Bridging Logical Reasoning and Machine Learning in Program Synthesis

Yanju Chen
Computer Science Department
University of California, Santa Barbara
12/03/2020
Overview

• Program Synthesis in a Nutshell
• MARS: Encoding Multi-Layer Specifications
• CONCORD: Deduction-Guided Reinforcement Learning
• Related Works & Conclusions
Program Synthesis in a Nutshell

- Problem Formalization
- Related Works
- Program Synthesis with Machine Learning (I)
- A Data Wrangling Example & DSL
- NEO: A Brief Overview

- Observations & Motivations
  - Q1: Why logical reasoning?
  - Q2: Why machine learning?
  - Q3: Why bridging?
Problem Formalization

I need to reformat the data so that there is just one row per site visit (i.e. in a given site name and date combo) with columns for total found by species and the fish status (i.e. speciesA_pos, SpeciesA_neg, Sp_B_pos.. etc).

\[
\text{occurs}(\text{unite}) \land \\
\text{occurs}(\text{group_by}) \land \\
\text{hasChild}(\text{group_by}, \text{unite}) \land \\
\ldots
\]

logical constraints

... logical constraints

natural languages

examples

multi-modal

multi-paradigm

Find a program \( P \) that satisfies all the specifications \( \phi \).
<table>
<thead>
<tr>
<th>synthesizer</th>
<th>domain evaluated</th>
<th>specification</th>
<th>logical reasoning</th>
<th>machine learning</th>
<th>bridging level</th>
<th>multi-modal</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEEPCODER (Balog et al. 2017)</td>
<td>list</td>
<td>IO</td>
<td>😞</td>
<td>😞</td>
<td>NA</td>
<td>😞</td>
</tr>
<tr>
<td>SEQ2SQL (Zhong et al. 2017)</td>
<td>SQL</td>
<td>I + NL</td>
<td>😞</td>
<td>😞</td>
<td>NA</td>
<td>😞</td>
</tr>
<tr>
<td>DIALSQL (Gur et al. 2018)</td>
<td>SQL</td>
<td>NL</td>
<td>😞</td>
<td>😞</td>
<td>NA</td>
<td>😞</td>
</tr>
<tr>
<td>EXEC (Chen et al. 2018)</td>
<td>Karel</td>
<td>IO</td>
<td>😞</td>
<td>😞</td>
<td>NA</td>
<td>😞</td>
</tr>
<tr>
<td>NEO (Feng et al. 2018)</td>
<td>table + list</td>
<td>IO</td>
<td>😞</td>
<td>😞</td>
<td>★</td>
<td>😞</td>
</tr>
<tr>
<td>SKETCHADAPT (Nye et al. 2018)</td>
<td>list + string + Algolisp</td>
<td>IO / IO + NL</td>
<td>😞</td>
<td>😞</td>
<td>★</td>
<td>★★★</td>
</tr>
<tr>
<td>SQLIZER (Yaghmazadeh et al. 2018)</td>
<td>SQL</td>
<td>NL</td>
<td>😞</td>
<td>😞</td>
<td>★</td>
<td>😞</td>
</tr>
<tr>
<td>AutoPandas (Bavishi et al. 2019)</td>
<td>table</td>
<td>IO</td>
<td>😞</td>
<td>😞</td>
<td>NA</td>
<td>😞</td>
</tr>
<tr>
<td>MARS (Chen et al. 2019)</td>
<td>table</td>
<td>IO + NL</td>
<td>😞</td>
<td>😞</td>
<td>★★★</td>
<td>★★★</td>
</tr>
<tr>
<td>METAL (Si et al. 2019)</td>
<td>circuit</td>
<td>logical formula</td>
<td>😞</td>
<td>😞</td>
<td>★★★</td>
<td>★★★</td>
</tr>
<tr>
<td>CONCORD (Chen et al. 2020)</td>
<td>list</td>
<td>IO</td>
<td>😞</td>
<td>😞</td>
<td>★★★</td>
<td>★★★</td>
</tr>
<tr>
<td>PROBE (Barke et al. 2020)</td>
<td>string + circuit + bitvector</td>
<td>IO</td>
<td>😞</td>
<td>😞</td>
<td>★★★</td>
<td>★★★</td>
</tr>
<tr>
<td>REGEL (Chen et al. 2020)</td>
<td>regex</td>
<td>IO + NL</td>
<td>😞</td>
<td>😞</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>VISER (Wang et al. 2020)</td>
<td>visualization</td>
<td>IO + visual sketch</td>
<td>😞</td>
<td>😞</td>
<td>★</td>
<td>★</td>
</tr>
</tbody>
</table>

*The table only lists some of the recent related works.

IO: Input-Output Example | NL: Natural Language | ☒: Yes | ☐: Not explicitly claimed
Program Synthesis in a Nutshell

Program Synthesis with Machine Learning (I)

Predictions of Language Constructs / Partial Programs

Natural Language

DSL

Examples

User

Neural Encoder

Program State

Input: 

[-17, -3, 4, 11, 0, -5, -9, 13, 6, 6, -8, 11]

Output: 

[-12, -20, -32, -36, -68]

[1] DEEPCODER (Balog et al. 2017); EXEC (Chen et al. 2018); AutoPandas (Bavishi et al. 2019); METAL (Si et al. 2019); CONCORD (Chen et al. 2020);

[2] SEQ2SQL (Zhong et al. 2017); DIALSQL (Gur et al. 2018); SQLIZER (Yaghmazadeh et al. 2018); MARS (Chen et al. 2019); REGEL (Chen et al. 2020); VISER (Wang et al. 2020);
A Running Example from StackOverflow[1]

I need to reformat the data so that there is just one row per site visit (i.e. in a given site name and date combo) with columns for total found by species and the fish status (i.e. speciesA_pos, SpeciesA_neg, Sp_B_pos.. etc).

figured I need to sum within site. My thoughts were to use split/apply/aggregate/for loops etc but tried various combinations and not getting anywhere. apologies I'm not familiar with R. any comments appreciated!
Program Synthesis in a Nutshell

A Running DSL for Data Wrangling[1]

\[ t \rightarrow x_i \] (input table)

- \( \text{select}(t, \tilde{c}_{\text{arg}}) \) (column projection)
- \( \text{unite}(t, c_{\text{tgt}}, \tilde{c}_{\text{arg}}) \) (column merging)
- \( \text{separate}(t, \tilde{c}_{\text{tgt}}, c_{\text{arg}}) \) (column splitting)
- \( \text{mutate}(t, c_{\text{tgt}}, op, \tilde{c}_{\text{arg}}) \) (column arithmetic)
- \( \text{group}_by(t, \tilde{c}_{\text{arg}}) \) (row grouping)
- \( \text{summarise}(t, c_{\text{tgt}}, a, \tilde{c}_{\text{arg}}) \) (row aggregation)
- \( \text{filter}(t, f, \tilde{c}_{\text{arg}}) \) (row filtering)

\( op \rightarrow + \mid - \mid \times \mid \div \)

\( a \rightarrow \min \mid \max \mid \sum \mid \text{count} \mid \text{avg} \)

\( x_i \): the \( i \)-th input table

\( t \): table

\( c, \tilde{c} \): column(s) of table

\( op \): arithmetic operation

\( a \): aggregation function

\( f \): higher-order boolean function

[1] DSL adapted from Wang, C. et al. Visualization by Example. POPL'20

Bridging Logical Reasoning and Machine Learning in Program Synthesis
Program Synthesis in a Nutshell

A Running Example from StackOverflow

[Example]

<table>
<thead>
<tr>
<th>sample_ID</th>
<th>site</th>
<th>coll_date</th>
<th>species</th>
<th>TOT</th>
<th>inf_status</th>
</tr>
</thead>
<tbody>
<tr>
<td>382870</td>
<td>site1</td>
<td>27/10/2007</td>
<td>SpeciesB</td>
<td>1</td>
<td>positive</td>
</tr>
<tr>
<td>382872</td>
<td>site2</td>
<td>27/10/2007</td>
<td>SpeciesB</td>
<td>1</td>
<td>negative</td>
</tr>
<tr>
<td>487405</td>
<td>site3</td>
<td>28/10/2007</td>
<td>SpeciesA</td>
<td>1</td>
<td>positive</td>
</tr>
<tr>
<td>487405</td>
<td>site3</td>
<td>28/10/2007</td>
<td>SpeciesA</td>
<td>1</td>
<td>positive</td>
</tr>
</tbody>
</table>

[Solution]

T0 = unite( input, "cat", ["species", "inf_status"] )
T1 = group_by( T0, ["site", "cat"] )
output = summarise( T1, "sts", sum, ["TOT"] )

Labels of some AST nodes are yet to be determined.

Bridging Logical Reasoning and Machine Learning in Program Synthesis
Program Synthesis in a Nutshell > Deductive Program Synthesis

**NEO[1]**: A Brief Overview

Bridging Logical Reasoning and Machine Learning in Program Synthesis

---

**SMT-Based Deduction[1]** & **Analyze Conflicts[2]**

**Equivalent Modulo Conflict (EMC)[1]**

- select: `out.row==in.row ∧ out.col==in.col-1`
- unite: `out.row==in.row ∧ out.col==in.col+1`
- separate: `out.row==in.row ∧ out.col==in.col-1`
- mutate: `out.row==in.row ∧ out.col==in.col+1`
- group_by: `out.row==in.row ∧ out.col==in.col`
- summarise: `out.row==in.row ∧ out.col<in.col+1`
- filter: `out.row<in.row ∧ out.col==in.col`

Component-Based Specifications[2] for Data Wrangling DSL

---

*Feng, Y. et al. Program Synthesis using Conflict-Driven Learning. PLDI'18*

*Feng, Y. et al. Component-based Synthesis of Table Consolidation and Transformation Tasks from Examples. PLDI'17*

---

**NEO**

- **Decide**
  - Specification
  - DSL grammar & semantics
  - Knowledge base (initially empty)
  - New partial program
  - All holes filled
- **Deduce**
  - Partial program
  - New partial program
  - Conflict
- **Analyze**
  - Solution
  - No solution

**proposed sketch**

**generated constraints**

- input.row==4 ∧ input.col==6 ∧ N8.row==input.row ∧ N8.col==input.col ∧ N5.row==N8.row ∧ N5.col==N8.col-1 ∧ N1.row==N5.row ∧ N1.col==N5.col-1 ∧ N0.row==N1.row ∧ N0.col==N1.col+1 ∧ output.row==N0.row ∧ output.col==N0.col ∧ output.row==3 ∧ output.col==3

---

(input example)

(input alignment)

(select semantics)

(unite semantics)

(mutate semantics)

(output alignment)

(output example)
Program Synthesis in a Nutshell

Observations & Motivations

- **Q1: Why logical reasoning?**
  - Example: EXEC\[^1\]
    - Concrete interpretation is less efficient, especially for complex problems
    - Logical reasoning results generalize better in pruning search space

- **Q2: Why machine learning?**
  - Example: AutoPandas\[^2\]
    - Machine learning backend provides better estimations prioritizing search order

- **Q3: Why bridging?**
  - Example: NEO\[^3\]
    - Programs are precise, but specifications can be vague
    - Statistical components can’t reflect deduction feedbacks on the fly

---

We need both, and better!

---

\[^1\] Chen, X. et al. Execution-Guided Neural Program Synthesis. ICLR’18
\[^2\] Bavishi, R. et al. AutoPandas: Neural-backed Generators for Program Synthesis. OOPSLA’19
\[^3\] Feng, Y. et al. Program Synthesis using Conflict-Driven Learning. PLDI’18

Bridging Logical Reasoning and Machine Learning in Program Synthesis
Observations & Motivations

- Existing tools do have logical and statistical components combined
  - Example: NEO\textsuperscript{[1]} / TRINITY\textsuperscript{[2]}
  - But they are no more than "wired" together: still talk in different languages, act independently

Two Bridging Directions

- MARS\textsuperscript{[3]}: Encode multi-layer specifications (via machine learning) into logical components
  - Talk in logical language!
  - Encode specifications as soft/hard constraints in maximum satisfiability modulo theory (Max-SMT)

- CONCORD\textsuperscript{[4]}: Guide the statistical components using deductions
  - Talk in statistical language!
  - Generate training samples for machine learning models by explaining deduction results

---

1. Feng, Y. et al. Program Synthesis using Conflict-Driven Learning. PLDI’18
**MARS**\(^1\): Encoding Multi-Layer Specifications

- Motivations
- Formalization
- Framework Overview
- Multi-Layer Specification Encoding
  - Encoding Examples as Hard Constraints
  - Encoding Natural Language Specifications
- Evaluations
  - Evaluation Setup
  - Evaluation Results & Analysis
- Discussions

\(^1\) Chen, Y. et al. Maximal Multi-layer Specification Synthesis. FSE’19
Bridging Logical Reasoning and Machine Learning in Program Synthesis
Maximal Multi-Layer Specification Synthesis

**Motivations**
- Examples can be imprecise
- Multi-modal specifications contain more useful information

**Title** r script to count columns within dataset

**Example**

<table>
<thead>
<tr>
<th>sample_ID</th>
<th>site</th>
<th>coll_date</th>
<th>species</th>
<th>TOT</th>
<th>inf_status</th>
</tr>
</thead>
<tbody>
<tr>
<td>382870</td>
<td>site1</td>
<td>27/10/2007</td>
<td>SpeciesB</td>
<td>1</td>
<td>positive</td>
</tr>
<tr>
<td>382872</td>
<td>site2</td>
<td>27/10/2007</td>
<td>SpeciesB</td>
<td>1</td>
<td>negative</td>
</tr>
<tr>
<td>487405</td>
<td>site3</td>
<td>28/10/2007</td>
<td>SpeciesA</td>
<td>1</td>
<td>positive</td>
</tr>
<tr>
<td>487405</td>
<td>site3</td>
<td>28/10/2007</td>
<td>SpeciesA</td>
<td>1</td>
<td>positive</td>
</tr>
</tbody>
</table>

**Description**
I need to reformat the data so that there is just one row per site visit (i.e. in a given site name and date combo) with columns for total found by species and the fish status (i.e. speciesA_pos, SpeciesA_neg, Sp_B_pos.. etc).

figured I need to sum within site. My thoughts were to use split/apply/aggregate/for loops etc but tried various combinations and not getting anywhere. apologies I'm not familiar with R. any comments appreciated!
Maximal Multi-Layer Specification Synthesis

- Motivations
  - Examples can be imprecise
  - Multi-modal specifications contain more useful information

**[Title]** r script to count columns within dataset

**[Example]**

<table>
<thead>
<tr>
<th>sample_ID</th>
<th>site</th>
<th>coll_date</th>
<th>species</th>
<th>TOT</th>
<th>inf_status</th>
</tr>
</thead>
<tbody>
<tr>
<td>382870</td>
<td>site1</td>
<td>27/10/2007</td>
<td>SpeciesB</td>
<td>1</td>
<td>positive</td>
</tr>
<tr>
<td>382872</td>
<td>site2</td>
<td>27/10/2007</td>
<td>SpeciesB</td>
<td>1</td>
<td>negative</td>
</tr>
<tr>
<td>487405</td>
<td>site3</td>
<td>28/10/2007</td>
<td>SpeciesA</td>
<td>1</td>
<td>positive</td>
</tr>
<tr>
<td>487405</td>
<td>site3</td>
<td>28/10/2007</td>
<td>SpeciesA</td>
<td>1</td>
<td>positive</td>
</tr>
</tbody>
</table>

**[Description]**
I need to reformat the data so that there is just one row per site visit (i.e. in a given site name and date combo) with columns for total found by species and the fish status (i.e. speciesA_pos, SpeciesA_neg, Sp_B_pos.. etc).

figured I need to sum within site. My thoughts were to use split/apply/aggregate/for loops etc but tried various combinations and not getting anywhere. apologies I'm not familiar with R. any comments appreciated!
Maximal Multi-Layer Specification Synthesis

- **Motivations**
  - Examples can be imprecise
  - Multi-modal specifications contain more useful information

---

**[Title]** r script to count columns within dataset

**[Example]**

<table>
<thead>
<tr>
<th>sample_ID</th>
<th>site</th>
<th>coll_date</th>
<th>species</th>
<th>TOT</th>
<th>inf_status</th>
</tr>
</thead>
<tbody>
<tr>
<td>382870</td>
<td>site1</td>
<td>27/10/2007</td>
<td>SpeciesB</td>
<td>1</td>
<td>positive</td>
</tr>
<tr>
<td>382872</td>
<td>site2</td>
<td>27/10/2007</td>
<td>SpeciesB</td>
<td>1</td>
<td>negative</td>
</tr>
<tr>
<td>487405</td>
<td>site3</td>
<td>28/10/2007</td>
<td>SpeciesA</td>
<td>1</td>
<td>positive</td>
</tr>
<tr>
<td>487405</td>
<td>site3</td>
<td>28/10/2007</td>
<td>SpeciesA</td>
<td>1</td>
<td>positive</td>
</tr>
</tbody>
</table>

**[Description]**

I need to reformat the data so that there is just one row per site visit (i.e. in a given site name and date combo) with columns for total found by species and the fish status (i.e. speciesA_pos, SpeciesA_neg, Sp_B_pos.. etc).

figured I need to sum within site. My thoughts were to use split/apply/aggregate/for loops etc but tried various combinations and not getting anywhere. apologies I'm not familiar with R. any comments appreciated!
Maximal Multi-Layer Specification Synthesis

- Motivations
  - Examples can be imprecise
  - Multi-modal specifications contain more useful information

**MARS: Encoding Multi-Layer Specifications**

[Description]
I need to reformat the data so that there is just one row per site visit (i.e. in a given site name and date combo) with columns for total found by species and the fish status (i.e. speciesA_pos, SpeciesA_neg, Sp_B_pos... etc).

figured I need to sum within site. My thoughts were to use split/apply/aggregate for loops etc but tried various combinations and not getting anywhere. apologies I'm not familiar with R. any comments appreciated!

[Example]

<table>
<thead>
<tr>
<th>sample_ID</th>
<th>site</th>
<th>coll_date</th>
<th>species</th>
<th>TOT</th>
<th>inf_status</th>
</tr>
</thead>
<tbody>
<tr>
<td>382870</td>
<td>site1</td>
<td>27/10/2007</td>
<td>SpeciesB</td>
<td>1</td>
<td>positive</td>
</tr>
<tr>
<td>382872</td>
<td>site2</td>
<td>27/10/2007</td>
<td>SpeciesB</td>
<td>1</td>
<td>negative</td>
</tr>
<tr>
<td>487405</td>
<td>site3</td>
<td>28/10/2007</td>
<td>SpeciesA</td>
<td>1</td>
<td>positive</td>
</tr>
<tr>
<td>487405</td>
<td>site3</td>
<td>28/10/2007</td>
<td>SpeciesA</td>
<td>1</td>
<td>positive</td>
</tr>
</tbody>
</table>

[Title] r script to count columns within dataset

Natural language provides hints to problem solutions.
Formalization

- Maximal Multi-Layer Specification Synthesis

Given specification \((\mathcal{E}, \Psi, \Sigma)\) where \(\mathcal{E} = (T_{in}, T_{out})\), \(\Psi = \cup(\chi_i, \omega_i)\), and \(\Sigma\) represents all symbols in the DSL, the Maximal Multi-Layer Specification Synthesis problem is to infer a program \(\mathcal{P}\) such that:

- \(\mathcal{P}\) is a well-typed expression over symbols in \(\Sigma\),
- \(\mathcal{P}(T_{in}) = T_{out}\) and
- \(\sum \omega_i\) is maximized.

We model the problem using maximum satisfiability modulo theory (Max-SMT) and solve it with an off-the-shelf SMT solver.
MARS: Encoding Multi-Layer Specifications

Framework Overview

- User
- Natural languages
- Neural
- Examples
- Logical constraints
- Logical constraints
- Deductive
- Hard
- Soft
- DSL
- Max-SMT
- Program P

Encoding statistical estimations

Bridging Logical Reasoning and Machine Learning in Program Synthesis
MARS: Encoding Multi-Layer Specifications

Encoding Examples as Hard Constraints

symbolic program

output

summarise

N0

N1

N2

N3

N4

N5

N6

N7

N8

N9

input

unite

group_by

N7.row == 4
N7.col == 5

N5.row == 4
N5.col == 5

N1.row == 4
N1.col == 5

N0.row <= 4
N0.col <= 6

output.row == 4
output.col == 6

output.row == N0.row
output.col == N0.col

satisfiable?

YES

input

summarise

group_by

unite

site
cat
sts
site1 SpeciesB_positive 1
site2 SpeciesB_negative 1
site3 SpeciesA_positive 2

sample_ID site coll_date species TOT inf_status
382870 site1 27/10/2007 SpeciesB 1 positive
382872 site2 27/10/2007 SpeciesB 1 negative
487405 site3 28/10/2007 SpeciesA 1 positive
487405 site3 28/10/2007 SpeciesA 1 positive
MARS: Encoding Multi-Layer Specifications

Encoding Examples as Hard Constraints

symbolic program

output

mutate

unite

select

input

\[ \text{sample ID} \quad \text{site} \quad \text{coll date} \quad \text{species} \quad \text{TOT} \quad \text{inf status} \]

<table>
<thead>
<tr>
<th>sample ID</th>
<th>site</th>
<th>coll date</th>
<th>species</th>
<th>TOT</th>
<th>inf status</th>
</tr>
</thead>
<tbody>
<tr>
<td>382870</td>
<td>site1</td>
<td>27/10/2007</td>
<td>SpeciesB</td>
<td>1</td>
<td>positive</td>
</tr>
<tr>
<td>382872</td>
<td>site2</td>
<td>27/10/2007</td>
<td>SpeciesB</td>
<td>1</td>
<td>negative</td>
</tr>
<tr>
<td>487405</td>
<td>site3</td>
<td>28/10/2007</td>
<td>SpeciesA</td>
<td>1</td>
<td>positive</td>
</tr>
<tr>
<td>487405</td>
<td>site3</td>
<td>28/10/2007</td>
<td>SpeciesA</td>
<td>1</td>
<td>positive</td>
</tr>
</tbody>
</table>

Bridging Logical Reasoning and Machine Learning in Program Synthesis
MARS: Encoding Multi-Layer Specifications

Encoding Natural Language Specifications

- The Hybrid Neural Architecture
  - seq2seq model (supervised): capture common natural language semantics
  - association rule module (unsupervised): capture frequent patterns and refine the preference

Bridging Logical Reasoning and Machine Learning in Program Synthesis
MARS: Encoding Multi-Layer Specifications

Encoding Natural Language Specifications

• The seq2seq model

\[ \ell(P|D) = \sum_{i} \log \pi(P_{i}|D) \]

*\( \pi \) is the seq2seq model
MARS: Encoding Multi-Layer Specifications

Encoding Natural Language Specifications

- The association rule module

**Algorithm 1** Symbolic Program Score Refinement Algorithm

1. procedure **REFINEMENT**(R, D, S, c, θ)
2. input: association rule set R, question D, solution S with its corresponding score c and weight function θ
3. output: refined score \( c_r \)
4. \( b \leftarrow 0 \) \quad \rightarrow \text{accumulative boosting ratio}
5. for rule \( r_j \in R \) do
6. \( b \leftarrow b + \theta(r_j) \cdot \text{match}(r_j, D, S) \)
7. \( c_r \leftarrow c + b \cdot |c| \) \quad \rightarrow \text{update score}
8. return \( c_r \)

Association rules captures local user intents.
MARS: Encoding Multi-Layer Specifications

Encoding Natural Language Specifications

- Encoding refined preference scores

\[
\begin{align*}
\text{occurs predicates} & : \text{occurs}(\text{summarise}, 2.3) \land \\
& \text{occurs}(\text{group_by}, 2.3) \land \\
& \text{occurs}(\text{unite}, 2.3) \\
\text{weight} & : 2.3 \\
\text{hasChild predicates} & : \text{hasChild}(\text{summarise}, \text{group_by}, 2.3) \land \\
& \text{hasChild}(\text{group_by}, \text{unite}, 2.3) \\
\text{weight} & : 2.3 \\
\end{align*}
\]

\[
(N0 = \text{idx(}\text{summarise}\text{)} \lor N1 = \text{idx(}\text{summarise}\text{)} \lor N5 = \text{idx(}\text{summarise}\text{)}) \land \\
(N0 = \text{idx(}\text{group_by}\text{)} \lor N1 = \text{idx(}\text{group_by}\text{)} \lor N5 = \text{idx(}\text{group_by}\text{)}) \land \\
(N0 = \text{idx(}\text{unite}\text{)} \lor N1 = \text{idx(}\text{unite}\text{)} \lor N5 = \text{idx(}\text{unite}\text{)})
\]

encoding \(\text{occurs}(p_i, \omega_i)\)

\[
\bigwedge_{p_i \in \Lambda} \bigvee_{N_i \in \mathbb{N}} N_i = \text{idx}(p_i)
\]

encoding \(\text{hasChild}(p_i, p_j, \omega_i)\)

\[
p_i, p_j \in \Lambda, N_i \in \mathbb{N} \\
N_i = \text{idx}(p_i) \Rightarrow \bigwedge_{N_j \in \text{Ch}(N_i)} N_j = \text{idx}(p_j)
\]
MARS: Encoding Multi-Layer Specifications

Evaluation Setup

• Research Questions
  • Q1: Do our multi-layer specification and neural architecture suggest candidates that are close to the user intent?
  • Q2: What is the impact of the neural architecture in MARS on the performance of a state-of-the-art synthesizer for data wrangling tasks?
  • Q3: How is the performance of MARS affected by the quality of the corpus?

• Experiment Setup
  • Benchmarks: 80 Real-World Challenging Data Wrangling Tasks
  • Dataset: 20,640 StackOverflow Pages of Data Wrangling Tasks
    • 16,459 question-solution pairs for seq2seq model
    • 37,748 transactions for association rule mining (Apriori algorithm); we obtain 187 valid\(^1\) rules

• Comparison to MORPHEUS\(^2\)

\(^1\) A rule is valid if its confidence ≥ 0.9 or support ≥ 0.003, and satisfies all the criteria defined in Chen, Y. et al.. Maximal Multi-layer Specification Synthesis. FSE’19
\(^2\) Feng, Y. et al.. Component-based Synthesis of Table Consolidation and Transformation Tasks from Examples. PLDI’17
MARS: Encoding Multi-Layer Specifications

Evaluation Results & Analysis

• Timeout: 5 mins

• Ablation Variants
  • \textit{ngram}: built-in statistical model in MORPHEUS
  • \textit{seq2seq}: MARS with \textit{seq2seq} model
  • \textit{hybrid}: MARS with \textit{seq2seq} model and preference score refinement (association rules)

<table>
<thead>
<tr>
<th>Table 1: Statistics for different model rankings.</th>
<th>Table 2: Counts of top-1s and top-3s in different models.</th>
<th>Table 3: Statistics of running time.</th>
</tr>
</thead>
<tbody>
<tr>
<td>model</td>
<td>\textit{n-gram}</td>
<td>\textit{seq2seq}</td>
</tr>
<tr>
<td>average (^1)</td>
<td>42</td>
<td>25</td>
</tr>
<tr>
<td>std. (^1)</td>
<td>70</td>
<td>39</td>
</tr>
</tbody>
</table>

\(^1\) standard deviation.
\(\ast\) computed based on the rankings of the correct solutions.

<table>
<thead>
<tr>
<th>model</th>
<th>\textit{n-gram}</th>
<th>\textit{seq2seq}</th>
<th>\textit{hybrid}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-1 total (^\ast)</td>
<td>0</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>Top-3 total (^\ast)</td>
<td>2</td>
<td>18</td>
<td>29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>model</th>
<th>avg. speedup (^1)</th>
<th>#timeouts (^\ast)</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{ngram}</td>
<td>1x</td>
<td>11</td>
</tr>
<tr>
<td>\textit{seq2seq}</td>
<td>6x</td>
<td>8</td>
</tr>
<tr>
<td>\textit{hybrid}</td>
<td>15x</td>
<td>2</td>
</tr>
</tbody>
</table>

\(^1\) average speedup on challenging solved benchmarks.
\(^\ast\) number of timeouts on all benchmarks.
MARS: Encoding Multi-Layer Specifications

Discussions

• Limitations
  • Insufficient Text
    • Description of the question is barely useful
  • Contextual Text
    • Some questions require understanding of *pragmatic* contexts, not only semantic
  • Misleading Text
    • User specifies functionality not supported by the DSL

• Threats to Validity
  • Quality of the Corpus
  • Benchmark Selection

---

“... *I can solve my problem using dplyr’s mutate but it’s a time-intensive, roundabout way to achieve my goal.* ...”

“... *I want to use mutate to make variable d which is mean of a, b and c.* ...”
Bridging the Logical and Statistical Lands

MARS

Intertitles
CONCORD\cite{1}: Deduction-Guided Reinforcement Learning

- Motivations
- Pure Deductive & Statistical Approaches
- Framework Overview
- Formalization
- A Running Example

- Deduction-Guided Reinforcement Learning
  - Deduction Engine
  - Off-Policy Sampling
  - Importance Weighting

- Evaluations
  - Evaluation Setup
  - Evaluation Results & Analysis

\cite{1} Chen, Y. et al. Program Synthesis Using Deduction-Guided Reinforcement Learning. CAV'20
Bridging Logical Reasoning and Machine Learning in Program Synthesis
Deduction-Guided Reinforcement Learning

Motivations

• Feedback of deduction cannot be seamlessly used by statistical model

-0.1 select unite mutate
-0.3 unite select mutate
-0.8 mutate select unite
-0.9 separate unite mutate
-1.2 select separate mutate

... 

-2.3 unite group_by summarise
-2.9 filter separate mutate

• Statistical estimation is not synchronized with deductive knowledge
• Maintenance of deductive knowledge creates overhead
CONCORD: Deduction-Guided Reinforcement Learning

Deductive Approach

- rich and accurate feedback
- efficient search space pruning

- KB maintenance can be difficult
- no feedback incorporation

<table>
<thead>
<tr>
<th>Decide</th>
<th>Deduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.1</td>
<td>select unite mutate</td>
</tr>
<tr>
<td>-0.3</td>
<td>unite select mutate</td>
</tr>
<tr>
<td>-0.8</td>
<td>mutate select unite</td>
</tr>
<tr>
<td>-0.9</td>
<td>separate unite mutate</td>
</tr>
<tr>
<td>-1.2</td>
<td>select separate mutate</td>
</tr>
</tbody>
</table>

...
CONCORD: Deduction-Guided Reinforcement Learning

Statistical Approach

- Infer
  - $\pi_0$
  - specs
  - DSL

- Check
  - select unite mutate: -0.1
  - unite select mutate: -0.3
  - mutate select unite: -0.8
  - separate unite mutate: -0.9
  - select separate mutate: -1.2
  - unite group_by summarise: -2.3
  - filter separate mutate: -2.9

- Solution

- Data-driven candidate list
- Can update policy seamlessly
- Less informative feedback
- Inefficient pruning
CONCORD: Deduction-Guided Reinforcement Learning

Framework Overview

Take Action ➔ Deduce

π₀ ➔ program ➔ solution

dsels ➔ Update Policy ➔ π

Bridging Logical Reasoning and Machine Learning in Program Synthesis
**Formalization**

- Program Synthesis as Markov Decision Process

**Objective:** Maximize Rewards

**Bridging Logical Reasoning and Machine Learning in Program Synthesis**

35
CONCORD: Deduction-Guided Reinforcement Learning

Running Example

- feedback from deduction flows seamlessly to the policy update
- not only prune the search space, but also promote good candidates
CONCORD: Deduction-Guided Reinforcement Learning

Synthesis Algorithm

Deduce

Take Action

Update Policy

-0.1 select unite mutate

sampled infeasible programs

select unite mutate
unite select mutate
mutate select unite
...

empty set $\emptyset$
CONCORD: Deduction-Guided Reinforcement Learning

Synthesis Algorithm

Deduction Engine

Sampler

- infeasible

feasible

empty set $\emptyset$

Take Action

Update Policy

Weighted future rewards

Importance weighting

Program distribution of policy

Program distribution of Sampler

$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{P \sim \mathcal{D}_{\pi_{\theta}}} [\ell(P)]$

$= \mathbb{E}_{P \sim \mathcal{D}} [\ell(P) \cdot \frac{\mathcal{D}_{\pi_{\theta}}(P)}{\mathcal{D}(P)}]$
CONCORD: Deduction-Guided Reinforcement Learning

Evaluation Setup

• Research Questions:
  • Q1: How does Concord compare against existing synthesis tools?
  • Q2: How effective is the off-policy RL algorithm compared to standard policy gradient?

• Experiment Setup
  • Deduction Engine: NEO’s (Feng et al. 2018) conflict-driven deduction engine
  • Policy: Gated Recurrent Unit (GRU)
  • Benchmarks: DEEPCODER benchmarks used in NEO
    • 100 challenging list processing problems
  • Comparison between:
    • NEO (Feng et al. 2018)
    • DEEPCODER (Balog et al. 2017)
CONCORD: Deduction-Guided Reinforcement Learning

Evaluation Results & Analysis

- Concord tightly couples statistical and deductive reasoning based on reinforcement learning.
- The off-policy reinforcement learning technique is effective.

### Table 1: Evaluation Results

<table>
<thead>
<tr>
<th>Tool</th>
<th>Solved</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONCORD</td>
<td>82%</td>
<td>36s</td>
</tr>
<tr>
<td>NEO</td>
<td>71%</td>
<td>99s</td>
</tr>
<tr>
<td>DEEPCODER</td>
<td>32%</td>
<td>205s</td>
</tr>
</tbody>
</table>

### Table 2: Speedup over NEO

<table>
<thead>
<tr>
<th>Tool</th>
<th>Solved</th>
<th>Speedup over NEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONCORD</td>
<td>82%</td>
<td>8.71x</td>
</tr>
</tbody>
</table>

### Figure 5: Comparison between CONCORD, NEO, and DEEPCODER

- The off-policy reinforcement learning technique is effective.
Related Works & Conclusions

- Program Synthesis with Machine Learning (II)
- Related Works
  - METAL
  - PROBE
  - ABL
- Challenges, Conclusions & Future Works
Program Synthesis with Machine Learning (II)

Related Works & Conclusions

Predictions of Language Constructs / Partial Programs

Neural Encoder

Examples

DSL

Proposed Program

Execution

Check Results & Update Policy

Dialog

User

Natural Language

representation learning

multi-modal encoding

Bridging Logical Reasoning and Machine Learning in Program Synthesis

Neural Encoder

Natural Language

Examples

Multi-modal encoding

Bridging Logical Reasoning and Machine Learning in Program Synthesis

Related Works & Conclusions

Program Synthesis with Machine Learning (II)

Predictions of Language Constructs / Partial Programs

Neural Encoder

Examples

DSL

Proposed Program

Execution

Check Results & Update Policy

Dialog

User

Natural Language

representation learning

multi-modal encoding

Bridging Logical Reasoning and Machine Learning in Program Synthesis

[1] SEQ2SQL (Zhong et al. 2017); EXEC (Chen et al. 2018); AutoPandas (Bavishi et al. 2019); REGEL (Chen et al. 2020); VISER (Wang et al. 2020)

[2] NEO (Feng et al. 2018); SQLIZER (Yaghmazadeh et al. 2018); MARS (Chen et al. 2019); PROBE (Barke et al. 2020); CONCORD (Chen et al. 2020)

[3] METAL (Si et al. 2019); SKETCHADAPT (Nye et al. 2018)

Related Works & Conclusions

Related Works

- **METAL**\(^1\)
  - Circuit Synthesis
  - Invoke a SAT solver to generate a counter-example which adds to the test cases
- **PROBE**\(^2\)
  - String Transformation & Bitvector & Circuit Synthesis
  - Just-in-Time Learning: updates a PCFG during synthesis by learning from partial solutions
- **ABL**\(^3\)
  - Handwritten Equation Decipherment
  - Improve machine learning models using abductive learning

\(^1\) Si, X. et al. Learning a Meta-Solver for Syntax-Guided Program Synthesis. ICLR’19
\(^2\) Barke, S. et al. Just-in-Time Learning for Bottom-up Enumerative Synthesis. OOPSLA’20

Bridging Logical Reasoning and Machine Learning in Program Synthesis
Related Works & Conclusions

**METAL[1]** (The Reinforcement Learning Part)

Reward smoothing by considering into reward design of counter-examples from SAT solver

---

Bridging Logical Reasoning and Machine Learning in Program Synthesis

\[ r = \frac{\sum_{b \in B_\phi \cup \hat{B}_b} [f(b) \equiv \phi(b)]}{|B_\phi \cup \hat{B}_b|} \]

\[ B_\phi \leftarrow B_\phi \cup \hat{B}_b \]
Related Works & Conclusions

PROBE\(^\text{[1]}\) (The Just-in-Time Learning Part)

Programs that satisfy a subset of the semantic specification often share syntactic similarity with the full solution.

Examples

<table>
<thead>
<tr>
<th>ID</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_0</td>
<td>&quot;+95 310-537-401&quot;</td>
<td>&quot;310&quot;</td>
</tr>
<tr>
<td>e_1</td>
<td>&quot;+72 001-050-856&quot;</td>
<td>&quot;001&quot;</td>
</tr>
<tr>
<td>e_2</td>
<td>&quot;+106 769-858-438&quot;</td>
<td>&quot;769&quot;</td>
</tr>
</tbody>
</table>

Proposed Solution

PCFG

(initially uniform)

Select Promising Partial Solutions

Update PCFG

where

\[
p(R) = \frac{p_{ul}(R) (1 - Frt)}{Z}
\]

\[
FIT = \max_{\{P \in PPSol \mid Retr(P)\}} \frac{|E \cap E[P]|}{|E|}
\]

*with different objectives and selection schemas


Bridging Logical Reasoning and Machine Learning in Program Synthesis
ABL\textsuperscript{[1]}: A Brief Overview

Bridging Logical Reasoning and Machine Learning in Program Synthesis
Related Works & Conclusions

Challenges, Conclusions & Future Works

**Scalability**
How do we speed up synthesis for given task?

**Interactivity**
How do we access and utilize extra information from users?

**Continuality**
How do we distill useful knowledge across synthesis?

**Robustness**
How do we tolerate specification mistakes/noises during synthesis?

**Multimodality**
How do we process multi-modal information?

---

**Scalability**

- DeepCoder (Balog et al. 2017)
- EXEC (Chen et al. 2018)
- NEO (Feng et al. 2018)
- SQLizer (Yaghmazadeh et al. 2018)
- AutoPandas (Bavishi et al. 2019)
- Metal (Si et al. 2019)
- SketchAdapt (Nye et al. 2018)
- Probe (Barke et al. 2020)
- Concord (Chen et al. 2020)
- ...

---

**Interactivity**

- InteractivePROSE (Le et al. 2017)
- DIALSQL (Gur et al. 2018)
- GIM (Peleg et al. 2020)
- SampleSy (Ji et al. 2020)
- ...

---

**Continuity**

- NELL (Mitchell et al. 2015)
- Net2Net (Chen et al. 2016)
- Parishi et al. 2019
- ...

---

**Robustness**

- FlashFill (Gulwani 2011)
- Rulesynth (Singh 2017)
- Bester (Peleg et al. 2020)
- ...

---

**Multimodality**

- Seq2SQL (Zhong et al. 2017)
- MARS (Chen et al. 2019)
- REGEL (Chen et al. 2020)
- Viser (Wang et al. 2020)
- ...

---

... and some more interesting dimensions?
References

- Gulwani, S. Automating String Processing in Spreadsheets using Input-Output Examples. In POPL'11
- Berant, J., Chou, A., Frostig, R., & Liang, P. Semantic Parsing on {F}reebase from Question-Answer Pairs. In EMNLP'13
- Feng, Y., Martins, R., Van Geffen, J., Dillig, I., & Chaudhuri, S. Component-based Synthesis of Table Consolidation and Transformation Tasks from Examples. In PLDI'17
- Le, V., Perelman, D., Polozov, O., Raza, M., Udupa, A., & Gulwani, S. Interactive Program Synthesis. CoRR, abs/1703.0
- Singh, R., Meduri, V. V., Elmagarmid, A., Madden, S., Papotti, P., Quiané-Ruiz, J.-A., ... Tang, N. Synthesizing Entity Matching Rules by Examples. In VLDB’17
- Zhong, V., Xiong, C., & Socher, R. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. CoRR, abs/1709.0
- Dong, L., & Lapata, M. Coarse-to-Fine Decoding for Neural Semantic Parsing. In ACL’18
- Feng, Y., Martins, R., Bastani, O., & Dillig, I. Program Synthesis using Conflict-Driven Learning. In PLDI’18
- Peleg, H., Shoham, S., & Yahav, E. Programming Not Only by Example. In ICSE’18
References II

- **Chen, Y., Martins, R., & Feng, Y. Maximal Multi-layer Specification Synthesis. In FSE’19**
- Peleg, H., & Polikarpova, N. Perfect is the Enemy of Good: Best-Effort Program Synthesis. In ECOOP’20
- Wang, C., Feng, Y., Bodik, R., Cheung, A., & Dillig, I. Visualization by Example. In POPL’20