ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Σχεδίαση Γλωσσών & Μεταγλωττιστές

Ενότητα 14: Συστήματα Τύπων

Επ. Καθ. Π. Κατσαρός
Τμήμα Πληροφορικής

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδεια χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος«Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Γλώσσες με Τύπους & Χωρίς Τύπους

Ορισμός 1 (Τύπος)

Τύπος είναι το εύρος τιμών που μπορούν να πάρουν κατά την εκτέλεση του προγράμματος οι μεταβλητές του.

- **Γλώσσες με Τύπους**: σε όλες τις μεταβλητές του προγράμματος αποδίδονται τύποι.
- **Γλώσσες Χωρίς Τύπους**: δεν περιορίζεται το εύρος τιμών των μεταβλητών, επειδή είτε δεν υπάρχουν τύποι, είτε υπάρχει ένας μόνο καθολικός τύπος με όλες τις πιθανές τιμές.
  Πράξεις με ακατάλληλα ορίσματα έχουν ως αποτέλεσμα:
  1. μία αυθαίρετα επιλεγμένη σταθερά ή
  2. μία σταθερά ή
  3. ένα λάθος ή
  4. απροσδιόριστη συμπεριφορά.

Συστήματα Τύπων & Ευρωστία Τύπου Γ.Π.

- Ένα από τα αντικείμενα σχεδίασης Γ.Π. αναφέρεται στην ταξινόμηση, την περιγραφή και τη μελέτη συστημάτων τύπων.

Σύστημα Τύπων & Ευρωστία Τύπου Γ.Π.

Το **σύστημα τύπων** μιας Γ.Π. αποσκοπεί στην αποτροπή λαθών κατά την εκτέλεση των προγραμμάτων (*λάθη εκτέλεσης*). Όταν αυτό συμβαίνει, τότε λέμε ότι η Γ.Π. χαρακτηρίζεται από **ευρωστία τύπου** (type soundness).

- Το σύστημα τύπων παρακολουθεί τους τύπους των μεταβλητών και των εκφράσεων στα προγράμματα. Επιτρέπεται η εκτέλεση μόνο των προγραμμάτων που συμμορφώνονται με το σύστημα τύπων.
- Για την αποφυγή εσφαλμένων υποθέσεων ως προς την ευρωστία τύπου απαιτείται ακριβής ανάλυση.

Συστήματα Τύπων & Ευρωστία Τύπου Γ.Π.

- Υιοθετούμε μαθηματικούς ορισμούς και συμβολισμούς, καθώς και αναλυτικές αποδείξεις για τυπικές ιδιότητες σχετικά με την καταλληλότητα των ορισμών.
- Όταν η περιγραφή του συστήματος τύπων μιας Γ.Π. δεν είναι διατυπωμένη με μαθηματική αυστηρότητα, αυτή είναι πηγή ασάφειας για την υλοποιήση της γλώσσας:
  1. Διαφορετικοί μεταγλωττιστές για την ίδια Γ.Π. υλοποιούν διαφορετικά συστήματα τύπων.
  2. Μπορεί ένας ορισμός Γ.Π. να μη διαθέτει ευρωστία τύπου. Δίνεται έτσι δυνατότητα σύνταξης προγραμμάτων που καταρρέουν, αν και αυτά περνάνε από το μεταγλωττιστή.

Εμφανείς & άδηλοι τύποι

- Μία Γ.Π. μπορεί να διαθέτει σύστημα τύπων ανεξάρτητα από το αν οι τύποι εφανίζονται στη σύνταξη ή όχι.

Ορισμός 2 (Εμφανείς & άδηλοι τύποι)

Αν οι τύποι είναι μέρος της σύνταξης μιας Γ.Π., τότε λέμε ότι η γλώσσα διαθέτει **εμφανείς τύπους** (explicitly typed), ενώ σε διαφορετική περίπτωση έχουμε **άδηλους τύπους** (implicitly typed).

- Οι περισσότερες Γ.Π. διαθέτουν αποκλειστικά εμφανείς τύπους.
- Οι Γ.Π. ML και Haskell έχουν μεικτό σύστημα τύπων: μπορεί σε ένα μέρος του προγράμματος να παραλείπονται οι πληροφορίες τύπων, οπότε αυτές συνάγονται αυτόματα.

Λάθη Εκτέλεσης & Ασφάλεια

- Δύο κατηγορίες λαθών χρόνου εκτέλεσης:
  1. **Λάθη κατάρρευσης**, που προκαλούν διακοπή της εκτέλεσης του προγράμματος (π.χ. διαίρεση με το μηδέν, προσπέλαση μη έγκυρης δ/νσης μνήμης κ.α.)
  2. **Λάθη ασταθούς συμπεριφοράς** που μπορεί να περνούν απαρατήρητα, αλλά εκτρέπουν την εκτέλεση σε ανακόλουθη συμπεριφορά (π.χ. προσπέλαση δεδομένου πέρα από το όριο πίνακα όταν δε γίνεται έλεγχος ορίων σε χρόνο εκτέλεσης).
- Ένα πρόγραμμα θεωρείται **ασφαλές** αν δεν περιέχει λάθη ασταθούς συμπεριφοράς. Αν όλα τα πιθανά προγράμματα μιας Γ.Π. είναι ασφαλή, τότε λέμε ότι η γλώσσα είναι **ασφαλής**.
- Οι χωρίς τύπους Γ.Π. μπορεί να επιβάλουν την ασφάλεια με **ελέγχους σε χρόνο εκτέλεσης**.

Λάθη Εκτέλεσης & Ασφάλεια

- Οι Γ.Π. με τύπους επιβάλουν την ασφάλεια απορρίπτοντας με στατική ανάλυση όλα τα προγράμματα που πιθανώς δεν είναι ασφαλή. Επίσης, μπορεί να χρησιμοποιούν ένα μείγμα από **στατικούς ελέγχους** και ελέγχους σε χρόνο εκτέλεσης.
- Για τις περισσότερες Γ.Π. όλα τα λάθη ασταθούς συμπεριφοράς, καθώς και ένα υποσύνολο των λαθών κατάρρευσης θεωρούνται **απαγορευμένα λάθη**.

Ορισμός 3 (Καλώς συμπεριφερόμενα προγράμματα)

Ένα πρόγραμμα λέμε ότι επιδεικνύει **καλή συμπεριφορά** αν δεν περιλαμβάνει απαγορευμένα λάθη (άρα είναι ασφαλές).

Ορισμός 4 (Ισχυρά ελεγχόμενη Γ.Π.)

Μία Γ.Π. της οποίας όλα τα επιτρεπόμενα προγράμματα έχουν καλή συμπεριφορά ονομάζεται **ισχυρά ελεγχόμενη** (strongly checked).

Λάθη Εκτέλεσης & Ασφάλεια

Ισχυρά Ελεγχόμενη Γ.Π.

Για μία ισχυρά ελεγχόμενη Γ.Π. σε σχέση με το σύστημα τύπων της ισχύουν τα εξής:

- Δε συμβαίνουν λάθη ασταθούς συμπεριφοράς (άρα είναι ασφαλής).
- Δε συμβαίνει κανένα από τα λάθη κατάρρευσης που θεωρούνται απαγορευμένα (τα επιτρεπτά λάθη κατάρρευσης είναι ευθύνη του προγραμματιστή).

Ορισμός 5 (Έλεγχος Τύπων)

Ο Έλεγχος Τύπων είναι ένας αλγόριθμος που σε χρόνο μεταγλώττισης απορρίπτει τα κακώς συμπεριφερόμενα προγράμματα. Ένα πρόγραμμα που περνά τον έλεγχο τύπων λέμε ότι έχει **καλώς ορισμένους τύπους** (well typed).

Λάθη Εκτέλεσης & Ασφάλεια

- Παραδείγματα Γ.Π. με τύπους και στατικό έλεγχο: ML, Java, Pascal (η Pascal διαθέτει κάποια μη ασφαλή χαρακτηριστικά).
- Οι Γ.Π. χωρίς τύπους επιβάλουν την καλή συμπεριφορά (άρα και την ασφάλεια) μέσω ελέγχων σε χρόνο εκτέλεσης, όπως π.χ. η LISP.
  Π.χ. ελέγχονται δυναμικά όλα τα όρια των πινάκων και οι διαιρέσεις και ενδεχόμενη εμφάνιση απαγορευμένου λάθους σηματοδοτεί την ενεργοποιήση διαχειρίσιμης εξαίρεσης.
- Ακόμη και οι Γ.Π. με στατικό έλεγχο χρειάζεται να διεξάγουν κάποιους ελέγχους σε χρόνο εκτέλεσης για να εγγυηθούν την ασφαλή εκτέλεση, π.χ. οι έλεγχοι των ορίων πινάκων γενικά γίνονται σε χρόνο εκτέλεσης.
- Αρκετές Γ.Π. αξιοποιούνε τις στατικές δομές τύπων για να διεξάγουνε προηγμένους δυναμικούς ελέγχους, π.χ. την εντολή `TYPECASE` στη Modula-3 και την εντολή `instanceof` στη Java.

Μη ασφαλή προγράμματα

- Ο πρωταρχικός σκοπός ενός συστήματος τύπων είναι να απορρίπτει **όλα** τα λάθη ασταθούς συμπεριφοράς για όλες τις πιθανές εκτελέσεις του προγράμματος (ασφάλεια).
- Ο δεδηλωμένος σκοπός για τα περισσότερα συστήματα τύπων είναι η πιο γενική ιδιότητα των καλώς συμπεριφερόμενων προγραμμάτων, που έχει ως επακόλουθο την ασφάλεια. Άρα, αυτά τα συστήματα τύπων μπορούν θεωρητικά να αναγνωρίσουν αν ένα πρόγραμμα έχει καλώς ορισμένους τύπους.
- Κάποιες Γ.Π. με στατικό έλεγχο δεν εγγυώνται ασφαλή εκτέλεση του προγράμματος: το σύνολο των απαγορευμένων λαθών δεν περιλαμβάνει όλα τα πιθανά λάθη ασταθούς συμπεριφοράς. Λέμε ότι αυτές οι Γ.Π. είναι **δεν είναι ισχυρά ελεγχόμενες**. Π.χ. στη γλώσσα C τέτοια χαρακτηριστικά είναι η αριθμητική δεικτών και οι μετατροπές (casting).

Μη ασφαλή προγράμματα

- Οι περισσότερες Γ.Π. χωρίς τύπους είναι ασφαλείς (π.χ. η LISP), γιατί αλλιώς ο προγραμματισμός θα ήταν πολύ δύσκολος. Οι Γ.Π. assembly είναι χωρίς τύπους και χωρίς ασφάλεια.

	Με τύπους	**Χωρίς τύπους**
Ασφαλείς	`ML, Java`	`LISP`
Μη ασφαλείς	`C`	`Assembly`

Ιδιότητες Συστημάτων Τύπων

Ένα σύστημα τύπων θα πρέπει:

- να **καθιστά αποφασίσιμη την επαλήθευση** προγραμμάτων (decidable verifiable), δηλ. να υπάρχει αλγόριθμος που να εξασφαλίζει ότι ένα πρόγραμμα είναι καλώς συμπεριφερόμενο
- να είναι **διάφανο ως προς την ανάπτυξη προγραμμάτων** (transparent), δηλ. να μπορούμε να διακρίνουμε αν ένα πρόγραμμα περνάει τον έλεγχο τύπων
- να **επιβάλλει τη λειτουργία του** (enforceable)
  - οι δηλώσεις τύπων να ελέγχονται στατικά αν γίνεται ή όταν αυτό δεν είναι εφικτό να ελέγχονται δυναμικά
  - να επαληθεύεται η συνέπεια των προγραμμάτων με τις δηλώσεις τύπων (εξαιρούνται τα σχόλια και άλλες συμβάσεις των Γ.Π.)

Θεώρημα Ευρωστίας Τύπων

Τυπικός ορισμός συστήματος τύπων (formal type system) είναι η μαθηματική προδιαγραφή της άτυπης περιγραφής που περιέχεται στο εγχειρίδιο μιας Γ.Π. Αν ένα σύστημα τύπων έχει τυποποιηθεί, τότε μπορεί να αποδειχθεί (αν ισχύει) το:

Θεώρημα 6 (Ευρωστίας Τύπων)

Όλα τα προγράμματα με καλώς ορισμένους τύπους είναι καλώς συμπεριφερόμενα (άρα και ασφαλή). Ένα τέτοιο σύστημα τύπων λέμε ότι είναι ***εύρωστο****.*

Διαδικασία Ορισμού Συστήματος Τύπων

Βήματα για τον τυπικό ορισμό ενός συστήματος τύπων:

1. Ορίζουμε τη σύνταξη των τύπων και των **όρων**. Οι τύποι εκφράζουν στατική γνώση για τα προγράμματα ενώ οι όροι (εντολές, εκφράσεις κ.α.) εκφράζουν την αλγοριθμική τους συμπεριφορά.

2. Ορίζουμε τους **κανόνες εμβέλειας** (scoping rules), που αντιστοιχούν την κάθε εμφάνιση ονόματος στο σημείο της δήλωσής του.
   Οι Γ.Π. με τύπους χρειάζονται κανόνες **στατικής εμβέλειας** (static scoping), δηλ. οι αντιστοιχίσεις γίνονται πριν από την εκτέλεση. Αν οι αντιστοιχίσεις καθορίζονται απευθείας από τη σύνταξη, τότε λέμε ότι έχουμε **λεξική εμβέλεια**.
   Αν δεν υπάρχει στατική εμβέλεια, τότε έχουμε **δυναμική εμβέλεια** (dynamic scoping).

Διαδικασία Ορισμού Συστήματος Τύπων

Παράδειγμα Κανόνων Εμβέλειας της C

- Ένα όνομα έχει **τοπική εμβέλεια**, αν ορίζεται μέσα σε ένα block, οπότε μπορεί να χρησιμοποιηθεί στο block αυτό και σε όσα περικλείει αρκεί να έχει προηγουμένως δηλωθεί. Κατά την έξοδο από ένα bock τα ονόματα που δηλώνονται σε αυτό δεν είναι πλέον διαθέσιμα.
- Τα ονόματα των παραμέτρων συνάρτησης έχουν εμβέλεια όλο το block με τον κώδικα της συνάρτησης. Αν η συνάρτηση δηλώνεται αλλά δεν ορίζεται, τότε η εμβέλεια περιορίζεται στο πρωτότυπο της συνάρτησης.
- Αν ένα block περιέχει ένα άλλο block, οι μεταβλητές του είναι ορατές στο ένθετο block. Αν όμως μια μεταβλητή του ένθετου δηλώνεται με το όνομα μεταβλητής του περικλείοντος block, τότε η ένθετη δήλωση αποκρύπτει μέρος της εμβέλειας της δήλωσης του περικλείοντος block.
- Μόνο τα ονόματα ετικετών έχουν **εμβέλεια συνάρτησης**. Η δήλωση ετικέτας υπονοείται από την εμφάνιση της στο πρόγραμμα και είναι ορατή στη συνάρτηση που αυτή εμφανίζεται. Μία ετικέτα μπορεί να χρησιμοποιηθεί σε εντολή goto πριν από την δήλωσή της.
- Ένα όνομα έχει **εμβέλεια αρχείου** αν δεν εμφανίζεται μέσα σε block.

Διαδικασία Ορισμού Συστήματος Τύπων

3. Ορίζουμε τους **κανόνες τύπων**. Αυτοί περιγράφουν μία σχέση $M : A$, που εκφράζει ότι οι όροι M έχουν τύπους A.
Οι κανόνες τύπων ορίζονται πάντα ως προς ένα **στατικό περιβάλλον τύπων**: περιλαμβάνει τους τύπους των ελεύθερων μεταβλητών των διαφόρων μερών του προγράμματος (δηλ. τις πληροφορίες του πίνακα συμβόλων). Έτσι, η σχέση *has-type* που γράφεται ως
$$\Gamma \vdash M : A$$
σημαίνει ότι «οι όροι M έχουν τύπους A στο περιβάλλον Γ».
Τα σύνθετα συστήματα τύπων απαιτούν και τον ορισμό μιας σχέσης subtype-of.

4. Ορίζουμε τη σημασία της γλώσσας ως μία σχέση *has-value* μεταξύ των όρων της και μιας συλλογής **αποτελεσμάτων**. Η σημασία της Γ.Π. και το σύστημα τύπων της είναι αλληλένδετα: οι τύποι ενός όρου και το αποτέλεσμά του πρέπει ή να είναι ίδιοι ή να σχετίζονται κατάλληλα (π.χ. ισοδύναμοι τύποι από μία κατάλληλα ορισμένη σχέση equal-type).

Συμβολισμός Κανόνων Τύπων

Συμβολισμός για τον ορισμό συστήματος τύπων

- Χρησιμοποιούμε εκφράσεις της μορφής

  $\Gamma \vdash \mathcal{J}$ $\mathcal{J}$: ισχυρισμός με ελεύθερες μεταβλητές δηλωμένες στο Γ που διαβάζεται ως «Γ συνεπάγεται $\mathcal{J}$».

  Το Γ είναι ένα περιβάλλον τύπων, π.χ. μία διατεταγμένη λίστα μεταβλητών με τους τύπους τους, της μορφής $\emptyset, x_1 : A_1, ..., x_n : A_n$. Συμβολίζουμε με $\emptyset$ το κενό περιβάλλον και με $dom(\Gamma)$ τη συλλογή των μεταβλητών $x_1, ..., x_n$ που δηλώνονται στο Γ.

- Εκφράσεις τύπων

  $\Gamma \vdash M : A$: ο όρος M έχει τύπο A στο Γ

  Παραδείγματα:

  $\emptyset \vdash true : Bool$: το $true$ έχει τύπο $Bool$

  $\emptyset, x : Nat \vdash x + 1 : Nat$: $x + 1$ έχει τύπο Nat, αν το x έχει τύπο Nat

- Λέμε ότι «ένα περιβάλλον είναι καλώς ορισμένο» αν $\Gamma \vdash \diamond$.

- Μία δοθείσα έκφραση μπορεί να είναι έγκυρη (π.χ. $\Gamma \vdash true : Bool$) ή άκυρη (π.χ. $\Gamma \vdash true : Nat$). Η εγκυρότητα των εκφράσεων τύπων προσδιορίζει αν τα προγράμματα έχουν καλώς ορισμένους τύπους.

Συμβολισμός Κανόνων Τύπων

Κανόνες τύπων

- Συνάγουν την εγκυρότητα εκφράσεων τύπων με βάση κάποιες άλλες εκφράσεις, που είναι αποδεδειγμένα έγκυρες. Η διαδικασία ξεκινάει από μία τετρριμμένα έγκυρη έκφραση, συνήθως την $\emptyset \vdash \diamond$.
- Γενική μορφή κανόνων τύπων:

$$\frac{\Gamma \vdash \mathcal{J}_1 \quad \ldots \quad \Gamma \vdash \mathcal{J}_n}{\Gamma \vdash \mathcal{J}} \quad \text{(όνομα κανόνα)}$$

Περιλαμβάνουν τις εκφράσεις της υπόθεσης (πάνω από τη γραμμή) και τις εκφράσεις του συμπεράσματος (κάτω από τη γραμμή). Όταν ικανοποιούνται όλες οι υποθέσεις, τότε ισχύει το συμπερασμα.

Παραδείγματα:

$$\frac{\Gamma \vdash \diamond}{\Gamma \vdash n{:}Nat}\ (\text{Val } n)\ \ (\text{n=0,1,...}) \qquad \frac{\Gamma \vdash M{:}Nat \quad \Gamma \vdash N{:}Nat}{\Gamma \vdash M{+}N{:}Nat}\ (\text{Val } +)$$

Ο Val n λέει ότι κάθε αριθμός είναι τύπου Nat σε ένα καλώς ορισμένο περιβάλλον. Ο Val+ δίνει τον τύπο του αθροίσματος αριθμών Nat.

Τυπικός Ορισμός Συστήματος Τύπων

- Ένας θεμελιώδης κανόνας είναι ότι το κενό περιβάλλον είναι καλώς ορισμένο, χωρίς υποθέσεις:

$$\frac{}{\emptyset \vdash \diamond} \quad (\text{Env } \emptyset)$$

Ορισμός 7 (Τυπικά ορισμένο σύστημα τύπων)

Ένα σύστημα τύπων είναι τυπικά ορισμένο αν αυτό περιγράφεται από μία συλλογή κανόνων τύπων, που υποστηρίζει τη βήμα προς βήμα απόδειξη του τύπου για ένα οποιοδήποτε μέρος προγράμματος.

- Ορίζουμε το σύστημα τύπων μιας Γ.Π. ανεξάρτητα από τον αλγόριθμο ελέγχου τύπων, σε αντιστοιχία με τη σύνταξη, που ορίζεται ανεξάρτητα από τον αλγόριθμο συντακτικής ανάλυσης.
- Αυτό σημαίνει ότι είναι πιθανό να ορίσουμε ένα σύστημα τύπων που καθιστά ανέφικτο τον έλεγχο τύπων με έναν αλγόριθμο.

Παραγωγές Τύπων

- Παραγωγή σε ένα σύστημα τύπων είναι ένα δέντρο εκφράσεων με τα φύλλα στο πάνω μέρος και τη ρίζα στο χαμηλότερο επίπεδο, έτσι ώστε κάθε έκφραση προκύπτει από τις εκφράσεις στο αμέσως υψηλότερο επίπεδο με κάποιο κανόνα τύπων. Παράδειγμα:

$$\dfrac{\dfrac{\dfrac{}{\emptyset \vdash \diamond}\ \text{by (Env } \emptyset)}{\emptyset \vdash 1 : \mathit{Nat}}\ \text{by (Val } n) \qquad \dfrac{\dfrac{}{\emptyset \vdash \diamond}\ \text{by (Env } \emptyset)}{\emptyset \vdash 2 : \mathit{Nat}}\ \text{by (Val } n)}{\emptyset \vdash 1{+}2 : \mathit{Nat}}\ \text{by (Val +)}$$

- Μία **έγκυρη έκφραση** είναι αυτή που μπορεί να ληφθεί ως ρίζα μιας παραγωγής σε ένα δοθέν σύστημα τύπων.

Καλώς Ορισμένοι Τύποι & Συναγωγή Τύπων

Ορισμός 8 (Καλώς Ορισμένος Τύπος)

Σε ένα δοθέν σύστημα τύπων ο όρος M έχει καλώς ορισμένο τύπο στο περιβάλλον Γ, αν υπάρχει τύπος A ώστε η έκφραση $\Gamma \vdash M : A$ να είναι έγκυρη, δηλ. τελικά μπορεί να αποδοθεί ένας τύπος στον όρο M.

- **Συναγωγή τύπου** (type inference) είναι η ανακάλυψη παραγωγής τύπου για έναν όρο του προγράμματος.
  Παράδειγμα:
  Στο σύστημα τύπων με κανόνες (Env $\emptyset$), (Val n), (Val+) συνάγεται τύπος για τον όρο $1 + 2$ στο κενό περιβάλλον: ο τύπος *Nat* από την παραγωγή της προηγούμενης διαφάνειας.
- **Λάθος τύπων** (typing error)
  Στο σύστημα τύπων του παραδείγματος προσθέτουμε τον κανόνα:
  $$\frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathit{true} : \mathit{Bool}}$$
  Στο νέο σύστημα δε μπορούμε να συνάγουμε τύπο για τον όρο $1 + \mathit{true}$, επειδή δεν υπάρχει κανόνας για άθροισμα *Nat* με *Bool*. Λέμε ότι ο όρος $1 + \mathit{true}$ δεν έχει καλώς ορισμένο τύπο (ή λάθος τύπου).

Καλώς Ορισμένοι Τύποι & Συναγωγή Τύπων

- Το πρόβλημα της συναγωγής τύπου για έναν όρο είναι αλληλένδετο με τον ορισμό του συστήματος τύπων.

  Παράδειγμα: Έστω ότι προσθέτουμε τον κανόνα

  $$\frac{\Gamma \vdash M : Nat \quad \Gamma \vdash N : Bool}{\Gamma \vdash M + N : Nat}$$

  στο σύστημα τύπων του προηγούμενου παραδείγματος, με το σκεπτικό ότι ο όρος *true* ερμηνεύεται ως 1. Τότε, στο νέο σύστημα τύπων μπορούμε να συνάγουμε τύπο για το όρο $1 + true$ (καλώς ορισμένος).
- Αν και τα συστήματα τύπων σχεδιάζονται σε αφαιρετικό επίπεδο ως προς τη Γ.Π., η χρησιμότητά τους εξαρτάται από το αν υπάρχουν καλοί, δηλ. αποδοτικοί αλγόριθμοι συναγωγής τύπου.
  - Το πρόβλημα της συναγωγής τύπου για διαδικασιακές Γ.Π. με εμφανείς τύπους (π.χ. Pascal) έχει επιλυθεί με αλγορίθμους που έχουν υλοποιηθεί με επιτυχία.
  - Το πρόβλημα έχει λυθεί και για τα πολυμορφικά χαρακτηριστικά με εμφανείς τύπους σε Γ.Π. όπως οι Ada, ML.
  - Υπάρχουν ακόμη δυσκολίες στην υλοποίηση για χαρακτηριστικά Γ.Π. με υπονοούμενους τύπους.

Τυπικά Ορισμένο Σύστημα Τύπων & Ευρωστία Τύπου

- Ένα χρήσιμο σύστημα τύπων δεν είναι απλά ένα σύνολο κανόνων. Πρέπει να ελεγχθεί η εσωτερική συνέπεια του συστήματος αποδεικνύοντας ένα θεώρημα ευρωστίας τύπου.
- Έτσι θεμελιώνεται ότι τα προγράμματα με καλώς ορισμένους τύπους αντιστοιχούν σε κάποια σημασιολογική ερμηνεία καλής συμπεριφοράς και με αυτό τον τρόπο τα συστήματα τύπων συναρτώνται με τη σημασία των Γ.Π.

Λογισμός λ με τύπους

Ορισμός 9 (Συστήματα Τύπων Πρώτης Τάξης)

Είναι τα συστήματα τύπων που διαθέτουν οι κλασσικές διαδικασιακές Γ.Π. (Pascal, FORTRAN, Algol68) και δεν περιλμβάνουν δυνατότητες παραμετροποιήσης τύπου, καθώς και αφαιρετικούς τύπους.

Λογισμός λ με τύπους

- Είναι ένα μαθηματικό σύστημα επίδειξης εννοιών Γ.Π. με έναν απλό τρόπο. Περιλαμβάνει ένα συμβολισμό ορισμού συναρτήσεων, ένα αποδεικτικό σύστημα με το οποίο μπορούμε να δείξουμε εξισώσεις μεταξύ εκφράσεων και ένα σύνολο κανόνων υπολογισμού που λέγονται (reductions). Τα reductions χρησιμοποιούνται για υπολογισμό του αποτελέσματος μιας συνάρτησης για μία ή περισσότερες παραμέτρους.
- Η έκφραση $\lambda x.M$ αναπαριστά μία συνάρτηση με παράμετρο x και αποτέλεσμα M.
- Εισάγουμε επισημειώσεις τύπων χρησιμοποιώντας της σύνταξη $\lambda x : A.M$, που σημαίνει ότι η x είναι παράμετρος της συνάρτησης, το A είναι ο τύπος της και M το σώμα της.

Σύστημα Τύπων F_1

Ορισμός 10 (Σύστημα Τύπων F_1 για τον λογισμό λ με τύπους)

Περιλαμβάνει ένα σύνολο *Basic* βασικών τύπων, όπως οι τύποι *Bool* και *Nat*, καθώς και τύπους συναρτήσεων της μορφής $A \to B$, δηλ. συναρτήσεων με παραμέτρους τύπου A και αποτελέσματα τύπου B.

- Σύνταξη συστήματος τύπων F_1
  Θεωρούμε ότι τα A, B είναι τύποι, ενώ τα M, N είναι όροι.

$$
\begin{array}{lll}
A,B ::= & K & K \in \mathit{Basic} \\
\mid & A \to B & \text{τύπος συνάρτησης} \\
M,N ::= & x & \text{μεταβλητή} \\
\mid & \lambda x : A.M & \text{συνάρτηση} \\
\mid & M\ N & \text{εφαρμογή}
\end{array}
$$

Σύστημα Τύπων F_1

- Ο ρόλος της Γ.Χ.Σ. σε ένα σύστημα τύπων είναι ο καθορισμός των κανόνων εμβέλειας, δηλαδή να διακρίνονται οι ελεύθερες μεταβλητές στις συμβολοσειρές.
  Παράδειγμα:
  Με βάση της γραμματική του F_1 οι εκφράσεις $\lambda x : K.x$ και $\lambda y : K.y$ δε διαφέρουν στη σύνταξη.
- Η συμβολοσειρά $\lambda x : K.x(y)$ είναι συντακτικά σωστή σύμφωνα με τη Γ.Χ.Σ. του F_1, αλλά δεν έχει καλώς ορισμένους τύπου αφού το K δεν είναι τύπος συνάρτησης.

Κανόνες Τύπων

Εκφράσεις στο σύστημα τύπων F_1:

$\Gamma \vdash \diamond$ το Γ είναι καλώς ορισμένο

$\Gamma \vdash A$ ο A είναι καλώς ορισμένος τύπος στο Γ

$\Gamma \vdash M : A$ ο M είναι καλώς ορισμένος όρος τύπου A στο Γ

(Env $\emptyset$)
$$\frac{}{\emptyset \vdash \diamond}$$

(Env x)
$$\frac{\Gamma \vdash A \qquad x \notin dom(\Gamma)}{\Gamma, x{:}A \vdash \diamond}$$

(Type Const)
$$\frac{\Gamma \vdash \diamond \qquad K \in Basic}{\Gamma \vdash K}$$

(Type Arrow)
$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \to B}$$

(Val x)
$$\frac{\Gamma', x{:}A, \Gamma'' \vdash \diamond}{\Gamma', x{:}A, \Gamma'' \vdash x{:}A}$$

(Val Fun)
$$\frac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash \lambda x{:}A.M : A \to B}$$

(Val Appl)
$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\,N : B}$$

Κανόνες Τύπων

Ερμηνεία κανόνων του συστήματος τύπων F_1:

- **Κανόνας Env x**: Επεκτείνει το Γ σε ένα μεγαλύτερο περιβάλλον $\Gamma, x : A$ αν ο A είναι έγκυρος στο Γ. Η x δε θα πρέπει να έχει οριστεί στο Γ. Προσέχουμε να διατηρούμε τις μεταβλητές σε διακριτά περιβάλλοντα, έτσι ώστε π.χ. για την $\Gamma, x : A \vdash M : A$ από τον Val Fun να ξέρουμε ότι η x δεν εμφανίζεται στο $dom(\Gamma)$.
- **Κανόνες Type Const, Type Arrow**: Δημιουργούνε τύπους.
- **Κανόνας Val x**: Εξάγει την υπόθεση $x : A$ από ένα περιβάλλον (τα Γ' και Γ'' αναφέρονται - παράτυπα - στο ίδιο περιβάλλον).
- **Κανόνας Val Fun**: Αποδίδει τον τύπο $A \to B$ σε μία συνάρτηση, υπό την προϋπόθεση ότι η συνάρτηση παίρνει τον τύπο B όταν δέχεται παράμετρο A. Το περιβάλλον Γ μεγαλώνει.
- **Κανόνας Val App**: Εφαρμόζει μία συνάρτηση σε μία παράμετρο.

Παραγωγή στο Σύστημα Τύπων

Παραγωγή στο σύστημα τύπων F_1:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\emptyset \vdash \diamond}\ \text{by (Env } \emptyset)}{\emptyset \vdash K}\ \text{by (Type Const)} \quad \dfrac{\dfrac{}{\emptyset \vdash \diamond}\ \text{by (Env } \emptyset)}{\emptyset \vdash K}\ \text{by (Type Const)}}{\emptyset \vdash K{\to}K}\ \text{by (Type Arrow)}}{\emptyset,\ y{:}K{\to}K \vdash \diamond}\ \text{by (Env } x)}{\emptyset,\ y{:}K{\to}K \vdash K}\ \text{by (Type Const)}}{\emptyset,\ y{:}K{\to}K,\ z{:}K \vdash \diamond}\ \text{by (Env } x)}{\emptyset,\ y{:}K{\to}K,\ z{:}K \vdash y : K{\to}K}\ \text{by (Val } x) \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\emptyset \vdash \diamond}\ \text{by (Env } \emptyset)}{\emptyset \vdash K}\ \text{by (Type Const)} \quad \dfrac{\dfrac{}{\emptyset \vdash \diamond}\ \text{by (Env } \emptyset)}{\emptyset \vdash K}\ \text{by (Type Const)}}{\emptyset \vdash K{\to}K}\ \text{by (Type Arrow)}}{\emptyset,\ y{:}K{\to}K \vdash \diamond}\ \text{by (Env } x)}{\emptyset,\ y{:}K{\to}K \vdash K}\ \text{by (Type Const)}}{\emptyset,\ y{:}K{\to}K,\ z{:}K \vdash \diamond}\ \text{by (Env } x)}{\emptyset,\ y{:}K{\to}K,\ z{:}K \vdash z : K}\ \text{by (Val } x)}{\emptyset,\ y{:}K{\to}K,\ z{:}K \vdash y(z) : K}\ \text{by (Val Appl)}}{\emptyset,\ y{:}K{\to}K \vdash \lambda z{:}K.y(z) : K{\to}K}\ \text{by (Val Fun)}$$

Κανόνες για Βασικούς Τύπους

Βασικοί τύποι στο F_1:

- Τύπος *Unit* για συναρτήσεις χωρίς παραμέτρους ή/και αποτέλεσμα (σε μερικές Γ.Π. ο τύπος λέγεται `Void` ή `Null`)

$$\frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathit{Unit}} \text{ (Type Unit)} \qquad \frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathit{unit}:\mathit{Unit}} \text{ (Val Unit)}$$

- Τύπος *Bool*

$$\frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathit{Bool}} \text{ (Type Bool)} \qquad \frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathit{true}:\mathit{Bool}} \text{ (Val True)} \qquad \frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathit{false}:\mathit{Bool}} \text{ (Val False)}$$

$$\text{(Val Cond)} \ \frac{\Gamma \vdash M:\mathit{Bool} \quad \Gamma \vdash N_1:A \quad \Gamma \vdash N_2:A}{\Gamma \vdash (\mathit{if}_A\ M\ \mathit{then}\ N_1\ \mathit{else}\ N_2):A}$$

Στην περίπτωση της if ο έλεγχος τύπων θα πρέπει να συνάγει χωριστά τους τύπους των N_1 και N_2 και στη συνέχεια να βρει έναν τύπο A που να είναι συμβατός και με τους δυο. Ο δείκτης A στο *if* σημειώνει την πληροφορία για τον έλεγχο τύπων ότι το αποτέλεσμα θα πρέπει να είναι τύπου A.

Κανόνες για Βασικούς Τύπους

Βασικοί τύποι στο F_1:

- Τύπος *Nat*

$$\text{(Type Nat)}\ \frac{\Gamma \vdash \diamond}{\Gamma \vdash Nat} \qquad \text{(Val Zero)}\ \frac{\Gamma \vdash \diamond}{\Gamma \vdash 0 : Nat} \qquad \text{(Val Succ)}\ \frac{\Gamma \vdash M : Nat}{\Gamma \vdash succ\ M : Nat}$$

$$\text{(Val Pred)}\ \frac{\Gamma \vdash M : Nat}{\Gamma \vdash pred\ M : Nat} \qquad \text{(Val IsZero)}\ \frac{\Gamma \vdash M : Nat}{\Gamma \vdash isZero\ M : Bool}$$

Ο τύπος *Nat* δημιουργείται από το 0 και το *succ* (επόμενος). Στους υπολογισμούς χρησιμοποιούνται τα *pred* (προηγούμενος) και *isZero* (έλεγχος για 0).

Κανόνες για Οριζόμενους Τύπους

- Τύπος *Product*:
  περιλαμβάνει ζεύγη τιμών, που η πρώτη είναι τύπου A_1 και η δεύτερη τύπου A_2. Τα συστατικά κάθε ζεύγους εξάγονται με τις λειτουργίες προβολής *first* και *second*.

$$\frac{\Gamma \vdash A_1 \quad \Gamma \vdash A_2}{\Gamma \vdash A_1 \times A_2} \text{ (Type Product)} \qquad \frac{\Gamma \vdash M_1 : A_1 \quad \Gamma \vdash M_2 : A_2}{\Gamma \vdash (M_1, M_2) : A_1 \times A_2} \text{ (Val Pair)}$$

$$\frac{\Gamma \vdash M : A_1 \times A_2}{\Gamma \vdash \mathit{first}\ M : A_1} \text{ (Val First)} \qquad \frac{\Gamma \vdash M : A_1 \times A_2}{\Gamma \vdash \mathit{second}\ M : A_2} \text{ (Val Second)}$$

$$\text{(Val With)} \ \frac{\Gamma \vdash M : A_1 \times A_2 \quad \Gamma, x_1 : A_1, x_2 : A_2 \vdash N : B}{\Gamma \vdash (\mathit{with}\ (x_1 : A_1, X_2 : A_2) := M\ \mathit{do}\ N) : B}$$

Η εντολή with αποσυνθέτει ένα ζεύγος M και συνδέει τα συστατικά του με δύο μεταβλητές x_1 και x_2 στην εμβέλεια του όρου N. Ο κανόνας αυτός περιγράφει τις λειτουργίες ταιριάσματος προτύπων της Γ.Π. ML και συνδέεται με το `with` της Pascal.

Κανόνες για Οριζόμενους Τύπους

- Τύπος *Union*:
  ένα στοιχείο τύπου $A_1 + A_2$ είτε είναι στοιχείο του A_1 με επισήμανση *left* (δημιουργείται με τον κανόνα *inLeft*), είτε στοιχείο του A_2 με επισήμανση *right* (κανόνας *inRight*)

$$\frac{\Gamma \vdash A_1 \quad \Gamma \vdash A_2}{\Gamma \vdash A_1 + A_2}$$
(Type Union)

$$\frac{\Gamma \vdash M_1 : A_1 \quad \Gamma \vdash A_2}{\Gamma \vdash inLeft_{A_2}\, M_1 : A_1 + A_2}$$
(Val inLeft)

$$\frac{\Gamma \vdash A_1 \quad \Gamma \vdash M_2 : A_2}{\Gamma \vdash inRight_{A_1}\, M_2 : A_1 + A_2}$$
(Val inRight)

$$\frac{\Gamma \vdash M : A_1 + A_2}{\Gamma \vdash isLeft\ M : Bool}$$
(Val isLeft)

$$\frac{\Gamma \vdash M : A_1 + A_2}{\Gamma \vdash isRight\ M : Bool}$$
(Val isRight)

$$\frac{\Gamma \vdash M : A_1 + A_2}{\Gamma \vdash asLeft\ M : A_1}$$
(Val asLeft)

$$\frac{\Gamma \vdash M : A_1 + A_2}{\Gamma \vdash asRight\ M : A_2}$$
(Val asRight)

$$\frac{\Gamma \vdash M : A_1 + A_2 \quad \Gamma, x_1 : A_1 \vdash N_1 : B \quad \Gamma, x_2 : A_2 \vdash N_2 : B}{\Gamma \vdash (case_B\, M\ of\ x_1 : A_1\ then\ N_1 \mid x_2 : A_2\ then\ N_2) : B}$$
(Val Case)

Η επισήμανση ελέγχεται με τους κανόνες *isLeft*,*isRight* και ο αντίστοιχος όρος εξάγεται με τους *asLeft*, *asRight*.

Κανόνες για Οριζόμενους Τύπους

- Τύπος *Union* (. . . συνέχεια)

  Αν ο κανόνας *asLeft* εφαρμοστεί λανθασμένα στην τιμή με επισήμανση *right*, τότε δημιουργείται ένα λάθος κατάρρευσης ή μία εξαίρεση. Το λάθος αυτό **δε κατατάσσεται από τη Γ.Π. στα απαγορευμένα**.

  Υποθέτουμε με ασφάλεια ότι οποιοδήποτε αποτέλεσμα του *asLeft* είναι A_1, επειδή η παράμετρος ή έχει επισήμανση *left* οπότε το αποτέλεσμα θα είναι A_1 ή έχει επισήμανση *right* οπότε δεν έχουμε αποτέλεσμα.

  Ο κανόνας (Val Case) είναι μία κομψή περιγραφή σε αντικατάσταση των *isLeft*, *isRight*, *asLeft*, *asRight* και των σχετικών λαθών κατάρρευσης (εξαλείφει την εξάρτηση από τον τύπο Bool).
  Εκτελείται μία από δύο πιθανές περιπτώσεις, ανάλογα με την επισήμανση του M με περιεχόμενο που μπορεί να αναφέρεται στο x_1 του όρου N_1 ή στο x_2 του όρου N_2
  (ο δείκτης B στο *case* είναι πληροφορία για τον έλεγχο τύπων ότι το αποτέλεσμα θα είναι τύπου B και οι τύποι που συνάγονται για τους N_1, N_2 θε πρέπει να συγκρίνονται ο καθένας απευθείας με τον B).

Κανόνες για Οριζόμενους Τύπους

- Τύπος *Record*:
  οριζόμενοι τύποι που υποστηρίζουν την εξαγωγή των συστατικών μιας τιμής ονοματίζοντας τα. Το with έχει αντίστοιχη λειτουργία με την περίπτωση του product.

(Type Record) (l_i distinct)

$$\frac{\Gamma \vdash A_1 \quad ... \quad \Gamma \vdash A_n}{\Gamma \vdash Record(l_1{:}A_1, ..., l_n{:}A_n)}$$

(Val Record) (l_i distinct)

$$\frac{\Gamma \vdash M_1 : A_1 \quad ... \quad \Gamma \vdash M_n : A_n}{\Gamma \vdash record(l_1{=}M_1, ..., l_n{=}M_n) : Record(l_1{:}A_1, ..., l_n{:}A_n)}$$

(Val Record Select)

$$\frac{\Gamma \vdash M : Record(l_1{:}A_1, ..., l_n{:}A_n) \qquad j \in 1..n}{\Gamma \vdash M.l_j : A_j}$$

(Val Record With)

$$\frac{\Gamma \vdash M : Record(l_1{:}A_1, ..., l_n{:}A_n) \qquad \Gamma, x_1{:}A_1, ..., x_n{:}A_n \vdash N : B}{\Gamma \vdash (with\ (l_1{=}x_1{:}A_1, ..., l_n{=}x_n{:}A_n) := M\ do\ N) : B}$$

Κανόνες για Οριζόμενους Τύπους

- Τύπος *Variant*:
  οριζόμενοι τύποι unions από διακριτούς ονοματισμένους τύπους. Η λειτουργία *is l* γενικεύει τις *isLeft*, *isRight*, ενώ η *as l* γενικεύει αντίστοιχα τις *asLeft*, *asRight*.

$$\frac{\Gamma \vdash A_1 \quad \ldots \quad \Gamma \vdash A_n}{\Gamma \vdash Variant(l_1{:}A_1,\ldots,l_n{:}A_n)}$$
(Type Variant) (διακριτό l_i)

$$\frac{\Gamma \vdash A_1 \quad \ldots \quad \Gamma \vdash A_n \quad \Gamma \vdash M_j{:}A_j \quad j{\in}1..n}{\Gamma \vdash variant_{(l_1:A_1,\ldots,l_n:A_n)}(l_j{=}M_j){:}Variant(l_1{:}A_1,\ldots,l_n{:}A_n)}$$
(Val Variant) (διακριτό l_i)

$$\frac{\Gamma \vdash M{:}Variant(l_1{:}A_1,\ldots,l_n{:}A_n) \quad j{\in}1..n}{\Gamma \vdash M \text{ is } l_j{:}Bool}$$
(Val Variant Is) (διακριτό l_i)

$$\frac{\Gamma \vdash M{:}Variant(l_1{:}A_1,\ldots,l_n{:}A_n) \quad j{\in}1..n}{\Gamma \vdash M \text{ as } l_j{:}Bool}$$
(Val Variant As) (διακριτό l_i)

(Val Variant Case)
$$\frac{\Gamma \vdash M{:}Variant(l_1{:}A_1,\ldots,l_n{:}A_n) \quad \Gamma,x_1{:}A_1 \vdash N_1{:}B \ \ldots \ \Gamma,x_n{:}A_n \vdash N_n{:}B}{\Gamma \vdash (case_B\, M \text{ of } l_1{=}x_1{:}A_1 \text{ then } N_1 \mid \ldots \mid l_n{=}x_n{:}A_n \text{ then } N_n){:}B}$$

Όπως και στα unions έτσι και εδώ τα *is l* και *as l* μπορούν να αντικατασταθούν από την εντολή case.

Κανόνες για Οριζόμενους Τύπους

- Τύπος *Reference*:
  χρησιμοποιούνται ως θεμελιώδεις τύποι μεταβλητών διεύθυνσης. Ένα στοιχείο τύπου Ref(A) είναι μία μεταβλητή κελιού που περιέχει στοιχείο τύπου *A*. Ένα νέο κελί μπορεί να εκχωρηθεί με τον Val Ref, ενώ ένα κελί ενημερώνεται με τον Val Assign ή αποαναφέρεται με τον Val Deref.

$$\text{(Type Ref)}\ \frac{\Gamma \vdash A}{\Gamma \vdash Ref\, A} \qquad \text{(Val Ref)}\ \frac{\Gamma \vdash M : A}{\Gamma \vdash ref\, M : Ref\, A}$$

$$\text{(Val Deref)}\ \frac{\Gamma \vdash M : Ref\, A}{\Gamma \vdash deref\, M : A} \qquad \text{(Val Assign)}\ \frac{\Gamma \vdash M : Ref\, A \quad \Gamma \vdash N : A}{\Gamma \vdash M := N : Unit}$$

Τύπος Array

Υλοποίηση Array:

$$Array(A) \triangleq$$

$$Nat \times (Nat \rightarrow Ref(A))$$

Μέγεθος & μία σχέση από δείκτες σε αναφορές λιγότερες από το μέγεθος

$$array_A(N, M) \triangleq$$

$$let\ cell_0 : Ref(A) = ref(M) and ... and\ cell_{N-1} : Ref(A) = ref(M)$$

$$in\ (N, \lambda x : Nat.if\ x = 0\ then\ cell_0\ else\ if ...$$

$$... else\ if\ x = N - 1\ then\ cell_{N-1}\ else\ error_{Ref(A)})$$

Κατασκευαστής Array για N αναφορές με αρχική τιμή M

$$bound(M) \triangleq$$

$$first\ M$$ Μέγεθος Array

$$M[N]_A \triangleq$$

$$if\ N < first\ M$$

$$then\ deref((second\ M)(N))$$

$$else\ error_A$$

Προσπέλαση στοιχείου

Τύπος Array

Υλοποίηση Array . . . . συνέχεια

$$M[N] := P \triangleq$$
$$\textit{if } N < \textit{first } M$$
$$\textit{then } ((\textit{second } M)(N)) := P$$
$$\textit{else } \textit{error}_{\textit{Unit}}$$
Ενημέρωση στοιχείου πίνακα

- Οι κανόνες τύπων για τις λειτουργίες του Array προκύπτουν από την υλοποιήση του τύπου με βάση τους κανόνες για τα products, τα functions και τα references:

$$\frac{\Gamma \vdash A}{\Gamma \vdash \textit{Array}(A)}$$
(Type Array)

$$\frac{\Gamma \vdash N{:}\textit{Nat} \quad \Gamma \vdash M{:}A}{\Gamma \vdash \textit{array}(N,M){:}\textit{Array}(A)}$$
(Val Array)

$$\frac{\Gamma \vdash M{:}\textit{Array}(A)}{\Gamma \vdash \textit{bound } M{:}\textit{Nat}}$$
(Val Array Bound)

$$\frac{\Gamma \vdash N{:}\textit{Nat} \quad \Gamma \vdash M{:}\textit{Array}(A)}{\Gamma \vdash M[N]{:}A}$$
(Val Array Index)

$$\frac{\Gamma \vdash N{:}\textit{Nat} \quad \Gamma \vdash M{:}\textit{Array}(A) \quad \Gamma \vdash P{:}A}{\Gamma \vdash M[N]{:=}P{:}\textit{Unit}}$$
(Val Array Update)

Τέλος ενότητας

Επεξεργασία: Εμμανουέλα Στάχτιαρη
Θεσσαλονίκη, 23/07/2014