = false**

ii) **not false = true**

Απόδειξη

(μόνο του πρώτου, παρόμοια γίνεται και του δεύτερου)

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

$\rightarrow_\beta$ **true false true**

$\equiv$ (λx. λy. x) **false true**

$\twoheadrightarrow_\beta$ **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** N **else** M = M

Απόδειξη (του πρώτου)

if true then N **else** M ≡ **cond** true N M

≡ (λz. λx. λy. z x y) **true** N M

$\twoheadrightarrow_\beta$ **true** N M

≡ (λx. λy. x) N M

$\twoheadrightarrow_\beta$ 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 $\equiv \lambda x.\ \lambda y.\ \lambda z.\ z\ x\ y$

$\langle N, M \rangle \equiv$ **pair** $N\ M$

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

fst $\equiv \lambda z.\ z$ **true**

snd $\equiv \lambda z.\ z$ **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 $\in$ Λ ισχύουν τα εξής:

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

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

Θεώρημα

Απόδειξη (μόνο του πρώτου)

fst $\langle N,M\rangle \equiv (\lambda z.z\ \mathbf{true})\ \langle N,M\rangle$

$\rightarrow_\beta \langle N,M\rangle\ \mathbf{true}$

$\equiv$ pair N M true

$\equiv (\lambda z.\ \lambda x.\ \lambda y.\ z\ x\ y)\ N\ M\ \mathbf{true}$

$\twoheadrightarrow_\beta \mathbf{true}\ N\ M$

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

$\twoheadrightarrow_\beta\ N$

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 $n \in N$ 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

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

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.

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 περάσματος 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