

Χ. Χατζώνας

Λογισμοί λ

Θεμελίωση & Σημασιολογία

Γλωσσών Προγραμματισμού

ΘΕΣΣΑΛΟΝΙΚΗ

 ΕΚΔΟΣΕΙΣ
ΖΗΤΗ

Αφιερώνεται με όλη μου την καρδιά στο Δημήτρη και στη Λένα, με τις ένοχες ευχαριστίες μου για την υπομονή και την κατανόησή τους για τις ατελείωτες ώρες μελέτης και απουσίας μου “στο γραφείο”!

Αφιερώνεται, επίσης, στους φοιτητές μου, για την προσπάθεια που καταβάλουν να ανταπεξέλθουν στις απαιτήσεις των μαθημάτων που διδάσκω στην περιοχή της Λογικής και της Θεμελίωσης του Προγραμματισμού.

Dedicated to the memory of my dearest teacher and friend, *Jon Barwise*, whose recent and premature loss has deeply touched most of those, including myself, who have had an opportunity to get to know him close enough.

Εισαγωγή

Μολονότι μια γλώσσα προγραμματισμού είναι συνήθως ένα συντακτικό οικοδόμημα μεγάλων διαστάσεων, προκύπτει εύκολα από την ανάλυσή της ότι αποτελείται από ένα βασικό γλωσσικό πυρήνα, επέκταση του οποίου (με ορισμούς) αποτελεί το υπόλοιπο της γλώσσας. Βέβαια, ο περιορισμός στον πυρήνα και μόνον της γλώσσας κάνει τη γλώσσα δύσκολη, αφού κάθε προγραμματιστής υποχρεώνεται να γράφει κώδικα για ρουτίνες οι οποίες υπεισέρχονται συχνά στην πρακτική της σύνταξης προγραμμάτων. Ωστόσο, όταν ο στόχος είναι όχι η χρήση της γλώσσας, αλλά η μελέτη των ιδιοτήτων της, τότε είναι σκόπιμο να περιοριστούμε απλά και μόνο στον πυρήνα, αφού συντακτικές κατασκευές που μπορούν να οριστούν από τα βασικά στοιχεία του γλωσσικού πυρήνα δεν προσθέτουν ή αφαιρούν τίποτε απολύτως στην κατανόησή μας των ιδιοτήτων της γλώσσας. Επιπλέον, διάφορες γλώσσες υλοποιούν ουσιαστικά τις ίδιες συντακτικές δομές, αλλά συνήθως η συγκεκριμένη σύνταξη επιφάνειας της κάθε γλώσσας διαφέρει. Αυτού του είδους οι συντακτικές παραλλαγές είναι επίσης άσχετες με το πρόβλημα της μελέτης της βασικής δομής μιας γλώσσας. Είναι, επομένως, σκόπιμο να υιοθετήσουμε μια αφηρημένη σύνταξη στην παρουσίαση και μελέτη μιας γλώσσας, απογυμνώνοντάς την από επιφανειακές συντακτικές συμβάσεις και ιδιομορφίες.

Μια γλώσσα γραμμένη σε αφηρημένη σύνταξη και η οποία περιορίζεται μόνο σε συντακτικές δομές του γλωσσικού πυρήνα μιας οικογένειας γλωσσών αποτελεί το πρότυπο γι αυτές τις γλώσσες ή, αλλιώς, ένα **Θεμελιωτικό Λογισμό** γι αυτές τις γλώσσες. Έτσι, παρά τη μεγάλη ποικιλία γλωσσών προγραμματισμού, υπάρχει ένας μικρός αριθμός λογισμών που μπορούν να θεωρηθούν ως θεμελιωτικοί λογισμοί. Οι διάφορες πα-

ραλλαγές του Λογισμού λ, όπως θα διαπιστώσουμε, αποτελούν τον κατ' εξοχή θεμελιωτικό λογισμό για τις λεγόμενες Συναρτησιακές Γλώσσες Προγραμματισμού (LISP, Scheme, ML, Haskell, C κλπ).

Περί τα μέσα της δεκαετίας του '30 προτάθηκαν, σχεδόν ταυτόχρονα, δύο διαφορετικά μοντέλα υπολογισιμότητας. Ο Alan Turing επινόησε μια έννοια αφηρημένης μηχανής, η οποία εκτελεί αριθμητικούς υπολογισμούς. Προβλήματα που είναι αλγοριθμικά στη φύση τους μπορούν να κωδικοποιηθούν στο μοντέλο του Turing, με την έννοια ότι μπορεί κανείς να περιγράψει μια Μηχανή Turing η οποία υλοποιεί τον αλγόριθμο επίλυσης του προβλήματος.

Παράλληλα, ο Alonzo Church επινόησε ένα τυπικό σύστημα, το Λογισμό λ, και υπέθεσε ότι μια συνάρτηση είναι αλγοριθμική αν και μόνον αν μπορεί να ορισθεί στο Λογισμό λ. Ο λεγόμενος Λογισμός λ χωρίς τύπους, προτάθηκε αρχικά με την πρόθεση να προσφέρει ένα Λογισμό Θεμελίωσης της Λογικής και των Μαθηματικών. Η αρχική αυτή πρόθεση δεν καρποφόρησε, αλλά σύντομα έγινε αντιληπτό ότι ο Λογισμός, στις διάφορες παραλλαγές του, μπορεί να χρησιμοποιηθεί με επιτυχία ως μοντέλο για τη μελέτη των υπολογίσιμων (computable) συναρτήσεων.

Ενα διαφορετικό μοντέλο αλγοριθμικότητας, για συναρτήσεις σε φυσικούς αριθμούς, προσέφερε επίσης και η Θεωρία των Αναδρομικών Συναρτήσεων, η οποία αναπτύχθηκε από τον Kleene στα πλαίσια της Μαθηματικής Λογικής. Έχει πλέον αποδειχθεί ότι τα τρία αυτά μοντέλα (καθώς και άλλα, που προτάθηκαν αργότερα) είναι ισοδύναμα.

Η σχεδίαση και σύνταξη μιας γλώσσας προγραμματισμού, είτε ακολουθεί άμεσα το πρότυπο κάποιου μοντέλου υπολογίσιμων (computable) συναρτήσεων, είτε μπορεί ναδειχθεί ότι, σε τελευταία ανάλυση, δεν αποτελεί παρά υλοποίηση ενός τέτοιου μοντέλου. Γλώσσες οι οποίες είναι σήμερα γνωστές ως Προστακτικές (Imperative Languages), όπως η Pascal, σχεδιάστηκαν με πρότυπο τις Μηχανές Turing. Γλώσσες οι οποίες είναι σήμερα γνωστές ως Συναρτησιακές (ML, Scheme, Hope, Miranda, C) σχεδιάστηκαν με πρότυπο το Λογισμό λ (με ή χωρίς τύπους).

Ωστόσο, με την επινόηση διαρκώς και πιο αποτελεσματικών μηχανικών μέσων υπολογισμού, το πρότυπο του συναρτησιακού και γενικότερα ακολουθιακού υπολογισμού (sequential computation) υποχώρησε προς όφελος νέων αντιλήψεων για παράλληλο υπολογισμό. Η ανάγκη για

θεμελίωση και αυστηρή ανάλυση της έννοιας της παραλληλίας (concurrent computation) οδήγησε στην επινοήση Θεμελιωτικών Λογισμών για παράλληλο υπολογισμό, όπως ο Λογισμός CCS (Concurrent Communicating Systems), που αναπτύχθηκε από το Robin Milner, και η CSP (Communicating Sequential Processes), που επινοήθηκε και μελετήθηκε από τον Tony Hoare. Επιπλέον, με την ανάπτυξη του διαδικτύου και τη δυνατότητα που δημιουργήθηκε για κατανομή υπολογισμών που εκτελούνται παράλληλα σε διαφορετικά και απομακρυσμένα φυσικά μέσα, δημιουργήθηκε αντίστοιχα, σχετικά πρόσφατα (στη δεκαετία του '90) και η ανάγκη για την επινοήση ενός αντίστοιχου Λογισμού Θεμελίωσης για Κατανεμημένους Υπολογισμούς. Το βασικό πρότυπο, που είναι ακόμη και σήμερα αντικείμενο έντονης μελέτης, είναι ο λεγόμενος Λογισμός π , ο οποίος επινοήθηκε από τους Robin Milner, Johacim Parrow και David Walker. Υπάρχει πλέον σήμερα ένας μεγάλος αριθμός γλωσσών οι οποίες υλοποιούν στοιχεία παράλληλου και κατανεμημένου υπολογισμού, όπως η Java, Concurrent ML (CML), Facile, Pict (υλοποίηση του Λογισμού π) κλπ.

Η μονογραφία αυτή απευθύνεται στον αναγνώστη που έχει τουλάχιστο μια βασική εξοικείωση με την πρακτική του προγραμματισμού, σε κάποια γλώσσα ή γλώσσες. Υποθέτω ότι ο αναγνώστης έχει πειραματιστεί τουλάχιστο για ένα εξάμηνο με γλώσσες όπως η Scheme, η ML, ή η Pascal, ώστε να γίνει αντιληπτό το πέρασμα από μια συγκεκριμένη γλώσσα προγραμματισμού σε ένα Λογισμό Θεμελίωσης για τέτοιες γλώσσες. Το υλικό που παρουσιάζεται στο παρόν τεύχος είναι μεν τεχνικό (μαθηματικό στη φύση του), αλλά ταυτόχρονα ιδιαίτερα κοντά στην πρακτική του προγραμματισμού. Στόχος της μονογραφίας αυτής είναι να βοηθήσει τον αναγνώστη να ανακαλύψει το κοινό θεμέλιο πίσω από την ποικιλία των γλωσσών προγραμματισμού. Επίσης, να τον διαπαιδαγωγήσει να αναζητά τη βασική, θεμελιωτική δομή πίσω από οποιαδήποτε γλώσσα χειρίζεται. Τέλος, να του υποδείξει, έστω και σε αρχικό επίπεδο, ποια είναι τα πρότυπα πάνω στα οποία πρέπει να στηρίζεται οποιαδήποτε δραστηριότητα που σχετίζεται με τη σχεδίαση γλωσσών προγραμματισμού.

X. Χαττώνας

hartonas@cs.teilar.gr

Οκτώβριος 2000

<http://www.cs.teilar.gr/~hartonas>

Περιεχόμενα

1	Λογισμοί λ χωρίς Τύπους	9
1.1	Εισαγωγή	9
1.2	Σύνταξη και Θεωρίες λ	10
1.2.1	Η Γλώσσα του Λογισμού λ	10
1.2.2	Οι Θεωρίες λ και $\lambda\eta$	14
1.2.3	Αριθμοσύμβολα, Ζεύγη και Τιμές Αλήθειας	20
1.2.4	Αναδρομή	24
1.3	Αναδρομικές Συναρτήσεις, λ -Definability	28
1.3.1	Πρωτογενείς Αναδρομικές Συναρτήσεις	30
1.3.2	Ολικές και Μερικές Αναδρομικές Συναρτήσεις . .	34
1.3.3	λ -definability Αναδρομικών Συναρτήσεων	38
1.4	Λειτουργική Ερμηνεία	42
1.4.1	Σχέσεις Αναγωγής	42
1.4.2	Προδιατάξεις και Ισοδυναμίες Προγραμμάτων . . .	45
1.4.3	Lazy Ερμηνεία	49
2	Λογισμοί λ με Τύπους	51
2.1	Εισαγωγή	51
2.2	Programming for Computable Functions	56
2.3	Λειτουργική Ερμηνεία της PCF	64
2.4	Παραλλαγές και Επεκτάσεις της PCF	69
2.4.1	Τύποι Unit και Αθροίσματος	69
2.4.2	Αναδρομικοί Τύποι	71
2.4.3	Ο Λογισμός FPC	73

3	Πολυμορφικοί Λογισμοί λ	83
3.1	Εισαγωγή	83
3.2	Predicative Πολυμορφισμός: $\lambda^{\rightarrow, \Pi}$	85
3.3	Impredicative Πολυμορφισμός: $\lambda^{\rightarrow, \forall}$	91
4	Δηλωτική Ερμηνεία	101
4.1	Εισαγωγή στα Cpo	101
4.2	Ερμηνεία του Κλασικού Λογισμού λ	111
4.3	Ερμηνεία της PCF	117
4.3.1	Ερμηνεία Τύπων και Εκφράσεων	117
4.3.2	Ιδιότητες της Ερμηνείας	125
	Βιβλιογραφία	128
	Ευρετήριο Συμβόλων και Ορων	131

Κεφάλαιο 1

Λογισμοί λ χωρίς Τύπους

1.1 Εισαγωγή

Όλες οι βασικές τυπικές συντακτικές δομές που συναντά κανείς σε μια συναρτησιακή γλώσσα μπορούν να αναπτυχθούν και να μελετηθούν στα πλαίσια (κατάλληλων επεκτάσεων) του Λογισμού λ . Με την έννοια αυτή, ο Λογισμός λ αποτελεί τον κατ' εξοχή Θεμελιωτικό Λογισμό για συναρτησιακό προγραμματισμό.

Στο Κεφάλαιο αυτό θα μελετήσουμε το Λογισμό λ χωρίς τύπους. Μετά από την παρουσίαση της γλώσσας του λογισμού, θα παρουσιάσουμε λ -θεωρίες, θα δείξουμε με ποιον τρόπο μπορούν να κωδικοποιηθούν οι φυσικοί αριθμοί ως λ -όροι και θα προχωρήσουμε στη διαπίστωση ότι κάθε αναδρομική συνάρτηση μπορεί να εκφραστεί στη γλώσσα του Λογισμού λ . Το αποτέλεσμα αυτό είναι μέρος ενός βασικού αποτελέσματος σύμφωνα με το οποίο μια συνάρτηση είναι αναδρομική αν και μόνον αν είναι λ -definable, αν και μόνον αν είναι υπολογίσιμη με μια Μηχανή Turing. Στη συνέχεια, θα προχωρήσουμε στη μελέτη σχέσεων αναγωγής λ -όρων (Λειτουργική Ερμηνεία), καθορίζοντας ένα συμβολικό ερμηνευτή για τη γλώσσα.

1.2 Σύνταξη και Θεωρίες λ

1.2.1 Η Γλώσσα του Λογισμού λ

Η γλώσσα του Λογισμού λ ορίζεται αναδρομικά, ως εξής: ¹

$$\Lambda \ni M := x (x \in X) \mid \lambda x.M \mid MM$$

όπου X είναι ένα απειραριθμήσιμο σύνολο μεταβλητών. Ως μεταβλητές θα χρησιμοποιούμε συνήθως τα γράμματα $x, y, z, f, g, h \dots$ του λατινικού αλφάβητου, εισάγοντας και δείκτες όποτε είναι απαραίτητο, π.χ. x_1, x_{23} και λοιπά. Θα γράφουμε συνήθως $\lambda xy.M$ αντί για $\lambda x.\lambda y.M$. Επίσης, θα γράφουμε $MNK \dots R$ αντί για $(\dots ((MN)K) \dots R)$. Για παράδειγμα, η έκφραση MNK είναι ταυτόσημη με την έκφραση $(MN)K$ και η έκφραση $xy(\lambda x.x)$ είναι ταυτόσημη με την έκφραση $((xy)(\lambda x.x))$. Παραδείγματα λ-όρων, σύμφωνα με την παραπάνω σύνταξη, είναι και τα εξής:

$$xx \quad \lambda x.xx \quad \lambda xy.yx \quad \lambda x f.fx(\lambda z.z) \quad y(\lambda uv.yvu)xy$$

Το **σώμα** ενός λ-όρου $\lambda x.M$ είναι η έκφραση M . Ωστε, για παράδειγμα, το σώμα του όρου $\lambda x.\lambda y.xy$ είναι η έκφραση $\lambda y.xy$. Επισημαίνουμε ότι στο Λογισμό λ *χωρίς τύπους* δεν υπάρχει διάκριση ανάμεσα σε συναρτήσεις και data, ή παραμέτρους, στις οποίες εφαρμόζονται συναρτήσεις.

Επισημαίνουμε, επίσης, την πλήρη απουσία σταθερών που υποδηλώνουν (φυσικούς) αριθμούς, καθώς και συναρτησιακών συμβόλων (σύμβολα πράξεων) που υποδηλώνουν αριθμητικές πράξεις. Θα διαπιστώσουμε αργότερα ότι ο Λογισμός λ έχει την απαραίτητη εκφραστική δύναμη, ώστε να κωδικοποιήσουμε στο εσωτερικό του τόσο τους φυσικούς αριθμούς, όσο και τις συνηθισμένες αριθμητικές πράξεις.

Ενας όρος της μορφής MN ονομάζεται όρος (ή έκφραση) **εφαρμογής** (application). Για παράδειγμα, η έκφραση $(\lambda x.xy)(\lambda z.z)$, αλλά και η έκφραση $x(\lambda u.uu)$ είναι εκφράσεις εφαρμογής. Ενας όρος της μορφής $\lambda x.M$ ονομάζεται όρος (ή έκφραση) **αφαίρεσης** (abstraction). Θεμιτή

¹Με άλλα λόγια, εκφράσεις της γλώσσας Λ είναι (1) οι μεταβλητές $x \in X$, και (2) αν $M, N \in \Lambda$ και $x \in X$, τότε και $\lambda x.M$, $MN \in \Lambda$.

έκφραση, στη γλώσσα του Λογισμού λ χωρίς τύπους, είναι και κάθε έκφραση της μορφής FF (εφαρμογή της F στον εαυτό της).

Στη γλώσσα του Λογισμού λ υπάρχουν μόνο δύο τελεστές

$$\lambda : X \times \Lambda \longrightarrow \Lambda \quad \text{apply} : \Lambda \times \Lambda \longrightarrow \Lambda$$

Συνηθίζεται όμως να γράφουμε MN αντί για $\text{apply}(M, N)$ και $\lambda x.M$ αντί για $\lambda(x, M)$.

Για κάθε μεταβλητή $x \in X$, προκύπτει ένας τελεστής $\lambda x : \Lambda \longrightarrow \Lambda$. Σε μια έκφραση $\lambda x.M$, λέμε ότι ο τελεστής λx **δεσμεύει** τη μεταβλητή x στο εσωτερικό του σώματος M του λ-όρου. Για παράδειγμα, η μεταβλητή x είναι δεσμευμένη στον όρο $\lambda x.xy$, ενώ η μεταβλητή y είναι **ελεύθερη** στον όρο αυτό. Μια συγκεκριμένη μεταβλητή μπορεί να εμφανίζεται ταυτόχρονα ελεύθερη και δεσμευμένη σε μια έκφραση, όπως η μεταβλητή y στην έκφραση $(\lambda y.yx)y$.

Ορίζουμε την έννοια της ελεύθερης μεταβλητής με αυστηρό τρόπο, ως εξής:

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

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

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

Ένας όρος M είναι **κλειστός**, αν και μόνον αν $FV(M) = \emptyset$ και **ανοιχτός** αν δεν είναι κλειστός. Το σύνολο των κλειστών όρων της γλώσσας Λ συμβολίζεται με Λ^0 .

Παράδειγμα 1.2.1 $FV(xy) = \{x, y\}$, $FV(\lambda z.xy) = \{x, y\} \setminus \{z\} = \{x, y\}$. Επίσης, $FV(\lambda x.\lambda y.xyz) = \{z\}$, αφού $FV(\lambda y.xyz) = \{x, z\}$ και η μεταβλητή x δεσμεύεται από τον τελεστή λx . Ο όρος $\lambda x.x$ είναι προφανώς κλειστός. Ο όρος $\lambda x.fx$ είναι ανοιχτός. Το κλείσιμό του προκύπτει με δέσμευση της μεταβλητής f , όπως στον όρο $\lambda f.\lambda x.fx$. ■

Αν M είναι μια έκφραση στη γλώσσα του Λογισμού λ, τότε με το συμβολισμό $M[N/x]$ εννοούμε την έκφραση που προκύπτει με αντικατάσταση της έκφρασης N στη θέση κάθε ελεύθερης εμφάνισης της μεταβλητής x

Πίνακας 1.1: Πράξη Αντικατάστασης

$$x[N/y] = \begin{cases} N & \text{αν } y = x \\ x & \text{αν } y \neq x \end{cases}$$

$$(\lambda x.M)[N/y] = \begin{cases} \lambda x.M[N/y] & \text{αν } y \neq x \text{ και } x \notin FV(N) \\ \text{δεν ορίζεται} & \text{αν } y = x \text{ η } x \in FV(N) \end{cases}$$

$$(FM)[N/x] = F[N/x]M[N/x]$$

στην M . Για παράδειγμα, αν $M = \lambda x.yx$ τότε $M[N/y] = \lambda x.Nx$, ενώ $M[N/x] = M$ (διότι η x δεν είναι ελεύθερη για αντικατάσταση, αφού δεσμεύεται από τον τελεστή λx). Είναι όμως δυνατό, στην εκτέλεση μιας αντικατάστασης, να προκληθεί τυχαία δέσμευση μιας μεταβλητής, πράγμα που είναι ανεπιθύμητο. Λόγου χάριν, χωρίς περιορισμούς στην πράξη της αντικατάστασης θα είχαμε $(\lambda x.yx)[(\lambda z.zx)/y] = \lambda x.(\lambda z.zx)x$. Έτσι, η μεταβλητή x που εμφανίζεται ελεύθερη στην έκφραση $\lambda z.zx$ δεσμεύεται, όταν εκτελείται η αντικατάσταση. Μπορούμε να αποφύγουμε φαινόμενα αυτού του είδους, ορίζοντας τη συντακτική πράξη της αντικατάστασης με αυστηρό τρόπο, όπως στον Πίνακα 1.1.

Έτσι, για παράδειγμα, η αντικατάσταση $(\lambda y.yx)[y/x]$ δεν ορίζεται. Το ίδιο ισχύει και για την αντικατάσταση $(\lambda y.yx)[(\lambda z.yz)/x]$.

Λέμε ότι ο όρος N είναι **ελεύθερος για τη μεταβλητή x στον όρο M** , όταν δεν υπάρχει μεταβλητή $y \in FV(N)$ η οποία να δεσμεύεται στον όρο $M[N/x]$ από έναν τελεστή λy ο οποίος εμφανίζεται στον M .

Παραδείγματα και Ασκήσεις

1.1 Υπολογίστε τα σύνολα $FV(M)$ για τους παρακάτω όρους M

$$\lambda xy.x\lambda z.xy(uz) \quad (\lambda u.u)(\lambda xy.x\lambda z.xy(uz)) \quad \lambda u.u(\lambda xy.x\lambda z.xy(uz))$$

1.2 Ποιές από τις παρακάτω αντικαταστάσεις είναι επιτρεπτές; (εξηγήστε)

$$y[(\lambda x.xy)/x] \quad y[(\lambda x.xy)/y] \quad (\lambda x.xy(xx))[zxz/y]$$

$$(\lambda x.xy(xx))[zxz/x] \quad (\lambda x. xyx)[(\lambda u. xux)/y]$$

1.3 (Barendregt) Ορίζουμε το **μήκος** $\ell(M)$ ενός όρου M ως τον αριθμό των συμβόλων που εμφανίζονται στον M . Με επαγωγή στο μήκος του M αποδείξτε ότι

1. Αν ο όρος M είναι όρος εφαρμογής (application), τότε είναι της μορφής $N_1 N_2 \cdots N_m$, για κάποιο $m \geq 2$, και όπου N_1 δεν είναι όρος εφαρμογής
2. Αν ο όρος M είναι όρος αφαίρεσης (λ-abstraction), τότε είναι της μορφής $\lambda x_1 \cdots x_n. N$, για κάποιο $n \geq 1$, και όπου ο N δεν είναι όρος αφαίρεσης.

Παράδειγμα 1.2.2 Χρησιμοποιώντας την προηγούμενη άσκηση δείχνουμε ότι αν ο όρος M δεν είναι μεταβλητή, τότε έχει μία από τις ακόλουθες μορφές

$$(\alpha) \lambda x_1 \cdots x_n. x M_1 \cdots M_m, n \geq 0, m \geq 0$$

$$(\beta) \lambda x_1 \cdots x_n. (\lambda x. M_0) M_1 \cdots M_m, n \geq 0, m \geq 1.$$

Απόδειξη: Ο όρος M θα είναι είτε όρος αφαίρεσης, είτε όρος εφαρμογής. Στην πρώτη περίπτωση είναι της μορφής $\lambda x_1 \cdots x_n. N$, όπου ο N δεν είναι όρος αφαίρεσης. Συνεπώς ο N θα είναι είτε μεταβλητή, είτε όρος εφαρμογής. Αν ο N είναι κάποια μεταβλητή x , τότε είναι στη μορφή που θέλουμε, για $m = 0$. Αλλιώς ο N είναι όρος εφαρμογής. Σ' αυτή την περίπτωση ο N είναι της μορφής $N_1 \cdots N_m$, για κάποιο $m \geq 2$, και όπου ο N_1 δεν είναι όρος εφαρμογής. Συνεπώς, ο N_1 θα είναι είτε κάποια μεταβλητή x , οπότε ο M είναι της μορφής $\lambda x_1 \cdots x_n. x N_2 \cdots N_m$, είτε όρος αφαίρεσης, δηλαδή της μορφής $\lambda x. M_0$. Σ' αυτή την περίπτωση ο όρος M παίρνει τη μορφή $\lambda x_1 \cdots x_n. (\lambda x. M_0) N_2 \cdots N_m$.

Εξετάζουμε τώρα και την περίπτωση όπου ο M είναι όρος εφαρμογής. Τότε ο M είναι της μορφής $N_1 \cdots N_m$, όπου ο N_1 δεν είναι όρος εφαρμογής. Επομένως ο N_1 είναι είτε κάποια μεταβλητή x , είτε όρος αφαίρεσης. Στην πρώτη περίπτωση ο M είναι της μορφής $\lambda x_1 \cdots x_n. x N_1 \cdots N_m$, για $n = 0$. Αλλιώς ο N_1 είναι όρος αφαίρεσης, οπότε ο M είναι της μορφής $\lambda x_1 \cdots x_n. (\lambda x. M_0) N_2 \cdots N_m$, για $n = 0$. ■

Πίνακας 1.2: Η Θεωρία λ

$\lambda x.M = \lambda y.M[y/x]$ (α -conversion, $y \notin FV(M)$)		
$(\lambda x.M)N = M[N/x]$ (β -reduction) (N ελεύθερος για την x στον M)		
$\frac{M = N}{\lambda x.M = \lambda x.N}$ (κανόνας ξ)		
$M = M$ (I)	$\frac{M = N}{N = M}$ (S)	$\frac{M = N \quad N = K}{M = K}$ (T)
$\frac{F = G}{FM = GM}$ (LA)	$\frac{M = N}{FM = FN}$ (RA)	

1.2.2 Οι Θεωρίες λ και $\lambda\eta$

Η Θεωρία λ είναι η θεωρία που παράγεται στη γλώσσα του Λογισμού λ από τα αξιώματα και τους κανόνες του Πίνακα 1.2. Γράφουμε $\lambda \vdash M = N$ ακριβώς όταν η ισότητα $M = N$ προκύπτει από τα αξιώματα και κανόνες της Θεωρίας λ . Σημειώνουμε ότι το αξίωμα ταυτότητας (I), καθώς και οι κανόνες (LA), (RA), (S) και (T), είναι κανόνες και αξιώματα της Λογικής της Ισότητας. Τα ιδιαίτερα αξιώματα και κανόνες που χαρακτηρίζουν τη Θεωρία λ περιορίζονται στα αξιώματα α -ισοδυναμίας, β -αναγωγής και τον κανόνα ξ .

Το αξίωμα της α -ισοδυναμίας καθορίζει ότι εκφράσεις που διαφέρουν μόνον κατά το όνομα δεσμευμένων μεταβλητών είναι ίσες. Λόγου χάριν $\lambda x.xy = \lambda z.zy$. Το αξίωμα της α -ισοδυναμίας μας επιτρέπει, επίσης, να χειριστούμε περιπτώσεις στις οποίες μια β -αναγωγή δε θα ήταν επιτρεπτή, για το λόγο ότι πιθανά δεν ορίζεται η απαραίτητη αντικατάσταση. Θα εξηγήσουμε με ένα κατάλληλο παράδειγμα παρακάτω.

Παραδείγματα

Παράδειγμα 1.2.3 Να δειχθεί ότι $\lambda \vdash ((\lambda f.\lambda x.fx)(\lambda z.z))(\lambda u.u) = \lambda y.y$.
Απόδειξη:

1. $(\lambda f.\lambda x.fx)(\lambda z.z) = \lambda x.(\lambda z.z)x$ (β - reduction)
2. $(\lambda z.z)x = x$ (β -reduction)
3. $\lambda x.(\lambda z.z)x = \lambda x.x$ (2, ξ)
4. $(\lambda f.\lambda x.fx)(\lambda z.z) = \lambda x.x$ (1-3, T)
5. $((\lambda f.\lambda x.fx)(\lambda z.z))(\lambda u.u) = (\lambda x.x)(\lambda u.u)$ (4, LA)
6. $(\lambda x.x)(\lambda u.u) = \lambda u.u$ (β -reduction)
7. $((\lambda f.\lambda x.fx)(\lambda z.z))(\lambda u.u) = \lambda u.u$ (5,6, T)
8. $\lambda u.u = \lambda y.y$ (α -conversion)
9. $((\lambda f.\lambda x.fx)(\lambda z.z))(\lambda u.u) = \lambda y.y$ (7,8, T)

Επομένως, δείξαμε ότι $\lambda \vdash ((\lambda f.\lambda x.fx)(\lambda z.z))(\lambda u.u) = \lambda y.y$.

Παράδειγμα 1.2.4 Να δείχθει ότι $\lambda \vdash (\lambda x.xx)((\lambda z.(\lambda x.xx)z)u) = (uu)(uu)$.
Απόδειξη:

1. $(\lambda x.xx)z = zz$ (β -reduction)
2. $\lambda z.(\lambda x.xx)z = \lambda z.zz$ (1, ξ)
3. $(\lambda z.(\lambda x.xx)z)u = (\lambda z.zz)u$ (2, LA)
4. $(\lambda z.zz)u = uu$ (β -reduction)
5. $(\lambda z.(\lambda x.xx)z)u = uu$ (3,4, T)
6. $(\lambda x.xx)((\lambda z.(\lambda x.xx)z)u) = (\lambda x.xx)(uu)$ (5, ξ)
7. $(\lambda x.xx)(uu) = (uu)(uu)$ (β -reduction)
8. $(\lambda x.xx)((\lambda z.(\lambda x.xx)z)u) = (uu)(uu)$ (6,7, T)

Επομένως, πράγματι $\lambda \vdash (\lambda x.xx)((\lambda z.(\lambda x.xx)z)u) = (uu)(uu)$.

Στη συνέχεια θα δίνουμε αποδείξεις χωρίς άμεση αναφορά στον κανόνα που εφαρμόστηκε, γράφοντας την απόδειξη, για παράδειγμα, στη μορφή:

$$\begin{aligned}
(\lambda x.xx)((\lambda z.(\lambda x.xx)z)u) &= (\lambda x.xx)((\lambda z.zz)u) \\
&= (\lambda x.xx)(uu) \\
&= (uu)(uu)
\end{aligned}$$

Παράδειγμα 1.2.5 Θεωρούμε την έκφραση $(\lambda x.\lambda y.yx)y$. Μια απρόσεκτη β -αναγωγή θα κατέληγε στην “απόδειξη” ότι $\lambda y.yy = (\lambda x.\lambda y.yx)y$. Ωστόσο,

$$(\lambda x.\lambda y.yx)y = (\lambda y.yx)[y/x] \quad (\beta - \text{reduction})$$

αλλά η αντικατάσταση του x από το y δεν είναι επιτρεπτή (βλ. τον ορισμό της πράξης της αντικατάστασης). Για να μπορέσουμε να εκτελέσουμε το βήμα της β -αναγωγής, χρειάζεται πρώτα να μεσολαβήσει ένα βήμα α -αναγωγής. Ωστε, η σωστή αναγωγή της αρχικής έκφρασης αντικαθιστά πρώτα την έκφραση $\lambda y.yx$ με μια α -ισοδύναμή της, έστω την $\lambda u.ux$, και εκτελεί κατόπιν το βήμα της β -αναγωγής. Ωστε, $(\lambda x.\lambda y.yx)y = (\lambda x.\lambda u.ux)y = (\lambda u.ux)[y/x] = \lambda u.uy$. ■

Στη μαθηματική (συνολοθεωρητική) μοντελοποίηση των συναρτήσεων, μια συνάρτηση είναι ένα σύνολο διατεταγμένων ζευγών. Είναι συνέπεια του αξιώματος της Εκτασης (Extensionality) της Θεωρίας Συνόλων ότι δύο συναρτήσεις f, g (με το ίδιο πεδίο ορισμού) ταυτίζονται αν και μόνον αν ισχύει $\text{dom}(f) = \text{dom}(g)$ και $\forall x \in \text{dom}(f) \ fx = gx$. Η θεωρία λ δεν ικανοποιεί το αξίωμα της Εκτασης, το οποίο στη γλώσσα του Λογισμού λ αναδιατυπώνεται με τη μορφή του κανόνα

$$\frac{Mx = Nx}{M = N}(\text{Ext}) \quad x \notin FV(M) \cup FV(N)$$

Η θεωρία που προκύπτει με προσθήκη του κανόνα Ext συμβολίζεται με $\lambda + \mathbf{Ext}$. Αποδεικνύεται ότι η θεωρία αυτή είναι ισοδύναμη με τη θεωρία λ_η που προκύπτει προσθέτοντας στη θεωρία λ το μοναδικό αξίωμα

$$\lambda x.Mx = M \quad x \notin FV(M) \text{ (κανόνας } \eta)$$

Θεώρημα 1.2.1 Οι Θεωρίες $\lambda + \mathbf{Ext}$ και λ_η είναι ισοδύναμες.

Απόδειξη: Ισοδυναμία των δύο θεωριών σημαίνει ότι αποδεικνύουν ακριβώς τις ίδιες ισότητες. Θα δείξουμε ότι $\lambda + \mathbf{Ext} \vdash \eta$ και ότι η Θεωρία

$\lambda\eta$ είναι κλειστή ως προς τον κανόνα Ext (δηλαδή, αν $\lambda\eta \vdash Mx = Nx$, όπου $x \notin FV(MN)$, τότε επίσης $\lambda\eta \vdash M = N$).

Εστω M τυχόν όρος και $N = \lambda x.Mx$. Θα δείξουμε ότι $\lambda + \mathbf{Ext} \vdash M = N$. Αλλά προφανώς, με β -αναγωγή, $Nx = (\lambda x.Mx)x = Mx$ ώστε, εφαρμόζοντας τον κανόνα Ext, προκύπτει $M = N$, δηλαδή $M = \lambda x.Mx$. Συνεπώς, $\lambda + \mathbf{Ext} \vdash \eta$.

Αντίστροφα, έστω ότι $\lambda\eta \vdash Mx = Nx$. Τότε $\lambda\eta \vdash \lambda x.Mx = \lambda x.Nx$, με χρήση του κανόνα ξ . Συνεπώς, επειδή $\lambda\eta \vdash \lambda x.Mx = M$, προκύπτει ότι $\lambda\eta \vdash M = N$. ■

Παράδειγμα 1.2.6 Στα πλαίσια της Θεωρίας λ η εξίσωση $\lambda x.Mx = M$ ισχύει μόνο σε ιδιαίτερες περιπτώσεις. Για παράδειγμα, αν $M = \lambda y.N[y]$, όπου ο συμβολισμός $N[y]$ σημαίνει ότι η υποδεικνυόμενη μεταβλητή εμφανίζεται στον όρο N , τότε έχουμε

$$\lambda x.Mx = \lambda x.(\lambda y.N[y])x = \lambda x.N[x] = \lambda y.N[y] = M \quad \blacksquare$$

Η έννοια του υποόρου ορίζεται με αναδρομή στη δομή ενός όρου. Ορίζουμε το σύνολο $Sub(M)$ των υποόρων ενός όρου M , ως εξής:

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

$$Sub(\lambda x.N) = \{\lambda x.N\} \cup Sub(N)$$

$$Sub(NK) = Sub(N) \cup Sub(K) \cup \{NK\}$$

Λέμε ότι ένας λ-όρος M είναι σε **β-κανονική μορφή** (normal form) αν και μόνον αν δεν έχει κανέναν υποόρο της μορφής $(\lambda x.N)K$. Για παράδειγμα, ο όρος $\lambda x.x$ είναι σε κανονική μορφή. Λέμε επίσης ότι ο όρος M έχει μια κανονική μορφή, αν και μόνον αν υπάρχει όρος M' σε κανονική μορφή για τον οποίο $\lambda \vdash M = M'$. Αντίστοιχα ορίζεται η έννοια της βη-κανονικής μορφής και για τη Θεωρία $\lambda\eta$, με τον πρόσθετο περιορισμό ότι δεν υπάρχει υποόρος της μορφής $\lambda x.Kx$, με $x \notin FV(K)$.

Για παράδειγμα, ο όρος $(\lambda x.xy)\lambda z.z$ δεν είναι σε κανονική μορφή. Ωστόσο,

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

ώστε $\lambda \vdash (\lambda x.xy)\lambda z.z = y$ και επομένως ο όρος $(\lambda x.xy)\lambda z.z$ έχει (β-) κανονική μορφή.

Ενας λ-όρος είναι σε **κανονική μορφή κεφαλής** (head normal form), με μεταβλητή κεφαλής τη μεταβλητή x , αν και μόνον αν είναι της μορφής $\lambda x_1 \cdots x_n.xM_1 \cdots M_m$, για $n, m \geq 0$ (δες επίσης την Ασκήση 1.2.2). Αν $n = 0$, τότε ο όρος είναι της μορφής $xM_1 \cdots M_m$. Αν επίσης $m = 0$, τότε ο όρος είναι απλά η μεταβλητή x . Ενας όρος M έχει κανονική μορφή κεφαλής αν υπάρχει όρος M' που είναι σε κανονική μορφή κεφαλής και για τον οποίο $\lambda \vdash M = M'$.

Δεν είναι όμως δυνατό να βρεθεί κανονική μορφή για όλους τους όρους. Για παράδειγμα, ο όρος $\Omega = (\lambda x.xx)(\lambda x.xx)$ δεν έχει κανονική μορφή.

Λήμμα 1.2.2 *Κάθε όρος σε κανονική μορφή, είναι και σε κανονική μορφή κεφαλής.*

Απόδειξη: Εστω N όρος σε κανονική μορφή. Αν ο N είναι μια μεταβλητή, $N = x$, τότε είναι σε κανονική μορφή κεφαλής. Αν είναι όρος αφαίρεσης, $N = \lambda x.A$, τότε θα πρέπει ο υποόρος A να είναι επίσης σε κανονική μορφή. Με επαγωγή, έστω $A = \lambda y_1 \cdots y_n.zB_1 \cdots B_m$. Τότε $N = \lambda xy_1 \cdots y_n.zB_1 \cdots B_m$ και είναι σε κανονική μορφή κεφαλής. Αν, τέλος, ο N είναι όρος εφαρμογής, τότε από την Ασκήση 1.3 γνωρίζουμε ότι έχει τη μορφή $N = N_1 \cdots N_k$, όπου ο N_1 δεν είναι όρος εφαρμογής. Αλλά ο N_1 δεν μπορεί να είναι ούτε όρος αφαίρεσης, διότι υποθέτουμε ότι ο N είναι κανονική μορφή. Συνεπώς ο N_1 είναι μεταβλητή, οπότε $N = xN_2 \cdots N_k$ και είναι σε κανονική μορφή κεφαλής. ■

Οι ακόλουθοι τελεστές είναι χρήσιμοι:

$$\begin{array}{ll} \mathbf{I} = \lambda x.x & \mathbf{K} = \lambda xy.x \\ \mathbf{K}_* = \lambda xy.y & \mathbf{S} = \lambda xyz.xz(yz) \\ \mathbf{B} = \lambda xyz.x(yz) & \mathbf{C} = \lambda xyz.xzy \end{array}$$

Συνεπώς ισχύουν οι ακόλουθες ισότητες στη Θεωρία λ .

$$\mathbf{IM} = M \quad \mathbf{KMN} = M \quad \mathbf{K}_*MN = N \quad \mathbf{SMNL} = \mathbf{ML(NL)}$$

Σημειώνουμε ιδιαίτερα ότι οι τελεστές **S**, **K**, **I** είναι η βάση της ονομαζόμενης Συνδυαστικής Λογικής (Combinatory Logic). Για την ανάπτυξη θεμάτων που σχετίζονται με την ισοδυναμία της Combinatory Logic με το Λογισμό λ, παραπέμπουμε τον αναγνώστη στο [2].

Παραδείγματα και Ασκήσεις

Παράδειγμα 1.2.7 Ισχύει $\lambda\eta \vdash \mathbf{C} = \mathbf{S}(\mathbf{BBS})(\mathbf{KK})$.

Απόδειξη: Για κάθε x, y, z ισχύει

$$\begin{aligned} \mathbf{S}(\mathbf{BBS})(\mathbf{KK})xyz &= \mathbf{BBS}x(\mathbf{KK}x)yz \\ &= \mathbf{B}(\mathbf{S}x)(\mathbf{KK}x)yz \\ &= \mathbf{B}(\mathbf{S}x)\mathbf{K}yz \\ &= \mathbf{S}x(\mathbf{K}y)z \\ &= xz(\mathbf{K}yz) \\ &= xzy \\ &= \mathbf{C}xyz \end{aligned}$$

και συνεπώς δείξαμε ότι $\lambda\eta \vdash \mathbf{C} = \mathbf{S}(\mathbf{BBS})(\mathbf{KK})$.

1.4 Ποιες από τις παρακάτω ισότητες προκύπτουν με θεμιτή εφαρμογή του αξιώματος της α-ισοδυναμίας (α-conversion);

- $\lambda x.yx = \lambda y.yy$
- $\lambda x.\lambda y.xy = \lambda y.\lambda x.yx$
- $\lambda x.xy = \lambda y.yx$
- $\lambda x.\lambda u.u = \lambda y.\lambda z.z$

1.5 Ποιες από τις παρακάτω β-αναγωγές είναι έγκυρες;

- $(\lambda y.yx)x = yx$
- $(\lambda z.\lambda y.zyx)y = \lambda y.yyx$
- $(\lambda x.\lambda y.xyx)(xy) = \lambda y.(xy)y(xy)$

1.6 Αποδείξτε ότι $\lambda \vdash (\lambda xyz.xyz)((\lambda u.u)(yz)) = \lambda ux.(yz)ux$.

1.7 Αποδείξτε ότι $\lambda \vdash \mathbf{SKK} = \mathbf{I}$

1.8 Δείξτε ότι $\lambda \vdash \mathbf{SKS} = \mathbf{I}$

1.9 Δείξτε ότι $\lambda \vdash \mathbf{SIII} = \mathbf{I}$

1.10 Εστω $\Omega = (\lambda x.xx)(\lambda x.xx)$. Βρείτε παραδείγματα λ-όρων M, N , έτσι ώστε ο όρος $(\lambda x.x\mathbf{I}\Omega)M$ να έχει κανονική μορφή, ενώ ο όρος $(\lambda x.x\mathbf{I}\Omega)N$ να μην έχει κανονική μορφή.

1.11 Εστω $X = \mathbf{SI}$. Δείξτε ότι $\lambda \vdash XXXX = X(X(XX))$

1.12 Δείξτε ότι ο όρος $(\lambda y.yyy)((\lambda ab.a)\mathbf{I}(\mathbf{SS}))$ έχει κανονική μορφή.

1.13 Δείξτε ότι ο όρος $\mathbf{SSSSSSS}$ έχει κανονική μορφή.

1.14 Δείξτε ότι ο όρος $(\lambda yz.zy)((\lambda x.xxx)(\lambda x.xxx))(\lambda u.\mathbf{I})$ έχει κανονική μορφή.

1.2.3 Αριθμοσύμβολα, Ζεύγη και Τιμές Αλήθειας

Οπως παρατηρήσαμε ήδη, η γλώσσα του Λογισμού λ δεν περιλαμβάνει σταθερές για την υποδήλωση (φυσικών) αριθμών, αλλά ούτε και συναρτησιακά σύμβολα για την υποδήλωση αριθμητικών πράξεων. Ωστόσο, η συντακτική δομή του Λογισμού λ μάς επιτρέπει να κωδικοποιήσουμε τόσο τους φυσικούς αριθμούς, όσο και τις αριθμητικές πράξεις. Εισάγουμε πρώτα ορισμούς για τιμές αλήθειας, καθώς και ένα μηχανισμό κωδικοποίησης διατεταγμένων ζευγών.

Για την κωδικοποίηση των τιμών αλήθειας θέτουμε $\mathbf{T}=\mathbf{K}$ και $\mathbf{F}=\mathbf{K}_*$. Το conditional **if** B **then** M **else** N μπορεί να αναπαρασταθεί με την έκφραση BMN . Πράγματι, ισχύει $\mathbf{T}MN = \mathbf{K}MN = M$ και $\mathbf{F}MN = \mathbf{K}_*MN = N$.

Για την κωδικοποίηση διατεταγμένων ζευγών θεωρούμε τη συνάρτηση **pair** $= \lambda xyz.zxy$. Για διευκόλυνση, γράφουμε (x, y) αντί για **pair** xy . Ωστε, $(M, N) = \lambda z.zMN$. Ορίζουμε τις συναρτήσεις **head** και **tail**, θέτοντας

$$\mathbf{head} = \lambda x.x\mathbf{T}$$

$$\mathbf{tail} = \lambda x.x\mathbf{F}$$

Συνεπώς έχουμε

$$\text{head}(M, N) = (M, N)\mathbf{T} = (M, N)\mathbf{K} = (\lambda z.zMN)\mathbf{K} = \mathbf{K}MN = M$$

$$\text{tail}(M, N) = (M, N)\mathbf{F} = (M, N)\mathbf{K}_* = (\lambda z.zMN)\mathbf{K}_* = \mathbf{K}_*MN = N$$

Ο λ-όρος $\lambda z.zMN$ μπορεί να θεωρηθεί επομένως ότι αναπαριστά πράγματι το διατεταγμένο ζεύγος (M, N)

Μπορούμε τώρα να εισάγουμε αριθμοσύμβολα στη γλώσσα του Λογισμού λ, θέτοντας

$$\mathbf{0} = \mathbf{I} \quad \mathbf{n} + \mathbf{1} = (\mathbf{F}, \mathbf{n})$$

Θεώρημα 1.2.3 Υπάρχουν τελεστές \mathbf{S}^+ , \mathbf{P}^- και $\mathbf{Z}?$ έτσι ώστε, για κάθε φυσικό αριθμό n να έχουμε

$$\mathbf{Z}?\mathbf{0} = \mathbf{T} \quad \mathbf{Z}?(n + \mathbf{1}) = \mathbf{F} \quad \mathbf{S}^+\mathbf{n} = n + \mathbf{1} \quad \mathbf{P}^-(n + \mathbf{1}) = n$$

Απόδειξη: Ορίζουμε

$$\mathbf{S}^+ = \lambda x.(\mathbf{F}, x) \quad \mathbf{P}^- = \lambda x.x\mathbf{F} \quad \mathbf{Z}? = \lambda x.x\mathbf{T}$$

και αφήνουμε στον αναγνώστη τον έλεγχο ότι ισχύουν οι ισότητες του Θεωρήματος. ■

Ασκήσεις

1.15 Δείξτε ότι οι ορισμοί των τελεστών \mathbf{S}^+ , \mathbf{P}^- και $\mathbf{Z}?$ που δόθηκαν στην απόδειξη του Θεωρήματος 1.2.3 ικανοποιούν τις εξισώσεις που αναφέρονται στην εκφώνηση του Θεωρήματος.

1.16 Έχουμε ορίσει $\mathbf{0} = \mathbf{I} = \lambda x.x$ και $\mathbf{n} + \mathbf{1} = (\mathbf{F}, \mathbf{n})$. Παίρνοντας υπόψη τον τρόπο με τον οποίο ορίστηκε το διατεταγμένο ζεύγος, γράψτε τους λ-όρους που αντιστοιχούν στα αριθμοσύμβολα **1**, **2**, **3**, **4**, **5**.

1.17 (Ζεύγη) Οι θεμελιώδεις ιδιότητες μιας συνάρτησης ζεύγους $\langle x, y \rangle$ είναι ότι υπάρχουν συναρτήσεις προβολής 1^{st} και 2^{nd} , έτσι ώστε

$$1. \ 1^{st}\langle x, y \rangle = x \text{ και } 2^{nd}\langle x, y \rangle = y$$

$$2. \langle 1^{st}(P), 2^{nd}(P) \rangle = P, \text{ αν } P = \langle M, N \rangle$$

$$3. \langle x, y \rangle = \langle x', y' \rangle \text{ ανν } x = x' \text{ και } y = y'$$

Επιβεβαιώστε ότι η συνάρτηση ζεύγους, όπως την υλοποιήσαμε για το Λογισμό λ, ικανοποιεί αυτές τις συνθήκες.

Παρατήρηση: Σημειώνουμε όμως ότι δεν ισχύει για έναν αυθαίρετο όρο M , ότι $\lambda \vdash \text{pair}(\text{head}M)(\text{tail}M) = M$.

1.18 Ορίστε πεπερασμένες ακολουθίες $\langle M_1, \dots, M_n \rangle$ και συναρτήσεις προβολής $\pi_i^n \langle M_1, \dots, M_n \rangle = M_i$.

1.19 (Λογικοί Σύνδεσμοί) Επιβεβαιώστε ότι ο λ-όρος $\lambda x.x\mathbf{FT}$ ορίζει τη λογική συνάρτηση **not**. Επιβεβαιώστε επίσης ότι ο λ-όρος $\lambda xy.xy\mathbf{F}$ ορίζει τη λογική συνάρτηση **and**. Ποιός λ-όρος ορίζει τη λογική συνάρτηση **or**?

1.20 Μεταφράστε σε κώδικα Scheme τους ορισμούς των τιμών αλήθειας και των λογικών συναρτήσεων και υποβάλτε για εκτίμηση προγράμματα της μορφής (AND FALSE TRUE), (NOT (OR FALSE FALSE)). Κάντε το ίδιο για το conditional, ορίζοντας το IF όπως στο Λογισμό λ χωρίς τύπους και υποβάλτε σχετικά προγράμματα για ερμηνευση από τον Ερμηνευτή της Scheme.

1.21 Αν είτε $\lambda \vdash M = \mathbf{T}$, είτε $\lambda \vdash M = \mathbf{F}$, τότε $\lambda \vdash \text{not}(\text{not}M) = M$.

1.22 Δείξτε ότι, αν $\lambda \vdash M = \mathbf{T}$ ή $\lambda \vdash M = \mathbf{F}$, και $\lambda \vdash N = \mathbf{T}$ ή $\lambda \vdash N = \mathbf{F}$, τότε $\lambda \vdash \text{not}(M\text{and}N) = (\text{not}M)\text{or}(\text{not}N)$.

1.23 (let εκφράσεις) Είναι συνηθισμένο σε συναρτησιακές γλώσσες προγραμματισμού να συμπεριλαμβάνεται ένας τελεστής αντικατάστασης, **let**, με τη σύνταξη **let** $x = M$ **in** N , όταν $x \notin FV(M)$. Εξηγήστε πώς μπορεί να οριστεί ο τελεστής **let** στο Λογισμό λ.

1.24 Υλοποιήστε στη Scheme τους ορισμούς των φυσικών αριθμών και των συναρτήσεων του επόμενου, του προηγούμενου, καθώς και του ελέγχου αν κάποιος αριθμός είναι μηδέν, με τον προφανή κώδικα

```

(define I (lambda (x) x))
(define 0 I)
(define SUCC (lambda (x) ((pair FALSE) x)))
(define HEAD (lambda (x) (x TRUE)))
(define TAIL (lambda (x) (x FALSE)))
(define Z? HEAD)
(define PRED TAIL)

```

και ελέγξτε τη συμπεριφορά του ερμηνευτή, γράφοντας μερικά απλά σχετικά προγράμματα.

1.25 (Αριθμοσύμβολα του Church) Η πρώτη κωδικοποίηση των φυσικών αριθμών ως λ-όρων προτάθηκε από το logician Alonzo Church και είναι διαφορετική από αυτήν που ήδη παρουσιάσαμε. Δίνουμε, στη συνέχεια, τους ορισμούς των αριθμοσυμβόλων Church.

Ορισμός 1.2.4 *Εστω $F, M \in \Lambda$. Ορίζουμε τον όρο $F^n M$ με επαγωγή στους φυσικούς, θέτοντας*

$$F^0 M = M \quad F^{n+1} M = F(F^n M)$$

Τα αριθμοσύμβολα \mathbf{c}_n , για κάθε φυσικό αριθμό n , ορίζονται από τη σχέση

$$\mathbf{c}_n = \lambda f. \lambda x. f^n x \quad (1.1)$$

Ορίζουμε, επίσης

$$\left. \begin{aligned} \mathbf{A}_+ &= \lambda x y p q. x p (y p q) \\ \mathbf{A}_* &= \lambda x y z. x (y z) \\ \mathbf{A}_{exp} &= \lambda x y. y x \end{aligned} \right\} \quad (1.2)$$

Θεώρημα 1.2.5 (Rosser) *Για κάθε φυσικούς $n, m \in \mathbb{N}$, ισχύει*

$$1. (\mathbf{c}_n x)^m y = x^{nm} y \text{ και, για } m > 0, (\mathbf{c}_n)^m x = \mathbf{c}_n x$$

$$2. \mathbf{A}_+ \mathbf{c}_m \mathbf{c}_n = \mathbf{c}_{m+n}$$

$$3. \mathbf{A}_* \mathbf{c}_m \mathbf{c}_n = \mathbf{c}_{mn}$$

$$4. \mathbf{A}_{exp} \mathbf{c}_n \mathbf{c}_m = \mathbf{c}_{n^m}, \text{ αν } m > 0.$$