Searching for Source Code with Constrained Semantic Search

Kathryn T. Stolee
Iowa State University
kstolee@iastate.edu

November 19, 2013

Programmers are Resourceful

Code Search is Common!

Programmers often search for code

89% of programmers search for code frequently or occasionally

[Sim et al. 2012]

85% search for code *at least weekly*

[Stolee et al., under review]

Programmers *use* the code they find

- copy/paste and modify (48%)
- reference example (67%)

[Stolee et al., under review]

State-of-the-Practice

State-of-the-Practice

Goal

extract alias from e-mail address in Java

State-of-the-Practice

Goal

extract alias from e-mail address in Java

State-of-the-Practice

Goal

extract alias from e-mail address in Java

Search

```
get alias from e-mail
address in Java
```

Results

```
public EmailAlias(String aliasName, String emailAddress) {
    this.aliasName = aliasName;
    this.emailAddress = emailAddress;
}
```

```
public boolean checkifAlias(String alias, String email){
    if(email.contains(alias) == true){
        return true;
    }else {
        return false;
    }
}
```

```
public static String validate(String addr) {
    Pattern pattern = Pattern
            .compile("^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
                    + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$");
    Matcher matcher = pattern.matcher(addr);
    if (matcher.matches()) {
        addr = addr.replaceAll(
                "@[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$", "");
        addr = addr.trim();
        return addr;
    } else {
        return addr;
    }
}
```

```
public String getFiveCharacterAlias(String name) {
    return name.substring(0, 5);
}
```

State-of-the-Practice

Goal

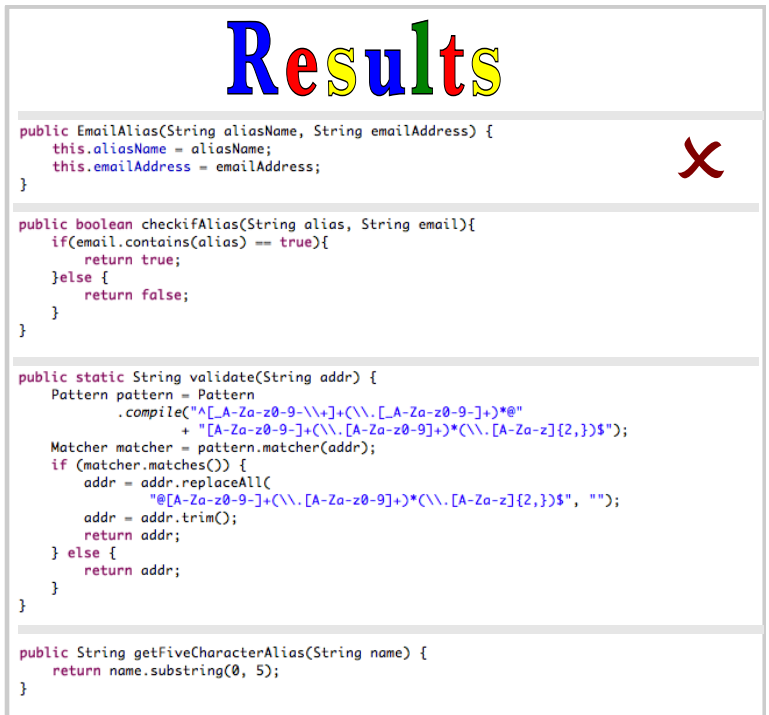
extract alias from e-mail address in Java


```
public EmailAlias(String aliasName, String emailAddress) {
    this.aliasName = aliasName;
    this.emailAddress = emailAddress;
}
```

State-of-the-Practice

Goal

extract alias from e-mail address in Java



State-of-the-Practice

Goal

extract alias from e-mail address in Java


```
public boolean checkifAlias(String alias, String email){
    if(email.contains(alias) == true){
        return true;
    }else {
        return false;
    }
}
```

State-of-the-Practice

Goal

extract alias from e-mail address in Java

State-of-the-Practice

Goal

extract alias from e-mail address in Java


```java
public static String validate(String addr) {
    Pattern pattern = Pattern
            .compile("^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
                    + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$");
    Matcher matcher = pattern.matcher(addr);
    if (matcher.matches()) {
        addr = addr.replaceAll(
                "@[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$", "");
        addr = addr.trim();
        return addr;
    } else {
        return addr;
    }
}
```

State-of-the-Practice

Goal

extract alias from e-mail address in Java

Search

get alias from e-mail address in Java

Results

```
public EmailAlias(String aliasName, String emailAddress) {
    this.aliasName = aliasName;
    this.emailAddress = emailAddress;
}
```

✗

```
public boolean checkifAlias(String alias, String email){
    if(email.contains(alias) == true){
        return true;
    }else {
        return false;
    }
}
```

✗

```
public static String validate(String addr) {
    Pattern pattern = Pattern
            .compile("^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
                    + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$");
    Matcher matcher = pattern.matcher(addr);
    if (matcher.matches()) {
        addr = addr.replaceAll(
                "@[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$", "");
        addr = addr.trim();
        return addr;
    } else {
        return addr;
    }
}
```

✓

```
public String getFiveCharacterAlias(String name) {
    return name.substring(0, 5);
}
```

State-of-the-Practice

Goal

extract alias from e-mail address in Java


```
public String getFiveCharacterAlias(String name) {
    return name.substring(0, 5);
}
```

State-of-the-Practice

Goal

extract alias from e-mail address in Java

Search

```
get alias from e-mail
address in Java
```

Results

```
public EmailAlias(String aliasName, String emailAddress) {
    this.aliasName = aliasName;
    this.emailAddress = emailAddress;
}
```

✗

```
public boolean checkifAlias(String alias, String email){
    if(email.contains(alias) == true){
        return true;
    }else {
        return false;
    }
}
```

✗

```
public static String validate(String addr) {
    Pattern pattern = Pattern
            .compile("^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
                    + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$");
    Matcher matcher = pattern.matcher(addr);
    if (matcher.matches()) {
        addr = addr.replaceAll(
                "@[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$", "");
        addr = addr.trim();
        return addr;
    } else {
        return addr;
    }
}
```

✓

```
public String getFiveCharacterAlias(String name) {
    return name.substring(0, 5);
}
```

?

\+ Easy to compose a query

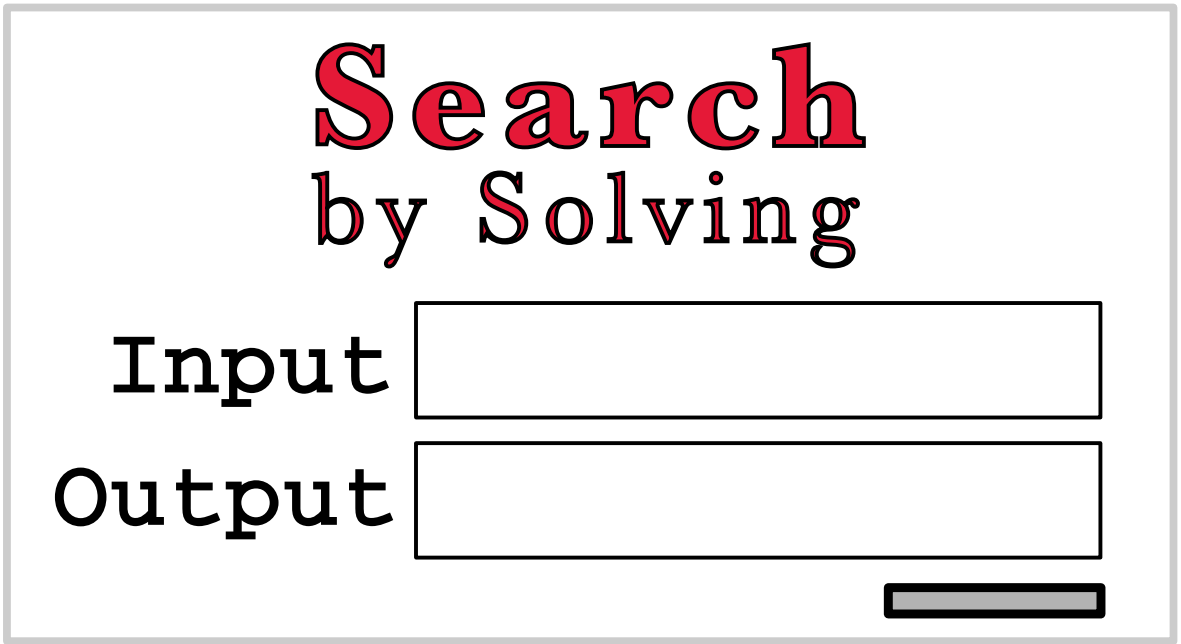
\- Many irrelevant results

Programmers are Resourceful

Programmers are Resourceful

...but the resources can be better

Semantic Approach



Semantic Approach

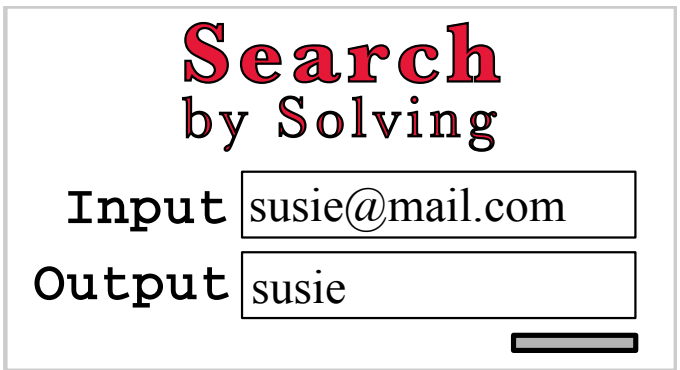
Goal

extract alias from e-mail address in Java

Semantic Approach

Goal

extract alias from e-mail address in Java



Semantic Approach

Goal

extract alias from e-mail address in Java

Search
by Solving

Input: susie@mail.com

Output: susie

Results

```
public String getAlias(String addr) {
    if (addr.indexOf('@') >= 0) {
        return addr.substring(0, addr.indexOf('@'));
    } else {
        return "";
    }
}
```

```
protected String returnHead(int limit, String str) {
    return str.substring(0, limit);
}
```

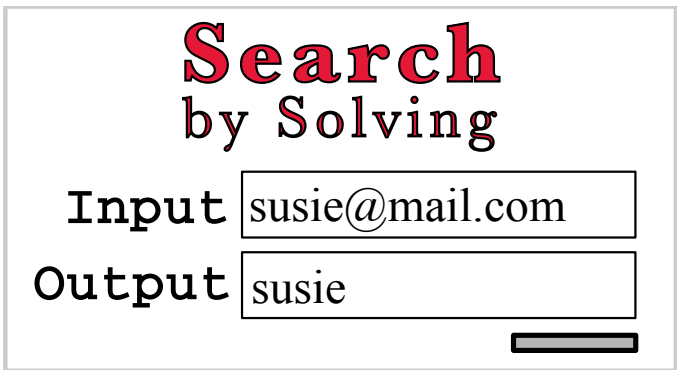
```
static String buildAliasLowercase(String email) {
    int index = 0;
    Character c;
    String alias = "";
    while (index < email.length()) {
        c = email.charAt(index);
        if (c == '@') {
            break;
        }
        alias = alias + c.toString().toLowerCase();
        index++;
    }
    return alias;
}
```

```
String untilChar(String asTyped, char special) {
    int idx1 = asTyped.lastIndexOf(special);
    String until = asTyped.substring(0, idx1);
    return until;
}
```

Semantic Approach

Goal

extract alias from e-mail address in Java




```
public String getAlias(String addr) {
    if (addr.indexOf('@') >= 0) {
        return addr.substring(0, addr.indexOf('@'));
    } else {
        return "";
    }
}
```

Semantic Approach

Goal

extract alias from e-mail address in Java

Search
by Solving

Input: susie@mail.com

Output: susie

Results

```
public String getAlias(String addr) {
    if (addr.indexOf('@') >= 0) {
        return addr.substring(0, addr.indexOf('@'));
    } else {
        return "";
    }
}
```

```
protected String returnHead(int limit, String str) {
    return str.substring(0, limit);
}
```

```
static String buildAliasLowercase(String email) {
    int index = 0;
    Character c;
    String alias = "";
    while (index < email.length()) {
        c = email.charAt(index);
        if (c == '@') {
            break;
        }
        alias = alias + c.toString().toLowerCase();
        index++;
    }
    return alias;
}
```

```
String untilChar(String asTyped, char special) {
    int idx1 = asTyped.lastIndexOf(special);
    String until = asTyped.substring(0, idx1);
    return until;
}
```

Semantic Approach

Goal

extract alias from e-mail address in Java

Search
by Solving

Input: susie@mail.com

Output: susie

Results

```
public String getAlias(String addr) {
    if (addr.indexOf('@') >= 0) {
        return addr.substring(0, addr.indexOf('@'));
    } else {
        return "";
    }
}
```

✓

```
protected String returnHead(int limit, String str) {
    return str.substring(0, limit);
}
```

✓

```
static String buildAliasLowercase(String email) {
    int index = 0;
    Character c;
    String alias = "";
    while (index < email.length()) {
        c = email.charAt(index);
        if (c == '@') {
            break;
        }
        alias = alias + c.toString().toLowerCase();
        index++;
    }
    return alias;
}
```

✓

```
String untilChar(String asTyped, char special) {
    int idx1 = asTyped.lastIndexOf(special);
    String until = asTyped.substring(0, idx1);
    return until;
}
```

✓

Semantic Approach

Goal

extract alias from e-mail address in Java

Search by Solving

Input: susie@mail.com

Output: susie

Results

```
public String getAlias(String addr) {
    if (addr.indexOf('@') >= 0) {
        return addr.substring(0, addr.indexOf('@'));
    } else {
        return "";
    }
}
```

✓

```
protected String returnHead(int limit, String str) {
    return str.substring(0, limit);
}
```

✓

```
static String buildAliasLowercase(String email) {
    int index = 0;
    Character c;
    String alias = "";
    while (index < email.length()) {
        c = email.charAt(index);
        if (c == '@') {
            break;
        }
        alias = alias + c.toString().toLowerCase();
        index++;
    }
    return alias;
}
```

✓

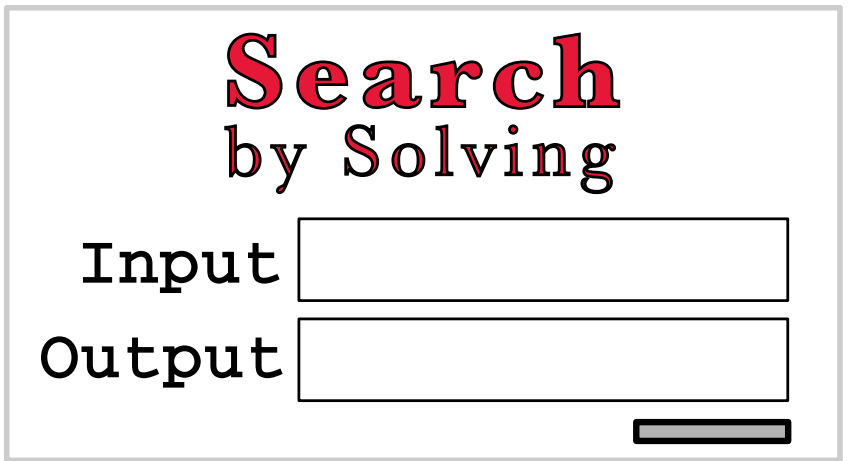
```
String untilChar(String asTyped, char special) {
    int idx1 = asTyped.lastIndexOf(special);
    String until = asTyped.substring(0, idx1);
    return until;
}
```

✓

\+ All results match example behavior

\- More expensive (structured) query

Comparison



+ Easy to compose a query
- Many irrelevant results

- More expensive query
+ All results match example behavior

Definitions

Query

A description, or specification, of what you want from the search. Queries can have various forms, e.g.:

- Keyword: “extract alias from email address in Java”
- Input/Output: “susie@mail.com”, “susie”

Definitions

Query

A description, or specification, of what you want from the search. Queries can have various forms, e.g.:

- Keyword: “extract alias from email address in Java”
- Input/Output: “susie@mail.com”, “susie”

Repository

The set of documents being searched over. In this work, these are source code snippets.

Definitions

Query

A description, or specification, of what you want from the search. Queries can have various forms, e.g.:

- Keyword: “extract alias from email address in Java”
- Input/Output: “susie@mail.com”, “susie”

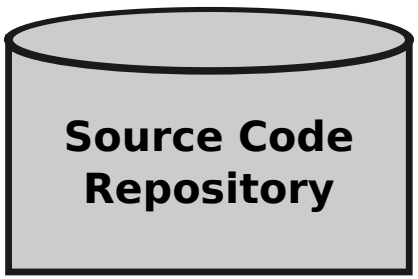
Repository

The set of documents being searched over. In this work, these are source code snippets.

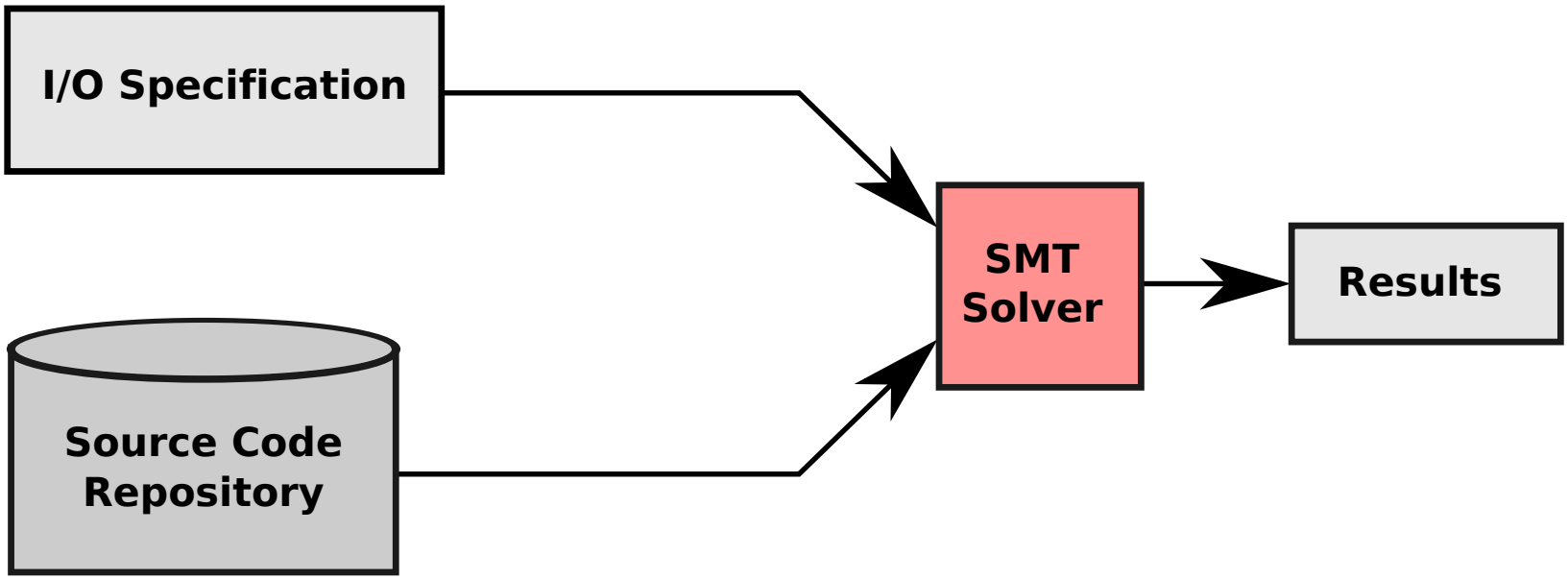
Result/Match

Source code, from a repository, that matches the query.

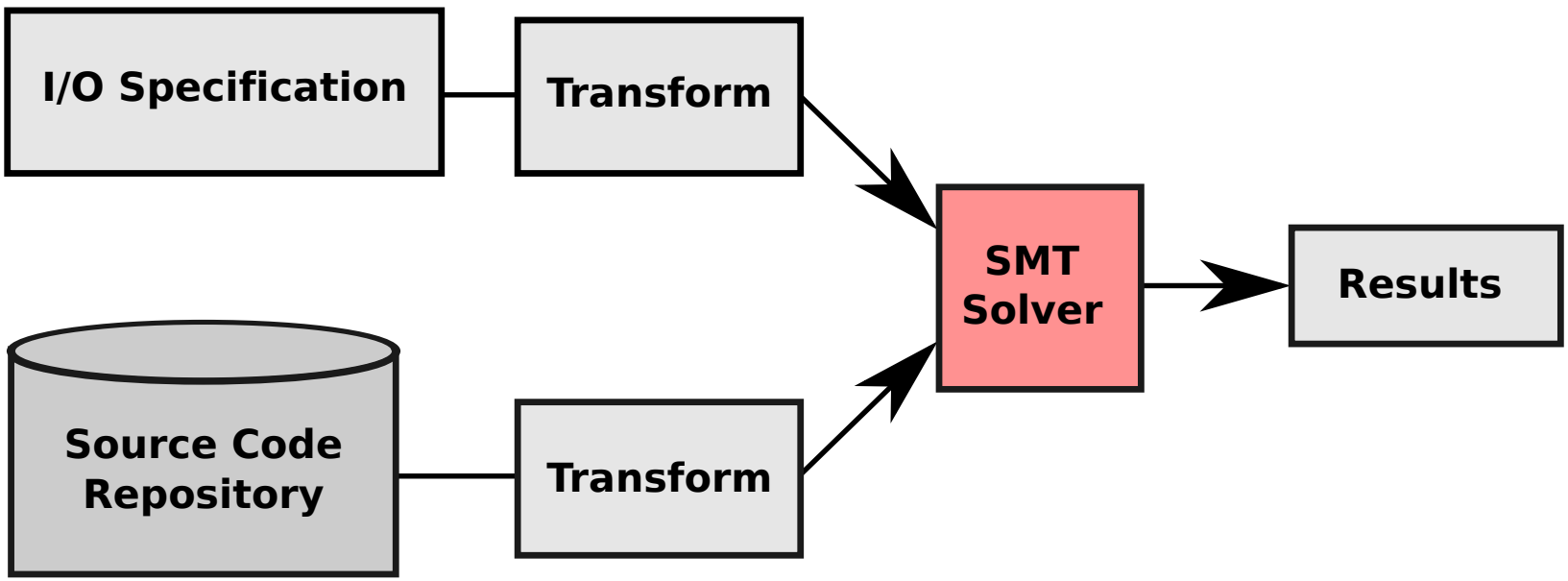
General Approach



General Approach



General Approach



Introduction to SMT Solvers

SMT Solvers

Satisfiability **M**odulo **T**heory Solvers determine if a logical formula is satisfiable

Introduction to SMT Solvers

SMT Solvers

Satisfiability **M**odulo **T**heory Solvers determine if a logical formula is satisfiable

Fact	**Constraint**
$a \geq 0$	`(assert (≥ a 0))`
$b = 2$	`(assert (= b 2))`
$c = 2$	`(assert (= c 2))`
$c = a * b$	`(assert (= (* a b) c))`

Introduction to SMT Solvers

SMT Solvers

Satisfiability **M**odulo **T**heory Solvers determine if a logical formula is satisfiable

Fact	**Constraint**
$a \geq 0$	`(assert (≥ a 0))`
$b = 2$	`(assert (= b 2))`
$c = 2$	`(assert (= c 2))`
$c = a * b$	`(assert (= (* a b) c))`

Result: **sat**, `a` $\mapsto$ `1`

Introduction to SMT Solvers

SMT Solvers

Satisfiability **M**odulo **T**heory Solvers determine if a logical formula is satisfiable

Fact	**Constraint**
$a \geq 0$	`(assert (≥ a 0))`
$b = ?$	`(assert (= b ?))`
$c = 2$	`(assert (= c 2))`
$c = a * b$	`(assert (= (* a b) c))`

Introduction to SMT Solvers

SMT Solvers

Satisfiability **M**odulo **T**heory Solvers determine if a logical formula is satisfiable

Fact	**Constraint**
$a \geq 0$	`(assert (≥ a 0))`
$b = ?$	`(assert (= b ?))`
$c = 2$	`(assert (= c 2))`
$c = a * b$	`(assert (= (* a b) c))`

Result: **sat**, `a ↦ 1 ∧ b ↦ 2`

Introduction to SMT Solvers

SMT Solvers

Satisfiability **M**odulo **T**heory Solvers determine if a logical formula is satisfiable

Fact	**Constraint**
$a \geq 0$	`(assert (≥ a 0))`
$b = ?$	`(assert (= b ?))`
$c = 2$	`(assert (= c 2))`
$c = a * b$	`(assert (= (* a b) c))`

Result: **sat**, `a` $\mapsto$ `2` $\wedge$ `b` $\mapsto$ `1`

Introduction to SMT Solvers

SMT Solvers

Satisfiability **M**odulo **T**heory Solvers determine if a logical formula is satisfiable

Fact	**Constraint**
$a = 0$	`(assert (= a 0))`
$b = ?$	`(assert (= b ?))`
$c = 2$	`(assert (= c 2))`
$c = a * b$	`(assert (= (* a b) c))`

Introduction to SMT Solvers

SMT Solvers

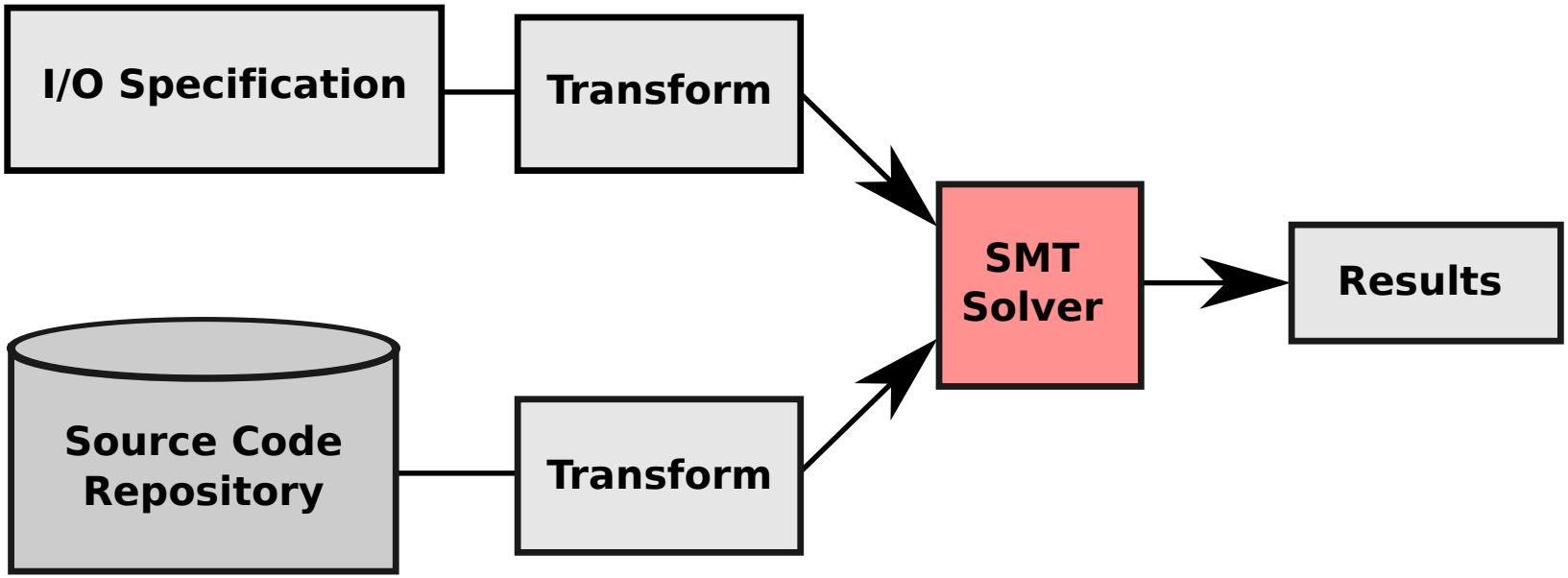
Satisfiability **M**odulo **T**heory Solvers determine if a logical formula is satisfiable

Fact	**Constraint**
$a = 0$	`(assert (= a 0))`
$b = ?$	`(assert (= b ?))`
$c = 2$	`(assert (= c 2))`
$c = a * b$	`(assert (= (* a b) c))`

Result: **unsat**

There does not exist a solution such that `0 * b = 2`.

Introduction to SMT Solvers



Related Work

Code Search

	Query	Matching
Satsy	I/O example	SMT solver
State of the Practice	keyword	syntax

Related Work

Code Search

	Query	Matching
Satsy	I/O example	SMT solver
State of the Practice	keyword	syntax
[Hill 2009] [Grechanik 2010]	keyword	NLP

Related Work

Code Search

	Query	Matching
Satsy	I/O example	SMT solver
State of the Practice	keyword	syntax
[Hill 2009] [Grechanik 2010]	keyword	NLP
[Zaremski 1997] [Penix 1999]	formal specification	theorem prover

Related Work

Code Search

	Query	Matching
Satsy	I/O example	SMT solver
State of the Practice	keyword	syntax
[Hill 2009] [Grechanik 2010]	keyword	NLP
[Zaremski 1997] [Penix 1999]	formal specification	theorem prover
[Podgurski 1993] [Lemos 2007] [Reiss 2009]	test cases	execute code

Related Work

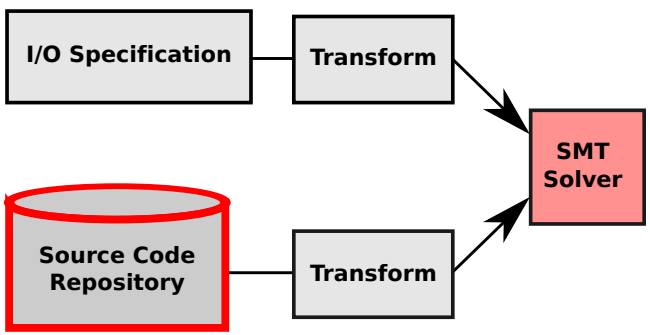
Code Search

	Query	Matching
Satsy	I/O example	SMT solver
State of the Practice	keyword	syntax
[Hill 2009] [Grechanik 2010]	keyword	NLP
[Zaremski 1997] [Penix 1999]	formal specification	theorem prover
[Podgurski 1993] [Lemos 2007] [Reiss 2009]	test cases	execute code

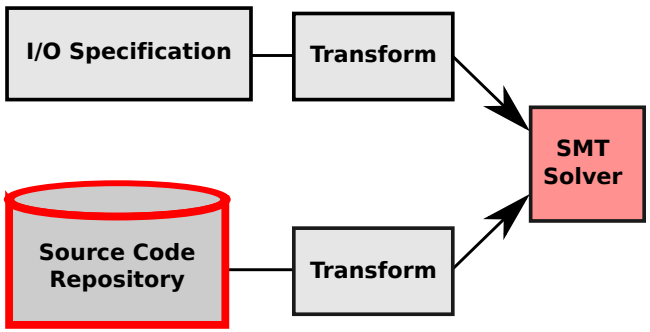
Other Areas

code reuse, symbolic execution, program synthesis

Repository Encoding



Repository Encoding



$RepP = \{P_1, P_2, P_3, P_4, P_5\}$

P_1.
```
String scheme = uri.substring(0, 5);
```

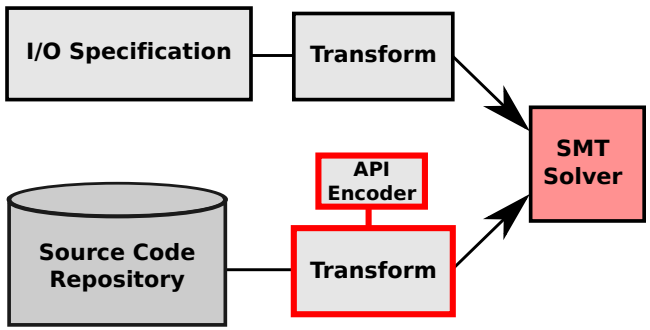
P_2.
```
return probando.substring(0, corte);
```

P_3.
```
int index = token.indexOf("://");
String label = token.substring(0,
    index);
```

P_4.
```
boolean sfx(String as, String bs) {
    return as.endsWith(bs);
}
```

P_5.
```
String getname(String addr) {
    int i = addr.indexOf("@");
    if(i > -1) {
        return addr.substring(0, i);
    }
    return "NoOne";
}
```

Repository Encoding



$RepP = \{P_1, P_2, P_3, P_4, P_5\}$

P_1.
```
String scheme = uri.substring(0, 5);
```

P_2.
```
return probando.substring(0, corte);
```

P_3.
```
int index = token.indexOf("://");
String label = token.substring(0,
    index);
```

P_4.
```
boolean sfx(String as, String bs) {
    return as.endsWith(bs);
}
```

P_5.
```
String getname(String addr) {
    int i = addr.indexOf("@");
    if(i > -1) {
        return addr.substring(0, i);
    }
    return "NoOne";
}
```

API Encoder, Java Preview

$s.substring(i,j) = sub$

ensures s between i and j equals sub	$(\forall k$ $(((k \geq i) \wedge (k < j))$ $\rightarrow (charAt(s,k) = charAt(sub,(k-i))))$
sets bounds on i, j	$(i \geq 0) \wedge (i < j) \wedge (j < length(s))$
sets constraints on string sizes	$(length(s) \geq length(sub))$ $\wedge((j-i) = length(sub))$

Encoding a Program

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

Encoding a Program

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

1. Declare the Variables

```
c1.  (declare-fun uri () String)
c2.  (declare-fun scheme () String)
c3.  (declare-fun resRHS () String)
c4.  (declare-fun upper () Int)
c5.  (declare-fun lower () Int)
```

Encoding a Program

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

```
( (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
```

1. Declare the Variables

Encoding a Program

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
```

1. Declare the Variables

2. Instantiate the Variables

```
c6.   (= lower 0)
c7.   (= upper 5)
c8.   (= scheme resRHS)
```

Encoding a Program

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= upper 5)
∧ (= scheme resRHS)
```

1. Declare the Variables
2. Instantiate the Variables

Encoding a Program

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= upper 5)
∧ (= scheme resRHS)
```

1. Declare the Variables

2. Instantiate the Variables

3. Set constraints on bounds

```
c9.   (<= lower upper)
c10.   (<= upper (length uri))
c11.   (= (- upper lower)(length resRHS))
```

Encoding a Program

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= upper 5)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
```

1. Declare the Variables
2. Instantiate the Variables
3. Set constraints on bounds

Encoding a Program

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= upper 5)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
```

1. Declare the Variables

2. Instantiate the Variables

3. Set constraints on bounds

4. Substring Logic

```
c12.  (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower)))))
```

Encoding a Program

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= upper 5)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
∧ (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower))))) )
```

1. Declare the Variables
2. Instantiate the Variables
3. Set constraints on bounds
4. Substring Logic

Encoding a Program

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

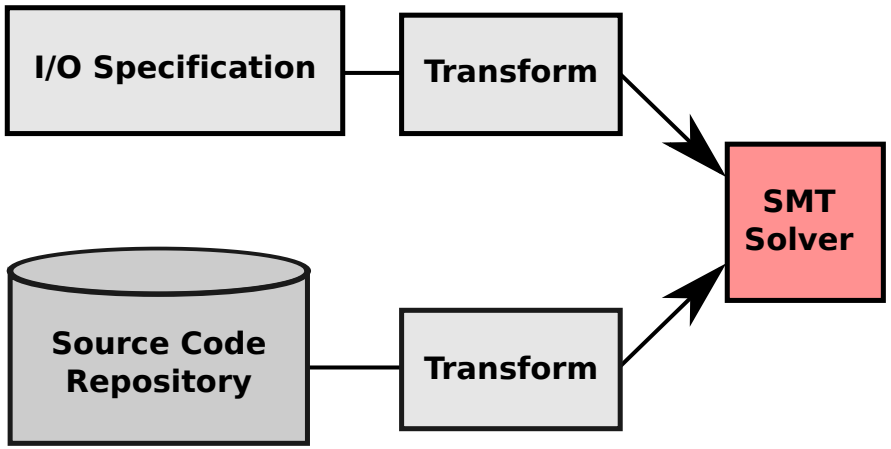
```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= upper 5)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
∧ (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower))))) )
```

1. Declare the Variables
2. Instantiate the Variables
3. Set constraints on bounds
4. Substring Logic

Repeat encoding $\forall P_i \in RepP$

$$RepP_{enc} = \{C_{P_1}, C_{P_2}, C_{P_3}, C_{P_4}, C_{P_5}\}$$

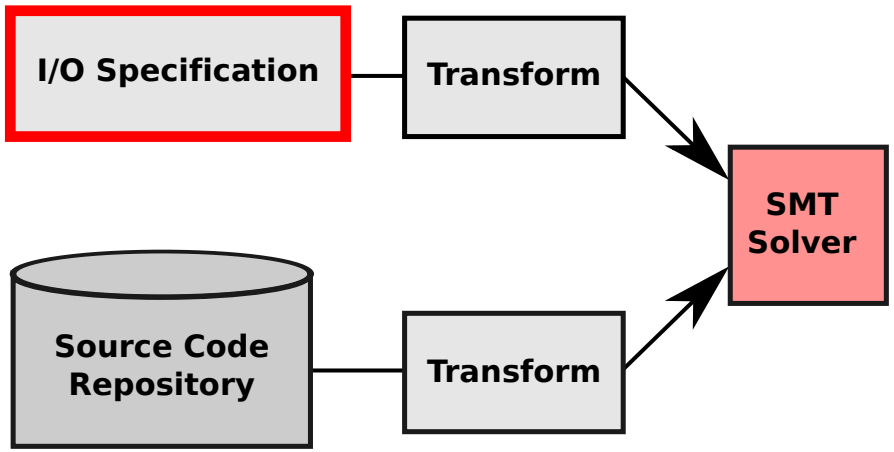
Lightweight Specifications



Goal

extract alias from e-mail address in Java

Lightweight Specifications



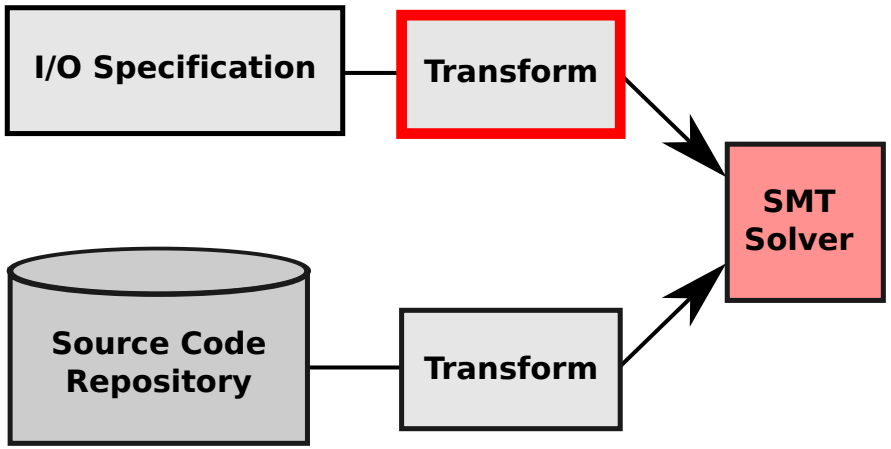
Goal

extract alias from e-mail address in Java

$LS =$

$\{(\textbf{susie@mail.com}, \textbf{susie})\}$

Lightweight Specifications



Goal

extract alias from e-mail address in Java

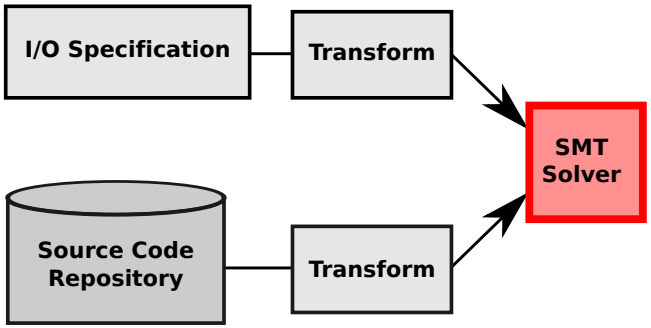
$LS =$

$\{(\textbf{susie@mail.com}, \textbf{susie})\}$

$C_{LS} =$

```
c1.  (declare-fun input () String)
c2.  (= input "susie@mail.com")
c3.  (= (length input) 14)
c4.  (declare-fun output () String)
c5.  (= output "susie")
c6.  (= (length output) 5)
```

Solving



RepP =

P_1.
```
String scheme = uri.substring(0, 5);
```

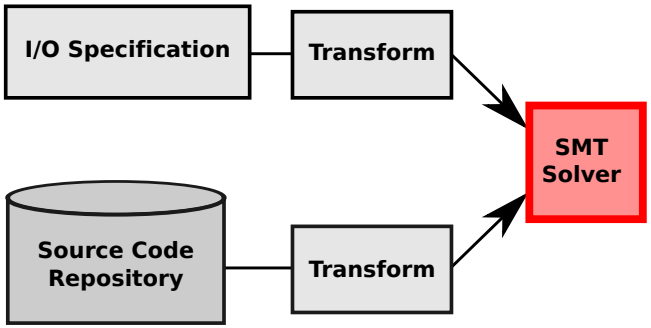
P_2.
```
return probando.substring(0, corte);
```

P_3.
```
int index = token.indexOf("://");
String label = token.substring(0, index);
```

P_4.
```
boolean sfx(String as, String bs) {
    return as.endsWith(bs);
}
```

P_5.
```
String getname(String addr) {
    int i = addr.indexOf("@");
    if(i > -1) {
        return addr.substring(0, i);
    }
    return "NoOne";
}
```

Solving



SMT Solver

$$Solve(C_P \wedge C_{LS}) =$$

$RepP =$

P_1.
```
String scheme = uri.substring(0, 5);
```

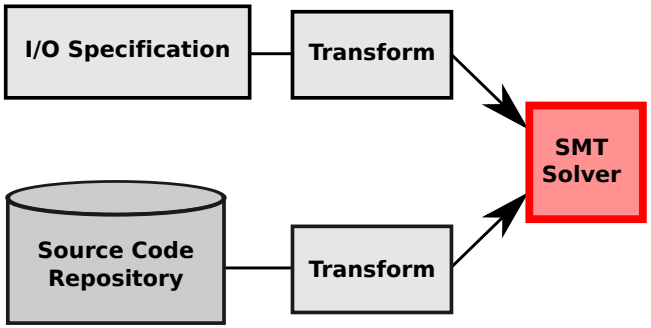
P_2.
```
return probando.substring(0, corte);
```

P_3.
```
int index = token.indexOf("://");
String label = token.substring(0, index);
```

P_4.
```
boolean sfx(String as, String bs) {
    return as.endsWith(bs);
}
```

P_5.
```
String getname(String addr) {
    int i = addr.indexOf("@");
    if(i > -1) {
        return addr.substring(0, i);
    }
    return "NoOne";
}
```

Solving



SMT Solver

$$Solve(C_P \wedge C_{LS}) =$$

$RepP =$

```
P1. String scheme = uri.substring(0, 5);
```

```
P2. return probando.substring(0, corte);
```

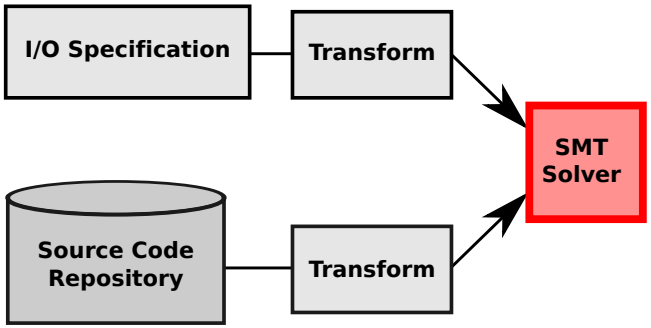
```
P3. int index = token.indexOf("://");
    String label = token.substring(0, index);
```

```
P4. boolean sfx(String as, String bs) {
      return as.endsWith(bs);
    }
```

```
P5. String getname(String addr) {
      int i = addr.indexOf("@");
      if(i > -1) {
         return addr.substring(0, i);
      }
      return "NoOne";
    }
```

$SatP =$

Solving



SMT Solver

$$Solve(C_{P_1} \wedge C_{LS}) =$$

$RepP =$

P_1.
```
String scheme = uri.substring(0, 5);
```

P_2.
```
return probando.substring(0, corte);
```

P_3.
```
int index = token.indexOf("://");
String label = token.substring(0, index);
```

P_4.
```
boolean sfx(String as, String bs) {
    return as.endsWith(bs);
}
```

P_5.
```
String getname(String addr) {
    int i = addr.indexOf("@");
    if(i > -1) {
        return addr.substring(0, i);
    }
    return "NoOne";
}
```

$SatP =$

Solving

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= upper 5)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
∧ (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower))))) )
```

C_{LS} =

```
(  (declare-fun input () String)
∧ (= input "susie@mail.com")
∧ (= (length input) 14)
∧ (declare-fun output () String)
∧ (= output "susie")
∧ (= (length output) 5)
```

Solving

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

```
( (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= upper 5)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
∧ (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower))))) )
```

C_{LS} =

```
( (declare-fun input () String)
∧ (= input "susie@mail.com")
∧ (= (length input) 14)
∧ (declare-fun output () String)
∧ (= output "susie")
∧ (= (length output) 5)
```

Map Input/Output to Snippet

```
c1.   (= input uri)
c2.   (= output scheme)
```

Solving

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= upper 5)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
∧ (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower))))) )
```

C_{LS} =

```
(  (declare-fun input () String)
∧ (= input "susie@mail.com")
∧ (= (length input) 14)
∧ (declare-fun output () String)
∧ (= output "susie")
∧ (= (length output) 5)
∧ (= input uri)
∧ (= output scheme) )
```

Solving

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= upper 5)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
∧ (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower))))) )
```

C_{LS} =

```
(  (declare-fun input () String)
∧ (= input "susie@mail.com")
∧ (= (length input) 14)
∧ (declare-fun output () String)
∧ (= output "susie")
∧ (= (length output) 5)
∧ (= input uri)
∧ (= output scheme) )
```

SMT Solver

$Solve(C_{P_1} \wedge C_{LS}) =$

Solving

P_1 =

```
String scheme = uri.substring(0, 5);
```

C_{P_1} =

```
( (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= upper 5)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
∧ (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower))))) )
```

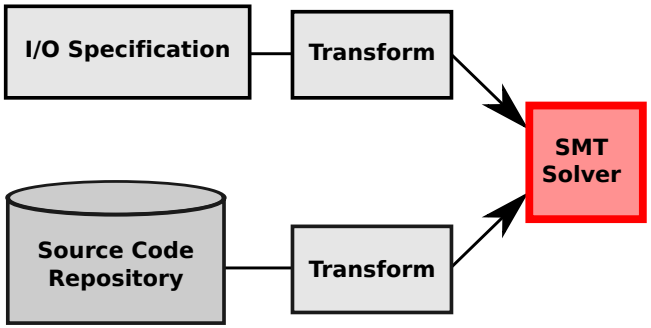
C_{LS} =

```
( (declare-fun input () String)
∧ (= input "susie@mail.com")
∧ (= (length input) 14)
∧ (declare-fun output () String)
∧ (= output "susie")
∧ (= (length output) 5)
∧ (= input uri)
∧ (= output scheme) )
```

SMT Solver

$Solve(C_{P_1} \wedge C_{LS}) =$ **sat**

Solving



SMT Solver

$$Solve(C_{P_1} \wedge C_{ls}) = \textbf{sat}$$

$RepP =$

```
P2. return probando.substring(0, corte);
```

```
P3. int index = token.indexOf("://");
    String label = token.substring(0, index);
```

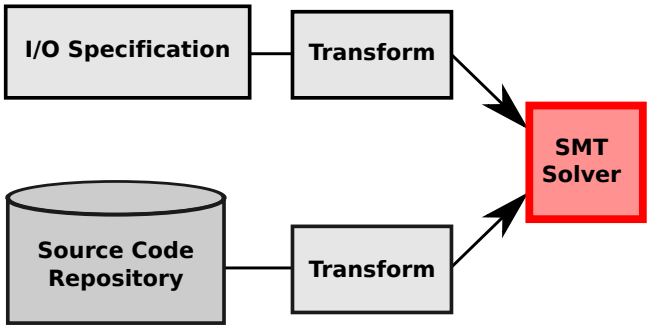
```
P4. boolean sfx(String as, String bs) {
      return as.endsWith(bs);
    }
```

```
P5. String getname(String addr) {
      int i = addr.indexOf("@");
      if(i > -1) {
         return addr.substring(0, i);
      }
      return "NoOne";
    }
```

$SatP =$

```
P1. String scheme = uri.substring(0, 5);
```

Solving



SMT Solver

$$Solve(C_{P_2} \wedge C_{ls}) =$$

$RepP =$

```
P2. return probando.substring(0, corte);
```

```
P3. int index = token.indexOf("://");
    String label = token.substring(0, index);
```

```
P4. boolean sfx(String as, String bs) {
      return as.endsWith(bs);
    }
```

```
P5. String getname(String addr) {
      int i = addr.indexOf("@");
      if(i > -1) {
         return addr.substring(0, i);
      }
      return "NoOne";
    }
```

$SatP =$

```
P1. String scheme = uri.substring(0, 5);
```

Solving

P_2 =

```
return probando.substring(0, corte);
```

C_{P_2} =

```
(  (declare-fun probando () String)
∧ (declare-fun return () String)
∧ (declare-fun result () String)
∧ (declare-fun corte () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= return result)
∧ (<= lower corte)
∧ (<= corte (length probando))
∧ (= (- corte lower)(length result))
∧ (forall ((index Int))
  (=>
    (and (< index corte)
      (>= index lower))
    (= (charOf probando index)
      (charOf result(- index lower))))) )
```

C_{LS} =

```
(  (declare-fun input () String)
∧ (= input "susie@mail.com")
∧ (= (length input) 14)
∧ (declare-fun output () String)
∧ (= output "susie")
∧ (= (length output) 5)
```

Solving

P_2 =

```
return probando.substring(0, corte);
```

C_{P_2} =

```
( (declare-fun probando () String)
∧ (declare-fun return () String)
∧ (declare-fun result () String)
∧ (declare-fun corte () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= return result)
∧ (<= lower corte)
∧ (<= corte (length probando))
∧ (= (- corte lower)(length result))
∧ (forall ((index Int))
  (=>
    (and (< index corte)
      (>= index lower))
    (= (charOf probando index)
      (charOf result(- index lower))))) )
```

C_{LS} =

```
( (declare-fun input () String)
∧ (= input "susie@mail.com")
∧ (= (length input) 14)
∧ (declare-fun output () String)
∧ (= output "susie")
∧ (= (length output) 5)
```

Map Input/Output to Snippet

```
c1.   (= input probando)
c2.   (= output return)
```

Solving

P_2 =

```
return probando.substring(0, corte);
```

C_{P_2} =

```
( (declare-fun probando () String)
∧ (declare-fun return () String)
∧ (declare-fun result () String)
∧ (declare-fun corte () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= return result)
∧ (<= lower corte)
∧ (<= corte (length probando))
∧ (= (- corte lower)(length result))
∧ (forall ((index Int))
  (=>
    (and (< index corte)
      (>= index lower))
    (= (charOf probando index)
      (charOf result(- index lower))))) )
```

C_{LS} =

```
( (declare-fun input () String)
∧ (= input "susie@mail.com")
∧ (= (length input) 14)
∧ (declare-fun output () String)
∧ (= output "susie")
∧ (= (length output) 5)
∧ (= input probando)
∧ (= output return) )
```

Solving

P_2 =

```
return probando.substring(0, corte);
```

C_{P_2} =

```
(  (declare-fun probando () String)
∧ (declare-fun return () String)
∧ (declare-fun result () String)
∧ (declare-fun corte () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= return result)
∧ (<= lower corte)
∧ (<= corte (length probando))
∧ (= (- corte lower)(length result))
∧ (forall ((index Int))
  (=>
    (and (< index corte)
      (>= index lower))
    (= (charOf probando index)
      (charOf result(- index lower))))) )
```

C_{LS} =

```
(  (declare-fun input () String)
∧ (= input "susie@mail.com")
∧ (= (length input) 14)
∧ (declare-fun output () String)
∧ (= output "susie")
∧ (= (length output) 5)
∧ (= input probando)
∧ (= output return) )
```

SMT Solver

$Solve(C_{P_2} \wedge C_{LS}) =$

Solving

P_2 =

```
return probando.substring(0, corte);
```

C_{P_2} =

```
(  (declare-fun probando () String)
∧ (declare-fun return () String)
∧ (declare-fun result () String)
∧ (declare-fun corte () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= return result)
∧ (<= lower corte)
∧ (<= corte (length probando))
∧ (= (- corte lower)(length result))
∧ (forall ((index Int))
  (=>
    (and (< index corte)
      (>= index lower))
    (= (charOf probando index)
      (charOf result(- index lower))))) )
```

C_{LS} =

```
(  (declare-fun input () String)
∧ (= input "susie@mail.com")
∧ (= (length input) 14)
∧ (declare-fun output () String)
∧ (= output "susie")
∧ (= (length output) 5)
∧ (= input probando)
∧ (= output return) )
```

SMT Solver

$Solve(C_{P_2} \wedge C_{LS}) =$ **sat**

Solving

P_2 =

```
return probando.substring(0, corte);
```

C_{P_2} =

```
(  (declare-fun probando () String)
∧ (declare-fun return () String)
∧ (declare-fun result () String)
∧ (declare-fun corte () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= return result)
∧ (<= lower corte)
∧ (<= corte (length probando))
∧ (= (- corte lower)(length result))
∧ (forall ((index Int))
  (=>
    (and (< index corte)
      (>= index lower))
    (= (charOf probando index)
      (charOf result(- index lower))))) )
```

C_{LS} =

```
(  (declare-fun input () String)
∧ (= input "susie@mail.com")
∧ (= (length input) 14)
∧ (declare-fun output () String)
∧ (= output "susie")
∧ (= (length output) 5)
∧ (= input probando)
∧ (= output return) )
```

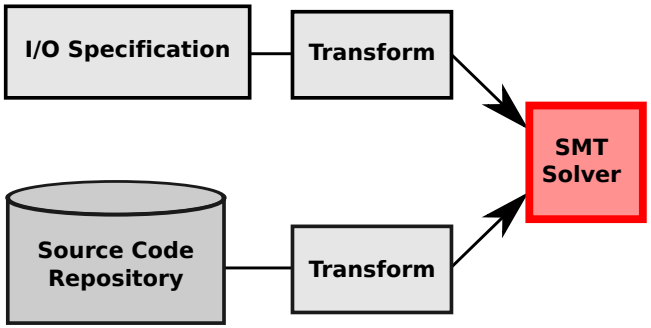
SMT Solver

$Solve(C_{P_2} \wedge C_{LS}) =$ **sat**

Solution:

$\{\textbf{corte} \mapsto 5\}$

Solving



SMT Solver

$$Solve(C_{P_2} \wedge C_{LS}) = \textbf{sat}$$

$RepP =$

```
P3. int index = token.indexOf("://");
    String label = token.substring(0, index);
```

```
P4. boolean sfx(String as, String bs) {
      return as.endsWith(bs);
    }
```

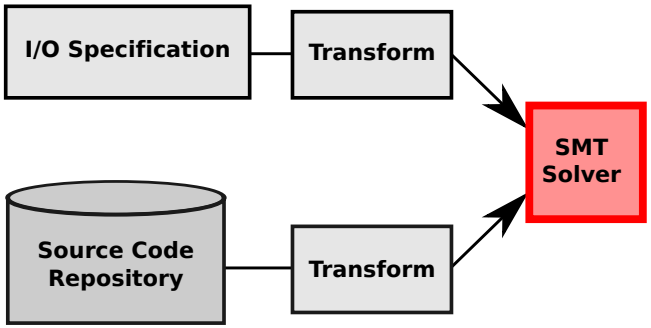
```
P5. String getname(String addr) {
      int i = addr.indexOf("@");
      if(i > -1) {
         return addr.substring(0, i);
      }
      return "NoOne";
    }
```

$SatP =$

```
P1. String scheme = uri.substring(0, 5);
```

```
P2. return probando.substring(0, corte);
```

Solving



SMT Solver

$$Solve(C_{P_3} \wedge C_{LS}) =$$

$RepP =$

```
P3. int index = token.indexOf("://");
    String label = token.substring(0, index);
```

```
P4. boolean sfx(String as, String bs) {
      return as.endsWith(bs);
    }
```

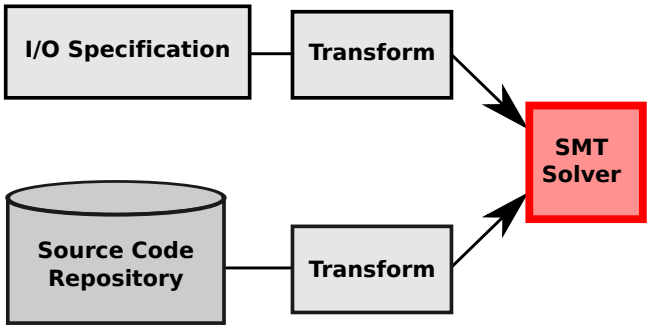
```
P5. String getname(String addr) {
      int i = addr.indexOf("@");
      if(i > -1) {
         return addr.substring(0, i);
      }
      return "NoOne";
    }
```

$SatP =$

```
P1. String scheme = uri.substring(0, 5);
```

```
P2. return probando.substring(0, corte);
```

Solving



SMT Solver

$$Solve(C_{P_3} \wedge C_{LS}) = \textbf{unsat}$$

$RepP =$

```
P3. int index = token.indexOf("://");
    String label = token.substring(0, index);
```

```
P4. boolean sfx(String as, String bs) {
      return as.endsWith(bs);
    }
```

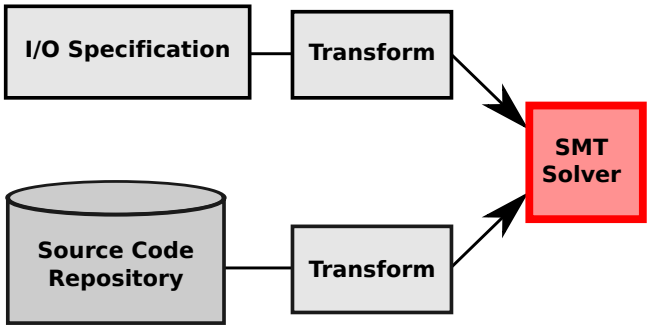
```
P5. String getname(String addr) {
      int i = addr.indexOf("@");
      if(i > -1) {
         return addr.substring(0, i);
      }
      return "NoOne";
    }
```

$SatP =$

```
P1. String scheme = uri.substring(0, 5);
```

```
P2. return probando.substring(0, corte);
```

Solving



SMT Solver

$$Solve(C_{P_4} \wedge C_{LS}) = \textbf{unsat}$$

$RepP =$

```
P3. int index = token.indexOf("://");
    String label = token.substring(0, index);
```

```
P4. boolean sfx(String as, String bs) {
      return as.endsWith(bs);
    }
```

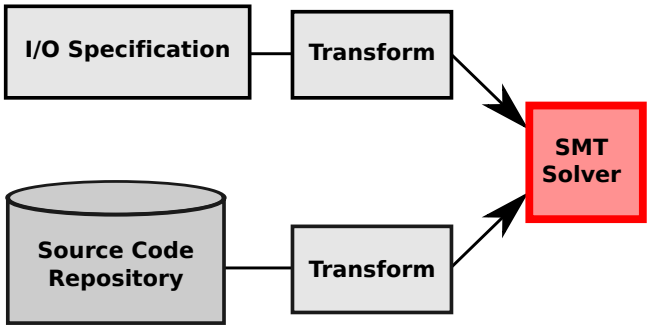
```
P5. String getname(String addr) {
      int i = addr.indexOf("@");
      if(i > -1) {
         return addr.substring(0, i);
      }
      return "NoOne";
    }
```

$SatP =$

```
P1. String scheme = uri.substring(0, 5);
```

```
P2. return probando.substring(0, corte);
```

Solving



SMT Solver

$$Solve(C_{P_5} \wedge C_{LS}) =$$

$RepP =$

```
P3. int index = token.indexOf("://");
    String label = token.substring(0, index);
```

```
P4. boolean sfx(String as, String bs) {
      return as.endsWith(bs);
    }
```

```
P5. String getname(String addr) {
      int i = addr.indexOf("@");
      if(i > -1) {
         return addr.substring(0, i);
      }
      return "NoOne";
    }
```

$SatP =$

```
P1. String scheme = uri.substring(0, 5);
```

```
P2. return probando.substring(0, corte);
```

Multi-Path Programs

```
1 String getname(String addr) {
2     int i = addr.indexOf("@");
3     if(i > -1) {
4         return addr.substring(0, i);
5     }
6     return "NoOne";
7 }
```

Multi-Path Programs

```
1 String getname(String addr) {
2     int i = addr.indexOf("@");
3     if(i > -1) {
4         return addr.substring(0, i);
5     }
6
7 }
```

```
1 String getname(String addr) {
2     int i = addr.indexOf("@");
3     if(i > -1) {
4
5     }
6     return "NoOne";
7 }
```

Multi-Path Programs

```
1  String getname(String addr) {
2      int i = addr.indexOf("@");
3      if(i > -1) {
4          return addr.substring(0, i);
5      }
6
7  }
```

q_1

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc1: i > -1
retVal = addr.substring(0, i);
return retVal;
```

```
1  String getname(String addr) {
2      int i = addr.indexOf("@");
3      if(i > -1) {
4
5      }
6      return "NoOne";
7  }
```

q_2

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc2: i <= -1
retVal = "NoOne";
return retVal;
```

Multi-Path Programs

```
1  String getname(String addr) {
2      int i = addr.indexOf("@");
3      if(i > -1) {
4          return addr.substring(0, i);
5      }
6
7  }
```

q_1

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc1: i > -1
retVal = addr.substring(0, i);
return retVal;
```

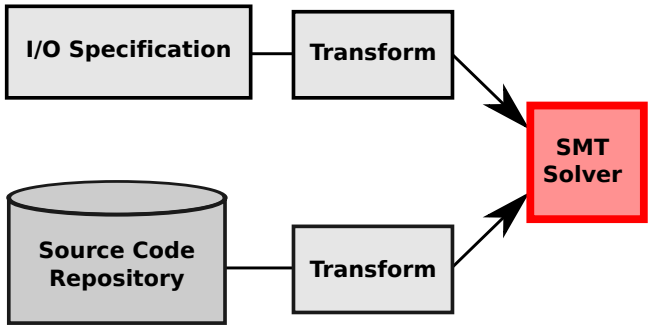
```
1  String getname(String addr) {
2      int i = addr.indexOf("@");
3      if(i > -1) {
4
5      }
6      return "NoOne";
7  }
```

q_2

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc2: i <= -1
retVal = "NoOne";
return retVal;
```

$$C_{P_5} = \{C_{q_1} \vee C_{q_2}\}$$

Specification Refinement



SMT Solver

$$Solve(C_{P_5} \wedge C_{LS}) =$$

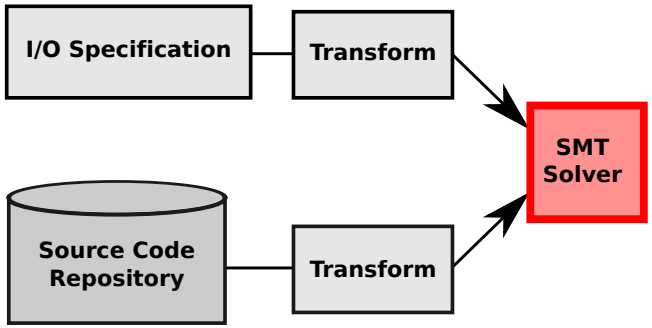
q_1

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc1: i > -1
retVal = addr.substring(0, i);
return retVal;
```

q_2

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc2: i <= -1
retVal = "NoOne";
return retVal;
```

Specification Refinement



SMT Solver

$$Solve(C_{P_5} \wedge C_{LS}) =$$

q_1

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc1: i > -1
retVal = addr.substring(0, i);
return retVal;
```

q_2

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc2: i <= -1
retVal = "NoOne";
return retVal;
```

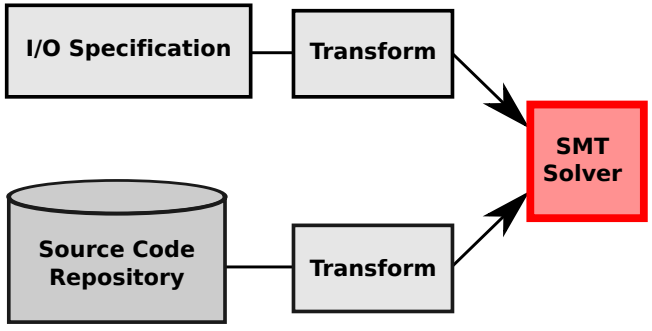
SMT Solver

$$Solve(C_{q_1} \wedge C_{LS}) =$$

SMT Solver

$$Solve(C_{q_2} \wedge C_{LS}) =$$

Specification Refinement



SMT Solver

$$Solve(C_{P_5} \wedge C_{LS}) =$$

q_1

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc1: i > -1
retVal = addr.substring(0, i);
return retVal;
```

q_2

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc2: i <= -1
retVal = "NoOne";
return retVal;
```

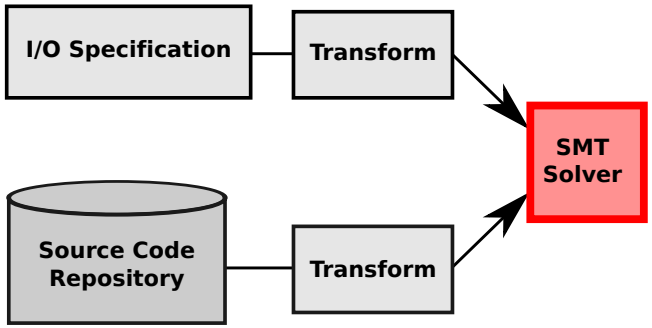
SMT Solver

$$Solve(C_{q_1} \wedge C_{LS}) = \textbf{sat}$$

SMT Solver

$$Solve(C_{q_2} \wedge C_{LS}) =$$

Specification Refinement



SMT Solver

$$Solve(C_{P_5} \wedge C_{LS}) =$$

q_1

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc1: i > -1
retVal = addr.substring(0, i);
return retVal;
```

q_2

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc2: i <= -1
retVal = "NoOne";
return retVal;
```

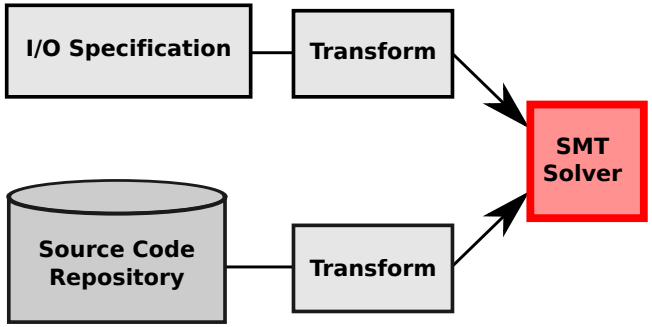
SMT Solver

$$Solve(C_{q_1} \wedge C_{LS}) = \textbf{sat}$$

SMT Solver

$$Solve(C_{q_2} \wedge C_{LS}) = \textbf{unsat}$$

Specification Refinement



SMT Solver

$$Solve(C_{P_5} \wedge C_{LS}) = \textbf{sat}$$

q_1

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc1: i > -1
retVal = addr.substring(0, i);
return retVal;
```

q_2

```
String addr;
int i;
String retVal;
i = addr.indexOf("@");
pc2: i <= -1
retVal = "NoOne";
return retVal;
```

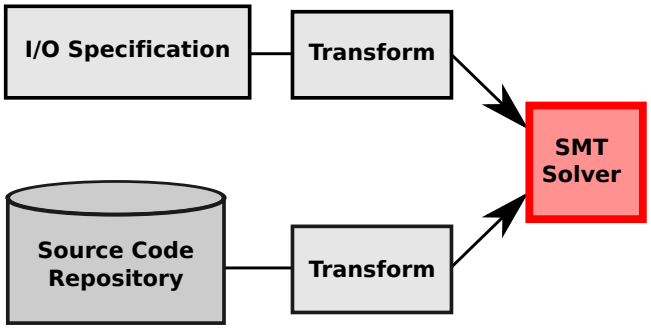
SMT Solver

$$Solve(C_{q_1} \wedge C_{LS}) = \textbf{sat}$$

SMT Solver

$$Solve(C_{q_2} \wedge C_{LS}) = \textbf{unsat}$$

Solving



SMT Solver

$$Solve(C_{P_5} \wedge C_{LS}) = \textbf{sat}$$

$RepP =$

```
P3. int index = token.indexOf("://");
    String label = token.substring(0, index);
```

```
P4. boolean sfx(String as, String bs) {
      return as.endsWith(bs);
    }
```

$SatP =$

```
P1. String scheme = uri.substring(0, 5);
```

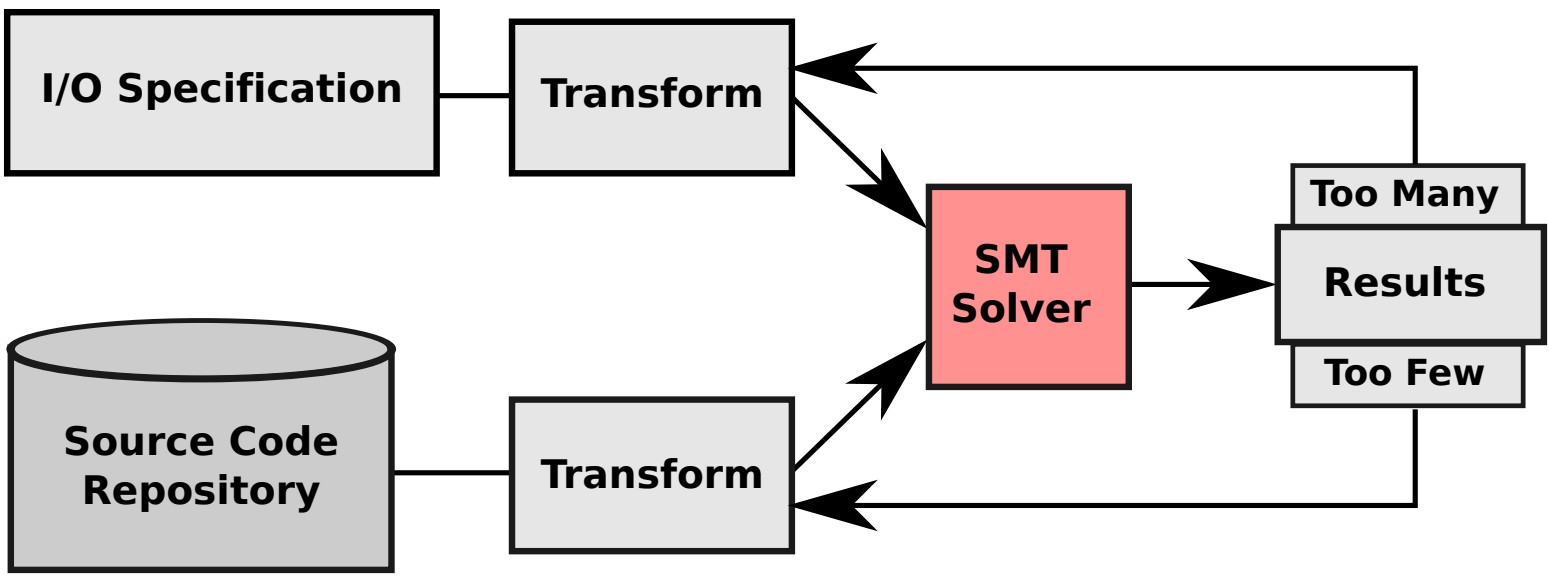
```
P2. return probando.substring(0, corte);
```

```
P5. String getname(String addr) {
      int i = addr.indexOf("@");
      if(i > -1) {
         return addr.substring(0, i);
      }
      return "NoOne";
    }
```

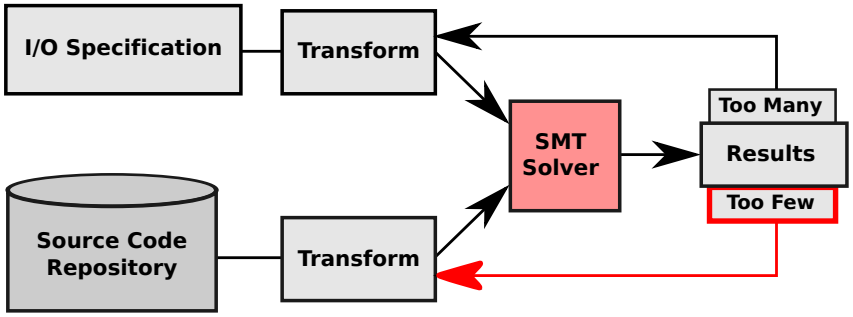
Revising the Results

Issues with Results:

1. *Too Many* $\rightarrow$ Refine lightweight specifications
2. *Too Few* $\rightarrow$ Abstract encodings



Encoding Abstraction



$LS =$

$\{(\textbf{susie@mail.com}, \textbf{mail.com})\}$

$RepP =$

```
P1. String scheme = uri.substring(0, 5);
```

```
P2. return probando.substring(0, corte);
```

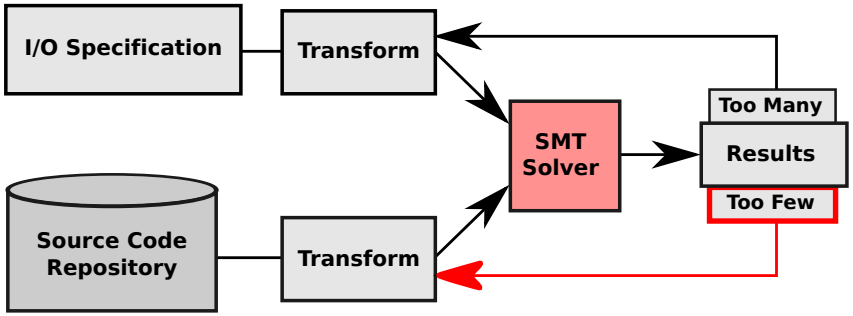
```
P3. int index = token.indexOf("://");
    String label = token.substring(0, index);
```

```
P4. boolean sfx(String as, String bs) {
      return as.endsWith(bs);
    }
```

```
P5. String getname(String addr) {
      int i = addr.indexOf("@");
      if(i > -1) {
         return addr.substring(0, i);
      }
      return "NoOne";
    }
```

$SatP =$

Encoding Abstraction



$LS =$

$\{(\textbf{susie@mail.com}, \textbf{mail.com})\}$

$RepP =$

```
P1. String scheme = uri.substring(0, 5);
```

```
P2. return probando.substring(0, corte);
```

```
P3. int index = token.indexOf("://");
    String label = token.substring(0, index);
```

```
P4. boolean sfx(String as, String bs) {
      return as.endsWith(bs);
    }
```

```
P5. String getname(String addr) {
      int i = addr.indexOf("@");
      if(i > -1) {
         return addr.substring(0, i);
      }
      return "NoOne";
    }
```

$SatP =$

Abstracting a Program Encoding

$P_1 =$

```
String scheme = uri.substring(0, 5);
```

$LS =$

$\{(\textbf{susie@mail.com}, \textbf{mail.com})\}$

Abstracting a Program Encoding

$P_1 =$

```
String scheme = uri.substring(?, ?);
```

$LS =$

$\{(\textbf{susie@mail.com}, \textbf{mail.com})\}$

Abstracting a Program Encoding

$P_1 =$

```
String scheme = uri.substring(?, ?);
```

$LS =$

$\{(\texttt{susie@mail.com}, \texttt{mail.com})\}$

$C_{P_1} =$

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower 0)
∧ (= upper 5)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
∧ (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower))))) )
```

Abstracting a Program Encoding

$P_1 =$

```
String scheme = uri.substring(?, ?);
```

$LS =$

$\{(\textbf{susie@mail.com}, \textbf{mail.com})\}$

$C_{P_1}{}' =$

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower ?)
∧ (= upper ?)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
∧ (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower))))) )
```

Abstracting a Program Encoding

$P_1 =$

```
String scheme = uri.substring(?, ?);
```

$LS =$

$\{(\text{susie@mail.com}, \text{mail.com})\}$

$C_{P_1}{}' =$

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower ?)
∧ (= upper ?)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
∧ (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower))))) )
```

SMT Solver

$Solve(C_{P_1}{}' \wedge LS_{enc}) =$

Abstracting a Program Encoding

$P_1 =$

```
String scheme = uri.substring(?, ?);
```

$LS =$

$\{(\text{susie@mail.com}, \text{mail.com})\}$

$C_{P_1}{}' =$

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower ?)
∧ (= upper ?)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
∧ (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower))))) )
```

SMT Solver

$Solve(C_{P_1}{}' \wedge LS_{enc}) = \textbf{sat}$

Abstracting a Program Encoding

$P_1 =$

```
String scheme = uri.substring(?, ?);
```

$LS =$

$\{(\textbf{susie@mail.com}, \textbf{mail.com})\}$

$C_{P_1}{}' =$

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower ?)
∧ (= upper ?)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
∧ (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower))))) )
```

SMT Solver

$Solve(C_{P_1}{}' \wedge LS_{enc}) = \textbf{sat}$

Solution:

$\{\textbf{lower} \mapsto 6, \textbf{upper} \mapsto 14\}$

Abstracting a Program Encoding

$P_1 =$

```
String scheme = uri.substring(?, ?);
```

$LS =$

$\{(\textbf{susie@mail.com}, \textbf{mail.com})\}$

$C_{P_1}{}' =$

```
(  (declare-fun uri () String)
∧ (declare-fun scheme () String)
∧ (declare-fun resRHS () String)
∧ (declare-fun upper () Int)
∧ (declare-fun lower () Int)
∧ (= lower ?)
∧ (= upper ?)
∧ (= scheme resRHS)
∧ (<= lower upper)
∧ (<= upper (length uri))
∧ (= (- upper lower)(length resRHS))
∧ (forall ((index Int))
  (=>
    (and (< index upper)
      (>= index lower))
    (= (charOf uri index)
      (charOf resRHS(- index lower))))) )
```

SMT Solver

$Solve(C_{P_1}{}' \wedge LS_{enc}) = \textbf{sat}$

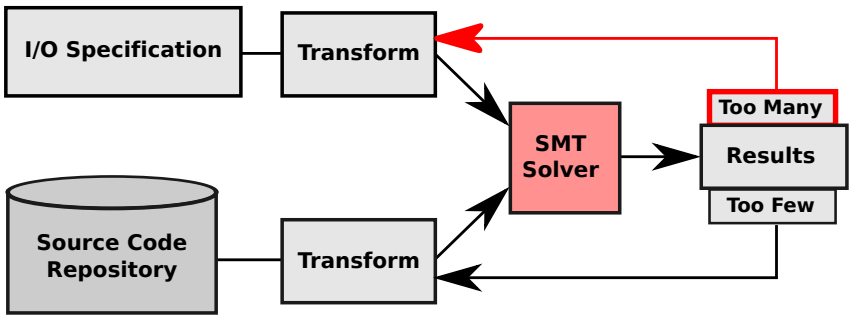
Solution:

$\{\textbf{lower} \mapsto 6, \textbf{upper} \mapsto 14\}$

$P_1{}' =$

```
String scheme = uri.substring(6, 14);
```

Specification Refinement



Goal

extract alias from e-mail address in Java

LS =

{(**susie@mail.com**, **susie**)}

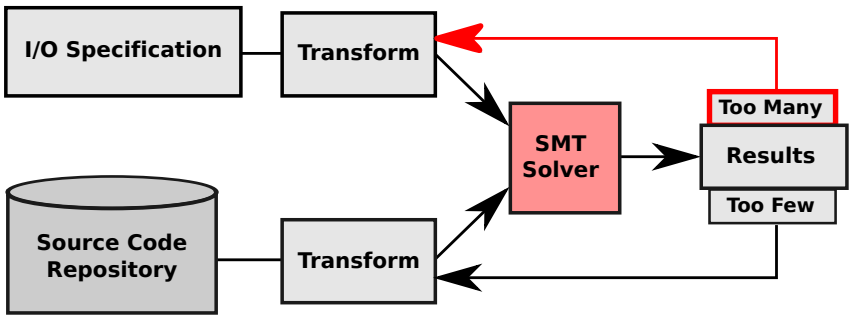
$SatP$ =

P_1.
```
String scheme = uri.substring(0, 5);
```

P_2.
```
return probando.substring(0, corte);
```

P_5.
```
String getname(String addr) {
    int i = addr.indexOf("@");
    if(i > -1) {
        return addr.substring(0, i);
    }
    return "NoOne";
}
```

Specification Refinement



Goal

extract alias from e-mail address in Java

$LS' =$

$\{$(**susie@mail.com**, **susie**), (**al@me.us**, **al**)$\}$

$SatP =$

```
P1. String scheme = uri.substring(0, 5);
```

```
P2. return probando.substring(0, corte);
```

```
P5. String getname(String addr) {
       int i = addr.indexOf("@");
       if(i > -1) {
          return addr.substring(0, i);
       }
       return "NoOne";
    }
```

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_1,P_2,P_5

```
P1. String scheme = uri.substring(0, 5);
```

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_1, P_2, P_5

```
P1. String scheme = uri.substring(0, 5);
```

SMT Solver

$Solve(C_{P_1} \wedge C_{ls_1}) =$

SMT Solver

$Solve(C_{P_1} \wedge C_{ls_2}) =$

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_1, P_2, P_5

```
P1. String scheme = uri.substring(0, 5);
```

SMT Solver

$Solve(C_{P_1} \wedge C_{ls_1}) = \textbf{sat}$

SMT Solver

$Solve(C_{P_1} \wedge C_{ls_2}) =$

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_1, P_2, P_5

```
P1. String scheme = uri.substring(0, 5);
```

SMT Solver

$Solve(C_{P_1} \wedge C_{ls_1}) = \textbf{sat}$

SMT Solver

$Solve(C_{P_1} \wedge C_{ls_2}) = \textbf{unsat}$

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_2, P_5

```
P1. String scheme = uri.substring(0, 5);
```

SMT Solver

$Solve(C_{P_1} \wedge C_{ls_1}) = \textbf{sat}$

SMT Solver

$Solve(C_{P_1} \wedge C_{ls_2}) = \textbf{unsat}$

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_2, P_5

```
P2. return probando.substring(0, corte);
```

SMT Solver

$Solve(C_{P_2} \wedge C_{ls_1}) =$

SMT Solver

$Solve(C_{P_2} \wedge C_{ls_2}) =$

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_2, P_5

```
P2. return probando.substring(0, corte);
```

SMT Solver

$Solve(C_{P_2} \wedge C_{ls_1}) = \textbf{sat}$

SMT Solver

$Solve(C_{P_2} \wedge C_{ls_2}) =$

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_2,P_5

```
P2. return probando.substring(0, corte);
```

SMT Solver

$Solve(C_{P_2} \wedge C_{ls_1}) = \textbf{sat}$

SMT Solver

$Solve(C_{P_2} \wedge C_{ls_2}) = \textbf{sat}$

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_2,P_5

```
P2. return probando.substring(0, corte);
```

SMT Solver

$Solve(C_{P_2} \wedge C_{ls_1}) = \textbf{sat}$
Solution:
$\{\textbf{corte} \mapsto 5\}$

SMT Solver

$Solve(C_{P_2} \wedge C_{ls_2}) = \textbf{sat}$

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_2,P_5

```
P2. return probando.substring(0, corte);
```

SMT Solver

$Solve(C_{P_2} \wedge C_{ls_1}) = \textbf{sat}$
Solution:
$\{\textbf{corte} \mapsto 5\}$

SMT Solver

$Solve(C_{P_2} \wedge C_{ls_2}) = \textbf{sat}$
Solution:
$\{\textbf{corte} \mapsto 2\}$

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_5

```
P2. return probando.substring(0, corte);
```

SMT Solver

$Solve(C_{P_2} \wedge C_{ls_1}) = \textbf{sat}$
Solution:
$\{\textbf{corte} \mapsto 5\}$

SMT Solver

$Solve(C_{P_2} \wedge C_{ls_2}) = \textbf{sat}$
Solution:
$\{\textbf{corte} \mapsto 2\}$

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_5

```
P5. String getname(String addr) {
      int i = addr.indexOf("@");
      if(i > -1) {
         return addr.substring(0, i);
      }
      return "NoOne";
   }
```

SMT Solver

$Solve(C_{q_1} \wedge C_{ls_1}) =$
$Solve(C_{q_2} \wedge C_{ls_1}) =$

SMT Solver

$Solve(C_{q_1} \wedge C_{ls_2}) =$
$Solve(C_{q_2} \wedge C_{ls_2}) =$

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_5

```
P5. String getname(String addr) {
       int i = addr.indexOf("@");
       if(i > -1) {
          return addr.substring(0, i);
       }
       return "NoOne";
    }
```

SMT Solver

$Solve(C_{q_1} \wedge C_{ls_1}) = \textbf{sat}$
$Solve(C_{q_2} \wedge C_{ls_1}) = \textbf{unsat}$

SMT Solver

$Solve(C_{q_1} \wedge C_{ls_2}) =$
$Solve(C_{q_2} \wedge C_{ls_2}) =$

Specification Refinement

$LS' = \{ls_1, ls_2\}$

$ls_1 = (\textbf{susie@mail.com}, \textbf{susie})$
$ls_2 = (\textbf{al@me.us}, \textbf{al})$

$SatP =$

P_5

```
P5. String getname(String addr) {
      int i = addr.indexOf("@");
      if(i > -1) {
         return addr.substring(0, i);
      }
      return "NoOne";
   }
```

SMT Solver

$Solve(C_{q_1} \wedge C_{ls_1}) = \textbf{sat}$
$Solve(C_{q_2} \wedge C_{ls_1}) = \textbf{unsat}$

SMT Solver

$Solve(C_{q_1} \wedge C_{ls_2}) = \textbf{sat}$
$Solve(C_{q_2} \wedge C_{ls_2}) = \textbf{unsat}$

Ranking

Let LS be a set of specifications

Let Q_P be the paths in a program P

Let $LS_{sat} = \bigcup_{ls \in LS} \exists q \in Q_P \wedge Solve(C_q \wedge C_{ls}) = sat$

Let $Q_{sat} = \bigcup_{q \in Q_P} \exists ls \in LS \wedge Solve(C_q \wedge C_{ls}) = sat$

Ranking

Let LS be a set of specifications

Let Q_P be the paths in a program P

Let $LS_{sat} = \bigcup_{ls \in LS} \exists q \in Q_P \wedge Solve(C_q \wedge C_{ls}) = sat$

Let $Q_{sat} = \bigcup_{q \in Q_P} \exists ls \in LS \wedge Solve(C_q \wedge C_{ls}) = sat$

Path Matching Ranking

	$LS_{sat} \subset LS$	$LS_{sat} = LS$
$Q_{sat} \subset Q_P$	Splintered	Over Match
$Q_{sat} = Q_P$	Under Match	Full Match

Ranking Example

SatP =

P_1. `String scheme = uri.substring(0, 5);`

P_2.
```
return probando.substring(0, corte);
```

P_5.
```
String getname(String addr) {
    int i = addr.indexOf("@");
    if(i > -1) {
        return addr.substring(0, i);
    }
    return "NoOne";
}
```

LS′ =

{(**susie@mail.com**, **susie**), (**al@me.us**, **al**)}

Ranking Example

$SatP$ =

P_1.
```
String scheme = uri.substring(0, 5);
```

P_2.
```
return probando.substring(0, corte);
```

P_5.
```
String getname(String addr) {
    int i = addr.indexOf("@");
    if(i > -1) {
        return addr.substring(0, i);
    }
    return "NoOne";
}
```

LS' =

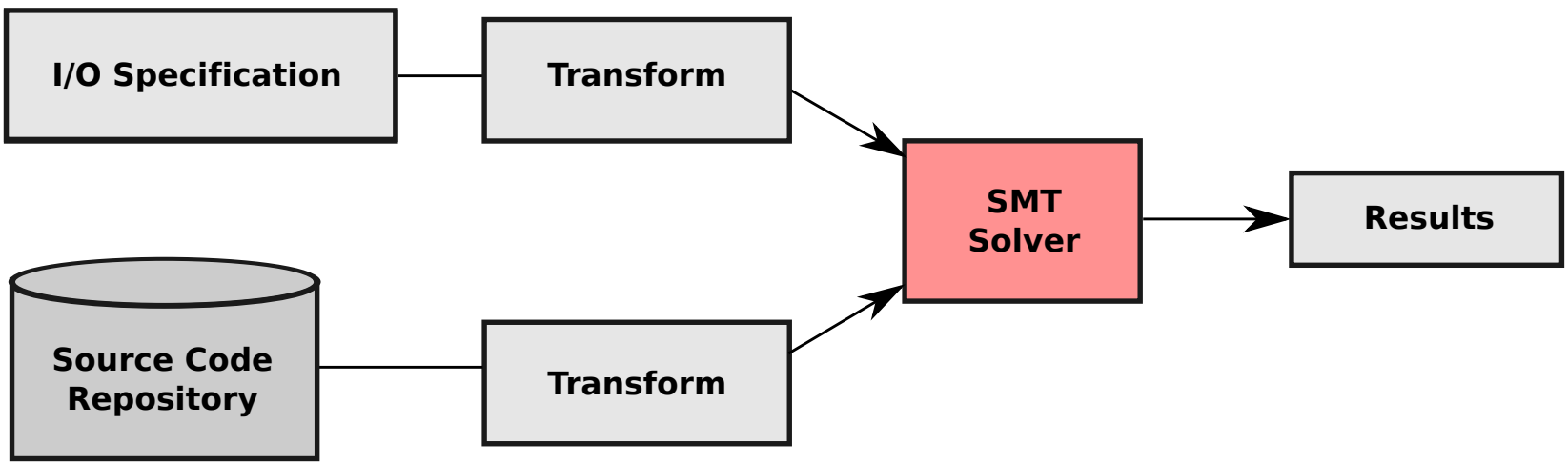
{(**susie@mail.com**, **susie**), (**al@me.us**, **al**)}

Path Matching Ranking

	$LS_{sat} \subset LS$	$LS_{sat} = LS$
$Q_{sat} \subset Q_P$		P_5
$Q_{sat} = Q_P$	P_1	P_2

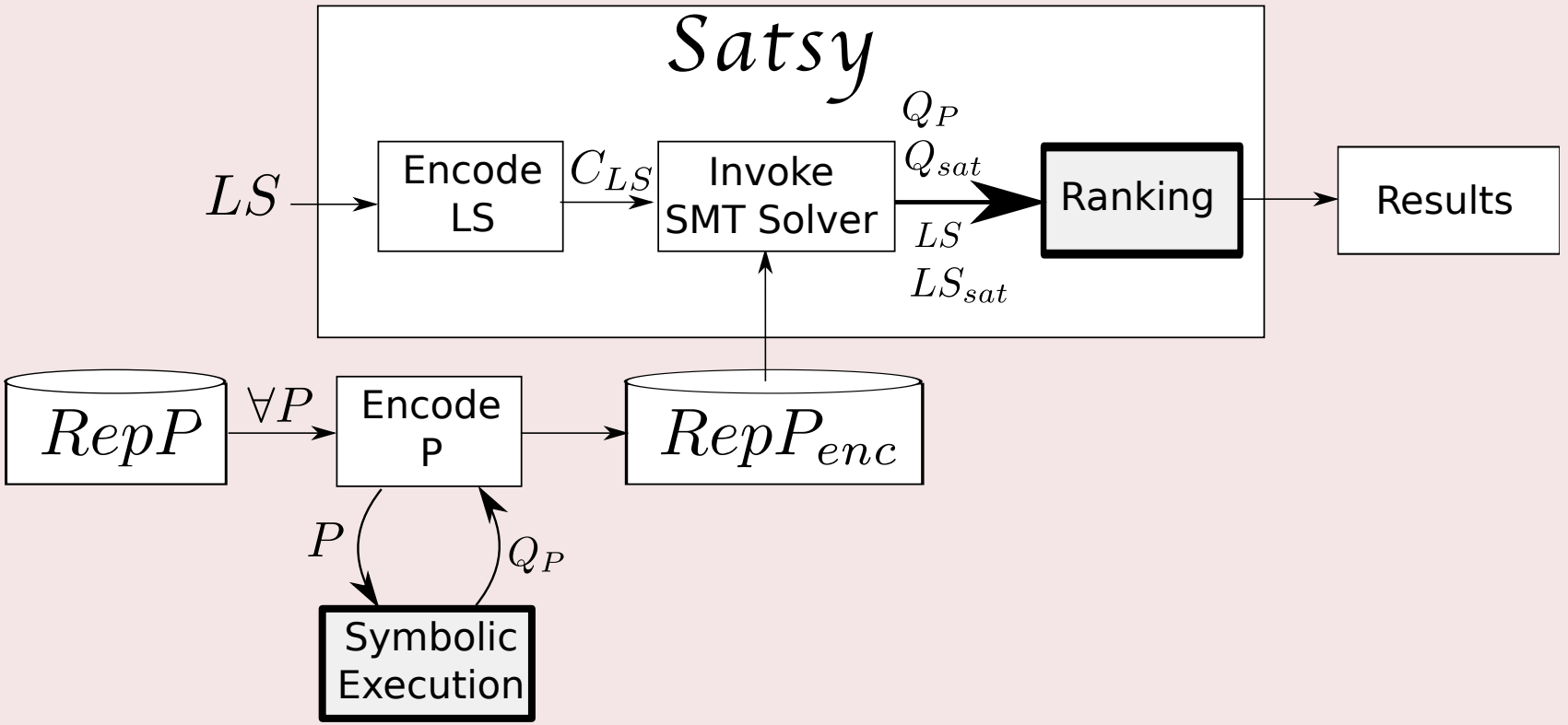
Three Instantiations of Approach

1. Java String library
2. Yahoo! Pipes mashup programs
3. SQL Select statements



Introducing Satsy

Satsy



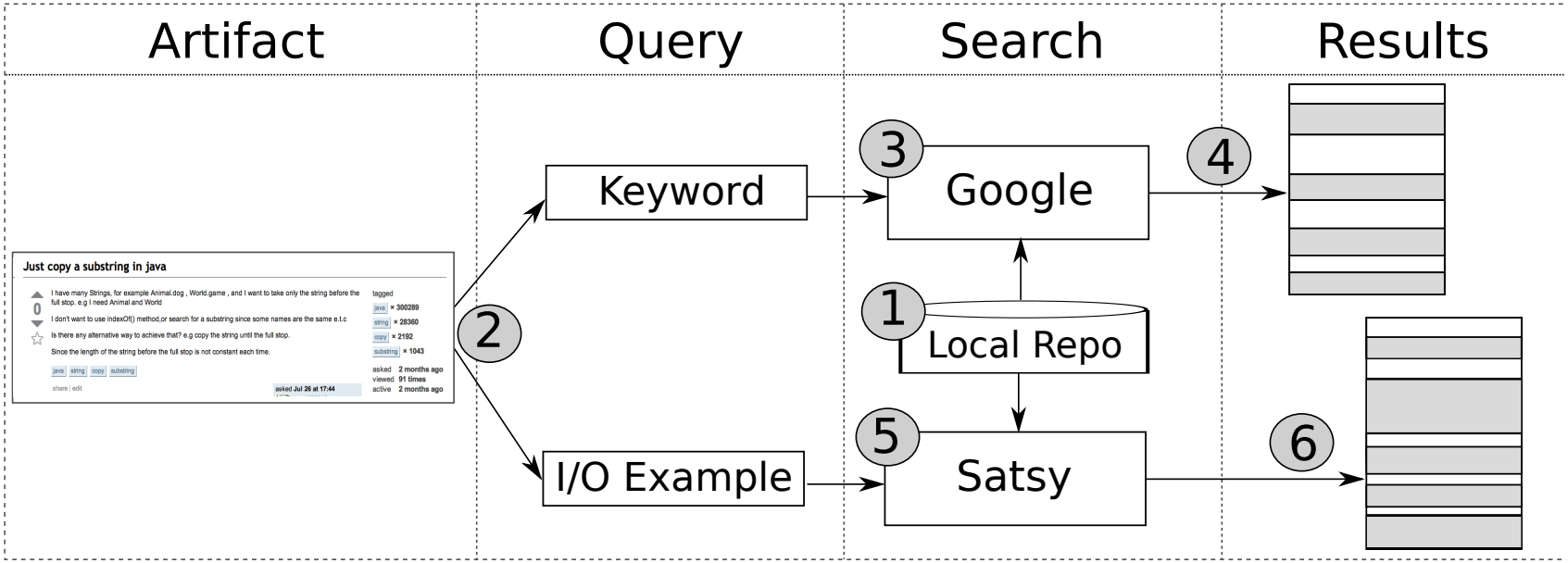
Research Questions

RQ1: How effective is Satsy at returning relevant source code?

	Repository	**Queries**	**Approach**	**Oracle**
(a)	Koders	Stackoverflow	Local Google, Satsy	Analysis, Humans
(b)	Web	Stackoverflow	Google + Satsy	Analysis
(c)	GitHub	Humans	Google, Merobase, Satsy	Humans

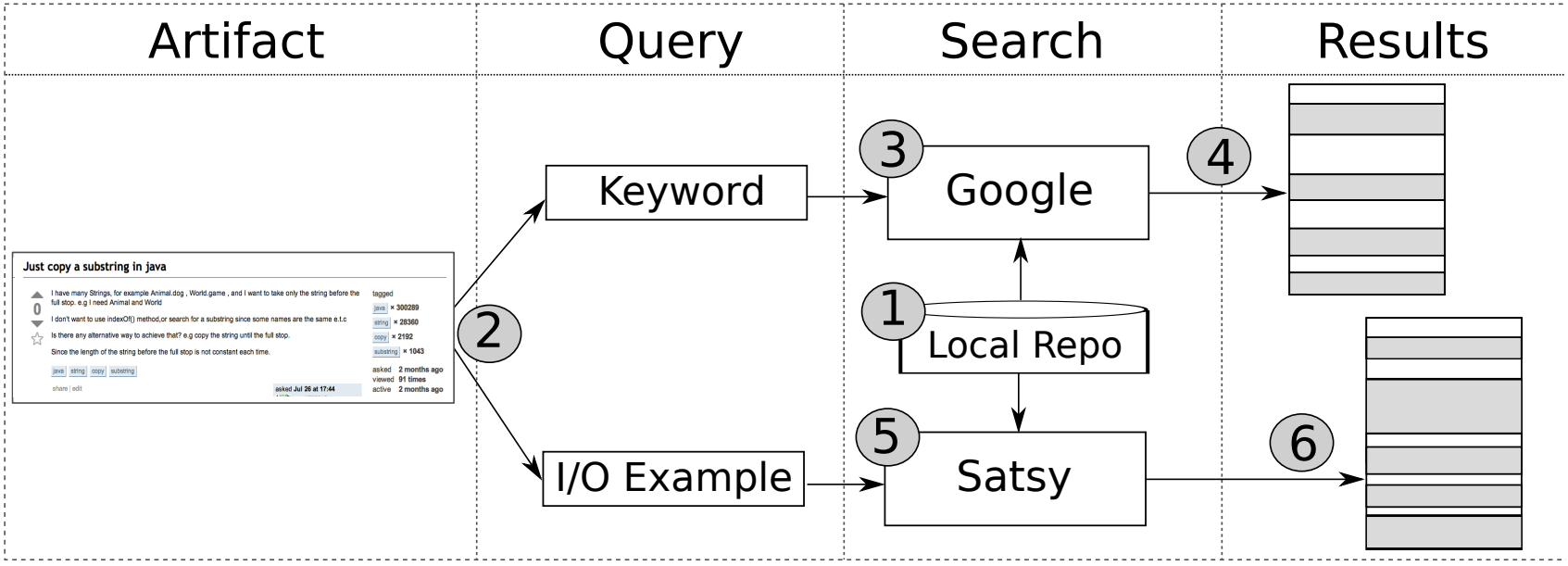
RQ1(a)

Satsy vs. Google, same repository?



RQ1(a)

Satsy vs. Google, same repository?

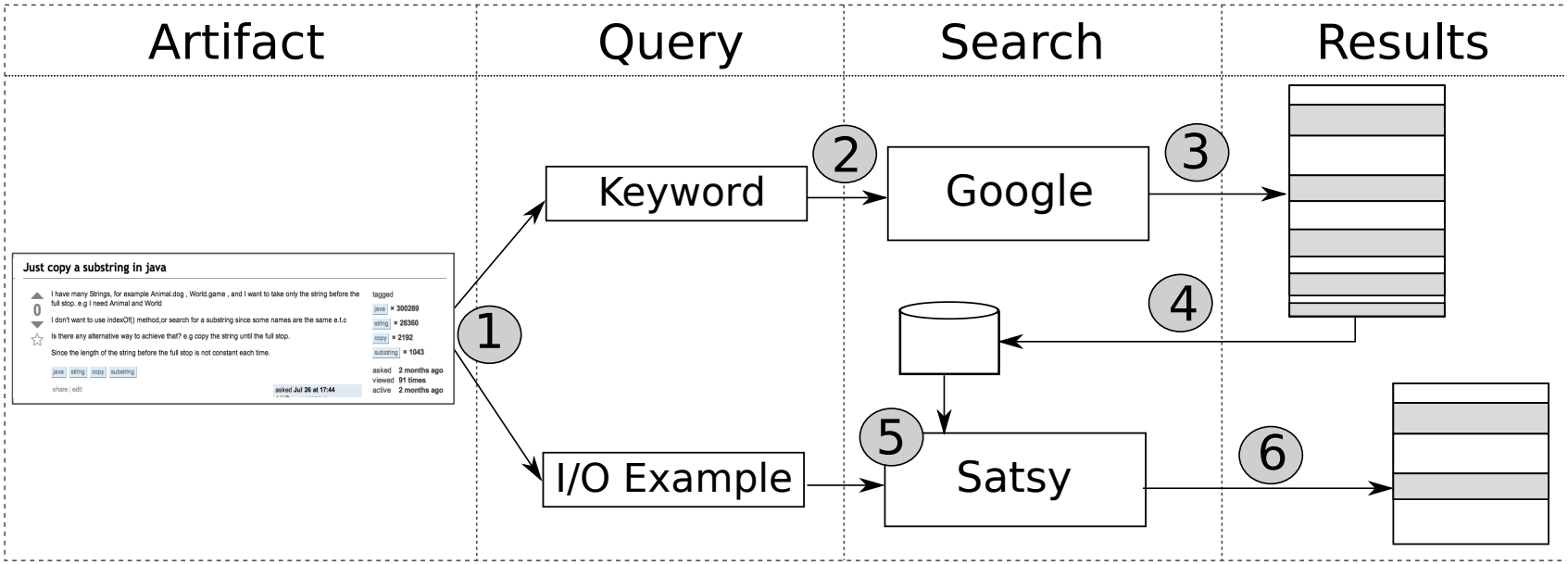


Analysis: Satsy is **5x** more effective at returning relevant results than Google.

Humans: Satsy is **3x** more effective at returning relevant results than Google.

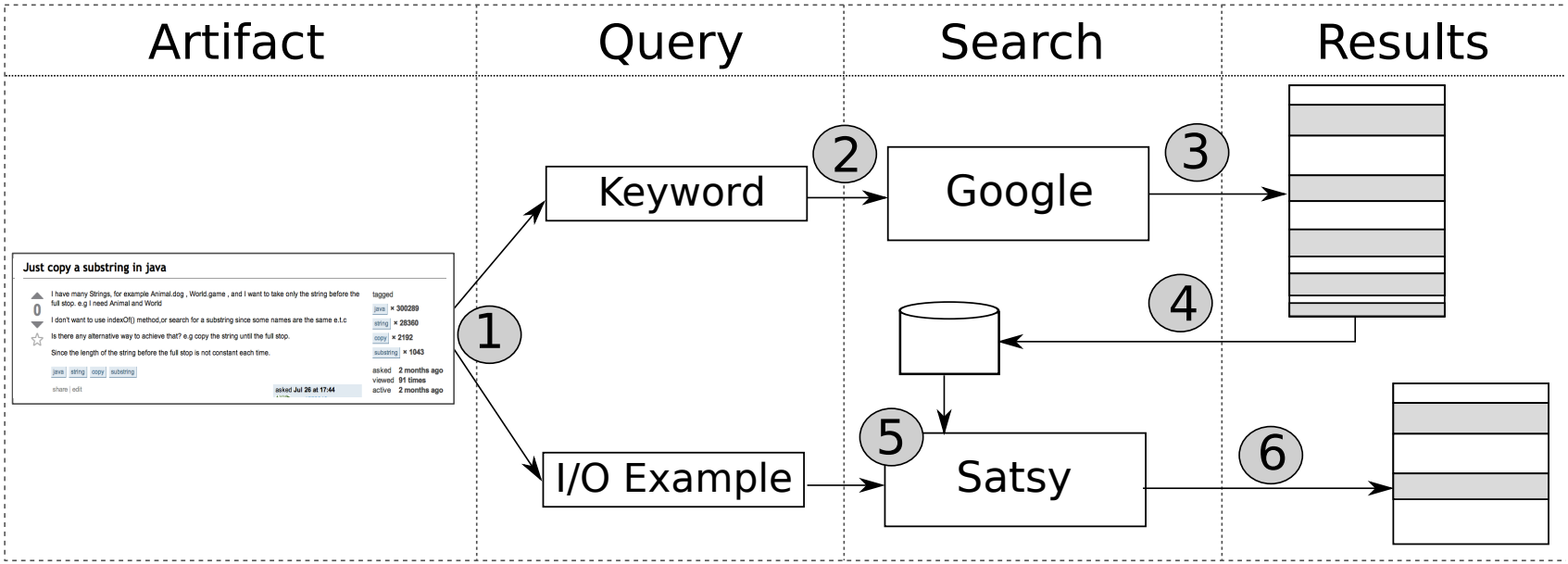
RQ1(b)

Google vs. Satsy + Google?



RQ1(b)

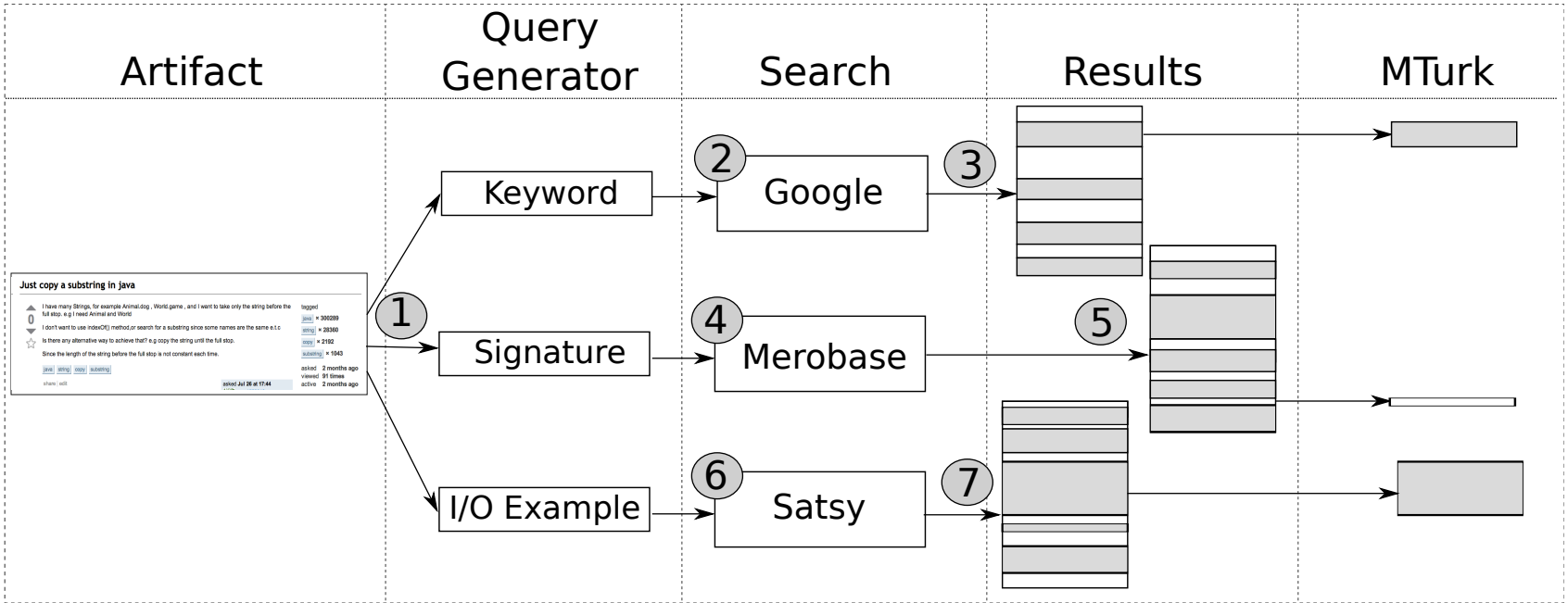
Google vs. Satsy + Google?



Satsy can reduce the snippets from a syntactic search by 34%.

RQ1(c)

Satsy vs. Merobase vs. Google?



Query Generators
Responses from a Generator

Programming Task

Check if one string contains another, case insensitive.

Query Generators
Responses from a Generator

Programming Task

Check if one string contains another, case insensitive.

Descriptive: *does string contain another string*

Query Generators
Responses from a Generator

Programming Task

Check if one string contains another, case insensitive.

Descriptive: *does string contain another string*

Signature: *boolean containsString(String, String)*

Query Generators
Responses from a Generator

Programming Task

Check if one string contains another, case insensitive.

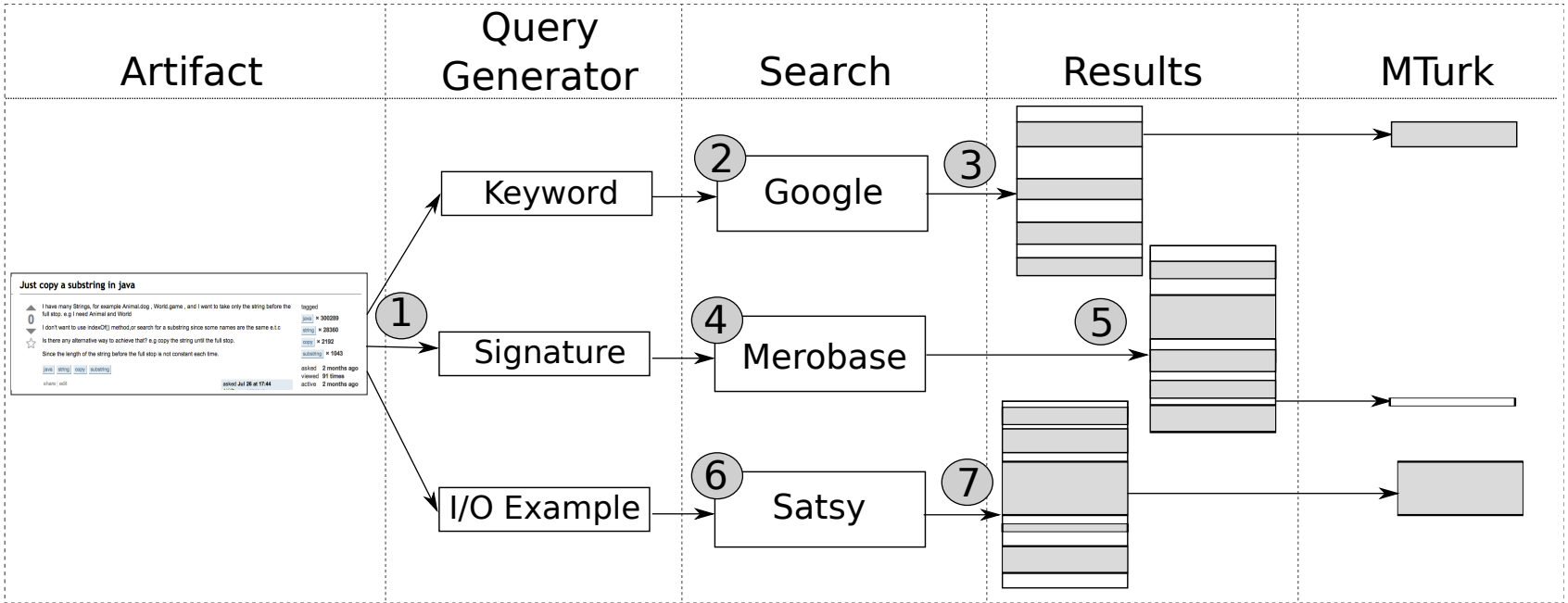
Descriptive: *does string contain another string*

Signature: *boolean containsString(String, String)*

Semantic:
$\{(\{\text{“hello”}, \text{“hello”}\}, \{true\}),$
$(\{\text{“jello”}, \text{“hello”}\}, \{false\}),$
$(\{\text{“hello world”}, \text{“hello”}\}, \{true\}),$
$(\{\text{“hello”}, \text{“hello world”}\}, \{true\})\}$

RQ1(c)

Satsy vs. Merobase vs. Google?



Implementation
Mechanical Turk

Task: Check if one string contains another, case insensitive

Code Snippet 1: Consider the following Java code:

```
boolean isRotation(String s1,String s2) {
return (s1.length() == s2.length()) && ((s1+s1).indexOf(s2) != -1);
}
```

1. Is this code *relevant* to the programming task (relevance means the source code can be easily adapted to solve the problem)?
   ○ Yes ○ No

   Why or why not? How could it be adapted? (requires 10+ word response)

2. Does this code *solve* the programming task (this means the code seems to work as is, without motification)?
   ○ Yes ○ No

   Why or why not? (requires a reasonable response)

Results

Approach	P@10
Google	0.633
Merobase	0.375
Satsy	0.533

Results

Approach	P@10
Google	0.633
Merobase	0.375
Satsy	0.533

Impact of Specification Size

	Number of Input/Output Pairs			
	1	**2**	**3**	**4**
Satsy	0.500	0.483	0.627	0.663

Impact of Specification Size

	Number of Input/Output Pairs			
	1	**2**	**3**	**4**
Satsy	0.500	0.483	0.627	0.663
Satsy (avg.)				0.533

Impact of Ranking

Path Matching Ranking

	$LS_{sat} \subset LS$	$LS_{sat} = LS$
$Q_{sat} \subset Q_P$	Splintered	Over Match
$Q_{sat} = Q_P$	Under Match	Full Match

Original Ranking, round-robin

Full, Over, Under, Splintered

Impact of Ranking

Path Matching Ranking

	$LS_{sat} \subset LS$	$LS_{sat} = LS$
$Q_{sat} \subset Q_P$	Splintered	Over Match
$Q_{sat} = Q_P$	Under Match	Full Match

Original Ranking, round-robin

Full, Over, Under, Splintered

New Ranking, greedy

Over, Full, Under, Splintered

Impact of Ranking

Path Matching Ranking

	$LS_{sat} \subset LS$	$LS_{sat} = LS$
$Q_{sat} \subset Q_P$	Splintered	Over Match
$Q_{sat} = Q_P$	Under Match	Full Match

Original Ranking, round-robin

Full, Over, Under, Splintered

New Ranking, greedy

Over, Full, Under, Splintered

Approach	**P@10**
Satsy	0.533
Satsy'	0.653

Conclusion

RQ1(c): Satsy vs. Merobase vs. Google?

Approach	**P@10**
Google	0.633
Merobase	0.375
Satsy	0.533
Satsy – Large Spec	0.663
Satsy – New Rank	0.653

Conclusion

RQ1(c): Satsy vs. Merobase vs. Google?

Approach	P@10
Google	0.633
Merobase	0.375
Satsy	0.533
Satsy – Large Spec	0.663
Satsy – New Rank	0.653

With smart ranking or larger specs, Satsy is competitive with Google.

Threats to Validity

- Experimental workflow not representative of practice
- Generalizability (e.g., other languages, other tasks)
- Reliability of $P@10$ measures
- ...

Why Input/Output?

Why Input/Output?

Top Question Types Asked on Stackoverflow

Question Type	SQL	YPipes	Java
How do I do X?	74	58	43
Can I do X with/without Y?	8	18	9
What's wrong with ...?	7	9	15
How does Y work?	6	4	19

Why Input/Output?

Top Question Types Asked on Stackoverflow

Question Type	SQL	YPipes	Java
How do I do X?	74 (53)	58 (40)	43 (29)
Can I do X with/without Y?	8 (4)	18 (14)	9 (4)
What's wrong with ...?	7 (2)	9 (8)	15 (14)
How does Y work?	6 (1)	4 (1)	19 (8)

Why Input/Output?

Top Question Types Asked on Stackoverflow

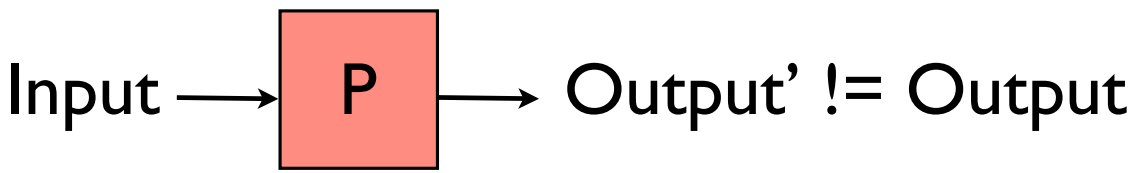
Question Type	SQL	YPipes	Java
How do I do X?	74 (53)	58 (40)	43 (29)
Can I do X with/without Y?	8 (4)	18 (14)	9 (4)
What's wrong with ...?	7 (2)	9 (8)	15 (14)
How does Y work?	6 (1)	4 (1)	19 (8)

"I've been looking for a simple java algorithm to generate a pseudo-random alpha-numeric string. ... Ideally I would be able to specify a length ... For example, a generated string of length 12 might look something like "AEYGF7K0DM1X"."

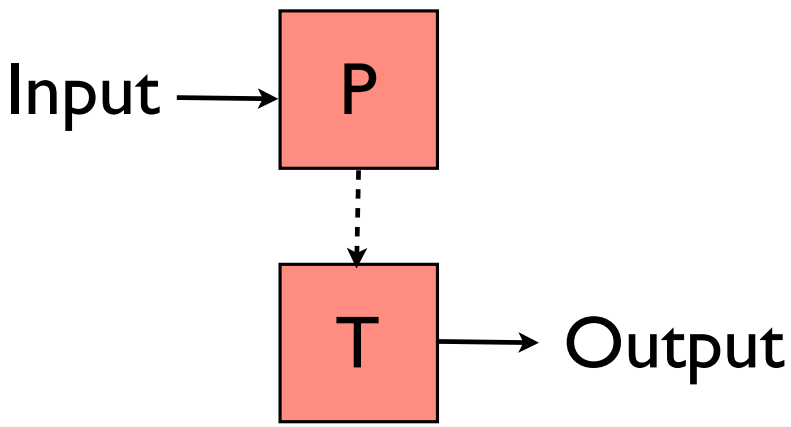
Future Work

1. Other specification models
2. Extending language support
3. Revise and evaluate ranking algorithms
4. End-to-end study with programmers
5. ...

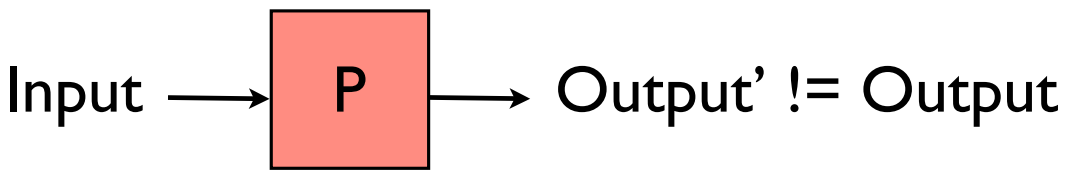
Program Composition



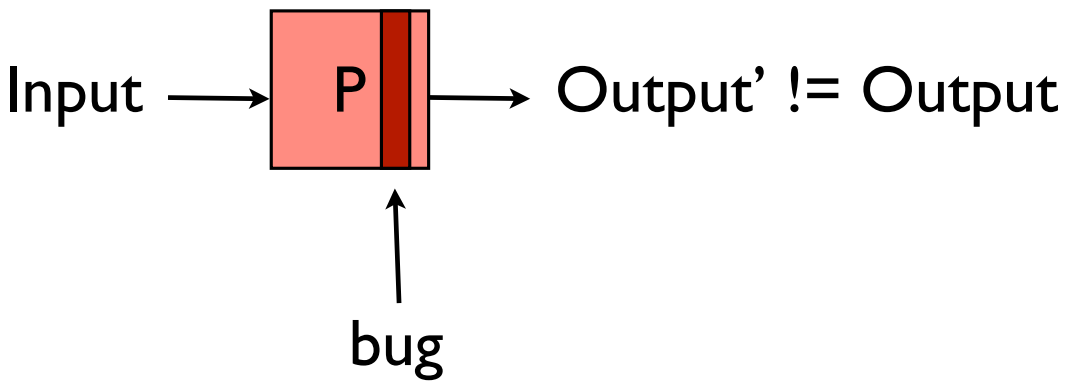
Program Composition



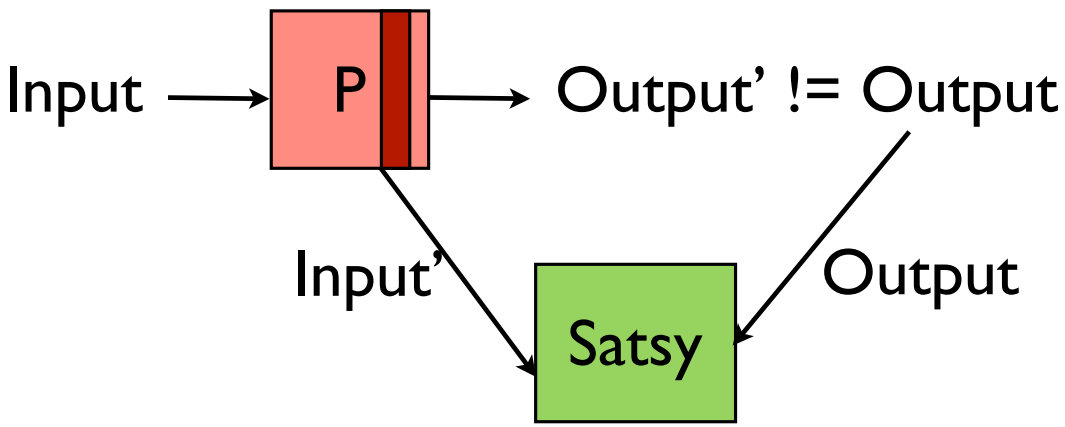
Automated Fault Fixing



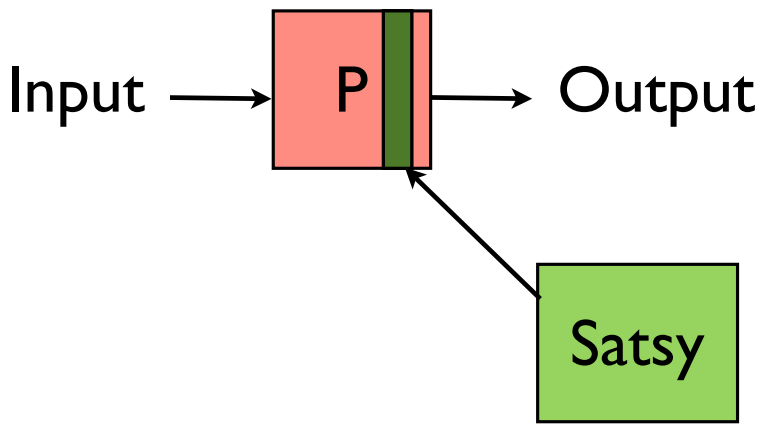
Automated Fault Fixing



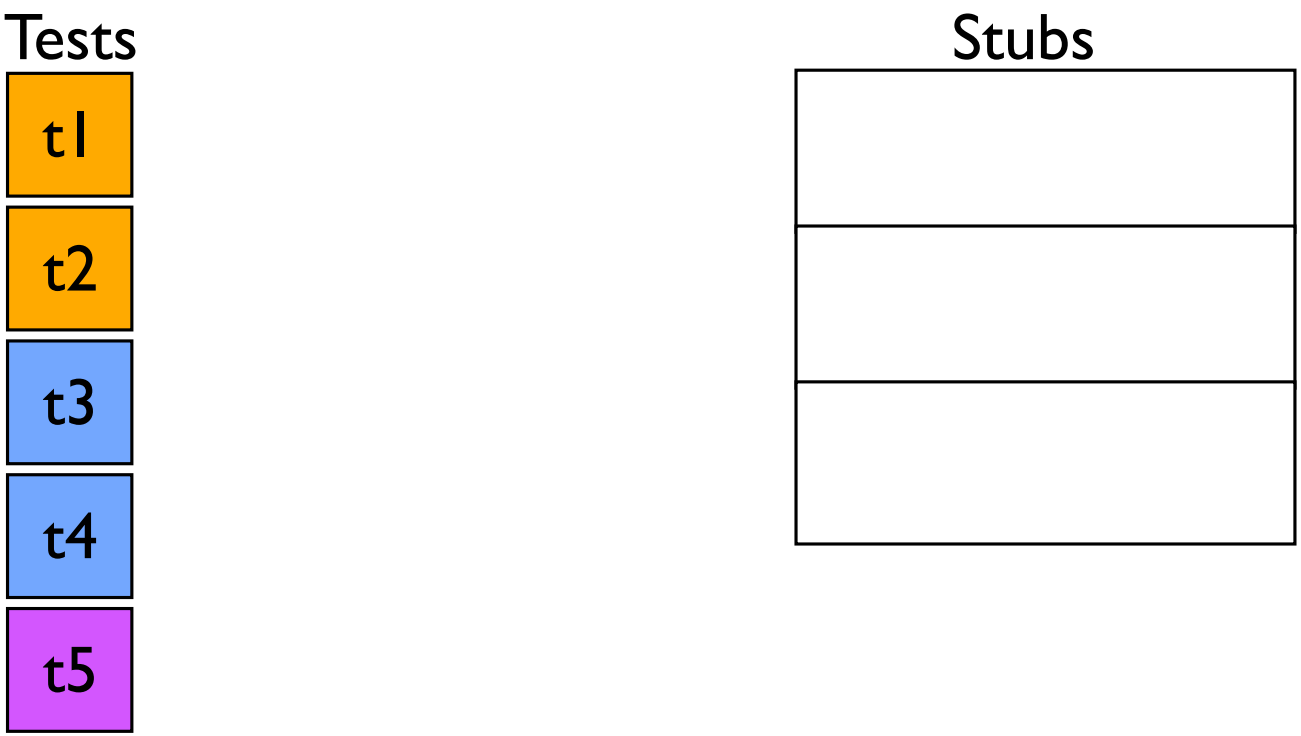
Automated Fault Fixing



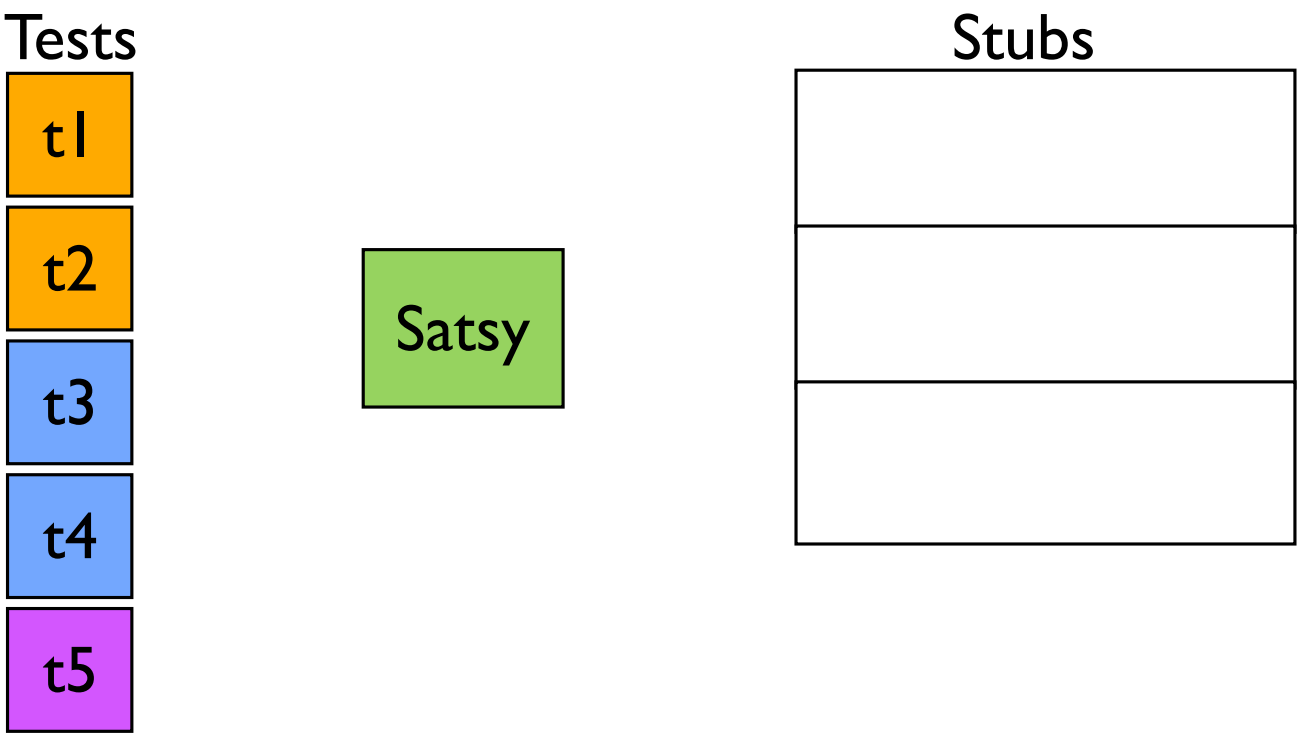
Automated Fault Fixing



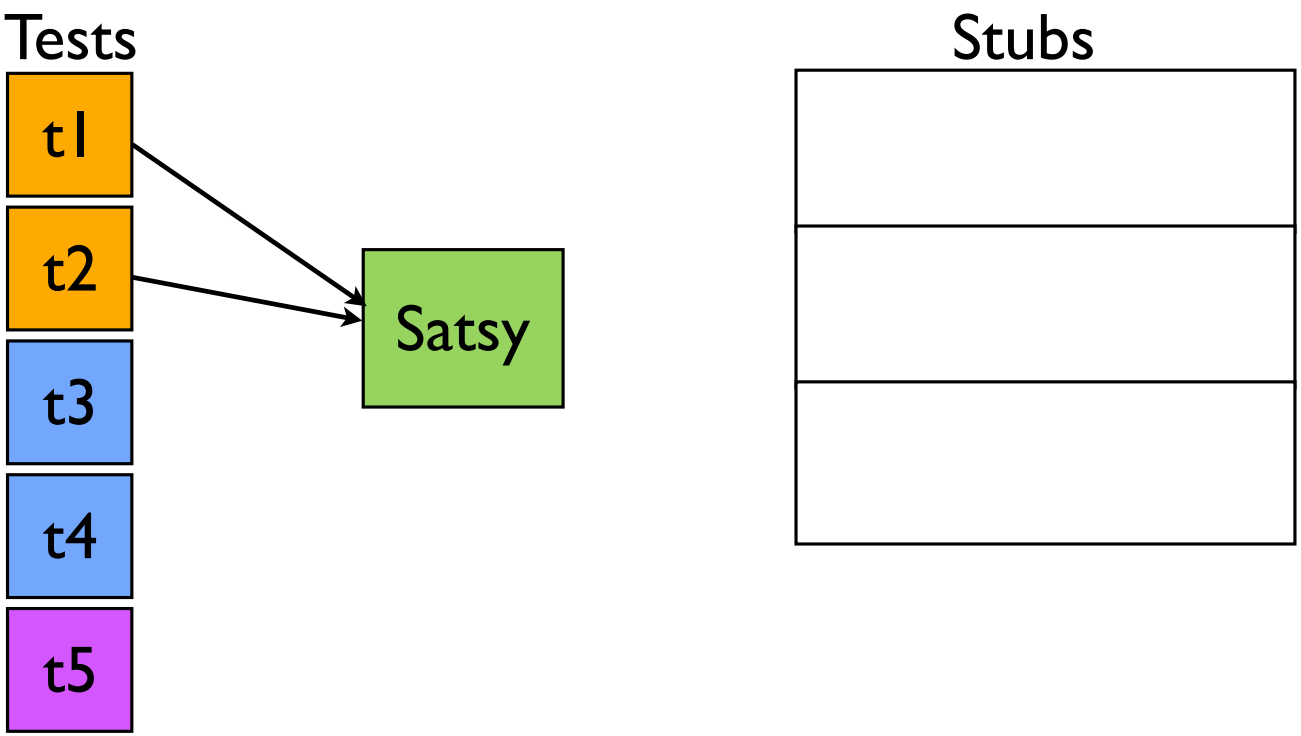
Test-Driven Development



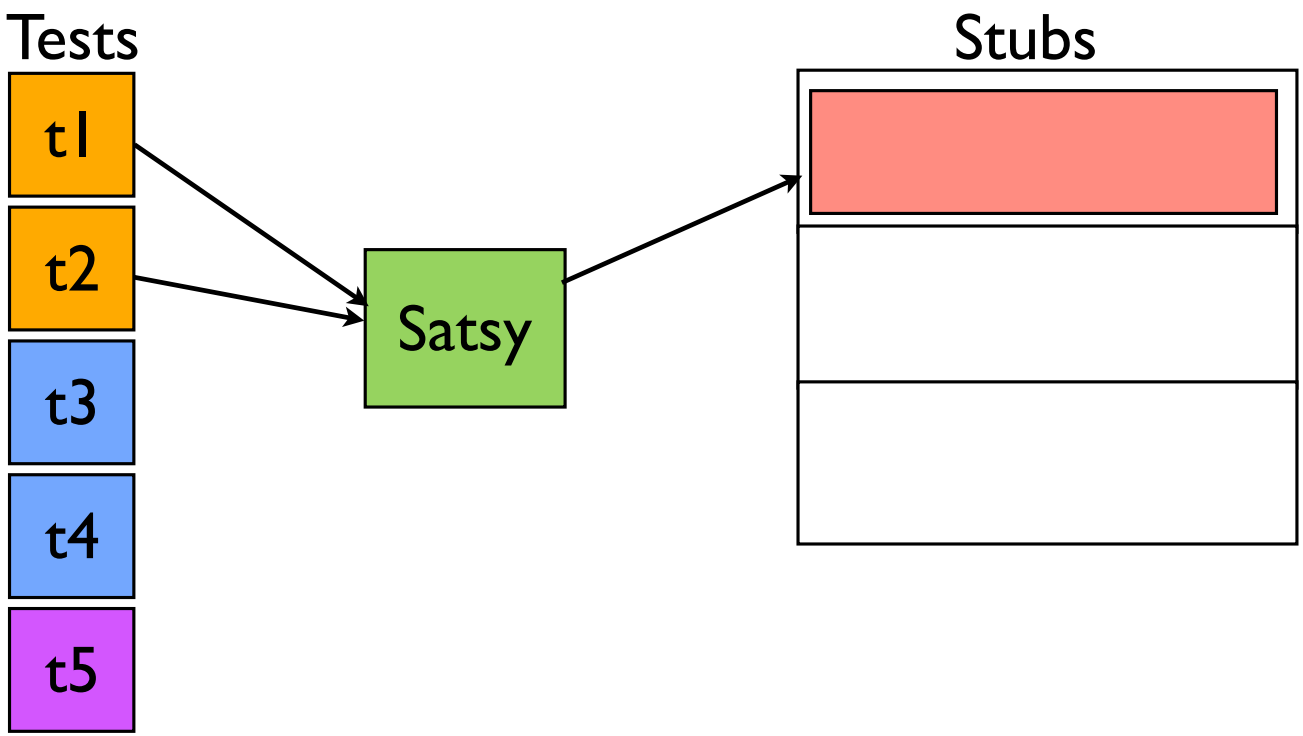
Test-Driven Development



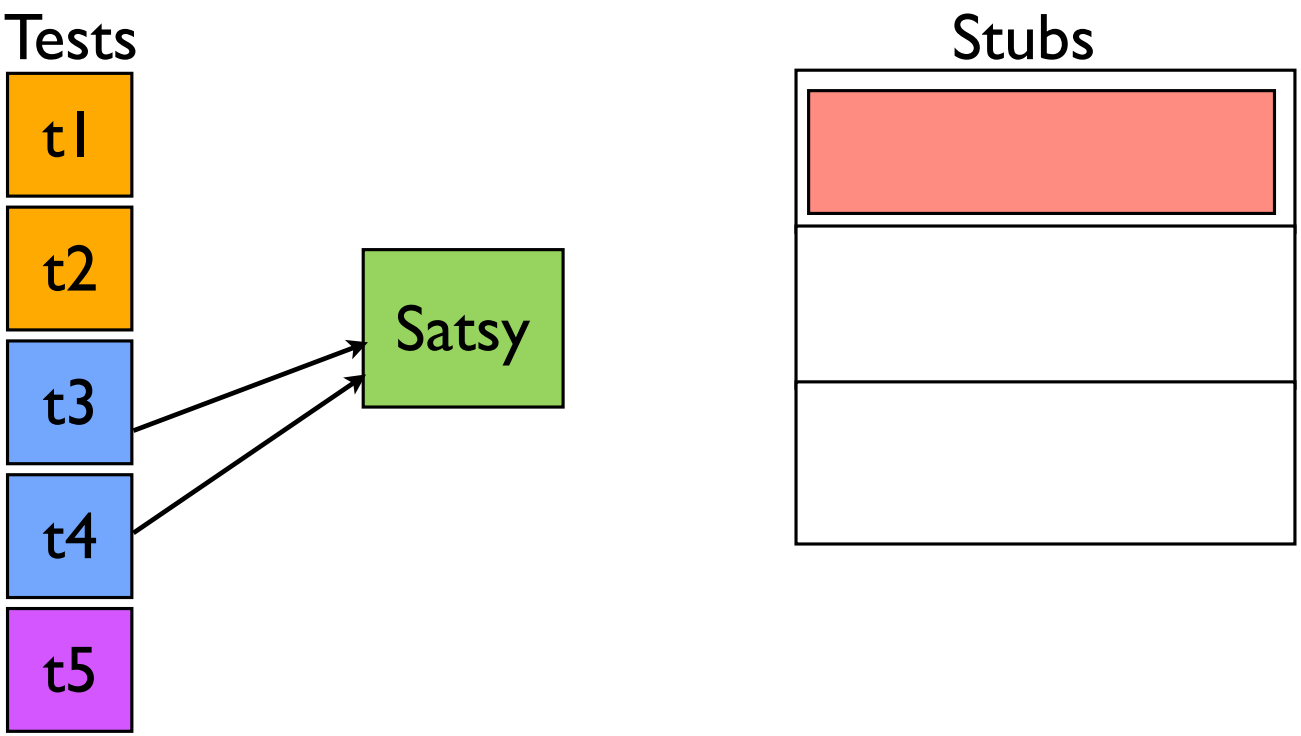
Test-Driven Development



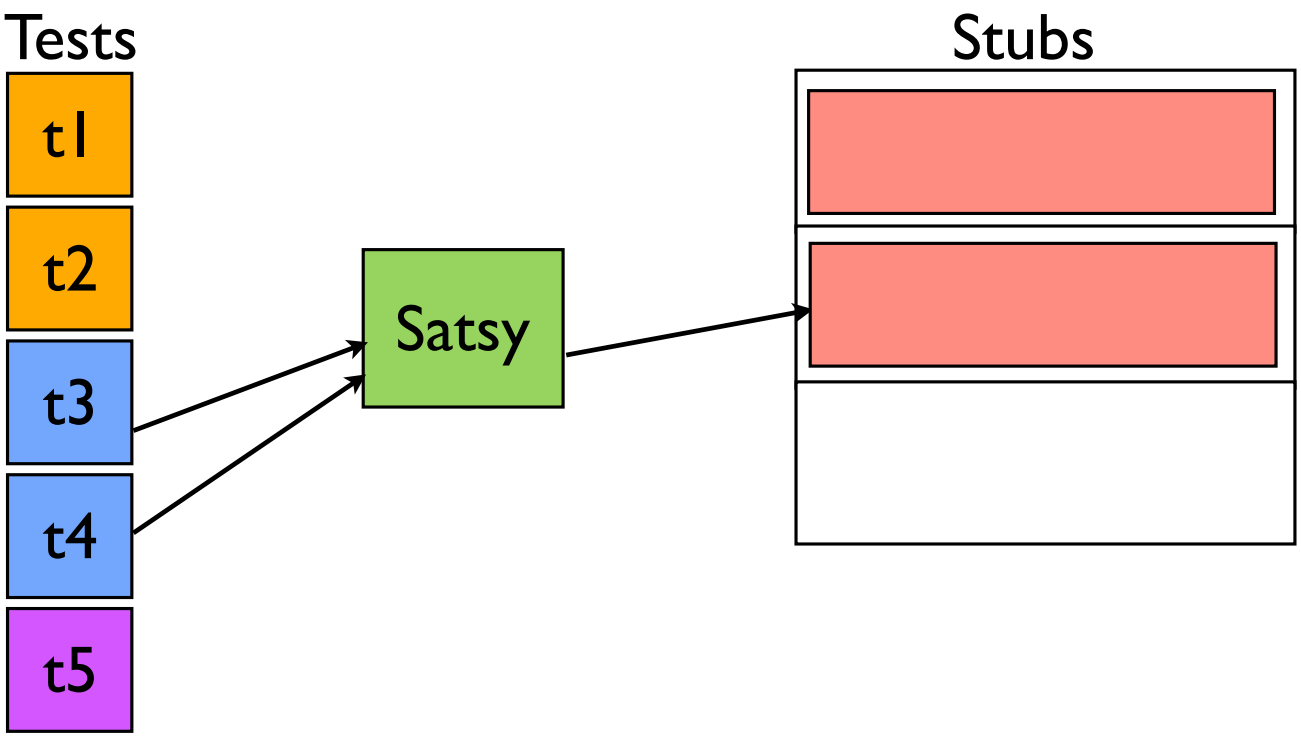
Test-Driven Development



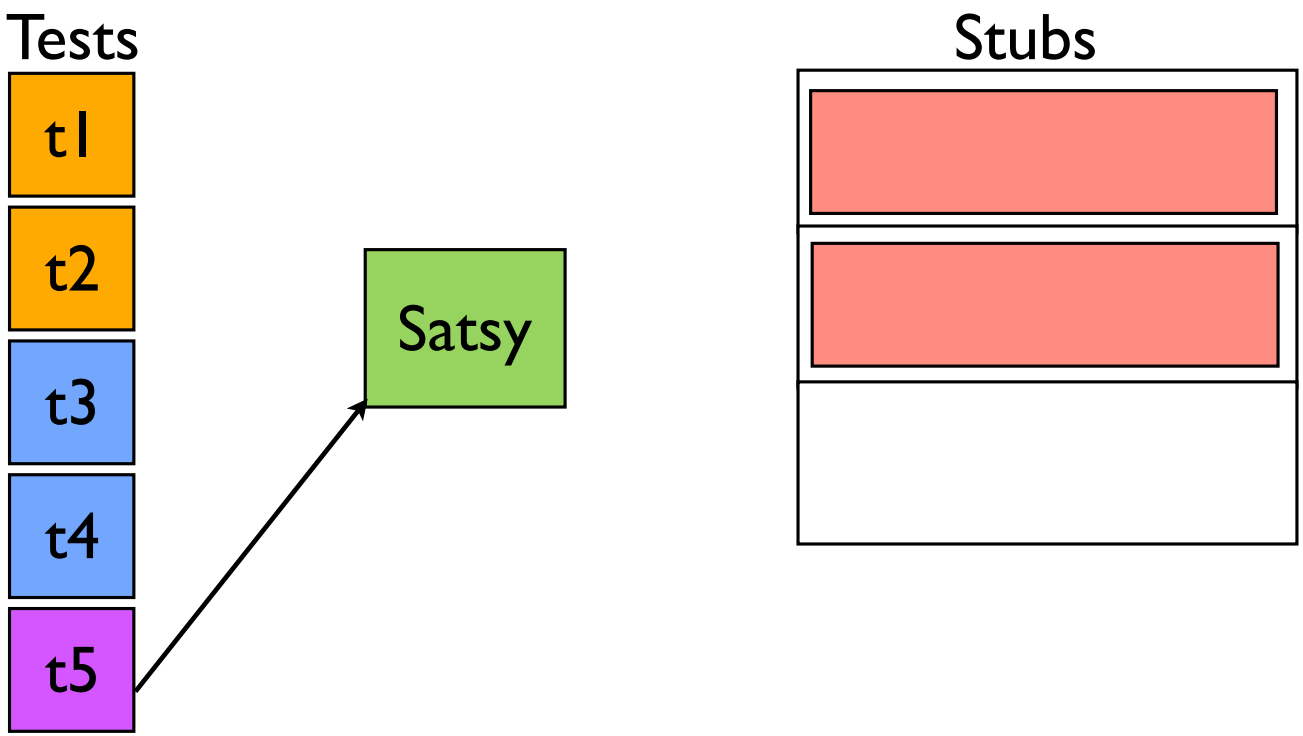
Test-Driven Development



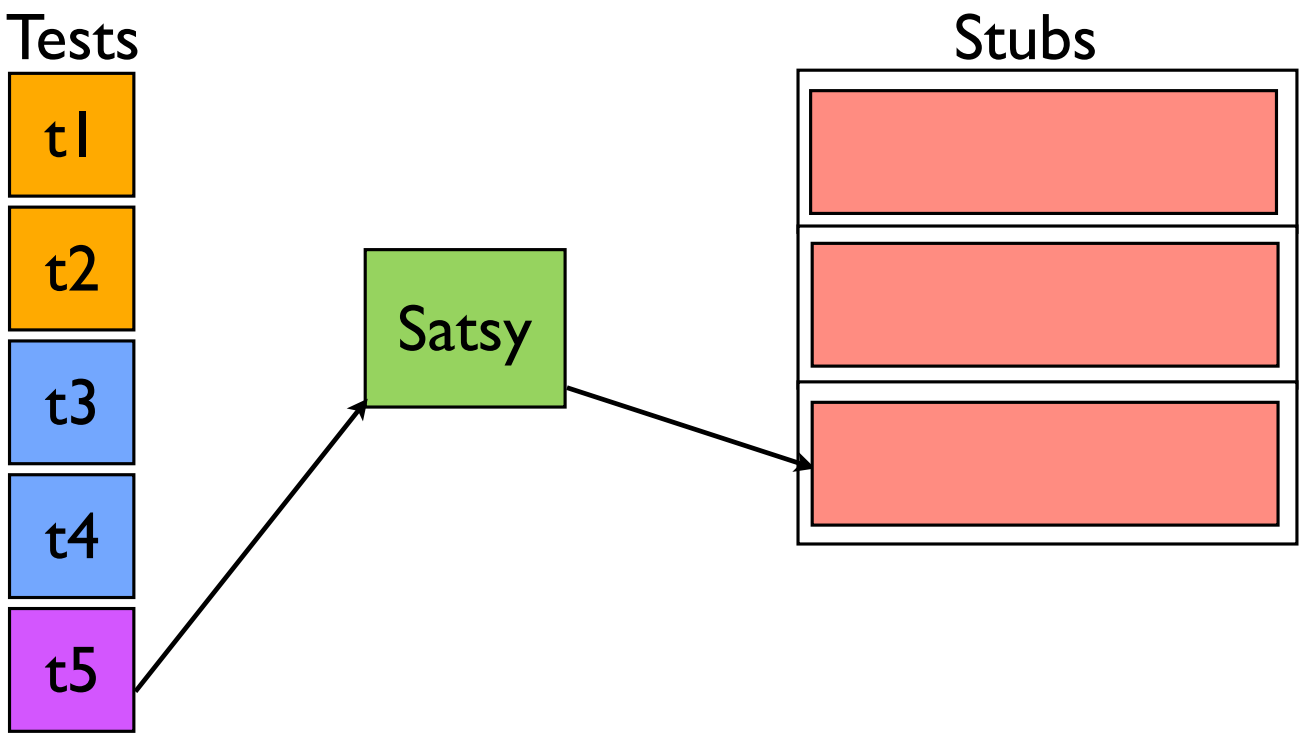
Test-Driven Development



Test-Driven Development



Test-Driven Development



Recap

- Introduced a code search approach using solvers and input/output
- Defined a notion of partial matching for ranking
- Evaluated the approach against Google and Merobase
- Discussed potential applications

Acknowledgements

- Collaborators: Sebastian Elbaum, Matthew Dwyer
- Funding:
  - NSF Graduate Research Fellowship
  - NSF SHF-1218265
  - AFOSR #9550-10-1-0406

Searching for Source Code with Constrained Semantic Search

Kathryn T. Stolee
Iowa State University
kstolee@iastate.edu

November 19, 2013