Programs that write themselves: Program synthesis for the masses

Yu Feng
UT Austin
New platforms impose demand for programming
New platforms impose demand for programming
New platforms impose demand for programming
New platforms impose demand for programming
New platforms impose demand for programming
New platforms impose demand for programming
Program Synthesis

Program synthesis makes programming more accessible.
Program Synthesis

Program synthesis make programming more accessible

Astroid, NDSS’17
Program Synthesis

Program synthesis make programming more accessible

Malware Signature

Astroid, NDSS’17

Algorithmic Program

SyPet, POPL’17
Program Synthesis

Program synthesis make programming more accessible

Malware Signature
Astroid, NDSS’17

Algorithmic Program
SyPet, POPL’17

Data Wrangling
Morpheus, PLDI’17
Outline
Outline

• Morpheus: automating data wrangling tasks in data science (Completed)
Outline

• Morpheus: automating data wrangling tasks in data science (Completed)

• SyPet: generating complex programs from non-trivial APIs (Completed)
Outline

• Morpheus: automating data wrangling tasks in data science (Completed)

• SyPet: generating complex programs from non-trivial APIs (Completed)

• Neo: a CDCL constraint solver for general synthesis problems (Ongoing)
Data science is important in many application domains
Data science is important in many application domains
Data science is important in many application domains
Data science is important in many application domains. 

AI

Market analysis

Intrusion Detection
Data science is important in many application domains

AI

Market analysis

Intrusion Detection

Recommender System
Data science in reality…
Data extraction

Data science in reality...
Data science in reality...
Data science in reality…

Data extraction
Data cleaning
Table transformation

Data science in reality…
Data science in reality...

Data scientists spend 80% of time on tedious data wrangling tasks
Morpheus at a glance

Morpheus Synthesizer
Morpheus at a glance
Morpheus at a glance
Morpheus at a glance

Table transformation
Table reshaping
Data consolidation
Data computation
...

Morpheus Synthesizer
A general framework

End user → Morpheus Synthesizer → R
A general framework
A general framework
A general framework
A general framework

Can be easily instantiated for any set of components!
Key contributions
Key contributions

 Novel component-based synthesis algorithm
that utilizes overapproximate component
specifications
Key contributions

- Novel component-based synthesis algorithm that utilizes overapproximate component specifications
- Core algorithm based on enumerative search, but uses SMT-based deduction over component specs to reject partial programs
Key contributions

 Novel component-based synthesis algorithm that utilizes overapproximate component specifications

 Core algorithm based on enumerative search, but uses SMT-based deduction over component specs to reject partial programs

 Further increases power of deductive reasoning by performing partial evaluation
Why incomplete specs
Why incomplete specs

Unlike prior approaches (Brahma, Synquid), Morpheus doesn’t require precise specs
Why incomplete specs

Unlike prior approaches (Brahma, Synquid), Morpheus doesn’t require precise specs

Difficult or impossible to write precise specs in many cases (e.g. table pivoting, reshaping)
High-level synthesis algorithm
High-level synthesis algorithm

Input

Hypothesis Generation
High-level synthesis algorithm
High-level synthesis algorithm

Input -> Hypothesis Generation -> Partial program -> SMT-based Deduction

Prune partial programs
High-level synthesis algorithm

Input

Hypothesis Generation

Prune partial programs

Partial program

SMT-based Deduction

Partial program

Type-directed Completion
High-level synthesis algorithm

Input

Hypothesis Generation

Partial program

Prune partial programs

SMT-based Deduction

Strengthen deduction via PE

Type-directed Completion

Partial Evaluation
High-level synthesis algorithm

Input

Hypothesis Generation

Prune partial programs

Partial program

SMT-based Deduction

Partial program

Type-directed Completion

Strengthen deduction via PE

Partial Evaluation

Full program
High-level synthesis algorithm

- **Input**
- **Hypothesis Generation**
- **Partial program**
- **Type-directed Completion**
- **Partial Evaluation**
- **Full program**
- **SMT-based Deduction**
- **Strengthen deduction via PE**
- **Prune partial programs**
Hypotheses

Input → Hypothesis Generation → Partial program → SMT-based Deduction → Partial Evaluation → Full program

- Prune partial programs
- Type-directed Completion
- Strengthen deduction via PE
Hypotheses

- Hypothesis Generation
- Partial program
- SMT-based Deduction
- Prune partial programs
- Partial Evaluation
- Type-directed Completion
- Strengthen deduction via PE
- Full program

Input

π
Hypotheses
SMT-based deduction
SMT-based deduction

Hypothesis Generation → Partial program → Type-directed Completion → Partial Evaluation → SMT-based Deduction → Prune partial programs → Partial program → Partial program → Strengthen deduction via PE → Full program → Input

\[ \pi_3 \]

\[ \sigma_2 \]

\[ ?_1 \]

\[ ? \]
SMT-based deduction

π

out.row = in.row &
out.col < in.col
SMT-based deduction

\[ \sigma \text{ out.row < in.row & out.col = in.col} \]

\[ \pi \text{ out.row = in.row & out.col < in.col} \]
SMT-based deduction
SMT-based deduction

\[ \pi_3 \sigma_2 ? \]

\[ \sigma_2 ? \]

\[ ?_1 ? \]

\[ T3.\text{row} = T2.\text{row} \land T3.\text{col} < T2.\text{col} \land T2.\text{row} < T1.\text{row} \land T2.\text{col} = T1.\text{col} \]
SMT-based deduction

| 2 | Bob  | 18  | 3.2 |
| 3 | Tom  | 2   | 3.0 |

T3 (output): 2X4

T3.row=2, T3.col=4

T3.row = T2.row &
T3.col < T2.col &
T2.row < T1.row &
T2.col = T1.col
SMT-based deduction

\[
T_3 \text{.row} = T_2 \text{.row} & T_3 \text{.col} < T_2 \text{.col} & T_2 \text{.row} < T_1 \text{.row} & T_2 \text{.col} = T_1 \text{.col}
\]

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>

T3 (output): 2X4

T3.row=2,T3.col=4

T3.row = T2.row & T3.col < T2.col & T2.row < T1.row & T2.col = T1.col
SMT-based deduction

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>8</td>
<td>4.0</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>

T1 (input): 3X4

\[ \pi_3 \circ_2 \]

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>

T3 (output): 2X4

\[ T3.\text{row}=2, T3.\text{col}=4 \]

\[ T3.\text{row} = T2.\text{row} \land T3.\text{col} < T2.\text{col} \land T2.\text{row} < T1.\text{row} \land T2.\text{col} = T1.\text{col} \]
SMT-based deduction

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>8</td>
<td>4.0</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>

T1 (input): 3X4

T3 (output): 2X4

T3.row = 2, T3.col = 4
T3.row = T2.row & T3.col < T2.col & T2.row < T1.row & T2.col = T1.col
T1.row = 3, T1.col = 4
SMT-based deduction

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>8</td>
<td>4.0</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>

T1 (input): 3X4

\[ T_3 \text{.row} = T_2 \text{.row} \& T_3 \text{.col} < T_2 \text{.col} \& T_2 \text{.row} < T_1 \text{.row} \& T_2 \text{.col} = T_1 \text{.col} \]

T1.row=3, T1.col=4

T3 (output): 2X4

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>

T3.row=2, T3.col=4
SMT-based deduction

1 sketch > 100 programs

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>8</td>
<td>4.0</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>

T1 (input): 3X4

T3 (output): 2X4

T3.row = 2, T3.col = 4
T3.row = T2.row & T3.col < T2.col & T2.row < T1.row & T2.col = T1.col
T1.row = 3, T1.col = 4
Partial evaluation

Given hole of type T, complete hole with possible inhabitants of T
Partial evaluation

Given hole of type T, complete hole with possible inhabitants of T
Partial evaluation

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>8</td>
<td>4.0</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>

\[ \pi_3 \sigma_2 ? \]

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ \text{id} \quad \text{name} \quad \text{age} \]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
</tr>
</tbody>
</table>
### Partial evaluation

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>8</td>
<td>4.0</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>
```

\[
\sigma_{2}^{?} \rightarrow \pi_{3}
\]

#### Query

\[
age > 8
\]

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>
```
Partial evaluation

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>8</td>
<td>4.0</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>

\[ \sigma_{age > 8} \]

\[ \pi_{3} \]

\[ T3.row=2, \; T3.col=3 \]

\[ T3.row = T2.row \; \& \; T3.col < T2.col \; \& \; \ldots \]
Partial evaluation

\[
\pi_3 \sigma_2 \text{age > 8}
\]

T3.row = 2, T3.col = 3
T3.row = T2.row &
T3.col < T2.col &
...

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>8</td>
<td>4.0</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>8</td>
<td>4.0</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>
Partial evaluation

\[ \pi_3 \]

Partial evaluation to strengthen current constraints

\( T3.row=2, \ T3.col=3 \)
\( T3.row = T2.row \ \& \ \ T3.col < T2.col \ \& \ \ldots \)
Partial evaluation

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
</tr>
</tbody>
</table>
```

\(\pi_3\)

Partial evaluation to strengthen current constraints

\(T_3.\text{row}=2, T_3.\text{col}=3\)
\(T_3.\text{row} = T_2.\text{row} \& T_3.\text{col} < T_2.\text{col} \& T_2.\text{row}=1, T_2.\text{col}=4\)
Partial evaluation

Partial evaluation + SMT-based deduction

Prune more programs!

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Bob</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>Tom</td>
<td>2</td>
</tr>
</tbody>
</table>

T3.row=2, T3.col=3
T3.row = T2.row &
T3.col < T2.col &
T2.row=1, T2.col=4

Partial evaluation to strengthen current constraints
Experiments
Experiments

RQ1: Can Morpheus efficiently automate real-world data wrangling tasks?
Experiments

★ RQ1: Can Morpheus efficiently automate real-world data wrangling tasks?
★ RQ2: How effective are SMT-based deduction and partial evaluation?
Experiments

★ RQ1: Can Morpheus efficiently automate real-world data wrangling tasks?
★ RQ2: How effective are SMT-based deduction and partial evaluation?
★ RQ3: How does Morpheus compare with other synthesis tools?
Data wrangling tasks

How to write a program to generate this table

How to reshape table using dplyr
Data wrangling tasks

- How to write a program to generate this table
- How to reshape table using dplyr

🌱 Collect 80 data wrangling tasks from Stackoverflow
Data wrangling tasks

- Collect 80 data wrangling tasks from Stackoverflow
- Morpheus successfully synthesized the correct implementation of 78 benchmarks
Data wrangling tasks

- Collect 80 data wrangling tasks from Stackoverflow
- Morpheus successfully synthesized the correct implementation of 78 benchmarks
- Median synthesis time 3.59 seconds
Deduction & Partial evaluation

Cumulative running time of Morpheus
Deduction & Partial evaluation

Cumulative running time of Morpheus
Deduction & Partial evaluation

Deduction and PE are important!

Cumulative running time of Morpheus
Compare with other tool
Compare with other tool

- No existing tool for synthesizing data wrangling tasks in R
Compare with other tool

- No existing tool for synthesizing data wrangling tasks in R
- Evaluate Morpheus on 28 benchmarks used in evaluating SQLSynthesizer (ASE’13)
Compare with other tool

- No existing tool for synthesizing data wrangling tasks in R
- Evaluate Morpheus on 28 benchmarks used in evaluating SQLSynthesizer (ASE’13)
- Each benchmark requires synthesizing SQL queries from input-output examples
Compare with other tool
Compare with other tool

Comparison with SQLSynthesizer

Percentage

0

0.25

0.5

0.75

1

SQL

SQLSynthesizer

Morpheus
Compare with other tool

Comparison with SQLSynthesizer

- SQLSynthesizer: 71.4%
- Morpheus: 96.4%
Compare with other tool

Morpheus outperforms state-of-the-art!

Comparison with SQLSynthesizer

- SQLSynthesizer: 96.4%
- Morpheus: 96.4%

Morpheus outperforms state-of-the-art!
Component-based Synthesis
Component-based Synthesis

Morpheus  =  Incomplete FOL
Component-based Synthesis

Morpheus = Incomplete FOL

SyPet = \( \lambda \)
Outline

• Morpheus: automating data wrangling tasks in data science (Completed)

• SyPet: generating complex programs from non-trivial APIs (Completed)

• Neo: a CDCL constraint solver for general synthesis problems (Ongoing)
A Motivating Example

• Consider rotating some object using a Java API

\textit{Area rotate(Area obj, Point 2D pt, double angle)}
A Motivating Example

• Consider rotating some object using a Java API

  ```java
  Area rotate(Area obj, Point 2D pt, double angle)
  ```

• Possible to do this using java.awt.geom, but not trivial:

  ```java
  AffineTransform at = new AffineTransform();

  double x = pt.getX();

  double y = pt.getY();

  at.setToRotation(angle, x, y);

  Area tr = obj.createTransformedArea(at);

  return tr;
  ```
A Motivating Example

• Consider rotating some object using a Java API

**Area rotate(Area obj, Point 2D pt, double angle)**

• Possible to do this using java.awt.geom, but not trivial:

```java
AffineTransform at = new AffineTransform();

double x = pt.getX();

double y = pt.getY();

at.setToRotation(angle, x, y);

Area tr = obj.createTransformedArea(at);

return tr;
```
A Motivating Example

• Consider rotating some object using a Java API

```java
Area rotate(Area obj, Point 2D pt, double angle)
```

• Possible to do this using java.awt.geom, but not trivial:

```java
AffineTransform at = new AffineTransform();

double x = pt.getX();

double y = pt.getY();

at.setToRotation(angle, x, y);

Area tr = obj.createTransformedArea(at);

return tr;
```
A Motivating Example

• Consider rotating some object using a Java API

\[
\text{Area rotate} \text{(Area obj, Point 2D pt, double angle)}
\]

• Possible to do this using java.awt.geom, but not trivial:

```java
AffineTransform at = new AffineTransform();

double x = pt.getX();

double y = pt.getY();

at.setToRotation(angle, x, y);

Area tr = obj.createTransformedArea(at);

return tr;
```
A Motivating Example

• Consider rotating some object using a Java API

\[
\text{Area rotate}(\text{Area } \text{obj}, \text{ Point 2D } \text{pt}, \text{ double } \text{angle})
\]

• Possible to do this using java.awt.geom, but not trivial:

\[
\text{AffineTransform } \text{at} = \new \text{AffineTransform}();
\]
\[
\text{double } \text{x} = \text{pt}.\text{getX}();
\]
\[
\text{double } \text{y} = \text{pt}.\text{getY}();
\]
\[
\text{at}.\text{setToRotation}(\text{angle}, \text{x}, \text{y});
\]
\[
\text{Area } \text{tr} = \text{obj}.\text{createTransformedArea}(\text{at});
\]
\[
\text{return } \text{tr};
\]

Finally get the object
Our Goal
Our Goal

• Using an API to achieve a conceptually simple task may be quite hard
Our Goal

• Using an API to achieve a conceptually simple task may be quite hard

• Our goal: Use type- and example-directed program synthesis to help programmers use APIs
Our Goal

• Using an API to achieve a conceptually simple task may be quite hard

• Our goal: Use type- and example-directed program synthesis to help programmers use APIs
Our Goal

• Using an API to achieve a conceptually simple task may be quite hard

• Our goal: Use type- and example-directed program synthesis to help programmers use APIs

• Want to use this approach to synthesize programs over any API (any set of components)
Why is this hard?
Why is this hard?

- **Number of components**: Much previous work on component-based synthesis, but typically work for a very small (< 20) number of components
Why is this hard?

- **Number of components**: Much previous work on component-based synthesis, but typically work for a very small (< 20) number of components

- **Generality**: Unlike many API completion tools, want to synthesize multi-statement code snippets and allow components with side effects and multiple arguments
Why is this hard?

- **Number of components**: Much previous work on component-based synthesis, but typically work for a very small (< 20) number of components.

- **Generality**: Unlike many API completion tools, want to synthesize multi-statement code snippets and allow components with side effects and multiple arguments.

Given type signature $T$, need smart way to enumerate all well-typed programs of type $T$ over the given components!
Our Solution
Our Solution

Use Petri net reachability analysis to look for well-typed programs of the desired type
Our Solution

Use Petri net reachability analysis to look for well-typed programs of the desired type

- Model relationships between components using Petri net
Our Solution

Use Petri net reachability analysis to look for well-typed programs of the desired type

• Model relationships between components using Petri net
• Use type signature of desired method to mark initial and target configurations
Our Solution

Use Petri net reachability analysis to look for well-typed programs of the desired type

- Model relationships between components using Petri net
- Use type signature of desired method to mark initial and target configurations
- Perform reachability analysis to find valid sequences of method calls
Primer on Petri nets
Primer on Petri nets

- Petri net is a generalized graph with two kinds of nodes: places and transitions
• Petri net is a generalized graph with two kinds of nodes: places and transitions
• Each place contains zero or more tokens; edges are labeled with a number (of tokens)
Primer on Petri nets

- Petri net is a generalized graph with two kinds of nodes: places and transitions.
- Each place contains zero or more tokens; edges are labeled with a number (of tokens).
- A transition T can fire if, for each edge (p,T) with label n, place p contains at least n tokens.
Primer on Petri nets

- Petri net is a generalized graph with two kinds of nodes: places and transitions
- Each place contains zero or more tokens; edges are labeled with a number (of tokens)
- A transition T can fire if, for each edge (p,T) with label n, place p contains at least n tokens
- Firing a transition T consumes (resp. produces) the indicated number of tokens at the source (resp. target) nodes
Primer on Petri nets

- Petri net is a generalized graph with two kinds of nodes: places and transitions.
- Each place contains zero or more tokens; edges are labeled with a number (of tokens).
- A transition $T$ can fire if, for each edge $(p,T)$ with label $n$, place $p$ contains at least $n$ tokens.
- Firing a transition $T$ consumes (resp. produces) the indicated number of tokens at the source (resp. target) nodes.
Reachability Problem in Petri nets
Reachability Problem in Petri nets

- Reachability problem: Given a Petri net with initial marking $M$ and a target marking $M'$, is it possible to obtain $M'$ by firing a sequencing of transitions?
Reachability Problem in Petri nets

- Reachability problem: Given a Petri net with initial marking $M$ and a target marking $M'$, is it possible to obtain $M'$ by firing a sequencing of transitions?
- Example: Consider marking $M'$: $[P1 \rightarrow 0, P2 \rightarrow 0, P3 \rightarrow 1]$. 
Reachability Problem in Petri nets

- Reachability problem: Given a Petri net with initial marking $M$ and a target marking $M'$, is it possible to obtain $M'$ by firing a sequencing of transitions?
- Example: Consider marking $M'$: $[P1 \rightarrow 0, P2 \rightarrow 0, P3 \rightarrow 1]$. This marking is reachable, and accepting run is $T1, T1, T2$. 
Reachability Problem in Petri nets

- Reachability problem: Given a Petri net with initial marking $M$ and a target marking $M'$, is it possible to obtain $M'$ by firing a sequencing of transitions?
- Example: Consider marking $M' : [P1 \rightarrow 0, P2 \rightarrow 0, P3 \rightarrow 1]$.
- This marking is reachable, and accepting run is $T1, T1, T2$. 
Reachability Problem in Petri nets

- Reachability problem: Given a Petri net with initial marking $M$ and a target marking $M'$, is it possible to obtain $M'$ by firing a sequencing of transitions?

- Example: Consider marking $M'$: $\left[ P1 \rightarrow 0, P2 \rightarrow 0, P3 \rightarrow 1 \right].$

- This marking is reachable, and accepting run is $T1, T1, T2.$
Reachability Problem in Petri nets

- Reachability problem: Given a Petri net with initial marking $M$ and a target marking $M'$, is it possible to obtain $M'$ by firing a sequencing of transitions?
- Example: Consider marking $M'$: $[P1 \rightarrow 0, P2 \rightarrow 0, P3 \rightarrow 1]$. This marking is reachable, and accepting run is $T1, T1, T2$. 
Algorithm Overview
Algorithm Overview

APIs

Construct
Petri net
Algorithm Overview

- APIs
- Construct Petri net
- Init/target markings
- Signature $\tau_1 \rightarrow \tau_2$
Algorithm Overview

- APIs
- Construct Petri net
- Init/target markings
- Signature $\tau_1 \rightarrow \tau_2$
- Reachability analysis
- Candidate Sketch
Algorithm Overview

APIs

Construct Petri net

Init/target markings

Signature $\tau_1 \rightarrow \tau_2$

Reachability analysis

Candidate Sketch

Candidate Program

Sketch Completion
Algorithm Overview

- API
- Construct Petri net
- Reachability analysis
- Candidate Sketch
- Sketch Completion
- Candidate Program
- Check Candidate
- JUnit

Init/target markings

Signature $\tau_1 \rightarrow \tau_2$
Algorithm Overview

- Construct Petri net
- APIs
- Reachability analysis
- Candidate Sketch
- Sketch Completion
- Candidate Program
- Check Candidate
- JUnit
- Init/target markings
- Init/target markings
- Signature $\tau_1 \rightarrow \tau_2$
- Backtrack
Algorithm Overview

- Construct Petri net
- APIs
- Reachability analysis
- Candidate Sketch
- Candidate Program
- Sketch Completion
- Check Candidate
- JUnit
- Java
- Init/target markings
- Signature \( \tau_1 \rightarrow \tau_2 \)
- Backtrack
Petri net Construction

class CPt {
    CPt(Int x, Int y, Color c);
    Int getX();
    void setColor(Color c);
    ...
}

Int

Color

CPt

void

32
class CPt {
    CPt(Int x, Int y, Color c);
    Int getX();
    void setColor(Color c);
    ...
}

Petri net Construction
class CPt {
    CPt(Int x, Int y, Color c);
    Int getX();
    void setColor(Color c);
    ...
}

Petri net Construction
class CPt {
    CPt(Int x, Int y, Color c);
    Int getX();
    void setColor(Color c);
    ...
}
Clone Transitions

Diagram:
- Int
- Color
- CPt
- CPt(..)
- void
- setColor
- getX

Relationships:
- Int to CPt
- Color to CPt
- CPt to CPt(..)
- CPt to void
- CPt(..) to setColor
- void to void

Arrows:
- 1

Numbers:
- 1
- 2
- 3
- 4

Legend:
- CPt
- CPt(..)
- void
- setColor
- Int
- Color
- void
Clone Transitions

• Our construction so far views objects as “resources” — every method “consumes” and “produces” objects
Clone Transitions

• Our construction so far views objects as “resources”– every method “consumes” and “produces” objects
• But in conventional languages, we can reuse objects!
Clone Transitions

- Our construction so far views objects as “resources”—every method “consumes” and “produces” objects
- But in conventional languages, we can reuse objects!
- Therefore, augment Petri net model with clone transitions
Clone Transitions

- Our construction so far views objects as “resources” – every method “consumes” and “produces” objects
- But in conventional languages, we can reuse objects!
- Therefore, augment Petri net model with clone transitions
Clone Transitions

• Our construction so far views objects as “resources” — every method “consumes” and “produces” objects
• But in conventional languages, we can reuse objects!
• Therefore, augment Petri net model with **clone transitions**
Clone Transitions

- Our construction so far views objects as “resources”—every method “consumes” and “produces” objects
- But in conventional languages, we can reuse objects!
- Therefore, augment Petri net model with **clone transitions**
Clone Transitions

• Our construction so far views objects as “resources” – every method “consumes” and “produces” objects
• But in conventional languages, we can reuse objects!
• Therefore, augment Petri net model with **clone transitions**
Initial and Target Markings

Use signature to determine initial and target markings of Petri net
Initial and Target Markings

Use signature to determine initial and target markings of Petri net

```cpp
CPt shift (CPt p, Int shiftX, Int shiftY)
```

![Petri net diagram]
Initial and Target Markings

Use signature to determine initial and target markings of Petri net

\[
\text{CPt shift (CPt p, Int shiftX, Int shiftY)}
\]
Initial and Target Markings

Use signature to determine initial and target markings of Petri net

\[ \text{CPt} \text{ shift} (\text{CPt} \ p, \ \text{Int} \ \text{shiftX}, \ \text{Int} \ \text{shiftY}) \]
Initial and Target Markings

Use signature to determine initial and target markings of Petri net

\[
\text{CPT shift (CPT p, Int shiftX, Int shiftY)}
\]
Initial and Target Markings

Use signature to determine initial and target markings of Petri net

\texttt{CPT \textbf{shift} (CPT \texttt{p}, \texttt{Int} \texttt{shiftX}, \texttt{Int} \texttt{shiftY})}

Target marking:
Initial and Target Markings

Use signature to determine initial and target markings of Petri net

\[
\text{CPt shift (CPt } p, \text{ Int shiftX, Int shiftY)}
\]

Target marking:
\[
\text{Cpt } = 1
\]
Initial and Target Markings

Use signature to determine initial and target markings of Petri net

CPt shift (CPt p, Int shiftX, Int shiftY)

Target marking:
Cpt = 1
void = *

[Diagram showing Petri net with labels and transitions]
Initial and Target Markings

Use signature to determine initial and target markings of Petri net

Cpt shift (Cpt p, Int shiftX, Int shiftY)

Target marking:

Cpt = 1
void = *
int = 0
Initial and Target Markings

Use signature to determine initial and target markings of Petri net

\[ \text{CPt shift (CPt p, Int shiftX, Int shiftY)} \]

Target marking:
- \( \text{Cpt} = 1 \)
- \( \text{void} = \ast \)
- \( \text{int} = 0 \)
- \( \text{color} = 0 \)
Initial and Target Markings

Use signature to determine initial and target markings of Petri net

C Pt shift (C Pt p, Int shiftX, Int shiftY)

Target marking:
- C Pt = 1
- void = *
- int = 0
- color = 0

All args must be used!
Next Step
Reachability Analysis
Reachability Analysis

All accepting runs of Petri net correspond to method call sequences with desired type signature!
Reachability Analysis

All accepting runs of Petri net correspond to method call sequences with desired type signature!

- Need to perform reachability analysis to identify accepting runs of the Petri Net
Reachability Analysis

All accepting runs of Petri net correspond to method call sequences with desired type signature!

- Need to perform reachability analysis to identify accepting runs of the Petri Net
- Furthermore, need to do this lazily because there may be many accepting runs
Reachability Analysis

All accepting runs of Petri net correspond to method call sequences with desired type signature!

• Need to perform reachability analysis to identify accepting runs of the Petri Net
• Furthermore, need to do this lazily because there may be many accepting runs
• Our solution reduces reachability analysis to integer linear programming (ILP) ⇒ solution corresponds to shortest sequence of method calls
Accepting Run as Program Sketch
Accepting Run as Program Sketch

Accepting run of Petri net corresponds to program sketch because it does not specify arguments of each method call.
Sketch Completion

```java
x = #1.getX(); y = #2.getY();
#3.setToRotation(#4, #5, #6);
a = #7.createTransformedArea(#8); return #9;
```
Sketch Completion

Given a program sketch with holes, need to instantiate each hole with a program variable such that program type checks

```java
x = #1.getX(); y = #2.getY();
#3.setToRotation(#4, #5, #6);
a = #7.createTransformedArea(#8);
return #9;
```
Sketch Completion

x = #1.getX(); y = #2.getY();
#3.setToRotation(#4, #5, #6);
a = #7.createTransformedArea(#8);
return #9;

- Given a program sketch with holes, need to instantiate each hole with a program variable such that program type checks
- Encode this as a boolean satisfiability problem:
Sketch Completion

• Given a program sketch with holes, need to instantiate each hole with a program variable such that program type checks

• Encode this as a boolean satisfiability problem:

\[
\forall \#i \in H. \forall v \in getV(V,\#i). \sum h_v^{\#i} = 1
\]

\[
\forall v \in V. \forall \#i \in getH(H,v). \sum h_v^{\#i} \geq 1
\]

```c
x = #1.getX(); y = #2.getY();
#3.setToRotation(#4, #5, #6);
a = #7.createTransformedArea(#8);
return #9;
```
Sketch Completion

- Given a program sketch with holes, need to instantiate each hole with a program variable such that program type checks
- Encode this as a boolean satisfiability problem:

\[
\forall #i \in H \cdot \forall v \in getV(V, #i). \sum h_v^{#i} = 1
\]

\[
\forall v \in V \cdot \forall #i \in getH(H, v). \sum h_v^{#i} \geq 1
\]
Sketch Completion

Given a program sketch with holes, need to instantiate each hole with a program variable such that program type checks.

Encode this as a boolean satisfiability problem:

$$\forall_{i \in H} \forall_{v \in \text{getV}(V, i)} \sum h_v^{#i} = 1$$

$$\forall_{v \in V} \forall_{i \in \text{getH}(H, v)} \sum h_v^{#i} \geq 1$$
Evaluation

• Collected 30 API-usage questions from Stackoverflow involving six different libraries with 751-9578 methods:
Evaluation

- Collected 30 API-usage questions from Stackoverflow involving six different libraries with 751-9578 methods:

- Extracted signature and test case from post if available, otherwise wrote it ourselves
Evaluation

• Collected 30 API-usage questions from Stackoverflow involving six different libraries with 751-9578 methods:

  - How do I determine if a given year is a leap year using joda library?
  - How can I solve a system of linear equations using the Apache math library?

• Extracted signature and test case from post if available, otherwise wrote it ourselves

• Used SyPet to automatically synthesize the implementation
Results

How do I determine if a given year is a leap year using Joda library?

How can I solve a system of linear equations using the Apache math library?
Results

- Our technique was able to successfully synthesize the correct implementation of all 30 benchmarks
Our technique was able to successfully synthesize the correct implementation of all 30 benchmarks.

Median synthesis time 1.57 seconds.
Results

• Our technique was able to successfully synthesize the correct implementation of all 30 benchmarks

• Median synthesis time 1.57 seconds

Our synthesis technique is useful to programmers
Comparison with Other Tools
Comparison with Other Tools

• Also compared SyPet with two other synthesis tools, CodeHint and InSynth
Comparison with Other Tools

- Also compared SyPet with two other synthesis tools, CodeHint and InSynth
Comparison with Other Tools

- Also compared SyPet with two other synthesis tools, CodeHint and InSynth.
Where are we?
Where are we?

State-of-the-art tools (Deepcoder, Morpheus) still cannot scale to large programs (< 5 components)
Where are we?

State-of-the-art tools (Deepcoder, Morpheus) still can not scale to large programs (< 5 components)

Machine learning (Deepcoder) Deduction (L2, Synquid)

Components \rightarrow Synthesizer \rightarrow Program

Specification
Where are we?

State-of-the-art tools (Deepcoder, Morpheus) still can not scale to large programs (< 5 components)

Most of existing tools focus on pruning search space

- Machine learning (Deepcoder)
- Deduction (L2, Synquid)
Where are we?

State-of-the-art tools (Deepcoder, Morpheus) still can not scale to large programs (< 5 components)

Most of existing tools focus on pruning search space

Machine learning (Deepcoder)
Deduction (L2, Synquid)

How do we learn from mistakes?
Outline

• Morpheus: automating data wrangling tasks in data science (Completed)

• SyPet: generating complex programs from non-trivial APIs (Completed)

• Neo: a CDCL constraint solver for general synthesis problems (Ongoing)
Conflict-driven clause learning

\[ \phi \]

- **Decide**
  - Current decision
  - No conflict
  - New lemma
- **Deduce**
  - Conflict
- **Analyze Conflict**
  - SAT
  - UNSAT
Conflict-driven clause learning

Diagram:
- Decide
- Deduce
- Analyze Conflict
  - New lemma
  - Conflict
Conflict-driven clause learning

Specification

DSL grammar & semantics

Knowledge base

Decide

Partial program

New partial program

Deduce

Conflict

Analyse Conflict

All holes filled

New lemma

Solution

No solution

No solution
Conflict-driven clause learning

Synthesizing $P$ that satisfies $\text{spec } \phi = \exists P. \phi(P)$

- Decide
- Deduce
- Analyze
- Conflict

Specification
DSL grammar & semantics
Knowledge base

Partial program
New partial program
New lemma
All holes filled
Solution
No solution
Conflict
Conflict-driven clause learning

Synthesizing $P$ that satisfies spec $\Phi = \exists P.\Phi(P)$

Solving the second-order constraint:

Specifying $P$ that satisfies $\Phi = \exists P.\Phi(P)$

Natural integration with machine learning

Decision

Deduction

Analyze

Specify unit

Deduce

All holes filled

New partial program

New lemma

Solution

No solution

Specification

DSL grammar & semantics

Knowledge base
A motivating example
A motivating example

\[ [1,2,3] \rightarrow [1,2] \]
A motivating example

\[ [1,2,3] \rightarrow [1,2] \]

map

\[ [1,2,3] \]

\[ g(\ldots) \]

\[ h(\ldots) \]
A motivating example

\[ [1,2,3] \rightarrow [1,2] \]
A motivating example

\[ [1,2,3] \rightarrow [1,2] \]
A motivating example

\[ [1,2,3] \rightarrow [1,2] \]
A motivating example

\[ [1,2,3] \rightarrow [1,2] \]

\[
\begin{align*}
\text{map} & \quad \text{f(\ldots)} \\
[1,2,3] & \quad [1,2,3] \\
\quad \text{g(\ldots)} & \quad \text{?(...)} \\
\quad \text{h(\ldots)} & \quad \text{?(...)}
\end{align*}
\]

\[ \text{spec(map)} = \text{spec(f)} \]
A motivating example

\[ [1,2,3] \quad \rightarrow \quad [1,2] \]

```
map
[1,2,3]  g(…)
    h(…)

f(…)
[1,2,3]  ?(…)
    ?(…)
```

\[ \text{spec(map)} = \text{spec(f)} \]

Analyze the root cause of failure and avoid making the same mistake!
Acknowledgement

Isil Dillig  Ruben Martins  Yuepeng Wang  Jacob Van Geffen

Osbert Bastani  Saswat Anand  Alex Aiken

Swarat Chaudhuri  Tom Reps  Xinyu Wang  Calvin Lin
https://utopia-group.github.io/morpheus/
Thank You!

https://utopia-group.github.io/morpheus/

http://fredfeng.github.io/sypet/