Fuzion — Safety through Simplicity

Fuzion Unifies Language Concepts

A features unifies concepts found in other languages:

⇨ class, interface, trait
⇨ method, constructor
⇨ field, argument, local variable
⇨ package, name space
⇨ record / struct
⇨ function
⇨ product or union type

Implementation details are the compiler's concern, not the developer's.

Design by Contract

```
sqrt(a i32) i32
  pre
    debug: a >= 0
  post
    debug: result * result <= a,
    debug: result + 1 > a / (result + 1)
is
 for
   r := 1, next
   next := (r + a/r) / 2
 until r = next
   r
```

Feature Declaration

⇨ a name
⇨ formal args, result type
⇨ an outer feature
⇨ formal generics
⇨ parent feature calls
⇨ a contract
⇨ an implementation
    → inner features

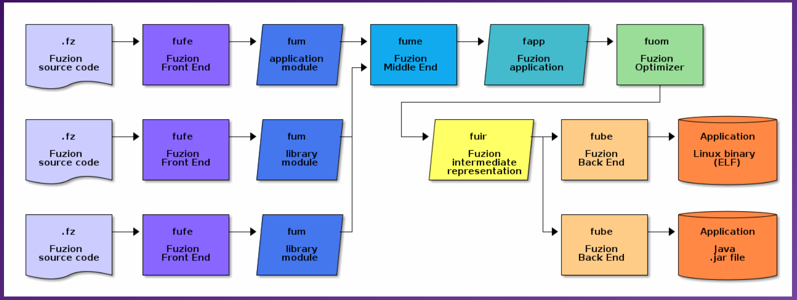
```
myUtils is
  myStack<T>(maxSize i64)
   : lifo<T>, streamable<T>
    pre
      maxSize > 0
    post
      isEmpty
  is
    size u64 => 0
    isEmpty => size = 0
    push(x T) myStack<T>
    is …
```

Simple Intermediate Code

Clazzes	Instructions
⇨ Routine	⇨ Assign
⇨ Field	⇨ Box
⇨ Abstract	⇨ Call
⇨ Choice	⇨ Current
⇨ Intrinsic	⇨ Const
	⇨ Match
	⇨ Tag
	⇨ Pop

Enbables powerful tools for analysis and optimization!

Toolchain



Compiler Decides

Where to store data
⇨ heap / stack / register
What runtime data to add
⇨ class / type descriptors
⇨ type tags
Trade-offs
⇨ Specialization vs. dynamic binding

Ellie

Tools Verify

Simple Intermediate Code allows
⇨ static analysis to verify correctness
⇨ absence of race conditions
⇨ contracts

flang.dev
github.com/fridis/fuzion

Dr. Fridtjof Siebert
Tokiwa Software GmbH
siebert@tokiwa.software