# *From Decision Procedures to Synthesis Procedures*

RUZICA PISKAC

YALE UNIVERSITY

# 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

# ComFuSy - Complete Functional Synthesis

WORK DONE DURING MY PHD STUDIES

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
```

# “choose” Construct

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

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

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 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 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]

# 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: $\gcd(\gamma_i) \mid C$

# Example

h * 3600 + m * 60 + s =  totalSeconds

$$S_H = \{(h, m, s) \mid \sum_{i=1}^{n} 3600\text{h} + 60\text{m} + \text{s} = 0\}$$

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

Any solution of h * 3600 + m * 60 + s =  totalSeconds

(h, m, s) = (0, 0, totalSeconds)

# Example

h * 3600 + m * 60 + s = totalSeconds

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

# 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

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

# Example

$3600\,h + 60\,m + s = 0$

$$S_H = \{(h, m, s) \mid \sum_{i=1}^{n} 3600\mathrm{h} + 60\mathrm{m} + \mathrm{s} = 0\}$$

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

Any solution of $1 * 3600 + m * 60 + s = 0$

$(h, m, s) = (1, 0, -3600)$

# Example

3600 h + 60 m + s = 0

$$S_H = \{(h, m, s) \mid \sum_{i=1}^{n} 3600\text{h} + 60\text{m} + \text{s} = 0\}$$

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

Any solution of 0 * 3600 + 1 * 60 + s = 0

(h, m, s) = (0, 1, - 60)

# Example

3600 h + 60 m + s = 0

$$S_H = \{(h, m, s) \mid \sum_{i=1}^{n} 3600\text{h} + 60\text{m} + \text{s} = 0\}$$

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

# 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:

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

# Example

$$s + h * 3600 + m * 60 = totalSeconds$$

$$s + 60 *( 60h + m) = totalSeconds$$

$$s + 60 * x = totalSeconds$$

$$(s, x) = (totalSeconds, 0)$$

$$60h + m = 0$$

# 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: $x_1 = p*C/\gcd(\gamma_1, \gamma_2)$

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

# Example

12 x + 8 y = a

12 *1 + 8 *(-1) = 4

3 *1 + 2 *(-1) = 1

3a + (-2)a = a

12 * (a / 4) + 8 (-a / 4) = a

# 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

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

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 the formula

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

Resulting formula (new specifications):

0 ≤ λ, 0 ≤ μ, μ ≤ 59, 0 ≤ totalSeconds – 3600λ - 60μ, totalSeconds – 3600λ - 60μ ≤ 59

# Processing Inequalities

process output variables one by one

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

expressing constraints as bounds on μ

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

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:

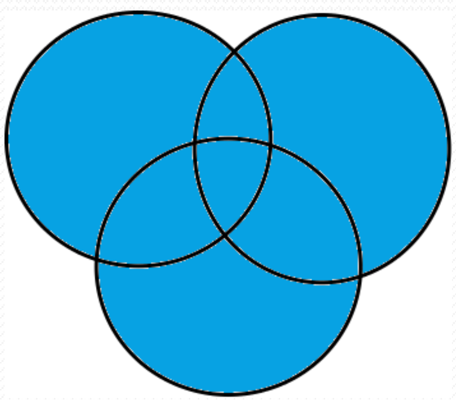
**val** lambda = totalSeconds **div** 3600

Preconditions: 0 ≤ totalSeconds

# From Data Structures to Numbers

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

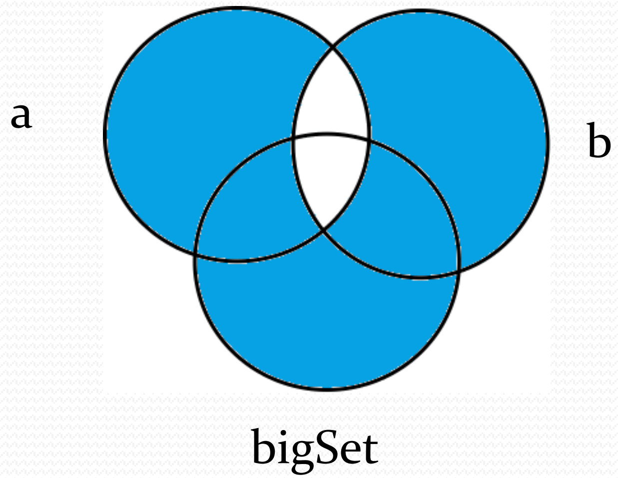
a.size == b.size && a union b == bigSet && a intersect b == empty



# From Data Structures to Numbers

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

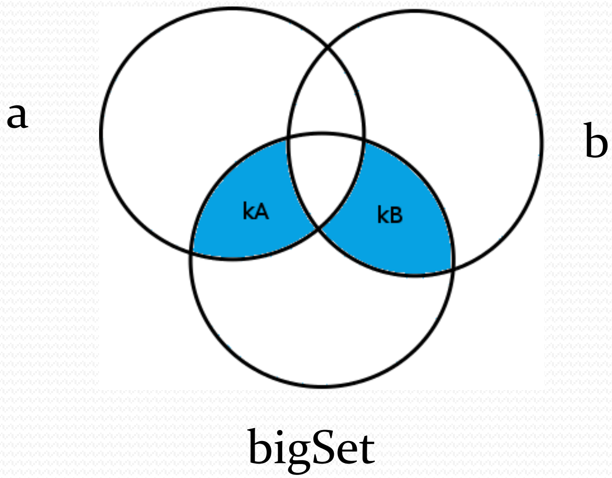
a.size == b.size && a union b == bigSet && a intersect b == empty



# From Data Structures to Numbers

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

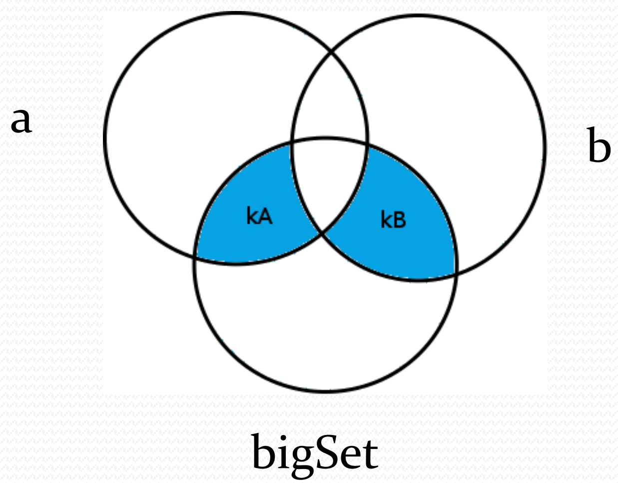
a.size == b.size && a union b == bigSet && a intersect b == empty



# 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



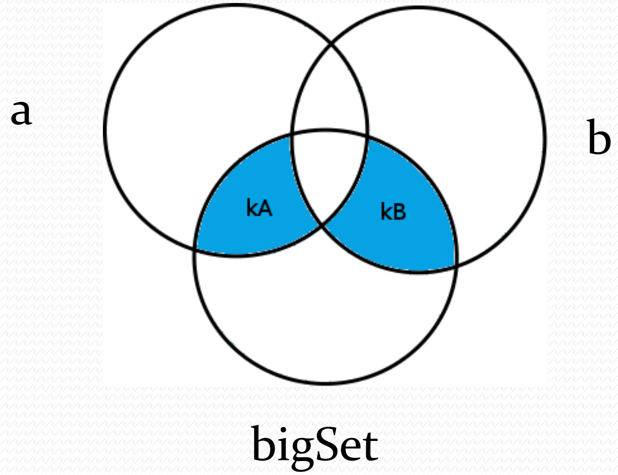

New specification:

kA = kB && kA +kB = |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




New specification:

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

because of quantifier elimination

# Summary: Comfusy

- Complete Functional Synthesis: extending decision procedures into synthesis algorithms
- A different synthesis procedure for a logic in which specification is given
- Completeness and correctness guarantees
- Writing complete specification can be a task harder than writing code

# Applications of Synthesis

CODE COMPLETION, CODE CORRECTION

# Example: Sequence of Streams

```
def main(args:Array[String]) = {
    var body:String = "email.txt"
    var sig:String = "signature.txt"
    var inStream:SeqInStr =
    …
}
```

# Example: Sequence of Streams

```
def main(args:Array[String]) = {
    var body:String = "email.txt"
    var sig:String = "signature.txt"
    var inStream:SeqInStr =   new SeqInStr(new FileInStr(sig), new FileInStr(sig))
                              new SeqInStr(new FileInStr(sig), new FileInStr(body))
                              new SeqInStr(new FileInStr(body), new FileInStr(sig))
                              new SeqInStr(new FileInStr(body), new FileInStr(body))
                              new SeqInStr(new FileInStr(sig), System.in)
    ...
}
```

# Example: Sequence of Streams

```
def main(args:Array[String]) = {
    var body:String = "email.txt"
    var sig:String = "signature.txt"
    var inStream:SeqInStr =
    ...
}
```

```
new SeqInStr(new FileInStr(sig), new FileInStr(sig))
new SeqInStr(new FileInStr(sig), new FileInStr(body))
new SeqInStr(new FileInStr(body), new FileInStr(sig))
new SeqInStr(new FileInStr(body), new FileInStr(body))
new SeqInStr(new FileInStr(sig), System.in)
```

# Example: Sequence of Streams

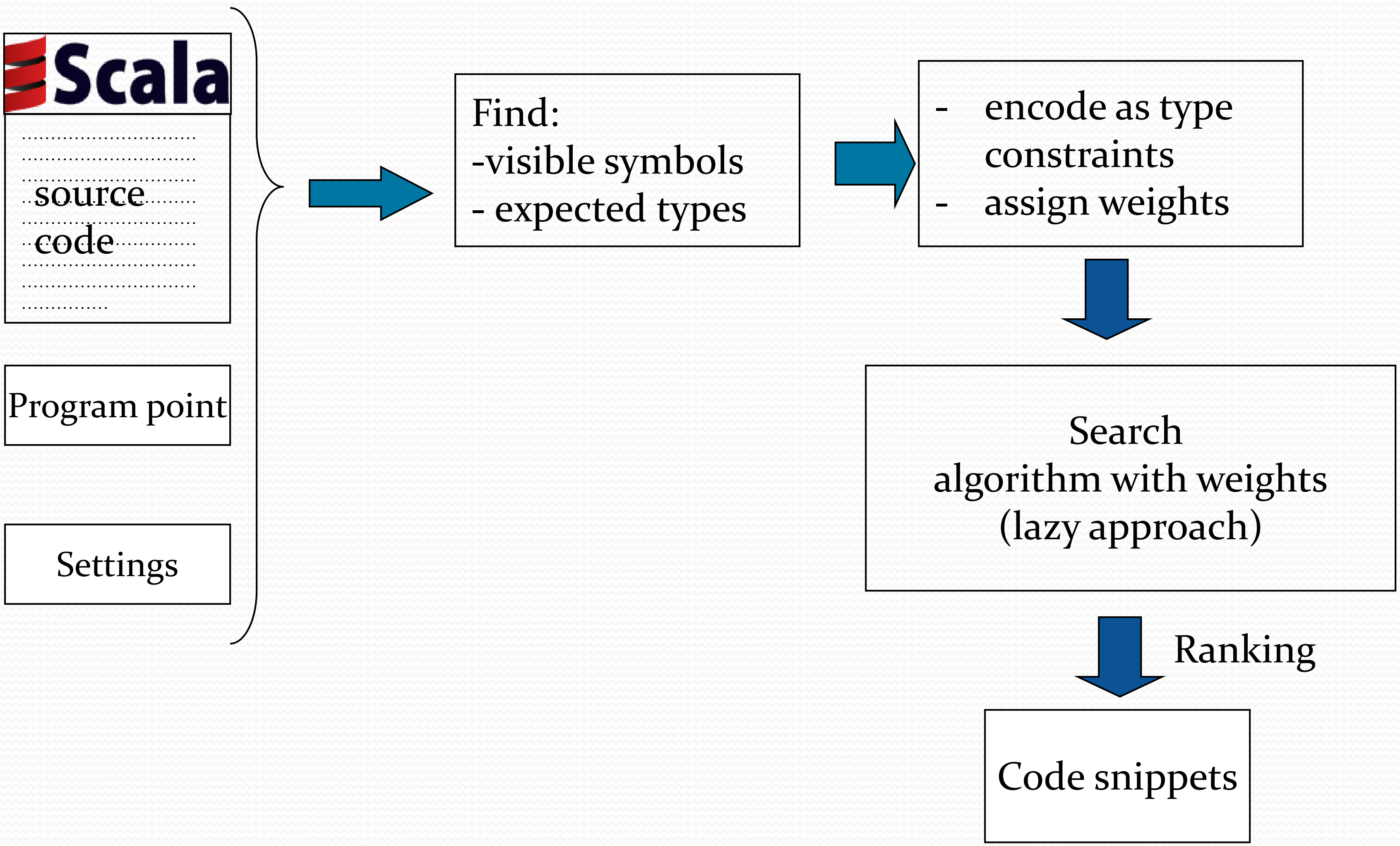
```
def main(args:Array[String]) = {
    var body:String = "email.txt"
    var sig:String = "signature.txt"
    var inStream:SeqInStr = new SeqInStr(new FileInStr(sig), new
FileInStr(body))

    ...
}
```

# 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

# Snippet Synthesis inside IDE



# Type Inhabitation Problem

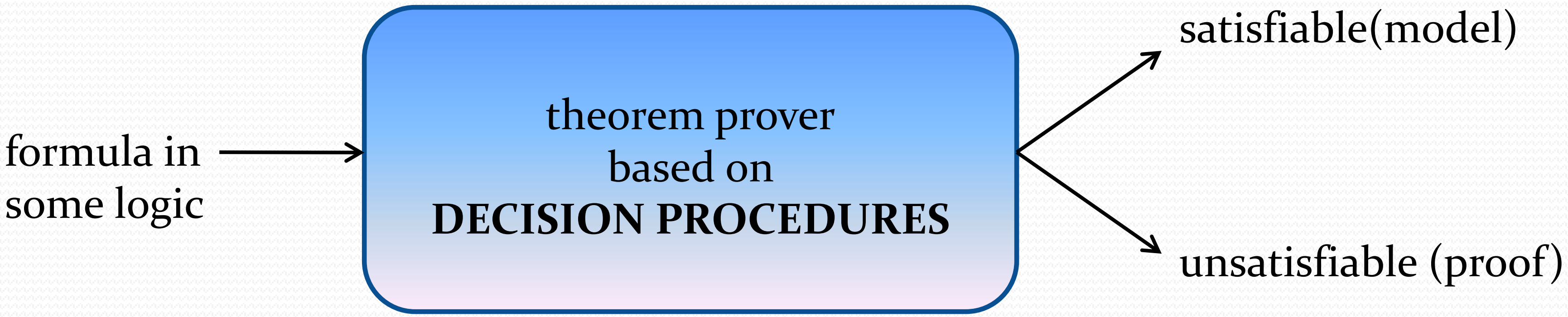
- Given a set of types *T* and a set of expressions *E*, a type environment is a set

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

**Type Inhabitation Problem**

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

# Automated Reasoning




- Comfusy – uses a model to for code extraction
- InSynth – extracts code from a proof of unsatisfiability

# From Code Synthesis to Code Repair

```
src@pldi:Repair$ javac DeflaterExample.java
DeflaterExample.java:15: error: no suitable constructor found for FileInputStream(no arguments)
        BufferedInputStream bis = new BufferedInputStream(buffSize,
          new DeflaterInputStream(new FileInputStream(), compLevel, true));

    constructor FileInputStream.FileInputStream(String) is not applicable
      (actual and formal argument lists differ in length)
    constructor FileInputStream.FileInputStream(File) is not applicable
      (actual and formal argument lists differ in length)
    constructor FileInputStream.FileInputStream(FileDescriptor) is not applicable
      (actual and formal argument lists differ in length)

Winston: info: Error detected! Attempting to repair...
Winston: info: try one of these:
    [0.90] BufferedInputStream bis = new BufferedInputStream(
               new DeflaterInputStream(new FileInputStream(FILE)), buffSize);
    [0.60] BufferedInputStream bis = new BufferedInputStream(
               new DeflaterInputStream(new FileInputStream(FILE),
                   new Deflater(compLevel, true)), buffSize);
1 error
src@pldi:Repair$
```

# From Code Synthesis to Code Repair

```
src@pldi:Repair$ javac DeflaterExample.java
DeflaterExample.java:15: error: no suitable constructor found for FileInputStream(no arguments)
        BufferedInputStream bis = new BufferedInputStream(buffSize,
          new DeflaterInputStream(new FileInputStream(), compLevel, true));

    constructor FileInputStream.FileInputStream(String) is not applicable
      (actual and formal argument lists differ in length)
    constructor FileInputStream.FileInputStream(File) is not applicable
      (actual and formal argument lists differ in length)
    constructor FileInputStream.FileInputStream(FileDescriptor) is not applicable
      (actual and formal argument lists differ in length)

Winston: info: Error detected! Attempting to repair...
Winston: info: try one of these:
    [0.90] BufferedInputStream bis = new BufferedInputStream(
               new DeflaterInputStream(new FileInputStream(FILE)), buffSize);
    [0.60] BufferedInputStream bis = new BufferedInputStream(
               new DeflaterInputStream(new FileInputStream(FILE),
                   new Deflater(compLevel, true)), buffSize);
1 error
src@pldi:Repair$
```

# From Code Synthesis to Code Repair

```
src@pldi:Repair$ javac DeflaterExample.java
DeflaterExample.java:15: error: no suitable constructor found for FileInputStream(no arguments)
        BufferedInputStream bis = new BufferedInputStream(buffSize,
          new DeflaterInputStream(new FileInputStream(), compLevel, true));

    constructor FileInputStream.FileInputStream(String) is not applicable
      (actual and formal argument lists differ in length)
    constructor FileInputStream.FileInputStream(File) is not applicable
      (actual and formal argument lists differ in length)
    constructor FileInputStream.FileInputStream(FileDescriptor) is not applicable
      (actual and formal argument lists differ in length)
```

```
Winston: info: Error detected! Attempting to repair...
Winston: info: try one of these:
    [0.90] BufferedInputStream bis = new BufferedInputStream(
               new DeflaterInputStream(new FileInputStream(FILE)), buffSize);
    [0.60] BufferedInputStream bis = new BufferedInputStream(
               new DeflaterInputStream(new FileInputStream(FILE),
                   new Deflater(compLevel, true)), buffSize);
1 error
src@pldi:Repair$
```

# Winston – Repair tool

- Based on type constraints
- A new data structure: synthesis graph, used to encode those constraints
- Edges have different weight based on the frequencies of code snippets
- Synthesis is a repair of the empty expression
- Extremely fast: synthesis is done within a couple of milliseconds, repair below half of the second

# Programming by Example

- Sometimes it is harder to write a specification or even to describe what the program should do
- A few representative examples can easy convey user's intentions

[1, 5, 2 ] ➔ [1, 2, 5]
[t, i, m, i, s, o, a, r, a] ➔ [a, a, i, i, m, o, r, s, t]

$$sorted\ (l, l') \equiv \forall i, j.\, i < j \rightarrow l'[i] \leq l'[j]$$

# Programming by Example

- Sometimes it is harder to write a specification or even to describe what the program should do
- A few representative examples can easy convey user's intentions

[1, 5, 2 ] ➔ [1, 2, 5]
[t, i, m, i, s, o, a, r, a] ➔ [a, a, i, i, m, o, r, s, t]

$sorted\ (l, l') \ \equiv \ \forall i, j. i \ < j \ \rightarrow l'[i] \leq l'[j] \ \wedge$ isPermutation(l, l')

```
sed/\(^[a-zA-Z0-9]+\)\.\([a-z]+\)/\<a href\=\"\1\.\2\" \>\1\<\/a\>/g
```

1

To make

```
SomeDocument1.docx
SomeDocument2.docx
```

To

```
<A HREF="SomeDocument1.docx" >SomeDocument1</A>
<A HREF="SomeDocument2.docx" >SomeDocument2</A>
```

Find

```
\(^[a-zA-Z0-9]+\)\.\([a-z]+\)
```

Replace with

```
\<A HREF\=\"\1\.\2\" \>\1\<\/A\>
```

Figuring out the `\(` and `\)` issue with Textpad for capture took time :)

share improve this answer

answered Apr 29 '09 at 4:51

Binoj Antony

8,634 ● 18 ● 74 ● 88

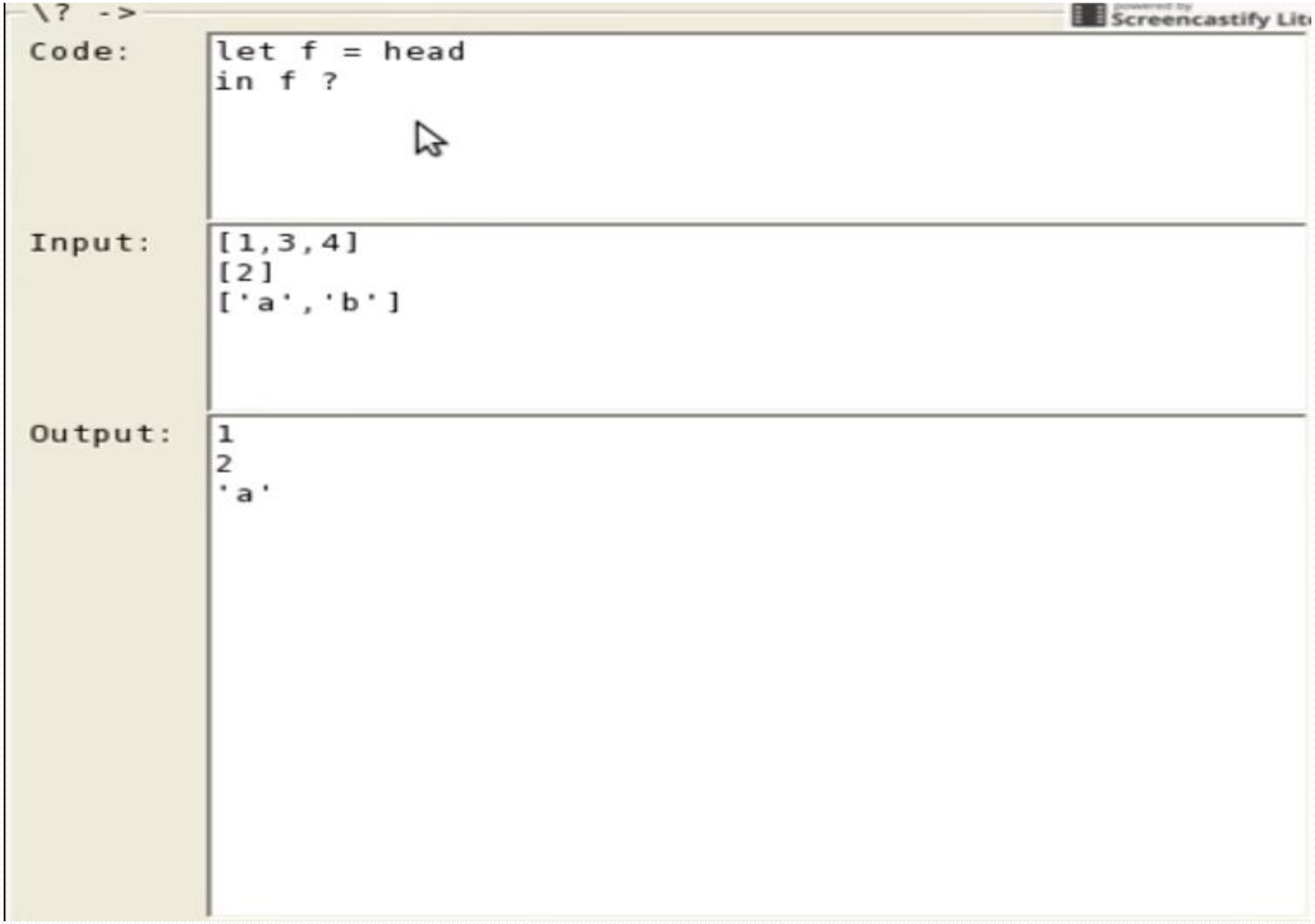
# Programming by Example

- Linkify

  test.doc ==> <a href=”test.doc”>test</a>

- We developed a tool that automatically generates scripts based on input/output examples
- Our tool supports other operations (besides the mapping): reduce, filter, partition
- We needed to extend the existing algorithms to support reasoning about counters

# Live Programming Environment



# Future Directions: Cooperative Programming

- Integrating Software Synthesis with the Live Paradigm
- Increasing Programmers Productivity
- Automating hard and complex task
- Goal: more reliable software

# Conclusions

Software Synthesis

- method to obtain *correct* software from the given specification
- Complete Functional Synthesis (Comfusy): extending decision procedures into synthesis algorithms
- Software synthesis can be used in various domains: for code completion, for code correction, for improving the programming experience in general