# Software Synthesis using Automated Reasoning

Güle güle, Turunç

# Complete Functional Synthesis

Joint work with V. Kuncak, M. Mayer and P. Suter

# Software Synthesis

```
val bigSet = ....

val (setA, setB) = choose((a: Set, b: Set) ) =>
  ( a.size == b.size  && a union b == bigSet && a intersect b == empty))
```

**Code**

```
val n = bigSet.size/2
val setA = take(n, bigSet)
val setB = bigSet -- setA
```

# Software Synthesis

```
val bigSet = ....

val (setA, setB) = choose((a: Set, b: Set) ) =>
   ( a.size == b.size  && a union b == bigSet && a intersect b == empty))
```

**Code**

```
assert (bigSet.size % 2  == 0)
val n = bigSet.size/2
val setA = take(n, bigSet)
val setB = bigSet –– setA
```

# Software Synthesis

- Software synthesis = a technique for automatically generating code given a specification
- Why?
  - ease software development
  - increase programmer productivity
  - fewer bugs
- Challenges
  - synthesis is often a computationally hard task
  - new algorithms are needed

# “choose” Construct

- specification is part of the Scala language
- two types of arguments: input and output

- a call of the form

$$\boldsymbol{val}\ x_1 = \boldsymbol{choose}(x \Rightarrow F(\ x,\ a\ ))$$

- corresponds to constructively solving the **quantifier elimination** problem

$$\exists x. F(x, a)$$

where $a$ is a parameter

# Complete Functional Synthesis

**Definition (Synthesis Procedure)**

A synthesis procedure takes as input formula F(x, a) and outputs:

1. a precondition formula *pre*(a)
2. list of terms Ψ

such that the following holds:

$$\exists x. F(x, a) \Leftrightarrow pre(a) \Leftrightarrow F[x := \Psi]$$

- Note: *pre(a)* is the “best” possible

# Synthesis for Linear Integer Arithmetic – Example / Overview

```
choose((h: Int, m: Int, s: Int) ⇒ (
        h * 3600 + m * 60 + s == totalSeconds
    && h ≥ 0
    && m ≥ 0 && m < 60
    && s ≥ 0 && s < 60   ))
```

Returned code:

```
assert (totalSeconds ≥ 0)
val h = totalSeconds div 3600
val temp = totalSeconds + (-3600) * h
val m = min(temp div 60, 59)
val s = totalSeconds + (-3600) * h + (-60) * m
```

# Synthesis Procedure - Overview

- process every equality: take an equality $E_i$, compute a parametric description of the solution set and insert those values in the rest of formula
  - for $n$ output variables, we need $n$-1 fresh new variables
  - number of output variables decreased for 1
  - compute preconditions
- at the end there are only inequalities – similar procedure as in [Pugh 1992]

# Synthesis Procedure by Example

- process every equality: take an equality $E_i$, compute a parametric description of the solution set and insert those values in the rest of formula

$$\begin{pmatrix} h \\ m \\ s \end{pmatrix} = \lambda \begin{pmatrix} 1 \\ 0 \\ -3600 \end{pmatrix} + \mu \begin{pmatrix} 0 \\ 1 \\ -60 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ totalSeconds \end{pmatrix} \mid \lambda, \mu \in Z$$

Code:
**<further code will come here>**
**val** h = lambda
**val** m = mu
**val val** s = totalSeconds + (-3600) * lambda + (-60) * mu

# Synthesis Procedure by Example

- process every equality: take an equality $E_i$, compute a parametric description of the solution set and insert those values in the rest of formula

$$\begin{pmatrix} h \\ m \\ s \end{pmatrix} = \lambda \begin{pmatrix} 1 \\ 0 \\ -3600 \end{pmatrix} + \mu \begin{pmatrix} 0 \\ 1 \\ -60 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ totalSeconds \end{pmatrix} \mid \lambda, \mu \in Z$$

Resulting formula (new specifications):

$0 \le \lambda$, $0 \le \mu$, $\mu \le 59$, $0 \le totalSeconds - 3600\lambda - 60\mu$, $totalSeconds - 3600\lambda - 60\mu \le 59$

# Processing Inequalities

process output variables one by one

$0 \leq \lambda$, $0 \leq \mu$, $\mu \leq 59$, $0 \leq \text{totalSeconds} - 3600\lambda - 60\mu$,
$\text{totalSeconds} - 3600\lambda - 60\mu \leq 59$

expressing constraints as bounds on μ

$0 \leq \lambda$, $0 \leq \mu$, $\mu \leq 59$, $\mu \leq \lfloor (\text{totalSeconds} - 3600\lambda)/60 \rfloor$ ,
$\lceil (\text{totalSeconds} - 3600\lambda - 59)/60 \rceil \leq \mu$

Code:

**val** mu = **min**(59, (totalSeconds -3600* lambda) **div** 60)

# Fourier-Motzkin-Style Elimination

$0 \le \lambda,\ 0 \le \mu,\ \mu \le 59,\ \mu \le \lfloor(\text{totalSeconds} - 3600\lambda)/60\rfloor\ ,$
$\lceil(\text{totalSeconds} - 3600\lambda - 59)/60\rceil \le \mu$

combine each lower and upper bound

$0 \le \lambda,\ 0 \le 59,\ 0 \le \lfloor(\text{totalSeconds} - 3600\lambda)/60\rfloor\ ,$
$\lceil(\text{totalSeconds} - 3600\lambda - 59)/60\rceil \le \lfloor(\text{totalSeconds} - 3600\lambda)/60\rfloor\ ,$
$\lceil(\text{totalSeconds} - 3600\lambda - 59)/60\rceil \le 59$

basic simplifications

$0 \le \lambda,\ 60\lambda \le \lfloor\text{totalSeconds}\ /60\rfloor,$
$\lceil(\text{totalSeconds} - 59)/60\rceil - 59 \le 60\lambda$

Code:

**val** lambda = totalSeconds **div** 3600

Preconditions: $0 \le$ totalSeconds

# Generated Code May Contain Loops

```
val (x1, y1) = choose(x: Int, y: Int =>
                    2*y − b =< 3*x + a &&  2*x − a =< 4*y + b)
```

```
val kFound = false
 for k = 0 to 5 do {
   val v1 = 3 * a + 3 * b – k
   if (v1 mod 6 == 0) {
     val alpha = ((k – 5 * a – 5 * b)/8).ceiling
     val l = (v1 / 6) + 2 * alpha
     val y = alpha
     val kFound = true
     break } }
 if (kFound)
   val x = ((4 * y + a + b)/2).floor
 else throw new Exception(”No solution exists”)
```

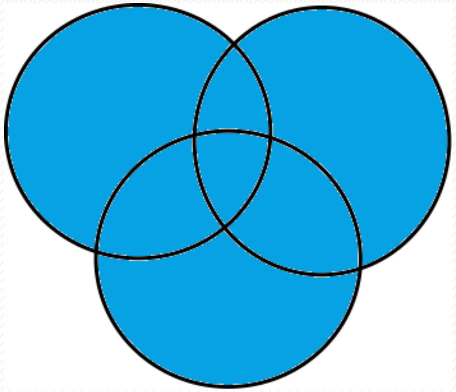
**Precondition**: $\exists k.\ 0 \le k \le 5 \wedge 6 | 3a + 3b - k$     **(true)**

# From Data Structures to Numbers

- Observation:
  - Reasoning about collections reduces to reasoning about linear integer arithmetic!

a.size == b.size && a union b == bigSet && a intersect b == empty

a



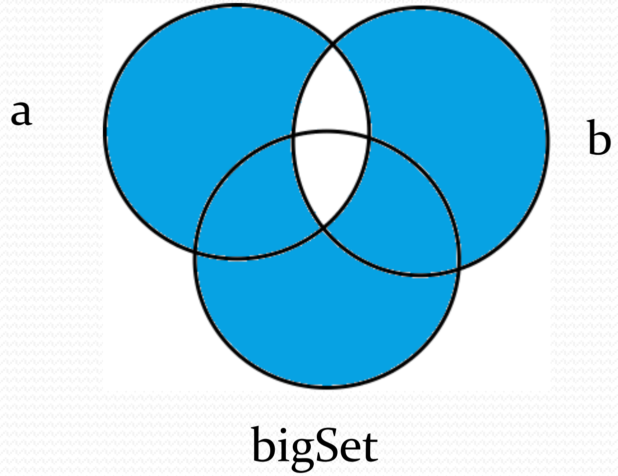
b

bigSet

# From Data Structures to Numbers

- Observation:
  - Reasoning about collections reduces to reasoning about linear integer arithmetic!

a.size == b.size && a union b == bigSet && a intersect b == empty

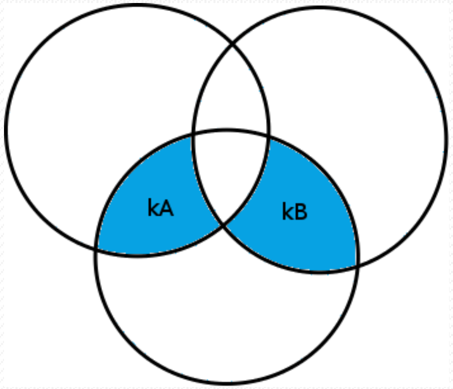


# From Data Structures to Numbers

- Observation:
  - Reasoning about collections reduces to reasoning about linear integer arithmetic!

a.size == b.size && a union b == bigSet && a intersect b == empty

a




b

bigSet

New specification:

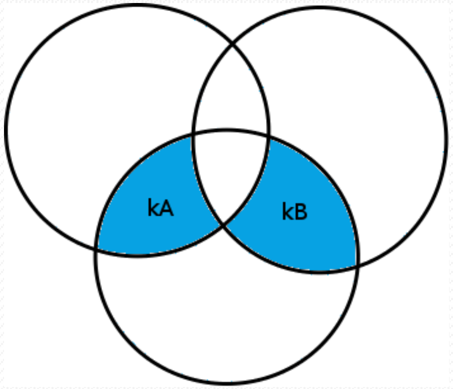
kA = kB

# From Data Structures to Numbers

- Observation:
  - Reasoning about collections reduces to reasoning about linear integer arithmetic!

a.size == b.size && a union b == bigSet && a intersect b == empty

a




b

bigSet

New specification:

kA = kB && kA +kB = |bigSet|

because of quantifier elimination

# Interactive Synthesis of Code Snippets

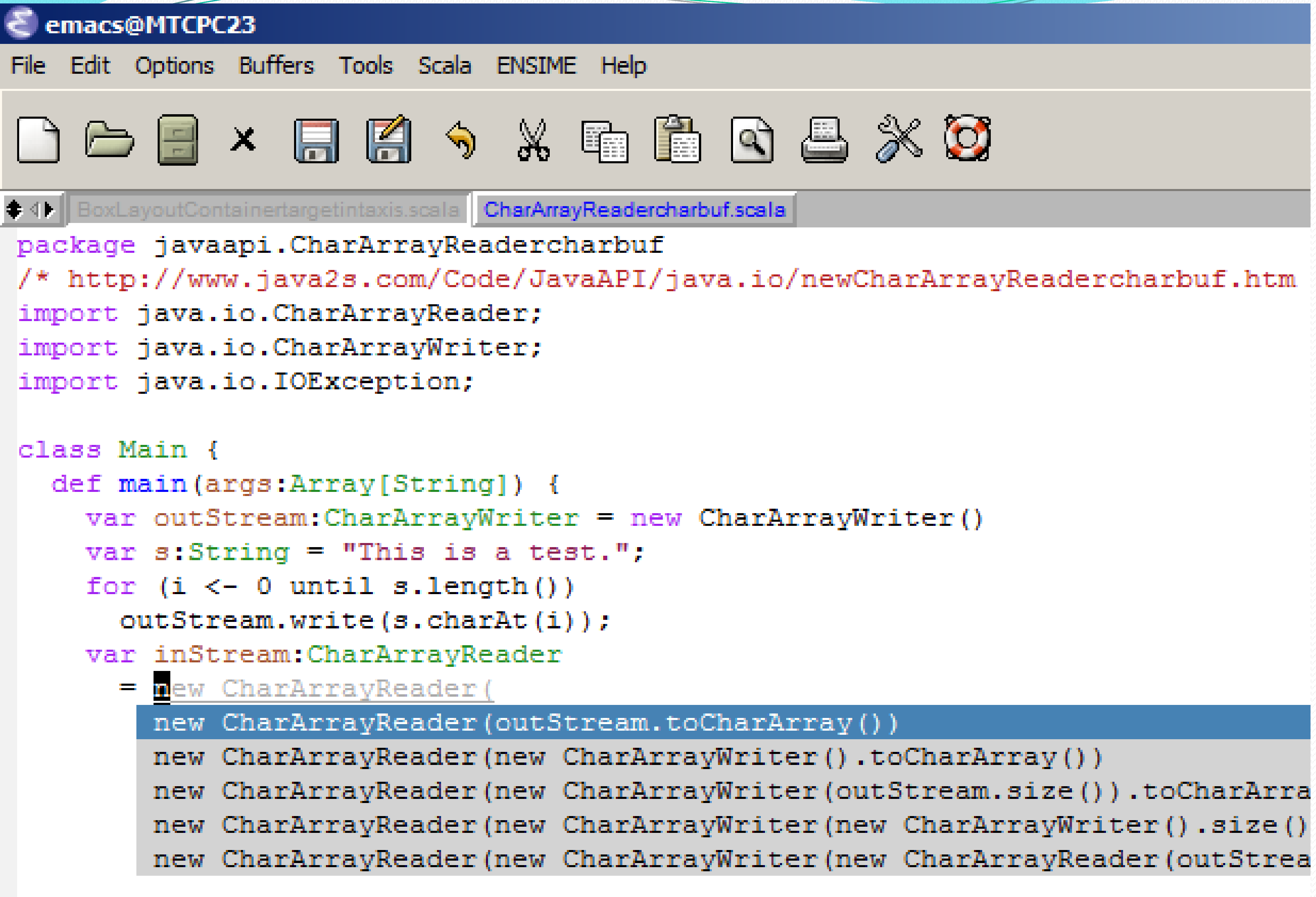
Joint work with Tihomir Gvero and Viktor Kuncak

emacs@MTCPC23

File Edit Options Buffers Tools Scala ENSIME Help

BoxLayoutContainertargetintaxis.scala CharArrayReadercharbuf.scala

```
package javaapi.CharArrayReadercharbuf
/* http://www.java2s.com/Code/JavaAPI/java.io/newCharArrayReadercharbuf.htm
import java.io.CharArrayReader;
import java.io.CharArrayWriter;
import java.io.IOException;

class Main {
  def main(args:Array[String]) {
    var outStream:CharArrayWriter = new CharArrayWriter()
    var s:String = "This is a test.";
    for (i <- 0 until s.length())
      outStream.write(s.charAt(i));
    var inStream:CharArrayReader
      = 
```


emacs@MTCPC23
File Edit Options Buffers Tools Scala ENSIME Help
BoxLayoutContainertargetintaxis.scala
CharArrayReadercharbuf.scala
package javaapi.CharArrayReadercharbuf
/* http://www.java2s.com/Code/JavaAPI/java.io/newCharArrayReadercharbuf.htm
import java.io.CharArrayReader;
import java.io.CharArrayWriter;
import java.io.IOException;
class Main {
def main(args:Array[String]) {
var outStream:CharArrayWriter = new CharArrayWriter()
var s:String = "This is a test.";
for (i <- 0 until s.length())
outStream.write(s.charAt(i));
var inStream:CharArrayReader
= new CharArrayReader(
new CharArrayReader(outStream.toCharArray())
new CharArrayReader(new CharArrayWriter().toCharArray())
new CharArrayReader(new CharArrayWriter(outStream.size()).toCharArra
new CharArrayReader(new CharArrayWriter(new CharArrayWriter().size()
new CharArrayReader(new CharArrayWriter(new CharArrayReader(outStrea

# InSynth - Interactive Synthesis of Code Snippets

- Software Synthesis = automatically deriving code from specifications
- InSynth – a tool for synthesis of code fragments (snippets)
  - interactive
    - getting results in a short amount of time
    - multiple solutions – a user needs to select
  - component based
    - assemble program from given components (local values, API)
  - partial specification
    - hard constraints – type constraints
    - soft constraints - use of components “most likely” to be useful

emacs@MTCPC23

File Edit Options Buffers Tools Scala ENSIME Help

Demo.scala DemoLib.scala SuperDemo.scala

```scala
package demo
import scala.Array

class Example {
  def createStringArray(name:String):Array[String] =
    Array[String]("Tihomir", "Gvero",
                  "Viktor", "Kuncak",
                  "Ruzica", "Piskac")

  def map[A,B](fun:A=>B, array:Array[A]):Array[B] =
    throw new Exception("implementation omitted")

  def createIntArray(fun:String=>Int, name:String):Array[Int]
    =
}
```

emacs@MTCPC23

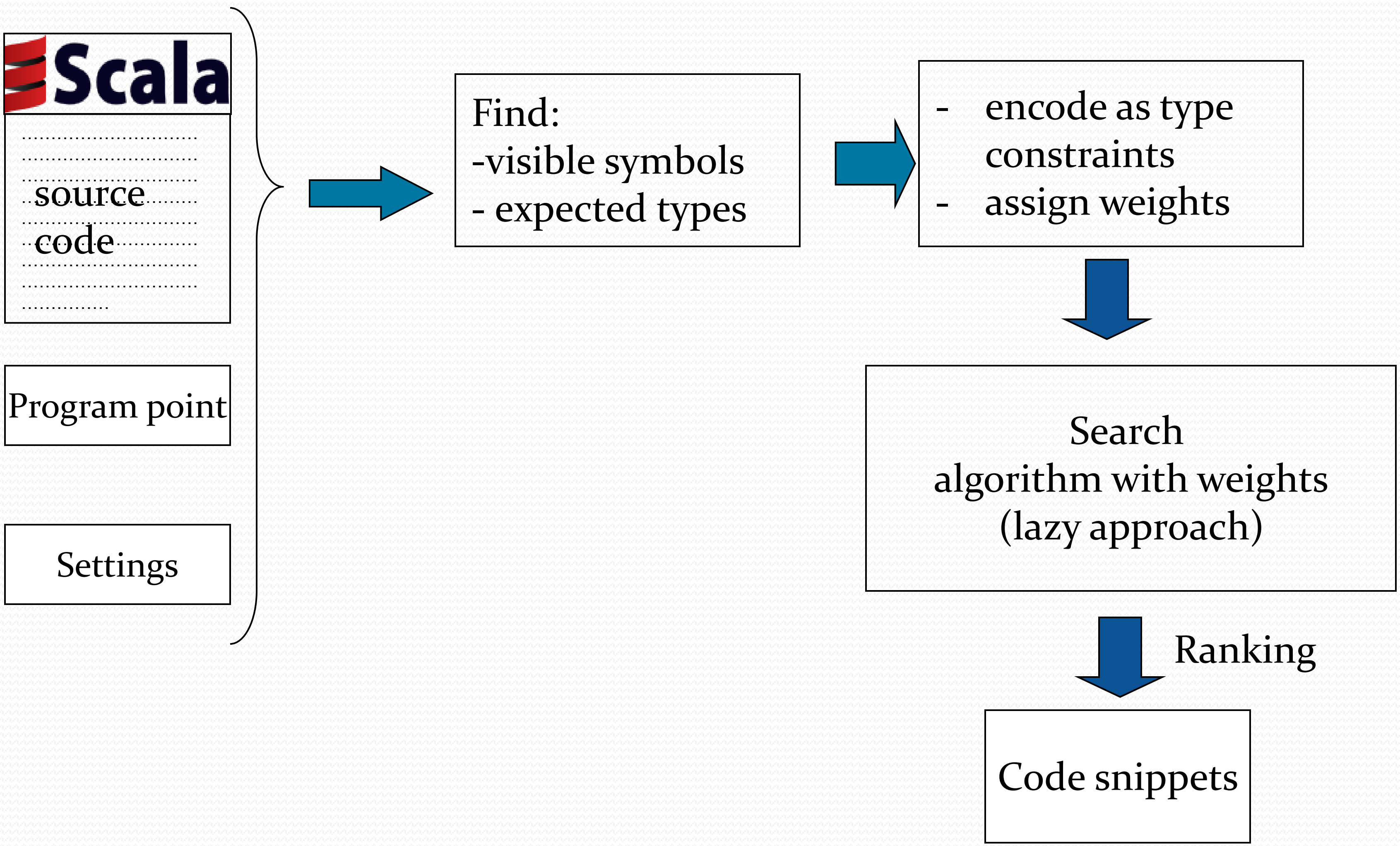
File Edit Options Buffers Tools Scala ENSIME Help

Demo.scala | DemoLib.scala | SuperDemo.scala

```scala
package demo
import scala.Array

class Example {
  def createStringArray(name:String):Array[String] =
    Array[String]("Tihomir", "Gvero",
                  "Viktor", "Kuncak",
                  "Ruzica", "Piskac")

  def map[A,B](fun:A=>B, array:Array[A]):Array[B] =
    throw new Exception("implementation omitted")

  def createIntArray(fun:String=>Int, name:String):Array[Int]
    =
}
```

```
map[String,Int](fun,createStringArray(name))        s
Array[Int]()                                         s
map[String,Int](fun,Array[String]())                 s
Array.concat[Int]()                                  s
map[String,Int](fun,Array.concat[String]())          s
```

# Snippet Synthesis inside IDE



# Encoding Type Constraints

$x : T \leftrightarrow x : \{\} \to T$

| TYPE DECLARATIONS | Translation |
|---|---|
| **val** n: Int | n : Int |
| **val** l: List[Int ] | $l : \{\} \to \mathrm{List(Int)}$ |
| **def** iTs(a: Int, b:Int): String | $\mathrm{iTs} : \{\mathrm{Int}\} \to \mathrm{String}$ |
| **def** el[A](a: A): List[A] | $\mathrm{el}: \forall\alpha.\ \{\alpha\} \to \mathrm{List}(\alpha)$ |

# Calculus for the Ground Types

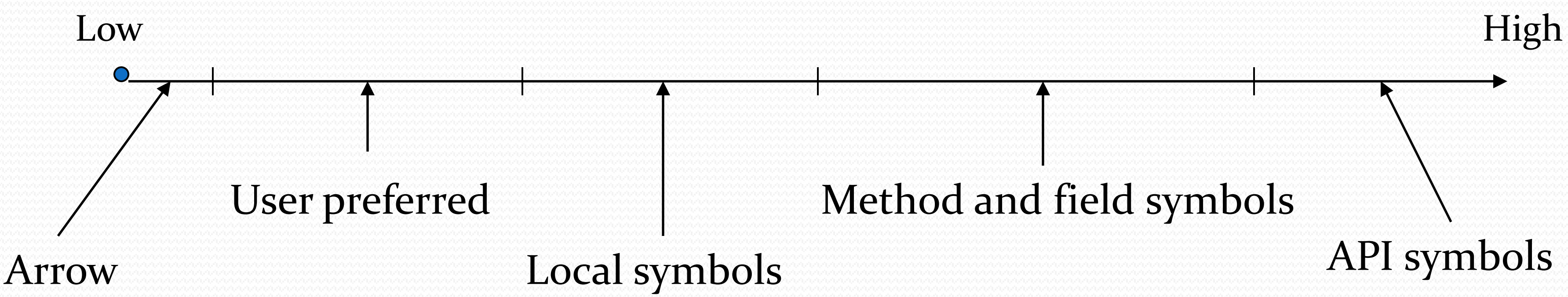
$$\frac{x{:}\ \tau \in \Gamma}{\Gamma \vdash x{:}\ \tau}\ \text{AX}$$

$$\frac{\Gamma \vdash f{:}\ \{\, S_1 \to T_1, \ldots, S_n \to T_n \} \to T \qquad \Gamma \cup \Gamma_{S_1} \vdash h_1{:}\ T_1 \qquad \ldots \qquad \Gamma \cup \Gamma_{S_n} \vdash h_n{:}\ T_n}{\Gamma \vdash f(\Lambda(\Gamma_{S_1}).h_1, \ldots, \Lambda(\Gamma_{S_n}).h_n){:}T}\ \text{APP}$$

- where:
  - $\Gamma_S = \{x_i{:}\tau_i \mid \tau_i \in S \text{ and } x_i \text{ fresh}\}$
  - $\Lambda(\emptyset) = \varepsilon$
  - $\Lambda(\{x{:}\tau\} \cup \Gamma_S) = \lambda x{:}\ \tau.\ \Lambda(\Gamma_S)$

# System of Weights

- Symbol weights – used for ranking solution and for directing the search
- Weight of a term is computed based on
  - precomputed term weights (based on analysis of over 100 examples taken from the Web) - frequency
  - proximity to the program point where the tool is invoked



# Subtyping using Coercions

- We model A <: B by introducing a coercion function
  c: A → B [Tannen etAL, 1991]

**class** ArrayList[T] **extends** AbstractList[T] **with** List[T]
  **with** RandomAccess **with** Cloneable **with** Serializable {...}
**abstract class** AbstractList[E] **extends** AbstractCollection[E]
  **with** List[E] {
  ....
  **def** iterator():Iterator[E] = {...}
}

# Subtyping using Coercions

- We model A <: B by introducing a coercion function
  c: A → B [Tannen etAL, 1991]

```
class ArrayList[T] extends AbstractList[T] with List[T]
   with RandomAccess  with Cloneable with Serializable {…}
abstract class AbstractList[E] extends AbstractCollection[E]
   with List[E] {
   ….
   def iterator():Iterator[E] = {…}
}
```

c1: $\forall\alpha$. ArrayList[$\alpha$]$\rightarrow$ AbstractList[$\alpha$]
c2: $\forall\beta$. AbstractList[$\beta$]$\rightarrow$ AbstractCollection[$\beta$]

# Subtyping Example

**val** a1: ArrayList[String] = …

…

**class** ArrayList[T] **extends** AbstractList[T] **with** List[T]
    **with** RandomAccess **with** Cloneable **with** Serializable {…}

**abstract class** AbstractList[E] **extends** AbstractCollection[E]
    **with** List[E] {

    ….

    **def** iterator():Iterator[E] = {…}

}

…

val i1: Iterator[String] = ■

# Subtyping Example

**val** a1: ArrayList[String] = …

…

**class** ArrayList[T] **extends** AbstractList[T] **with** List[T]
  **with** RandomAccess  **with** Cloneable **with** Serializable {…}

**abstract class** AbstractList[E] **extends** AbstractCollection[E]
  **with** List[E] {
  ….
  **def** iterator():Iterator[E] = {…}
}

…

val i1: Iterator[String] = ■

a1: ArrayList(String)
c1: $\forall\alpha$. ArrayList[$\alpha$]$\rightarrow$ AbstractList[$\alpha$]
c2: $\forall\beta$. AbstractList[$\beta$]$\rightarrow$ AbstractCollection[$\beta$]
iterator: $\forall\gamma$. AbstractList[$\gamma$] $\rightarrow$ Iterator[$\gamma$]
goal : Iterator[String] $\rightarrow$ ?

# Subtyping Example

**val** a1: ArrayList[String] = ...

...

**class** ArrayList[T] **extends** AbstractList[T] **with** List[T]

  **with** RandomAccess **with** Cloneable **with** Serializable {...}

**abstract class** AbstractList[E] **extends** AbstractCollection[E]

  **with** List[E] {

  ....

  **def** iterator():Iterator[E] = {...}

}

...

val i1: Iterator[String] = ■

goal(iterator(c1(a1)) ) : ?

a1: ArrayList(String)

c1: $\forall\alpha$. ArrayList[$\alpha$]$\rightarrow$ AbstractList[$\alpha$]

c2: $\forall\beta$. AbstractList[$\beta$]$\rightarrow$ AbstractCollection[$\beta$]

iterator: $\forall\ \gamma$. AbstractList[$\gamma$] $\rightarrow$ Iterator[$\gamma$]

goal : Iterator[String] $\rightarrow$ ?

# Subtyping Example

```
val a1: ArrayList[String] = …
…
class ArrayList[T] extends AbstractList[T] with List[T]
   with RandomAccess  with Cloneable with Serializable {…}
abstract class AbstractList[E] extends AbstractCollection[E]
   with List[E] {
   ….
   def iterator():Iterator[E] = {…}
}
…
val i1: Iterator[String] = a1.iterator()
```

goal(iterator(c1(a1)) ) : ?

# Evaluation

| Benchmark | Lenth | #Initial | #Derived | #Snip.Gen. | Rank | Time [ms] |
|---|---|---|---|---|---|---|
| BufferedReaderInputStreamReader | 3 | 370 | 5501 | 421 | 2 | 562 |
| DatagramSocketintport | 2 | 364 | 7712 | 243 | 5 | 702 |
| DataInputStreamFileInputStreamfileInputStream | 3 | 370 | 6020 | 385 | 3 | 562 |
| FileReaderFilefile | 3 | 371 | 4930 | 309 | 3 | 562 |
| GroupLayoutContainerhost | 2 | 1363 | 4556 | 166 | 4 | 608 |
| ObjectInputStreamInputStreamin | 3 | 373 | 5726 | 345 | 3 | 577 |
| PipedReaderPipedWritersrc | 2 | 370 | 9738 | 311 | 3 | 546 |
| ServerSocketintport | 2 | 723 | 8551 | 271 | 1 | 577 |
| StreamTokenizerReaderr | 4 | 370 | 5732 | 448 | 3 | 562 |
| URLStringspecthrowsMalformedURLException | 3 | 723 | 8691 | 276 | 1 | 624 |
| BufferedReaderReaderin | 4 | 49 | 1662 | 362 | 1 | 546 |
| ByteArrayInputStreambytebufintoffsetintlength | 4 | 22 | 4049 | 102 | 3 | 546 |
| CharArrayReadercharbuf | 3 | 26 | 782 | 343 | 1 | 546 |
| TimerintvalueActionListeneract | 3 | 28 | 921 | 1 | 1 | 531 |
| TransferHandlerStringproperty | 2 | 28 | 245 | 1154 | 1 | 640 |
| ArrayListtoArray | 2 | 24 | 647 | 400 | 1 | 655 |
| HashMapcontainsValueObjectvalue | 3 | 24 | 857 | 557 | 5 | 562 |
| HashMapentrySet | 2 | 24 | 3990 | 440 | 1 | 577 |
| HashMapvalues | 2 | 24 | 845 | 542 | 1 | 546 |
| HashSetiterator | 2 | 60 | 1832 | 201 | 1 | 546 |
| Hashtableelements | 2 | 32 | 869 | 445 | 1 | 546 |
| HashtableentrySet | 2 | 31 | 874 | 441 | 1 | 546 |
| HashtablekeySet | 2 | 32 | 968 | 492 | 3 | 546 |
| Hashtablekeys | 2 | 30 | 818 | 477 | 2 | 515 |
| PriorityQueuepoll | 2 | 27 | 1208 | 363 | 1 | 562 |
| TreeMapentrySet | 2 | 40 | 4267 | 29 | 1 | 562 |
| TreeMapvalues | 2 | 40 | 559 | 190 | 1 | 562 |
| Vectorelements | 2 | 35 | 1496 | 386 | 1 | 531 |
| VectortoArray | 2 | 35 | 1387 | 317 | 1 | 546 |

# Sample Results

| Benchmark | Lenth | #Initial | #Derived | #Snip.Gen. | Rank | Time [ms] |
|---|---|---|---|---|---|---|
| ByteArrayInputStreambytebufintoffsetintlength | 4 | 22 | 4049 | 102 | 3 | 546 |
| CharArrayReadercharbuf | 3 | 26 | 782 | 343 | 1 | 546 |
| HashSetiterator | 2 | 60 | 1832 | 201 | 1 | 546 |
| Hashtableelements | 2 | 32 | 869 | 445 | 1 | 546 |
| HashtableentrySet | 2 | 31 | 874 | 441 | 1 | 546 |
| HashtablekeySet | 2 | 32 | 968 | 492 | 3 | 546 |
| Hashtablekeys | 2 | 30 | 818 | 477 | 2 | 515 |
| PriorityQueuepoll | 2 | 27 | 1208 | 363 | 1 | 562 |

# Conclusions

Software Synthesis

- method to obtain *correct* software from the given specification
- Complete Functional Synthesis: extending decision procedures into synthesis algorithms
- Interactive Synthesis of Code Snippets: finding a term of a given type