Software Synthesis using Automated Reasoning

Ruzica Piskac
Yale University
How to obtain software reliability?

- software verification
How to obtain software reliability?

- software verification
  - specification
  - program
  - verifier
  - formulas
  - theorem prover
    - correct
    - no

- software synthesis
  - specification (formula)
  - constructive theorem prover
  - code
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
Automated Reasoning

• decision procedure answers whether the input formula is satisfiable or not
  - formula \( x \leq y \) is satisfiable for \( x=0, y=1 \)
  - formula \( x \leq y \land x + 1 > y + 1 \) is unsatisfiable
Talk Overview


ComFuSy - Complete Functional Synthesis

Joint work with Viktor Kuncak, Mikael Mayer and Philippe Suter
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
“choose” Construct

- specification is part of the Scala language
- two types of arguments: inputs and outputs
- a call of the form

\[
\text{val } x_1 = \text{choose}(x \Rightarrow F(x, a))
\]

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 a formula $F(x, a)$ and outputs:
1. a precondition formula $pre(a)$
2. list of terms $\Psi$

such that the following holds:

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

- 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 Boolean Algebra with Presburger Arithmetic (BAPA), for bit-vectors
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 the formula
  - for $n$ output variables, we need $n-1$ fresh new variables
  - number of output variables decreased by 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 the formula

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

Code:
<\text{further code will come here}>

```scala
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 the formula

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

Resulting formula (new specifications):

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

\[ 0 \leq \lambda, 0 \leq \mu, \mu \leq 59, \mu \leq \left\lfloor \frac{\text{totalSeconds} - 3600\lambda}{60} \right\rfloor, \]
\[ \left\lceil \frac{\text{totalSeconds} - 3600\lambda - 59}{60} \right\rceil \leq \mu \]

Code:

```plaintext
val mu = min(59, (totalSeconds - 3600* lambda) div 60)
```
Fourier-Motzkin-Style Elimination

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

combine each lower and upper bound

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

basic simplifications

Code:

```val```
lambda = totalSeconds \text{ div } 3600

Preconditions: \( 0 \leq \text{totalSeconds} \)
Observation:
- Reasoning about collections reduces to reasoning about linear integer arithmetic!

\[ \text{a.size} == \text{b.size} \land \text{a union b} == \text{bigSet} \land \text{a intersect b} == \text{empty} \]
From Data Structures to Numbers

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

\[ a.size == b.size \land a \cup b == \text{bigSet} \land a \cap b == \text{empty} \]
From Data Structures to Numbers

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

\[ a.\text{size} == b.\text{size} \ AndAlso a \cup b == \text{bigSet} \ AndAlso a \cap b == \text{empty} \]
Observation: Reasoning about collections reduces to reasoning about linear integer arithmetic!

\[ a.\text{size} == b.\text{size} \land a \cup b == \text{bigSet} \land a \cap b == \text{empty} \]

New specification:

\[ kA = kB \land kA + kB = |\text{bigSet}| \]
Observation:

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

$$a\text{.size} == b\text{.size} \&\& a\text{ union } b == \text{bigSet} \&\& a\text{ intersect } b == \text{empty}$$

New specification:

$$kA = kB \&\& kA + kB = |\text{bigSet}|$$

because of quantifier elimination
Interactive Synthesis of Code Snippets

Joint work with Tihomir Gvero, Viktor Kuncak and Ivan Kuraj
package javaapi.CharArrayReadercharbuf

import java.io.CharArrayReader;
import java.io.CharArrayWriter;
import java.io.IOException;

class Main {
  def main(args: Array[String]) {
    var outputStream: CharArrayWriter = new CharArrayWriter()
    var s: String = "This is a test."
    for (i <- 0 until s.length())
      outputStream.write(s.charAt(i))
    var inputStream: CharArrayReader
      =
package javaapi.CharArrayReadercharbuf

/* http://www.java2s.com/Code/JavaAPI/java.io/newCharArrayReadercharbuf.htm */

import java.ioCharArrayReader;
import java.ioCharArrayWriter;
import java.io.IOException;

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

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))
}

Array[Int]()
map[String, Int](fun, Array[String]()())
Array.concat[Int]()[()]
map[String, Int](fun, Array.concat[String]()())
Snippet Synthesis inside IDE

- Program point
  - source code

- Settings

Find:
- visible symbols
- expected types

- encode as type constraints
- assign weights

Search algorithm with weights (lazy approach)

Ranking

Code snippets

Scala
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 in Lambda Calculus

Type Inhabitation [Statman, 1979] for ground lambda calculus
- 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.
Succinct Lambda Calculus

- Succinct representation of type declarations
  - def iTs (a: Int, b: Int, c: Int): String
  - iTs : {Int} → String
- Reason: efficiency

<table>
<thead>
<tr>
<th>Without succinct types</th>
<th>With succinct types</th>
</tr>
</thead>
<tbody>
<tr>
<td>74% cases: desired snippet is among top 5 returned solution</td>
<td>94% cases: desired snippet is among top 5 returned solution</td>
</tr>
<tr>
<td>56% cases: desired snippet is top ranked</td>
<td>64% cases: desired snippet is top ranked</td>
</tr>
<tr>
<td>Average total time: 401ms (prover 266ms, reconstructor 135ms)</td>
<td>Average total time: 145ms (prover 9ms, reconstructor 136ms)</td>
</tr>
</tbody>
</table>
Succinct Lambda Calculus

- Efficient encoding of “patterns” - a witness that type $t$ is inhabited – finite graph representation of possibly infinite terms
- To derive the corresponding code snippets, we use a reconstruction function, combined with the weight function (to obtain the ranking of snippets)
- Succinct lambda calculus is sound and complete:

**Theorem**

A lambda term can be derived in the (standard) lambda calculus iff it can be “derived” in the succinct lambda calculus.
Quantitative Type Inhabitation Problem

Quantitative Type Inhabitation Problem
Given a type environment $\Gamma$, a type $\tau$ and some calculus, is there an expression $e$ such that $\Gamma \vdash e : \tau$, and such that $e$ is the “best possible solution”

- to all type assumptions we assign a weight
- lower weight indicates that term is more relevant
- 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 (obtained by analyzing a training set 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 $c: A \rightarrow B$ [Tannen et al., 1991]

```scala
class ArrayList[T] extends AbstractList[T] with List[T]
    with RandomAccess with Cloneable with Serializable {...}
    ....
    def iterator(): Iterator[E] = {...}
}
```
Subtyping using Coercions

- We model $A <: B$ by introducing a coercion function $c: A \rightarrow B$ [Tannen et al., 1991]

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

$c_1: \forall \alpha. \text{ArrayList}[^\alpha] \rightarrow \text{AbstractList}[^\alpha]$
$c_2: \forall \beta. \text{AbstractList}[\beta] \rightarrow \text{AbstractCollection}[\beta]$
Subtyping Example

```scala
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

```scala
val a1: ArrayList[String] = ...
...

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

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

val i1: Iterator[String] = ...
```
Subtyping Example

```scala
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] = ...
...

val i1: Iterator[String] = ...
```

**Subtyping Example:**

1. `val a1: ArrayList[String] = ...`
   ....
   def iterator():Iterator[E] = {...}
}
`
Subtyping Example

```scala
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
```

`iterator(c1(a1)): Iterator[String]`
<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Size</th>
<th>#Initial</th>
<th>Rank</th>
<th>Total</th>
<th>Rank</th>
<th>Prove</th>
<th>Recon</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWTPermissionStringname</td>
<td>2/2</td>
<td>5615</td>
<td>&gt;10</td>
<td>101</td>
<td>1</td>
<td>8</td>
<td>125</td>
<td>133</td>
</tr>
<tr>
<td>BufferedInputStreamFileInputStream</td>
<td>3/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>45</td>
<td>1</td>
<td>7</td>
<td>46</td>
<td>53</td>
</tr>
<tr>
<td>BufferedOutputStream</td>
<td>3/2</td>
<td>3367</td>
<td>&gt;10</td>
<td>18</td>
<td>1</td>
<td>7</td>
<td>11</td>
<td>19</td>
</tr>
<tr>
<td>BufferedReaderFileReaderfileReader</td>
<td>4/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>69</td>
<td>1</td>
<td>7</td>
<td>43</td>
<td>50</td>
</tr>
<tr>
<td>BufferedReaderInputStreamReader</td>
<td>4/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>66</td>
<td>1</td>
<td>7</td>
<td>42</td>
<td>49</td>
</tr>
<tr>
<td>BufferedReaderReaderin</td>
<td>5/4</td>
<td>4094</td>
<td>&gt;10</td>
<td>4760</td>
<td>6</td>
<td>7</td>
<td>237</td>
<td>244</td>
</tr>
<tr>
<td>ByteArrayInputStreambytebuf</td>
<td>4/4</td>
<td>3366</td>
<td>&gt;10</td>
<td>94</td>
<td>&gt;10</td>
<td>4</td>
<td>18</td>
<td>22</td>
</tr>
<tr>
<td>ByteArrayOutputStreamintsize</td>
<td>2/2</td>
<td>3363</td>
<td>&gt;10</td>
<td>51</td>
<td>2</td>
<td>8</td>
<td>63</td>
<td>70</td>
</tr>
<tr>
<td>DatagramSocket</td>
<td>1/1</td>
<td>3246</td>
<td>&gt;10</td>
<td>74</td>
<td>1</td>
<td>7</td>
<td>80</td>
<td>88</td>
</tr>
<tr>
<td>DataInputStreamFileInput</td>
<td>3/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>20</td>
<td>1</td>
<td>6</td>
<td>46</td>
<td>52</td>
</tr>
<tr>
<td>DataOutputStreamFileOutput</td>
<td>3/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>29</td>
<td>1</td>
<td>7</td>
<td>38</td>
<td>45</td>
</tr>
<tr>
<td>DefaultBoundedRangeModel</td>
<td>1/1</td>
<td>6673</td>
<td>&gt;10</td>
<td>220</td>
<td>1</td>
<td>10</td>
<td>257</td>
<td>266</td>
</tr>
<tr>
<td>DisplayModeintwidthintheightintbit</td>
<td>2/2</td>
<td>4999</td>
<td>&gt;10</td>
<td>136</td>
<td>1</td>
<td>6</td>
<td>147</td>
<td>154</td>
</tr>
<tr>
<td>FileInputStreamFileDescriptorfdObj</td>
<td>2/2</td>
<td>3366</td>
<td>&gt;10</td>
<td>24</td>
<td>2</td>
<td>6</td>
<td>17</td>
<td>23</td>
</tr>
<tr>
<td>FileInputStreamStringname</td>
<td>2/2</td>
<td>3363</td>
<td>&gt;10</td>
<td>125</td>
<td>1</td>
<td>9</td>
<td>100</td>
<td>109</td>
</tr>
<tr>
<td>FileOutputStreamFilefile</td>
<td>2/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>86</td>
<td>1</td>
<td>8</td>
<td>51</td>
<td>60</td>
</tr>
<tr>
<td>FileReaderFilefile</td>
<td>2/2</td>
<td>3365</td>
<td>&gt;10</td>
<td>37</td>
<td>2</td>
<td>7</td>
<td>13</td>
<td>20</td>
</tr>
<tr>
<td>FileStringname</td>
<td>2/2</td>
<td>3363</td>
<td>&gt;10</td>
<td>169</td>
<td>1</td>
<td>7</td>
<td>155</td>
<td>163</td>
</tr>
<tr>
<td>FileWriterFilefile</td>
<td>2/2</td>
<td>3366</td>
<td>&gt;10</td>
<td>40</td>
<td>1</td>
<td>8</td>
<td>28</td>
<td>36</td>
</tr>
<tr>
<td>FileWriterLPT1</td>
<td>2/2</td>
<td>3363</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GridBagConstraints</td>
<td>1/1</td>
<td>8402</td>
<td>&gt;10</td>
<td>3241</td>
<td>1</td>
<td>19</td>
<td>323</td>
<td>342</td>
</tr>
<tr>
<td>GridBagLayout</td>
<td>1/1</td>
<td>8401</td>
<td>&gt;10</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>GroupLayoutContainerhost</td>
<td>4/2</td>
<td>6436</td>
<td>&gt;10</td>
<td>24</td>
<td>1</td>
<td>10</td>
<td>26</td>
<td>36</td>
</tr>
<tr>
<td>ImageIconStringfilename</td>
<td>2/2</td>
<td>8277</td>
<td>&gt;10</td>
<td>495</td>
<td>1</td>
<td>13</td>
<td>154</td>
<td>167</td>
</tr>
<tr>
<td>InputStreamReaderInputStream</td>
<td>3/3</td>
<td>3363</td>
<td>&gt;10</td>
<td>90</td>
<td>4</td>
<td>7</td>
<td>177</td>
<td>184</td>
</tr>
<tr>
<td>JButtonStringtext</td>
<td>2/2</td>
<td>6434</td>
<td>&gt;10</td>
<td>117</td>
<td>1</td>
<td>9</td>
<td>85</td>
<td>95</td>
</tr>
<tr>
<td>JCheckBoxStringtext</td>
<td>2/2</td>
<td>8401</td>
<td>&gt;10</td>
<td>134</td>
<td>2</td>
<td>18</td>
<td>50</td>
<td>68</td>
</tr>
<tr>
<td>JTextFieldAbstractFormatter</td>
<td>3/2</td>
<td>10700</td>
<td>&gt;10</td>
<td>2048</td>
<td>4</td>
<td>21</td>
<td>101</td>
<td>122</td>
</tr>
<tr>
<td>JTextFieldTextFieldFormatter</td>
<td>2/2</td>
<td>9783</td>
<td>&gt;10</td>
<td>67</td>
<td>2</td>
<td>15</td>
<td>85</td>
<td>100</td>
</tr>
<tr>
<td>JTableObjectnameObjectdata</td>
<td>3/3</td>
<td>8280</td>
<td>&gt;10</td>
<td>109</td>
<td>2</td>
<td>13</td>
<td>129</td>
<td>142</td>
</tr>
</tbody>
</table>
## Evaluation

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Size</th>
<th>#Deriv.</th>
<th>Rank</th>
<th>Total</th>
<th>Rank</th>
<th>Prove [ms]</th>
<th>Recon [ms]</th>
<th>Total [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWTPermissionStringname</td>
<td>2/2</td>
<td>5615</td>
<td>&gt;10</td>
<td>101</td>
<td>1</td>
<td>8</td>
<td>125</td>
<td>133</td>
</tr>
<tr>
<td>BufferedInputStreamFileInputStream</td>
<td>3/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>45</td>
<td>1</td>
<td>7</td>
<td>46</td>
<td>53</td>
</tr>
<tr>
<td>BufferedOutputStream</td>
<td>3/2</td>
<td>3367</td>
<td>&gt;10</td>
<td>18</td>
<td>1</td>
<td>7</td>
<td>11</td>
<td>19</td>
</tr>
<tr>
<td>BufferedReaderFileReaderfileReader</td>
<td>4/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>69</td>
<td>1</td>
<td>7</td>
<td>43</td>
<td>50</td>
</tr>
<tr>
<td>BufferedReaderInputStreamReader</td>
<td>4/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>66</td>
<td>1</td>
<td>7</td>
<td>42</td>
<td>49</td>
</tr>
<tr>
<td>BufferedReaderReaderin</td>
<td>5/4</td>
<td>4094</td>
<td>&gt;10</td>
<td>4760</td>
<td>6</td>
<td>7</td>
<td>237</td>
<td>244</td>
</tr>
<tr>
<td>ByteArrayInputStreambytebuf</td>
<td>4/4</td>
<td>3366</td>
<td>&gt;10</td>
<td>94</td>
<td>&gt;10</td>
<td>4</td>
<td>18</td>
<td>22</td>
</tr>
<tr>
<td>ByteArrayOutputStreamintsize</td>
<td>2/2</td>
<td>3363</td>
<td>&gt;10</td>
<td>51</td>
<td>2</td>
<td>8</td>
<td>63</td>
<td>70</td>
</tr>
<tr>
<td>DatagramSocket</td>
<td>1/1</td>
<td>3246</td>
<td>&gt;10</td>
<td>74</td>
<td>1</td>
<td>7</td>
<td>80</td>
<td>88</td>
</tr>
<tr>
<td>DataInputStreamFileInput</td>
<td>3/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>20</td>
<td>1</td>
<td>6</td>
<td>46</td>
<td>52</td>
</tr>
</tbody>
</table>
## Evaluation

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Size</th>
<th>#Deriv.</th>
<th>Rank</th>
<th>Total</th>
<th>Size</th>
<th>#Deriv.</th>
<th>Rank</th>
<th>Total</th>
<th>Prove [ms]</th>
<th>Recon [ms]</th>
<th>Total [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWTPermissionStringname</td>
<td>2/2</td>
<td>5615</td>
<td>&gt;10</td>
<td>101</td>
<td>1/2</td>
<td>8053</td>
<td>&gt;10</td>
<td>101</td>
<td>1</td>
<td>8</td>
<td>125</td>
</tr>
<tr>
<td>BufferedReaderInputStreamFileInputOutputStream</td>
<td>3/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>45</td>
<td>1/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>45</td>
<td>1</td>
<td>7</td>
<td>46</td>
</tr>
<tr>
<td>BufferedReaderOutputStream</td>
<td>3/2</td>
<td>3367</td>
<td>&gt;10</td>
<td>18</td>
<td>1/2</td>
<td>3367</td>
<td>&gt;10</td>
<td>18</td>
<td>1</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>BufferedReaderFileReaderFileReader</td>
<td>4/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>69</td>
<td>1/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>69</td>
<td>1</td>
<td>7</td>
<td>43</td>
</tr>
<tr>
<td>BufferedReaderInputStreamReader</td>
<td>4/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>66</td>
<td>1/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>66</td>
<td>1</td>
<td>7</td>
<td>42</td>
</tr>
<tr>
<td>BufferedReaderReaderin</td>
<td>5/4</td>
<td>4094</td>
<td>&gt;10</td>
<td>4760</td>
<td>1/2</td>
<td>4094</td>
<td>&gt;10</td>
<td>4760</td>
<td>6</td>
<td>7</td>
<td>237</td>
</tr>
<tr>
<td>ByteArrayInputStreambytebuf</td>
<td>4/4</td>
<td>3366</td>
<td>&gt;10</td>
<td>94</td>
<td>1/2</td>
<td>3366</td>
<td>&gt;10</td>
<td>94</td>
<td>&gt;10</td>
<td>4</td>
<td>18</td>
</tr>
<tr>
<td>ByteArrayOutputStreamintsize</td>
<td>2/2</td>
<td>3363</td>
<td>&gt;10</td>
<td>51</td>
<td>1/2</td>
<td>3363</td>
<td>&gt;10</td>
<td>51</td>
<td>2</td>
<td>8</td>
<td>63</td>
</tr>
<tr>
<td>DatagramSocket</td>
<td>1/1</td>
<td>3246</td>
<td>&gt;10</td>
<td>74</td>
<td>1/2</td>
<td>3246</td>
<td>&gt;10</td>
<td>74</td>
<td>1</td>
<td>7</td>
<td>80</td>
</tr>
<tr>
<td>DataInputStreamFileInput</td>
<td>3/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>20</td>
<td>1/2</td>
<td>3364</td>
<td>&gt;10</td>
<td>20</td>
<td>1</td>
<td>6</td>
<td>46</td>
</tr>
</tbody>
</table>
Future Directions

- Program synthesis using natural language specifications
  - doctoral thesis of Tihomir Gvero
- Defining specification using examples – the MSR SEIF award 2014
  - on-going work with Sumit Gulwani et al.
- Program repair
  - on-going work with a Yale undergrad

```java
// error: BufferedReader has no such constructor
BufferedReader br = new BufferedReader (" file . txt ");
```
Future Directions

- Program synthesis using natural language specifications
  - doctoral thesis of Tihomir Gvero
- Defining specification using examples – the MSR SEIF award 2014
  - on-going work with Sumit Gulwani et al.
- Program repair
  - on-going work with a Yale undergrad

// correct : BufferedReader is built from a Reader implementation
BufferedReader br = new BufferedReader ( new FileReader ( " file . txt " ) ) ;
Additional Information

Conclusions and Contributions

Goal: create more reliable software
Focus of this talk: Software Synthesis

- method to obtain correct software from the given specification
- Complete Functional Synthesis (Comfusy): extending decision procedures into synthesis algorithms
- Interactive Synthesis of Code Snippets (InSynth): finding a term of a given type and a construction of the program snippet