Declare Your Language

Language Definition with the Spoofax Language Workbench

Eelco Visser

DSLDI Summer School, Lausanne
July 13, 2015

Delft University of Technology

MetaBorg – MetaBorg

metaborg.org

Search

METABORG NEWS DOWNLOAD SPOOFAX SDF3 NABL TS SPT STRATEGO DEV

SM.sdf3

```
System.Machine = [
  state machine [ID] [Extends]
    [{Element "\n"}*]
]

Extends.Extends =
  [extends [ID]]

Extends.NoExtends = []

Element.State =
  [state [ID]]

Element.Transition = [
  transition from [StateRef] to
```

names.nab

```
Machine(m, elems, extends) :
  defines Machine m
  scopes State, Variable

Extends(m) :
  imports State, Variable from
State(s) :
  defines State s

StateRef(s) :
  refers to State s

VarDef(x, c) :
  defines Variable x of type t
```

types.ts

```
False() : BoolType()
True()  : BoolType()

Var(x) : t
where definition of x : t

Or(e1, e2) + And(e1, e2) :
where e1 : BoolType()
      else error "bool expe
  and e2 : BoolType()
      else error "bool expe

Eq(e1, e2) + Gt(e1, e2) : E
where e1 : IntType()
      else error "int expec
```

generate.str

```
sm-to-java :
  machine@Machine(m, exter
    public class [m] [<ext
      String current = [<s
      [vardefs]

      String next(String e
        [cond-stat*]
        while(true) {
          [uncond-stat*]
        }
      }
    }
  ]
  where
```

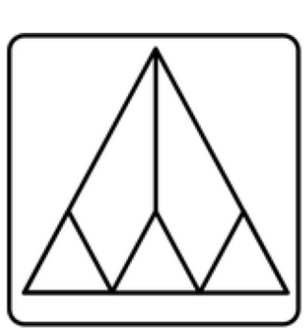
VendingMachine. | syntax-test.spt

```
state Vend_Drink
state Vend_Sweet
state Empty

transition from Waiting to Vend_Drink: V
  [ drinks > 0 ] / drinks := drinks - 1
transition from Vend_Drink to Waiting: V
  [ drinks > 0 or sweets > 0 ]
```

VendingMachine.aterm

```
Machine(
  "VendingMachine"
, NoExtends()
, [ VarDef("drinks", Int("10"))
  , VarDef("sweets", Int("20"))
  , State("Waiting")
```



Edit on GitHub

MetaBorg

MetaBorg is a collection of meta-programming tools including the following components:

- The Spoofax language workbench
- The SDF3 syntax definition formalism
- The NaBL name binding language
- The TS type specification language
- The Stratego transformation language

Software repositories

- https://github.com/metaborg
- https://svn.strategoxt.org/repoman/info/StrategoXT

MetaBorg provides generic technology for allowing a host language (collective) to incorporate and assimilate external domains (cultures) in order to strengthen itself. The ease of implementing embeddings makes resistance futile.

Part I: Hands-On Spoofax Tutorial

- Syntax Definition with SDF3
- Transformation with Stratego
- Name and type analysis with NaBL/TS

Part II: Ongoing Research

- A Theory of Name Resolution
- DynSem: A DSL for Dynamic Semantics Specification

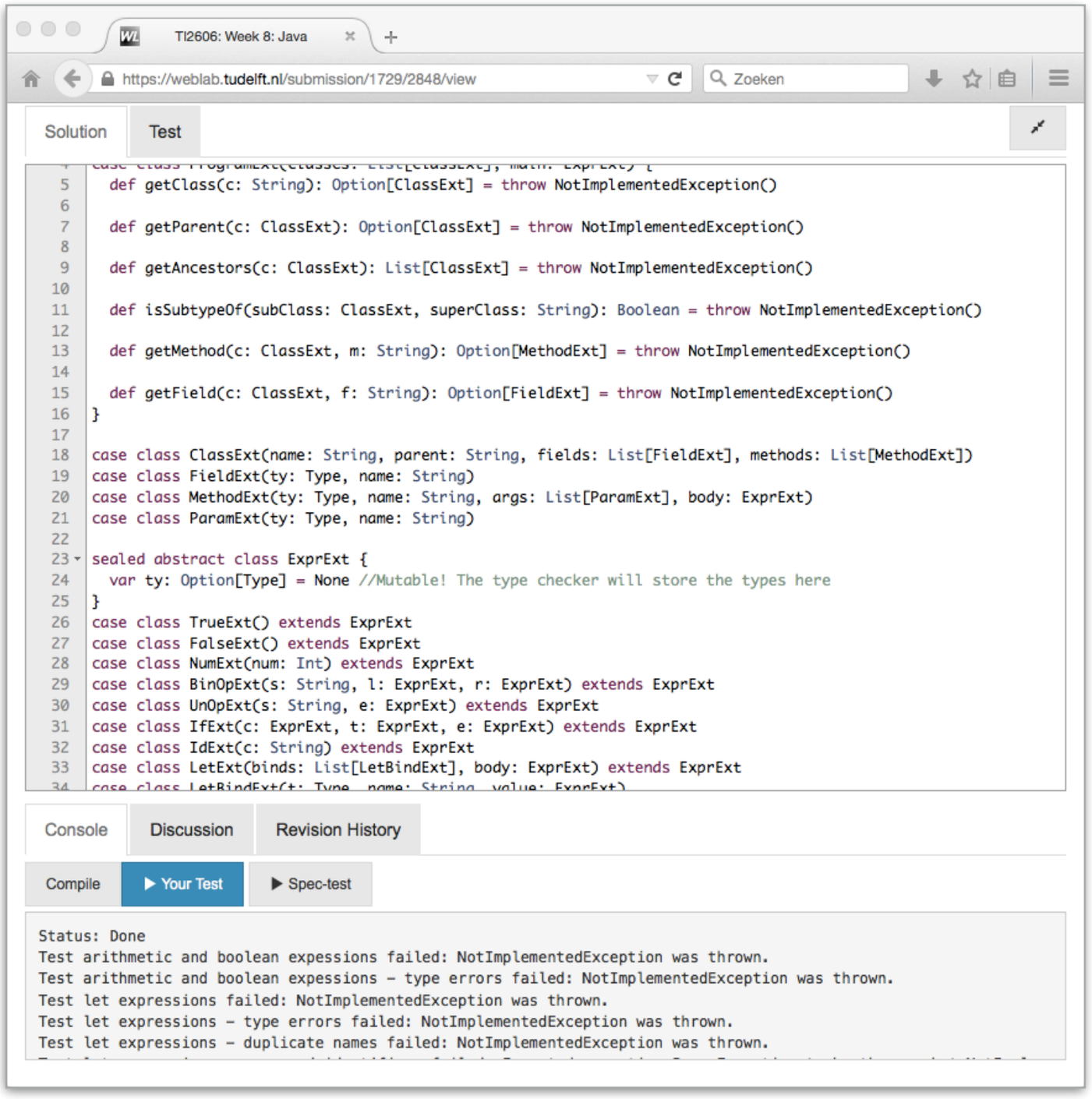
PAPLJ *

```
(program
   (classes
      (class Counter extends Object
         (fields
            (Num count))
         (methods
            (method Num count ()
               (set this count (+ (get this count) 1))))))

   (let ((Counter c1 (new Counter))
         (Counter c2 (new Counter))
         (Counter c3 (new Counter)))
      (do
        (call c1 count)
        (call c2 count)
        (call c1 count)
        (call c2 count)
        (call c3 count)
        (call c1 count)
        (* (get c1 count) (get c2 count))
      )
   )
)
```

*) Defined as an assignment for Shriram Krishnamurthi's Programming and Programming Languages (PAPL) book

PAPLJ in Scala in WebLab *



*) Final assignment for course on Concepts of Programming Languages using PAPL book

PAPLJ in Spoofax

```
program counter

  class Counter {
    Num count
    Num next() {
      this.count = this.count + 1
    }
    Counter reset(Num i) {
      this.count = i; this
    }
  }

run

  let Counter c1 = new Counter()
      Counter c2 = new Counter()
      Counter c3 = new Counter()
   in {
     c1.next();
     c2.next();
     c2.next();
     c2.next();
     c1.reset(c2.count);
     c1.count
   }
```

Syntax definition
- parser (error recovery)
- pretty-printer
- syntax highlighting
- abstract syntax
- editor with above

Transformation
- desugaring

Name binding
- name checking

Type rules
- type checking

Dynamic semantics
- interpreter