Software Synthesis using Automated Reasoning

Ruzica Piskac

Max Planck Institute for Software Systems, Germany

ComFuSy - Complete Functional Synthesis

Joint work with Viktor Kuncak, Mikael Mayer and Philippe 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

$$\textbf{\textit{val}}\ x1 = \textbf{\textit{choose}}(x \Rightarrow F(x, a))$$

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

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

where a is a parameter

Complete Functional Synthesis

complete = the synthesis procedure is guaranteed to find code that satisfies the given specification

functional = computes a function that satisfies a given input / output relation

Important features:

- code produced this way is correct by construction – no need for further verification
- a user does not provide hints on the structure of the generated code

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

From Decision Procedure to Synthesis Procedure

- based on quantifier elimination / model generating **decision procedures**
- fragment $\forall x.\exists y.F(x,y)$ in general undecidable
- decidable for logic of linear integer (rational, real) arithmetic, for Boolan Algebra with Presburger Arithmetic (BAPA)

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):

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

Processing Inequalities

process output variables one by one

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

expressing constraints as bounds on μ

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

Code:

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

Fourier-Motzkin-Style Elimination

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

combine each lower and upper bound

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

basic simplifications

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

Code:

val lambda = totalSeconds **div** 3600

Preconditions: $0 \le totalSeconds$

Handling of Equalities in Comfusy

Parametric Solution of Equation

Theorem

For an equation $\sum_{i=1}^{n} \gamma_i x_i + C = 0$ with S we denote the set of solutions.

- Let S_H be a set of solutions of the homogeneous equality:

$$S_H = \{ \mathbf{y} \mid \sum_{i=1}^{n} \gamma_i y_i = 0 \}$$

S_H is an "almost linear" set, i.e. can be represented as a linear combination of vectors:

$$S_H = \lambda_1 \mathbf{s}_1 + \ldots \lambda_{n-1} \mathbf{s}_{n-1}$$

- Let **w** be any solution of the original equation
- $\rightarrow$ $\mathbf{S} = w + \lambda_1 \mathbf{s}_1 + \ldots \lambda_{n-1} \mathbf{s}_{n-1}$ + preconditions: $\gcd(\gamma_i) \mid C$

Solution of a Homogenous Equation

Theorem

For an equation $\sum_{i=1}^{n} \gamma_i y_i = 0$ with S_H we denote the set of solutions.

$$S_H = \{\lambda_1 \begin{pmatrix} K_{11} \\ \vdots \\ K_{n1} \end{pmatrix} + \cdots + \lambda_{n-1} \begin{pmatrix} K_{1(n-1)} \\ \vdots \\ K_{n(n-1)} \end{pmatrix} \mid \lambda_i \in Z\}$$

where values K_{ij} are computed as follows:

- if i < j, K_{ij} = 0 (the matrix K is lower triangular)
- if i =j $K_{jj} = \frac{\gcd((\gamma_k)_{k \geq j+1})}{\gcd((\gamma_k)_{k \geq j})}$
- for remaining K_{ij} values, find any solution of the equation

$$\gamma_j K_{jj} + \sum_{i=j+1}^{n} \gamma_i z_{ij} = 0$$

Finding any Solution (*n* variables)

- Inductive approach
  - $\gamma_1 x_1 + \gamma_2 x_2 + \ldots + \gamma_n x_n = C$

    $\gamma_1 x_1 + \gcd(\gamma_2,\ldots,\gamma_n)[\lambda_2 x_2 + \ldots + \lambda_n x_n] = C$

    $\gamma_1 x_1 + \gamma x_F = C$
- find values for x_1 (w_1) and x_F (w_F) and then solve inductively:

  $\lambda_2 x_2 + \ldots + \lambda_n x_n = w_F$

Finding any Solution (*2* variables)

- based on Extended Euclidean Algorithm (EEA)
  - for every two integers *n* and *m* finds numbers *p* and *q* such that $n*p + m*q = gcd(n, m)$
- problem: $\gamma_1 x_1 + \gamma_2 x_2 = C$
- solution:
  - apply EEA to compute *p* and *q* such that

    $\gamma_1 p + \gamma_2 q = \gcd(\gamma_1, \gamma_2)$
  - solution: $x_1 = p*C/ \gcd(\gamma_1, \gamma_2)$

    $x_2 = q*C/ \gcd(\gamma_1, \gamma_2)$

Handling of Inequalities in Comfusy

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)**

Handling of Inequalities (1 variable)

- Solve for one by one variable:
  - separate inequalities depending on polarity of x:
    - $A_i \leq \alpha_i x$
    - $\beta_j x \leq B_j$
  - define values $a = \max_i[A_i/\alpha_i]$ and $b = \min_j[B_j/\beta_j]$
- if b is defined, return x = b else return x = a
- further continue with the conjunction of all formulas $[A_i/\alpha_i] \leq [B_j/\beta_j]$

Handling of Inequalities (more than 1 variable)

Consider the formula $2y - b \leq 3x + a \wedge 2x - a \leq 4y + b$

$$\lceil (2y - b - a)/3 \rceil \leq \lfloor (4y + a + b)/2 \rfloor$$

$$\Leftrightarrow \lceil (2y - b - a) * 2/6 \rceil \leq \lfloor (4y + a + b) * 3/6 \rfloor$$

$$\Leftrightarrow (4y - 2b - 2a)/6 \leq [(12y + 3a + 3b) - (12y + 3a + 3b) \bmod 6]/6$$

$$\Leftrightarrow (12y + 3a + 3b) \bmod 6 \leq 8y + 5a + 5b$$

$$\Leftrightarrow 12y + 3a + 3b = 6 * l + k \wedge k \leq 8y + 5a + 5b$$

Handling of Inequalities (more than 1 variable)

Consider the formula $2y - b \leq 3x + a \wedge 2x - a \leq 4y + b$

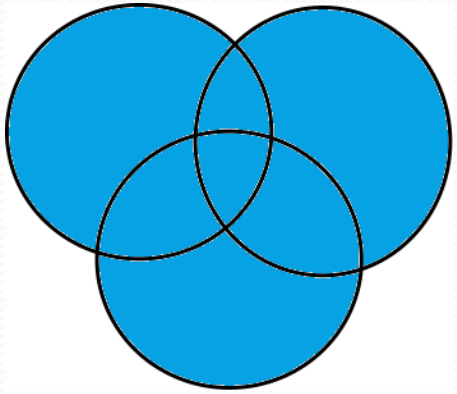
- $12y + 3a + 3b = 6 * l + k \wedge k \leq 8y + 5a + 5b$
- upon applying the equality, we obtain
  - preconditions: $6 | 3a + 3b - k$
  - solutions: $l = 2\lambda + (3a + 3b - k)/6$ and $y = \lambda$
- substituting those values in the inequality results in $k - 5a - 5b \leq 8\lambda$
- final solution: $\lambda = \lceil (k - 5a - 5b)/8 \rceil$

Handling of Data Structures in Comfusy

From Data Structures to Numbers

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

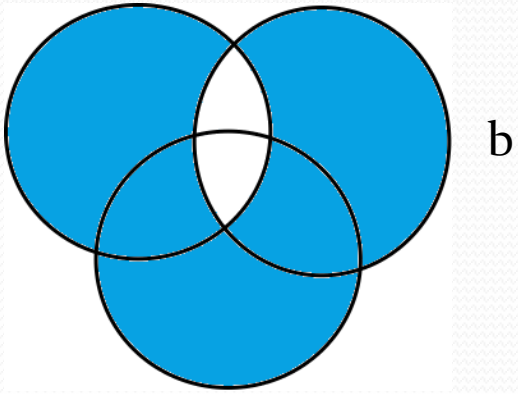
a.size == b.size && a union b == bigSet && a intersect b == empty



From Data Structures to Numbers

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

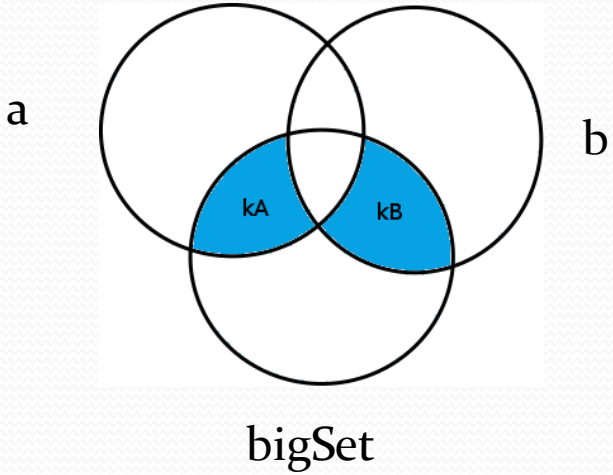
a.size == b.size && a union b == bigSet && a intersect b == empty



From Data Structures to Numbers

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

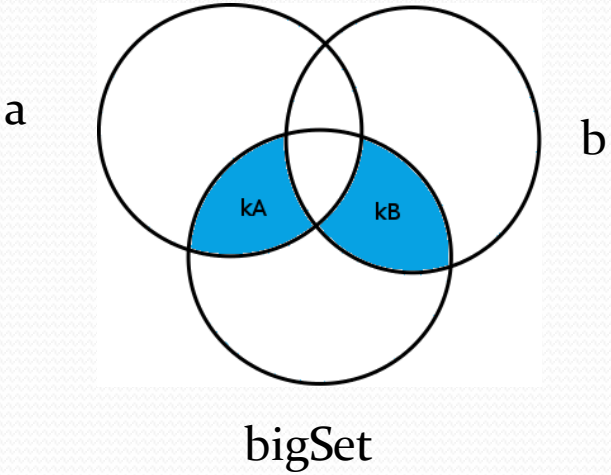
a.size == b.size && a union b == bigSet && a intersect b == empty



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



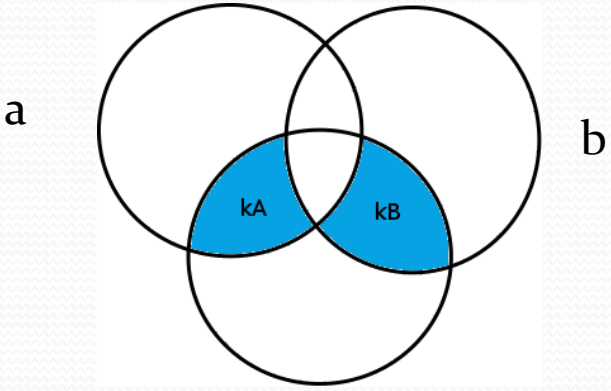
New specification:

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



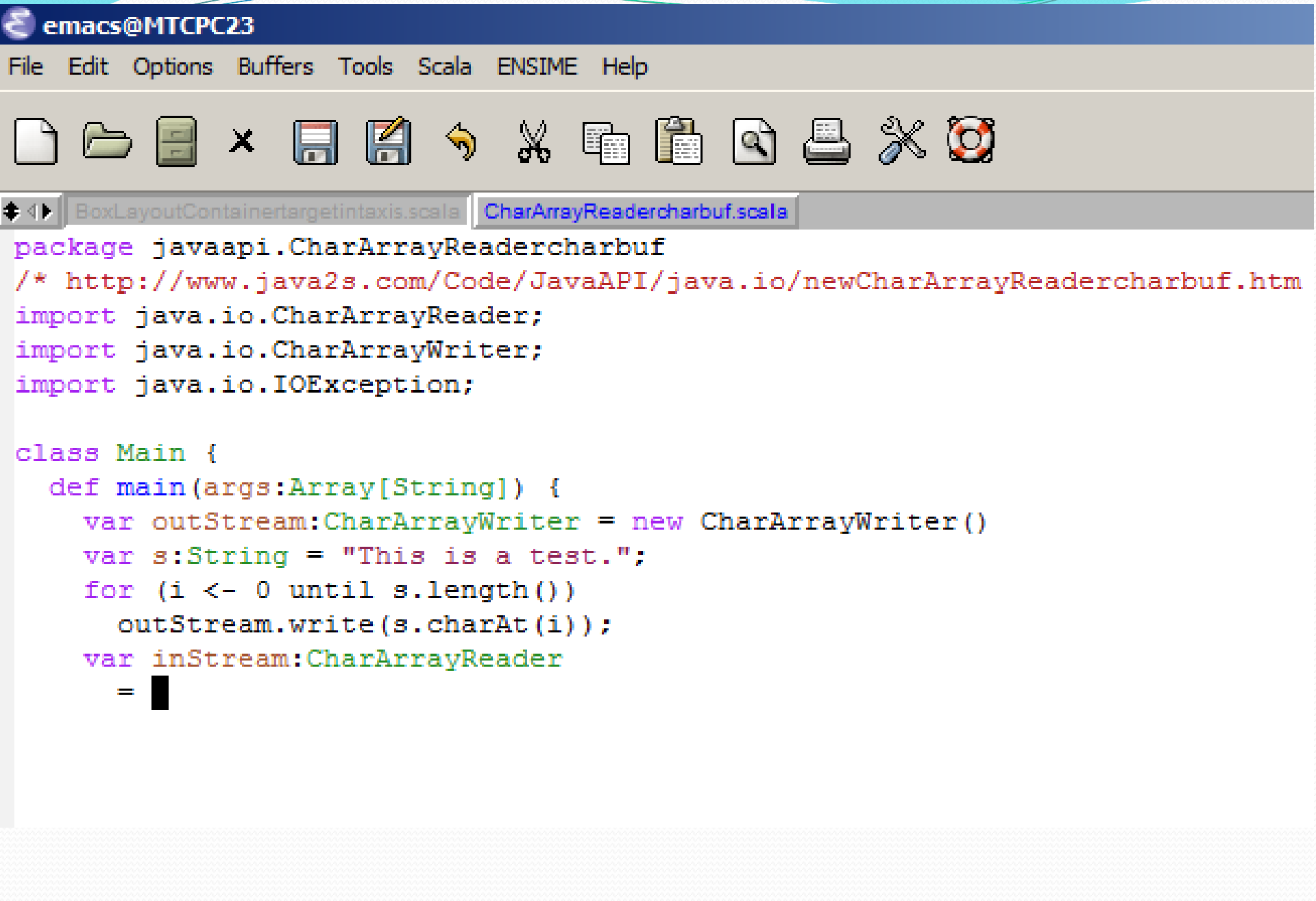
New specification:

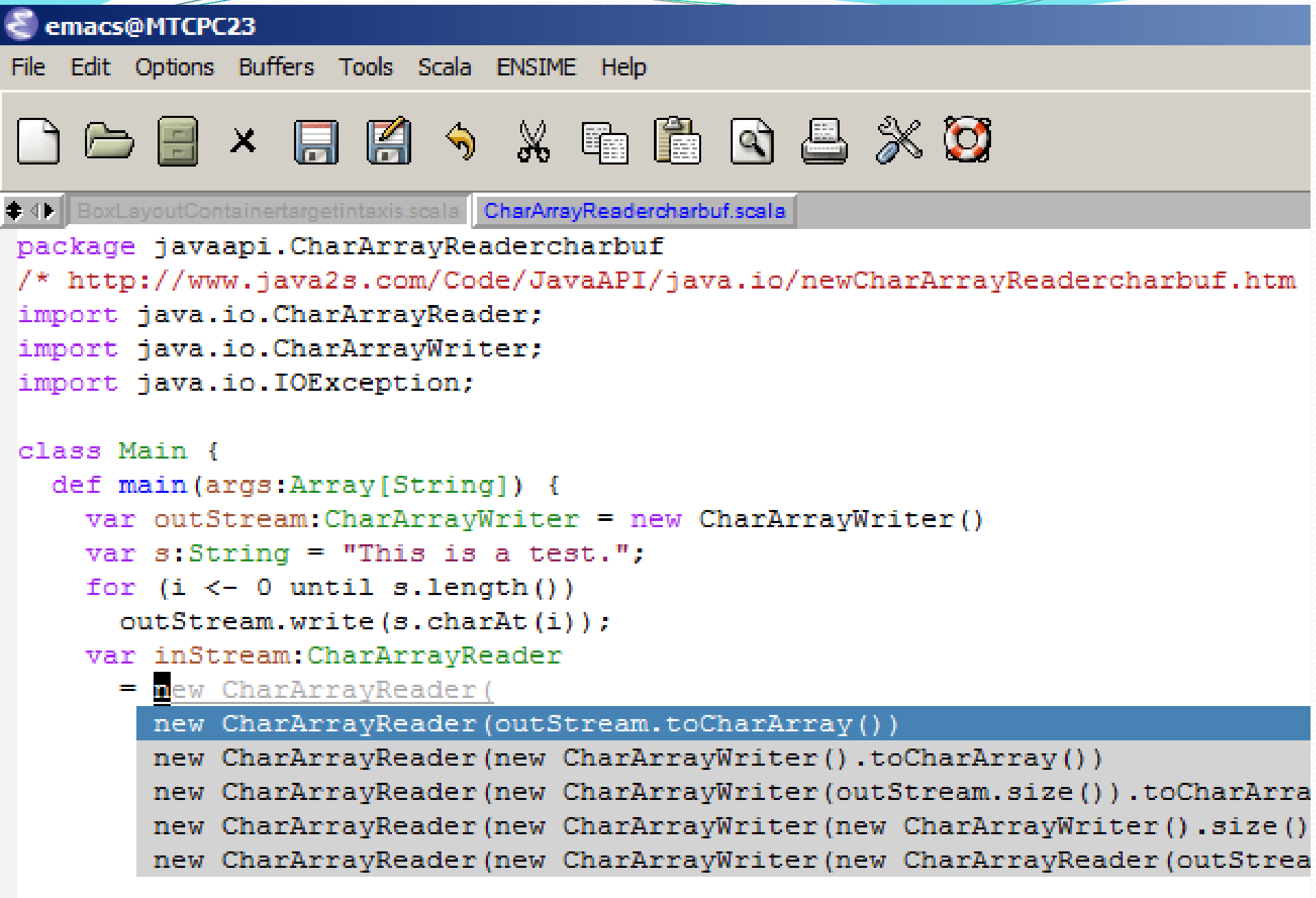
kA = kB && kA +kB = |bigSet|

because of quantifier elimination

Interactive Synthesis of Code Snippets

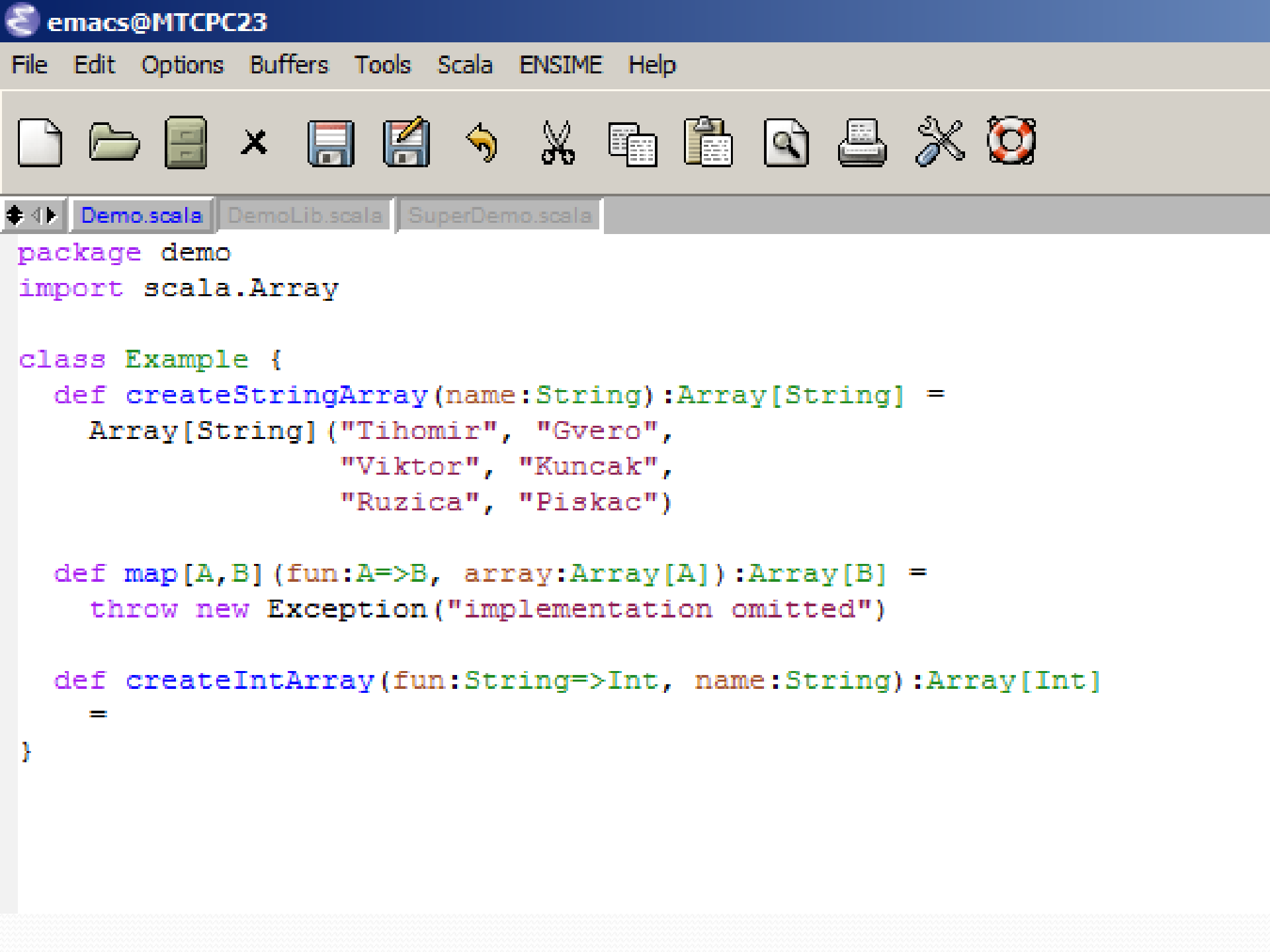
Joint work with Tihomir Gvero, Viktor Kuncak and Ivan Kuraj

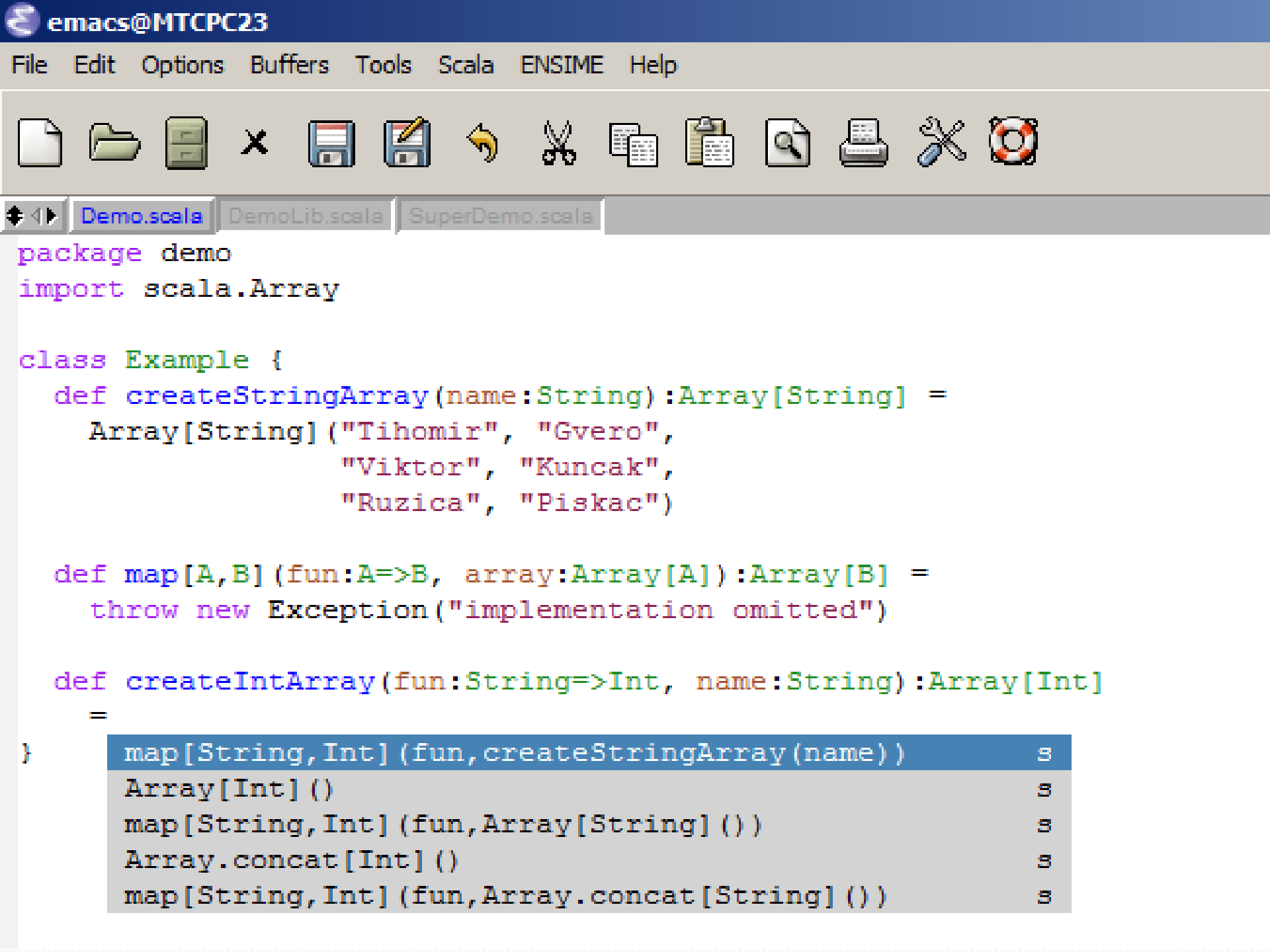

emacs@MTCPC23
File Edit Options Buffers Tools Scala ENSIME Help
BoxLayoutContainertargetintaxis.scala
CharArrayReadercharbuf.scala
package javaapi.CharArrayReadercharbuf
/* http://www.java2s.com/Code/JavaAPI/java.io/newCharArrayReadercharbuf.htm
import java.io.CharArrayReader;
import java.io.CharArrayWriter;
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

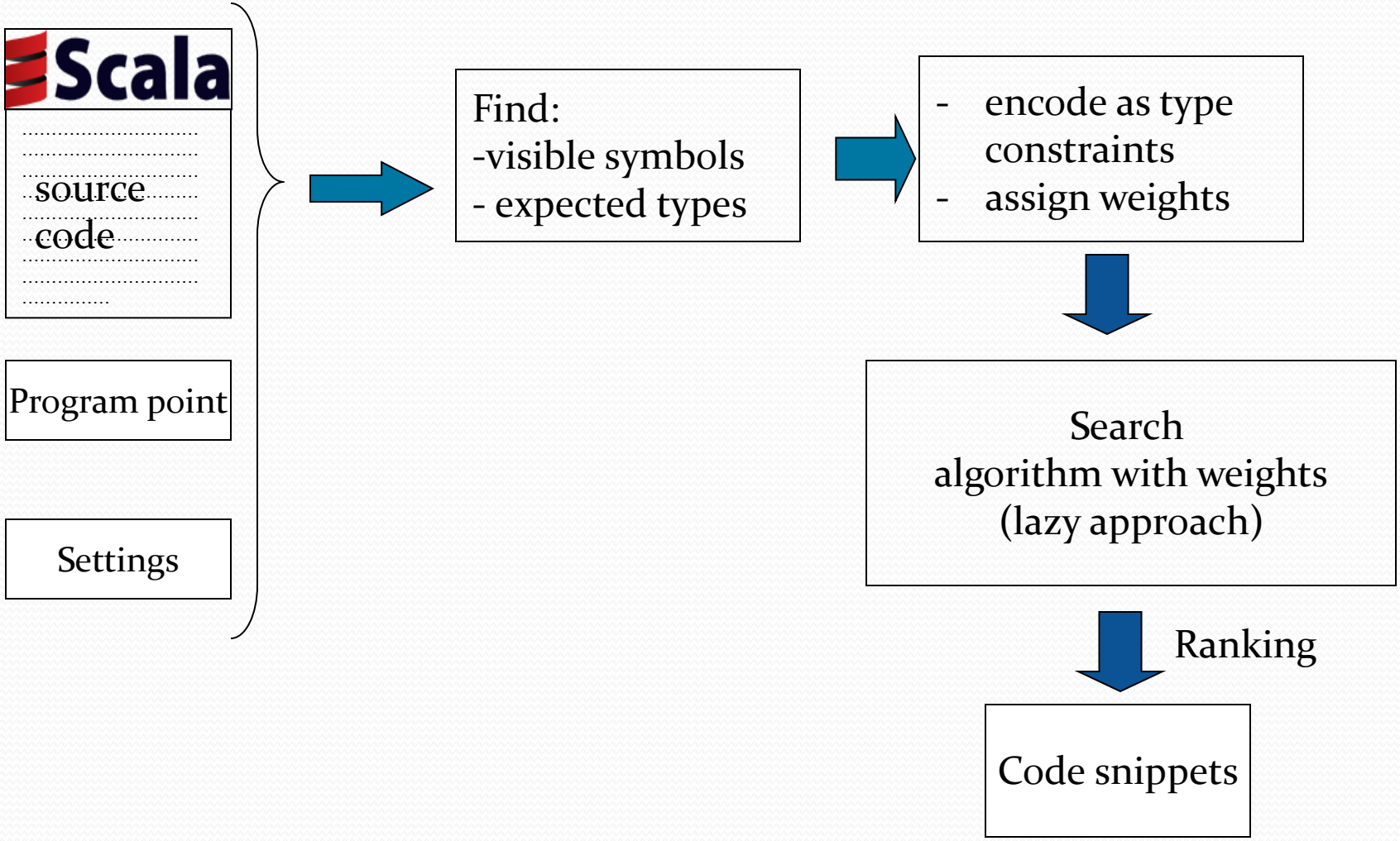
InSynth - Interactive Synthesis of Code Snippets

- Before: 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
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
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



Type Inhabitation Problem

- Given a set of types *T* and a set of expressions *E*, a type environment is a set

$$\Gamma = \{e_1 : \tau_1, e_2 : \tau_2, \ldots , e_n : \tau_n\}$$

Type Inhabitation Problem

Given a type environment Γ, a type τ and some calculus, is there are an expression *e* such that $\Gamma \vdash e : \tau$

Type Inhabitation in Lambda Calculus

Type Inhabitation [Statman, 1979] for ground

- the problem is PSPACE-complete

Theorem
The type inhabitation in ground applicative calculus without generating lambda expressions can be solved in polynomial time.

- For weak type polymorphism (quantifiers only on the top level), the type inhabitation problem is undecidable

Algorithm for TIP

TIP(Γ, τ) = switch (τ)

case $S \to \tau_1$: // $S \neq \emptyset$

val e = TIP($\Gamma \cup \Gamma_S$, τ_1)

if e == UNDEF **return** e

else return Reconstruct(Γ_{S_1}, e_1)

case $\emptyset \to \tau_1$:

val R = $\{f : \{A_1, ..., A_n\} \to \tau_1 \mid f : \{A_1, ..., A_n\} \to \tau_1 \in \Gamma\}$

if R == $\emptyset$ **return** UNDEF

run in **parallel** forall elements of R:

let r $\in$ R = g : $\{B_1, ..., B_m\} \to \tau_1$

if m == o **return** g

foreach B_i **do**

val e_i = TIP($\Gamma \setminus \{r\}$, τ)

if ($\forall$i: $e_i \neq$ UNDEF) **return** g $\{e_1, ..., e_m\}$

else return UNDEF

From the Lambda to the Succinct Representation

- Let C = {$c/_n$, ...} be a set of symbols. The elements of arity o are called *constants*.
- The set of all ground types is defined by the following grammar:

$$T_g ::= C(T_g, \ldots, T_g) \mid \{T_g, \ldots, T_g\} \rightarrow T_g$$

TYPE DECLARATIONS	TYPE ENVIRONMENT
val l: List[Int]	l : {} → List(Int)
def iTs(a: Int, b:Int): String	iTs : {Int} → String
def q(g : Int, f: Int=>Boolean): String	q: {{} →Int, {Int} → Boolean} → String

Calculus for Succinct Ground Types

$$\frac{\Gamma \cup S \vdash \pi : t}{\Gamma \vdash S \to t}\ \text{ABS}$$

$$\frac{\{t_1, \ldots, t_n\} \to t \in \Gamma \qquad \Gamma \vdash t_1 \quad \ldots \quad \Gamma \vdash t_n}{\Gamma \vdash @\{t_1, \ldots, t_n\} : t}\ \text{APP}$$

- where:
  - $@\{t_1, \ldots, t_n\}$ denotes a "pattern" – a witness that type *t* is inhabited, since all types t_i are also inhabited
  - $@\{t_1, \ldots, t_n\} : t$ says that an inhabitant of type *t* can be constructed from inhabitants of types t_i

Soundness and Completeness

- Let Γ_o be a set of standard lambda type declarations
- Let σ be a function converting lambda types into succinct types
- Let RCN be a function that reconstructs a lambda term (code snippet) from the succinct type environment

Theorem

$\Gamma_o \vdash_\lambda e : \tau$ iff $e \in$ RCN(Γ_o, $\sigma(\tau)$, L(e))

Quantitative Type Inhabitation Problem

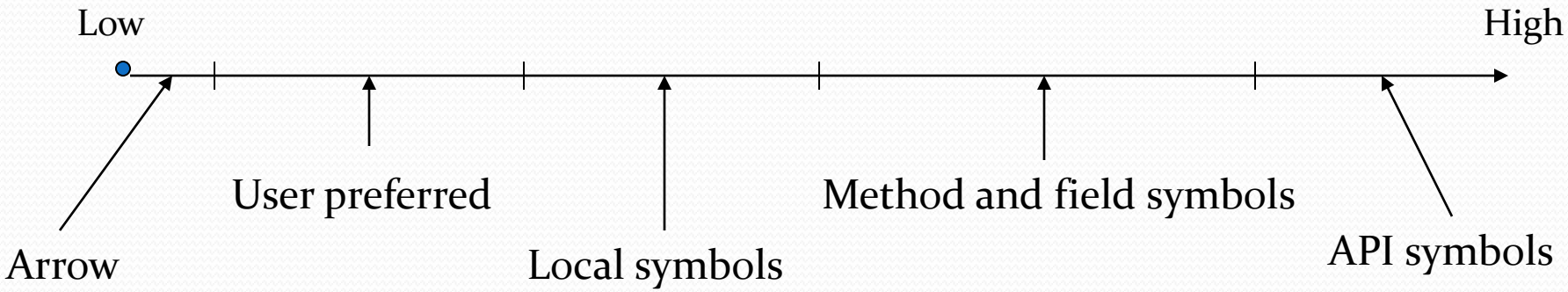
Quantitative Type Inhabitation Problem

Given a type environment Γ, a type τ and some calculus, is there are an expression *e* such that $\Gamma \vdash e : \tau$, and such that e is the "best possible"

- to all type assumptions we assign the weight
- a lower the weight indicates a more relevant term
- $w(e : \tau) = w(e) + w(\tau)$
- weight of a term or a type is computed as the sum of the weights of all symbols

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: ∀α. ArrayList[α]→ AbstractList[α]
c2: ∀β. AbstractList[β]→ AbstractCollection[β]

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[α]$\rightarrow$ AbstractList[α]
c2: $\forall\beta$. AbstractList[β]$\rightarrow$ AbstractCollection[β]
iterator: $\forall\ \gamma$. AbstractList[γ] $\rightarrow$ Iterator[γ]
goal : Iterator[String] $\rightarrow \bot$

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: ∀α. ArrayList[α]→ AbstractList[α]
c2: ∀β. AbstractList[β]→ AbstractCollection[β]
iterator: ∀ γ. AbstractList[γ] → Iterator[γ]
goal : 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.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

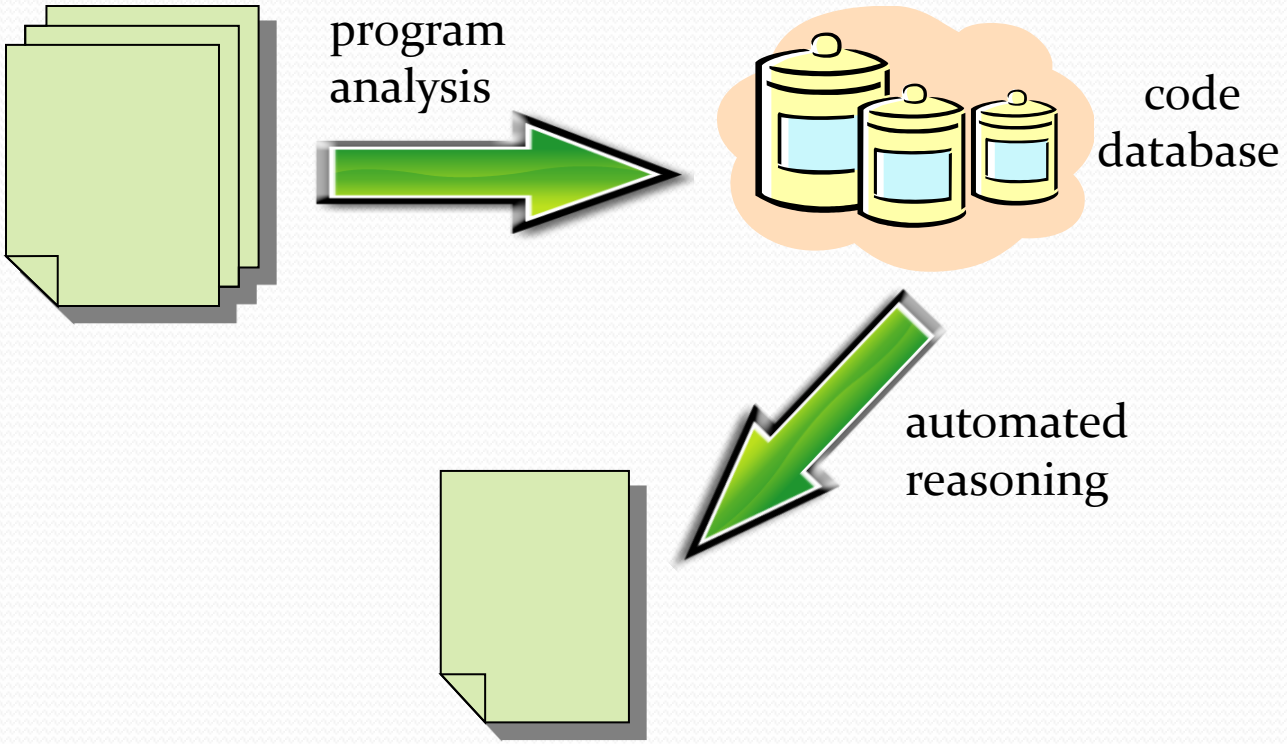
Benchmark	Lenth	#Initial	#Derived	#Snip.Gen.	Rank	Time [ms]
ByteArrayInputStreambytebufintoffsetintlength	4	22	4049	102	3	546
CharArrayReadercharbuf	3	26	782	343	1	546
HashSetiterator	2	60	1832	201	1	546
Hashtableelements	2	32	869	445	1	546
HashtableentrySet	2	31	874	441	1	546
HashtablekeySet	2	32	968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

Evaluation: the importance of weights and a corpus

Name	#Initial	No weights	No corpus	Both
AWTPermissionStringname	5615	>10	1	1
BufferedInputStreamFileInputStream	3364	>10	1	1
BufferedOutputStream	3367	>10	1	1
BufferedReaderFileReaderfileReader	3364	>10	2	1
BufferedReaderInputStreamReader	3364	>10	2	1
BufferedReaderReaderin	4094	>10	>10	6
ByteArrayInputStreambytebuf	3366	>10	3	>10
ByteArrayOutputStreamintsize	3363	>10	2	2
DatagramSocket	3246	>10	1	1
DataInputStreamFileInput	3364	>10	1	1
DataOutputStreamFileOutput	3364	>10	1	1
DefaultBoundedRangeModel	6673	>10	1	1

Future Directions

- Combination of program analysis, software synthesis and automated reasoning → use of code contracts



Contributions

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