Η γλώσσα Fortress

Γιώργος Κορφιάτης

Εργαστήριο Λογισμικού, ΕΜΠ

Ερευνητικά Θέματα Υλοποίησης
Γλωσσών Προγραμματισμού, 2009-10

Η γλώσσα Fortress

- Νέα γλώσσα από τη Sun
- Για υπολογισμούς υψηλής απόδοσης
- Προγραμματισμός σε υψηλό επίπεδο
- Γενικής χρήσης, έμφαση σε επιστημονικές εφαρμογές (εξού όνομα που θυμίζει Fortran...)

`http://projectfortress.sun.com`

Βασικά χαρακτηριστικά

- Ευρεία υποστήριξη παραλληλισμού (implicit)
- Σύνταξη βασισμένη στα μαθηματικά
- Μαθηματικές έννοιες (διανύσματα, πίνακες, σύνολα, …)
- Στατικό typing με type inference
- Αντικείμενα και traits
- Στοιχεία συναρτησιακού προγραμματισμού
- Επεκτασιμότητα

Ένα πρώτο παράδειγμα

```
component Factorial
  export Executable

  factorial(n: ℤ32) =
    if 0 ≤ n ≤ 1 then 1
    elif n > 1 then n(factorial(n − 1))
    else fail "Non-negative integer expected."
    end

  run() = println(factorial 5)
end
```

Σύνταξη

```
sum: RR64 := 0
for k<- 1:n do
  a[k] := (1-alpha)b[k]
  sum += c[k] x^k
end
```

$$
\begin{aligned}
&sum\colon \mathbb{R}64 := 0\\
&\texttt{for } k \leftarrow 1:n \texttt{ do}\\
&\quad a_k := (1-\alpha)b_k\\
&\quad sum \mathrel{+}= c_k x^k\\
&\texttt{end}
\end{aligned}
$$

- ελληνικοί κ.α χαρακτήρες unicode
- δείκτες, εκθέτες
- μεταβλητές σε πλάγια
- συμπαράθεση όρων (juxtaposition) υπερφορτωμένη (πολλαπλασιασμός αριθμών, εφαρμογή συνάρτησης σε όρισμα, string concatenation)
- 1: n γεννήτρια τιμών από 1 ως n

Σύνταξη

```
sum: RR64 := 0
for k<- 1:n do
   a[k] := (1-alpha)b[k]
   sum += c[k] x^k
end
```

$$sum\colon \mathbb{R}64 := 0$$
$$\texttt{for } k \leftarrow 1\colon n \texttt{ do}$$
$$\quad a_k := (1-\alpha)b_k$$
$$\quad sum \mathrel{+}= c_k x^k$$
$$\texttt{end}$$

- ελληνικοί κ.α χαρακτήρες unicode
- δείκτες, εκθέτες
- μεταβλητές σε πλάγια
- συμπαράθεση όρων (juxtaposition) υπερφορτωμένη (πολλαπλασιασμός αριθμών, εφαρμογή συνάρτησης σε όρισμα, string concatenation)
- 1: n γεννήτρια τιμών από 1 ως n

Παραλληλία

- Το πρόγραμμα έχει έμμεσα παραλληλία όπου είναι δυνατόν
- Πρέπει ρητά να σειριοποιηθεί όπου χρειάζεται
- πχ. ορίσματα συναρτήσεων αποτιμώνται παράλληλα
- το ίδιο και για το for loop

```
for i ← 1:5 do
  print(i " ")
  print(i " ")
end
```

```
------ Output ------
5 1 5 1 4 4 2 2 3 3
```

```
for i ← sequential(1:5) do
  print(i " ")
  print(i " ")
end
```

```
------ Output ------
1 1 2 2 3 3 4 4 5 5
```

Παραλληλία

- Το πρόγραμμα έχει έμμεσα παραλληλία όπου είναι δυνατόν
- Πρέπει ρητά να σειριοποιηθεί όπου χρειάζεται
- πχ. ορίσματα συναρτήσεων αποτιμώνται παράλληλα
- το ίδιο και για το for loop

```
for i ← 1:5 do
  print(i " ")
  print(i " ")
end
```

```
------ Output ------
5 1 5 1 4 4 2 2 3 3
```

```
for i ← sequential(1:5) do
  print(i " ")
  print(i " ")
end
```

```
------ Output ------
1 1 2 2 3 3 4 4 5 5
```

Παραλληλία

- Το πρόγραμμα έχει έμμεσα παραλληλία όπου είναι δυνατόν
- Πρέπει ρητά να σειριοποιηθεί όπου χρειάζεται
- πχ. ορίσματα συναρτήσεων αποτιμώνται παράλληλα
- το ίδιο και για το for loop

```
for i ← 1:5 do
  print(i " ")
  print(i " ")
end
```

Output

5 1 5 1 4 4 2 2 3 3

```
for i ← sequential(1:5) do
  print(i " ")
  print(i " ")
end
```

Output

1 1 2 2 3 3 4 4 5 5

Παραλληλία

- Η υλοποίηση αποφασίζει αν θα αποτιμήσει μία έκφραση παράλληλα ή σειριακά
- ...ανάλογα με τους διαθέσιμους πόρους
- Προσέγγιση work stealing, όπως στη Cilk

$$
\begin{array}{l}
fib(n{:}\,\mathbb{Z}32){:}\,\mathbb{Z}32 = \\
\quad \texttt{case}\ n\ \texttt{of} \\
\quad\quad 0 \Rightarrow 0 \\
\quad\quad 1 \Rightarrow 1 \\
\quad\quad \texttt{else} \Rightarrow fib(n-1) + fib(n-2) \\
\quad \texttt{end}
\end{array}
$$

Παραλληλία

- Η υλοποίηση αποφασίζει αν θα αποτιμήσει μία έκφραση παράλληλα ή σειριακά
- ...ανάλογα με τους διαθέσιμους πόρους
- Προσέγγιση work stealing, όπως στη Cilk

$$
\begin{array}{l}
fib(n\colon \mathbb{Z}32)\colon \mathbb{Z}32 = \\
\quad \texttt{case}\ n\ \texttt{of} \\
\quad\quad 0 \Rightarrow 0 \\
\quad\quad 1 \Rightarrow 1 \\
\quad\quad \texttt{else} \Rightarrow fib(n-1) + fib(n-2) \\
\quad \texttt{end}
\end{array}
$$

Παραλληλία

Transactional memory με ατομικές εκφράσεις

$factorial(n\colon \mathbb{Z}32) =$ do
  var $result\colon \mathbb{Z}32 = 1$
  for $i \leftarrow 1 : n$ do
    atomic $result := result\, i$
  end
  $result$
end

Λίστες

$$\texttt{trait}\ \mathrm{List}[\![E]\!]\ \texttt{extends}\ \{\ \mathrm{AnyList}, \mathrm{LexicographicOrder}[\![\mathrm{List}[\![E]\!], E]\!]\ \}$$

$$\texttt{getter}\ \mathit{extractLeft}(): \mathrm{Maybe}\,[\![(E, \mathrm{List}[\![E]\!])]\!]$$

$$\texttt{getter}\ \mathit{extractRight}(): \mathrm{Maybe}\,[\![(\mathrm{List}[\![E]\!], E)]\!]$$

$$\texttt{getter}\ \mathit{reverse}(): \mathrm{List}[\![E]\!]$$

$$\texttt{opr}\ \|\big(\texttt{self}, \mathit{other}\colon \mathrm{List}[\![E]\!]\big): \mathrm{List}[\![E]\!]$$

$$\mathit{addLeft}(\mathsf{e} : E): \mathrm{List}[\![E]\!]$$

$$\mathit{addRight}(\mathsf{e} : E): \mathrm{List}[\![E]\!]$$

$$\mathit{take}(n\colon \mathbb{Z}32): \mathrm{List}[\![E]\!]$$

$$\mathit{split}(): \big(\mathrm{List}[\![E]\!], \mathrm{List}[\![E]\!]\big)$$

$$\mathit{zip}[\![F]\!]\big(\mathit{other}\colon \mathrm{List}[\![F]\!]\big): \mathrm{Generator}\,[\![(E, F)]\!]$$

$$\mathit{filter}(p\colon E \to \mathrm{Boolean}): \mathrm{List}[\![E]\!]$$

$$\texttt{end}$$

Παράλληλη προσπέλαση λίστας

```
import List.{...}

noHello(xs: List[[String]]) : List[[String]] =
  case |xs| of
    0 ⇒ xs
    1 ⇒ if xs_0 = "Hello"
          then ⟨[[String]]⟩ else xs end
    else ⇒
      (ys, zs) = xs.split()
      noHello(ys) || noHello(zs)
  end

strings = ⟨[[String]] "Hello", "there", "how", "are", "you", "?"⟩
run() = println (noHello strings)
```

Παράλληλη προσπέλαση λίστας

```
noHello(xs: List[[String]]) : List[[String]] = do
  r : List[[String]] := ⟨[[String]]⟩
  for x ← xs do
    if x ≠ "Hello" then
      atomic r || = ⟨[[String]] x⟩
    end
  end
  r
end
```

```
————— Output —————
<|are, how, there, you, ?|>
```

Σειριοποίηση της προσπέλασης

$$
\begin{array}{l}
noHello\big(xs\colon \mathrm{List}[\![\mathrm{String}]\!]\big) : \mathrm{List}[\![\mathrm{String}]\!] = \mathsf{do} \\
\quad r : \mathrm{List}[\![\mathrm{String}]\!] := \big\langle [\![\mathrm{String}]\!] \big\rangle \\
\quad \mathsf{for}\ x \leftarrow seq(xs), x \neq \text{“}\texttt{Hello}\text{”}\ \mathsf{do} \\
\qquad r \mid\mid = \big\langle [\![\mathrm{String}]\!]\, x \big\rangle \\
\quad \mathsf{end} \\
\quad r \\
\mathsf{end}
\end{array}
$$

```
------ Output ------
  <|there , how, are , you , ?|>
```

List Comprehension

$$noHello\big(xs\colon \mathrm{List}[\![\mathrm{String}]\!]\big) : \mathrm{List}[\![\mathrm{String}]\!] =$$
$$\big\langle [\![\mathrm{String}]\!]\, x \;\big|\; x \leftarrow xs, x \neq \text{“}\texttt{Hello}\text{”} \big\rangle$$

Κατανομή δεδομένων

- χρήσιμο σε μεγάλα μηχανήματα για καλύτερη απόδοση
- ιεραρχία από regions δεδομένων
- υπολογισμός έχει φθηνή πρόσβαση σε κοντινά regions
- κατανομή παρέχεται αυτόματα από constructors της γλώσσας όπως generators

Contracts

```
component Fibonacci
  export Executable
  fib(n: ℤ32): ℤ32 requires { n ≥ 0 } =
    case n of
      0 ⇒ 0
      1 ⇒ 1
      else ⇒ fib(n − 1) + fib(n − 2)
    end
  run() = println (fib (− 1))
end
```

Η συνθήκη αποτιμάται πριν την κλήση

```
————————Fortress error appears below————————
./Fibonacci.fss:3:31~37: CallerViolation
```

Contracts

```
component Fibonacci
  export Executable
  fib(n: ℤ32): ℤ32 requires { n ≥ 0 } =
    case n of
      0 ⇒ 0
      1 ⇒ 1
      else ⇒ fib(n − 1) + fib(n − 2)
    end
  run() = println (fib (− 1))
end
```

Η συνθήκη αποτιμάται πριν την κλήση

```
————————Fortress error appears below————————
./Fibonacci.fss:3:31~37: CallerViolation
```

Traits

- Κάτι σαν τα interfaces στη Java
- Μπορεί να είναι πολυμορφικό (generic)
- Πολλαπλή κληρονομικότητα
- Μπορεί να περιέχει ορισμούς μεθόδων
- Δεν περιέχει πεδία

```
trait List⟦E⟧ extends { AnyList, LexicographicOrder⟦List⟦E⟧, E⟧ }
                excludes { Number, HasRank, String }
  getter extractLeft(): Maybe⟦(E, List⟦E⟧)⟧
  getter reverse(): List⟦E⟧ = ...
end
```

Traits

- Κάτι σαν τα interfaces στη Java
- Μπορεί να είναι πολυμορφικό (generic)
- Πολλαπλή κληρονομικότητα
- Μπορεί να περιέχει ορισμούς μεθόδων
- Δεν περιέχει πεδία

$$\texttt{trait}\ \mathrm{List}[\![E]\!]\ \texttt{extends}\ \{\ \mathrm{AnyList}, \mathrm{LexicographicOrder}[\![\mathrm{List}[\![E]\!], E]\!]\ \}$$
$$\texttt{excludes}\ \{\ \mathrm{Number}, \mathrm{HasRank}, \mathrm{String}\ \}$$
$$\texttt{getter}\ \mathit{extractLeft}() : \mathrm{Maybe}\,[\![(E, \mathrm{List}[\![E]\!])]\!]$$
$$\texttt{getter}\ \mathit{reverse}() : \mathrm{List}[\![E]\!] = \ldots$$
$$\texttt{end}$$

Αντικείμενα

- Κάτι σαν τις κλάσεις στη Java
- Μπορεί να είναι πολυμορφικό (generic)
- Πολλαπλή κληρονομικότητα
- Περιέχει ορισμούς μεθόδων και πεδία
- Δεν μπορεί να επεκταθεί (extend)

```
object ArrayList⟦E⟧(underlying, firstUsed, canExtendLeft,
                    firstUnused, canExtendRight)
    extends { List⟦E⟧, DelegatedIndexed⟦E, ℤ32⟧ }
  getter size(): ℤ32 = |self|
  getter extractLeft(): Maybe⟦(E, ArrayList⟦E⟧)⟧ = ...
end
```

Αντικείμενα

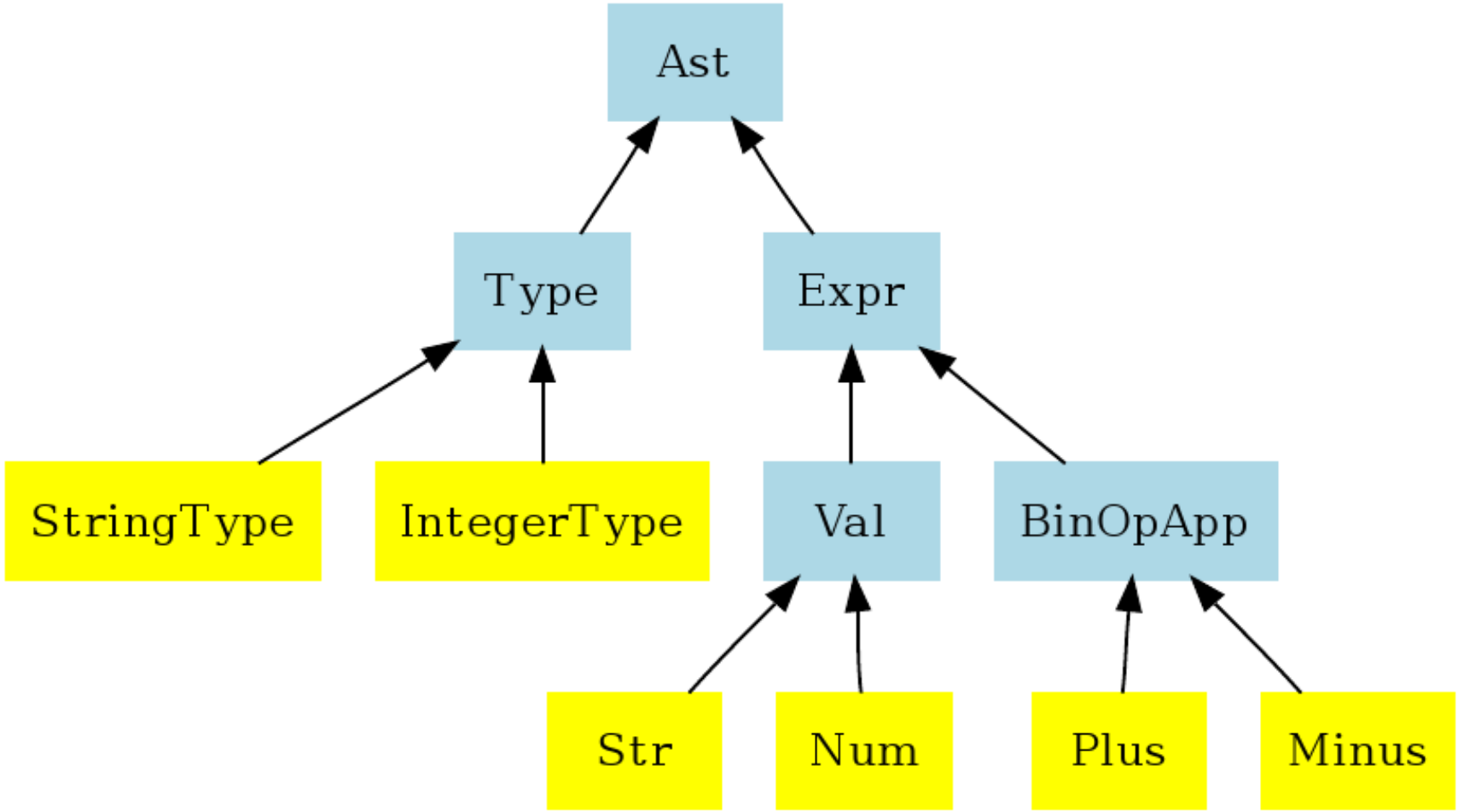
- Κάτι σαν τις κλάσεις στη Java
- Μπορεί να είναι πολυμορφικό (generic)
- Πολλαπλή κληρονομικότητα
- Περιέχει ορισμούς μεθόδων και πεδία
- Δεν μπορεί να επεκταθεί (extend)

$$\begin{aligned}&\texttt{object}\ \mathrm{ArrayList}[\![E]\!](\ \mathit{underlying}, \mathit{firstUsed}, \mathit{canExtendLeft},\\ &\qquad\qquad\qquad\qquad \mathit{firstUnused}, \mathit{canExtendRight})\\ &\qquad\texttt{extends}\ \{\ \mathrm{List}[\![E]\!], \mathrm{DelegatedIndexed}[\![E, \mathbb{Z}32]\!]\ \}\\ &\quad\texttt{getter}\ \mathit{size}(): \mathbb{Z}32 = |\texttt{self}|\\ &\quad\texttt{getter}\ \mathit{extractLeft}(): \mathrm{Maybe}[\![(E, \mathrm{ArrayList}[\![E]\!])]\!] = \ldots\\ &\texttt{end}\end{aligned}$$

Ιεραρχία τύπων

```
trait Ast comprises { Type, Expr }
  getter asString(): String end
trait Type extends Ast
    comprises { StringType, IntegerType }
    excludes Expr end
object StringType extends Type end
object IntegerType extends Type end
trait Expr extends Ast comprises { Val, BinOpApp }
  excludes Type end
trait Val extends Expr comprises { Str, Num }
  getter asValue(): Object end
object Str(string: String) extends Val
  getter asString() = string
  getter asValue() = string end
```

Ιεραρχία τύπων



Growing a language...

- Μικρός πυρήνας της γλώσσας
- Εκτεταμένες βιβλιοθήκες
- Πολλοί τύποι και constructs (for) ορίζονται σε βιβλιοθήκη

```
trait ℤ32 extends { ℤ64, Integral⟦ℤ32⟧ } comprises { Int, IntLiteral }
    getter zero(): ℤ32 = 0
    getter one(): ℤ32 = 1
    getter minimum(): ℤ32 = − 2147483647 − 1
    opr |self|: ℤ32 = if self ≥ 0 then self else − self end
...
```

Growing a language...

- Απαιτεί πολύ εκφραστικό σύστημα τύπων
- Συγγραφείς βιβλιοθηκών έχουν πρόσβαση στο AST
- Ανάλυση προγράμματος, βελτιστοποιήσεις
- ενθαρρύνει συνεργατικότητα και συγγραφή εξειδικευμένων βιβλιοθηκών