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?

Program Synthesis in a Nutshell

Problem Formalization

multi-modal

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

↳

site	coll_date	SP_A_pos	SP_A_neg	SP_B_pos	SP_B_neg
site1	27/10/2007	0	0	1	0
site2	27/10/2007	0	0	0	1
site3	28/10/2007	2	0	0	0

examples

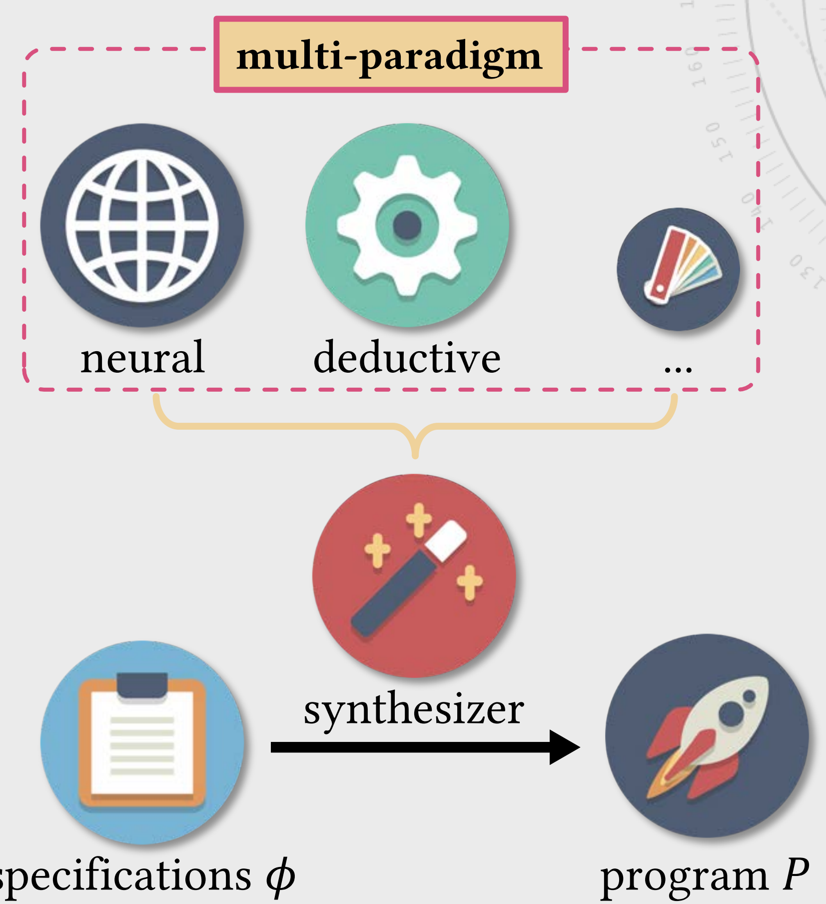
occurs(`unite`) ∧
occurs(`group_by`) ∧
hasChild(`group_by`, `unite`) ∧
...

logical constraints

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).

natural languages

...



Find a program P that satisfies all the specifications ϕ.

Related Works

*The table only lists some of the recent related works.
IO: Input-Output Example | NL: Natural Language | 😊: Yes | 😰: Not explicitly claimed

synthesizer	domain evaluated	specification	logical reasoning	machine learning	bridging level	multi-modal
DeepCoder (Balog et al. 2017)	list	IO	😰	😊	NA	😰
Seq2SQL (Zhong et al. 2017)	SQL	I + NL	😰	😊	NA	😊
DialSQL (Gur et al. 2018)	SQL	NL	😰	😊	NA	😊
Exec (Chen et al. 2018)	Karel	IO	😰	😊	NA	😰
Neo (Feng et al. 2018)	table + list	IO	😊	😊	★	😰
SketchAdapt (Nye et al. 2018)	list + string + Algolisp	IO / IO + NL	😊	😊	★★	😊
SQLizer (Yaghmazadeh et al. 2018)	SQL	NL	😊	😊	★	😰
AutoPandas (Bavishi et al. 2019)	table	IO	😰	😊	NA	😰
Mars (Chen et al. 2019)	**table**	**IO + NL**	😊	😊	★★★	😊
Metal (Si et al. 2019)	circuit	logical formula	😊	😊	★★★	😰
Concord (Chen et al. 2020)	**list**	**IO**	😊	😊	★★★	😰
Probe (Barke et al. 2020)	string + circuit + bitvector	IO	😊	😊	★★★	😰
Regel (Chen et al. 2020)	regex	IO + NL	😊	😊	★★	😊
Viser (Wang et al. 2020)	visualization	IO + visual sketch	😊	😊	★	😊

Program Synthesis with Machine Learning (I)

(+1)	(-1)	(*2)	(/2)	(*-1)	(**2)	(*3)	(/3)	(*4)	(/4)	(>0)	(>0)	(%2==1)	(%2==0)	HEAD	LAST	MAP	FILTER	SORT	REVERSE	TAKE	DROP	ACCESS	ZIPWITH	SCANL1	+	-	*	MIN	MAX	COUNT	MINIMUM	MAXIMUM	SUM
.0	.0	.1	.0	.0	.0	.0	.0	1.0	.0	.0	1.0	.0	.2	.0	.0	1.0	1.0	1.0	.7	.0	.1	.0	.4	.0	.0	.1	.0	.2	.1	.0	.0	.0	.0

Predictions of Language Constructs / Partial Programs

```
a ← [int]
b ← FILTER (<0) a
c ← MAP (*4) b
d ← SORT c
e ← REVERSE d
```

Proposed Program

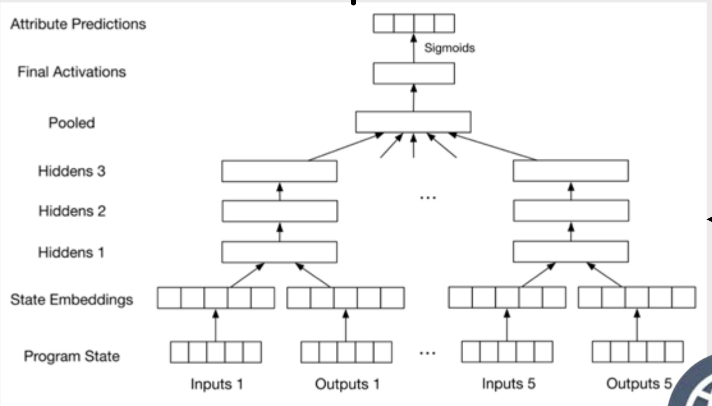


Figure 7: Schematic representation of our feed-forward encoder, and the decoder.

Neural Encoder

DSL

Natural Language

Input:
[-17, -3, 4, 11, 0, -5, -9, 13, 6, 6, -8, 11]
Output:
[-12, -20, -32, -36, -68]

Examples

User

representation learning[1]

multi-modal encoding[2]

[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]

[Title] r script to count columns within dataset

[Example]

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

➡

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

[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!

[1] Example adapted from https://stackoverflow.com/questions/39369502/r-script-to-reshape-and-count-columns-within-dataset

A Running DSL for Data Wrangling[1]

$$
\begin{array}{lll}
t & \rightarrow x_i & \text{(input table)} \\
 & \mid \ \texttt{select}(t, \vec{c}_{arg}) & \text{(column projection)} \\
 & \mid \ \texttt{unite}(t, c_{tgt}, \vec{c}_{arg}) & \text{(column merging)} \\
 & \mid \ \texttt{separate}(t, \vec{c}_{tgt}, c_{arg}) & \text{(column splitting)} \\
 & \mid \ \texttt{mutate}(t, c_{tgt}, op, \vec{c}_{arg}) & \text{(column arithmetic)} \\
 & \mid \ \texttt{group\_by}(t, \vec{c}_{arg}) & \text{(row grouping)} \\
 & \mid \ \texttt{summarise}(t, c_{tgt}, a, \vec{c}_{arg}) & \text{(row aggregation)} \\
 & \mid \ \texttt{filter}(t, f, \vec{c}_{arg}) & \text{(row filtering)} \\
op & \rightarrow + \mid - \mid \times \mid \div & \\
a & \rightarrow \texttt{min} \mid \texttt{max} \mid \texttt{sum} \mid \texttt{count} \mid \texttt{avg} &
\end{array}
$$

x_i: the i-th input table
t: table
$c, \vec{c}$: column(s) of table
op: arithmetic operation
a: aggregation function
f: higher-order boolean function

A	B	C	D
A1	B1	1	5
A2	B2	2	6
A3	B3	3	7
A4	B4	4	8

select →

A	C
A1	1
A2	2
A3	3
A4	4

A	B	C	D
A1	B1	1	5
A2	B2	2	6
A3	B3	3	7
A4	B4	4	8

unite → / ← separate

A_B	C	D
A1_B1	1	5
A2_B2	2	6
A3_B3	3	7
A4_B4	4	8

A	B	C	D
A1	B1	1	5
A2	B2	2	6
A3	B3	3	7
A4	B4	4	8

mutate →

A	B	C	D	C+D
A1	B1	1	5	6
A2	B2	2	6	8
A3	B3	3	7	10
A4	B4	4	8	12

A	B	C	D
X	B1	1	5
X	B2	2	6
Y	B3	3	7
Y	B4	4	8

group_by → summarise

A	avg.D
X	5.5
Y	7.5

A	B	C	D
A1	B1	1	5
A2	B2	2	6
A3	B3	3	7
A4	B4	4	8

↓ filter

A	B	C	D
A1	B1	1	5
A2	B2	2	6
A3	B3	3	7

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

Program Synthesis in a Nutshell

A Running Example from StackOverflow

[Example]

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

unite →

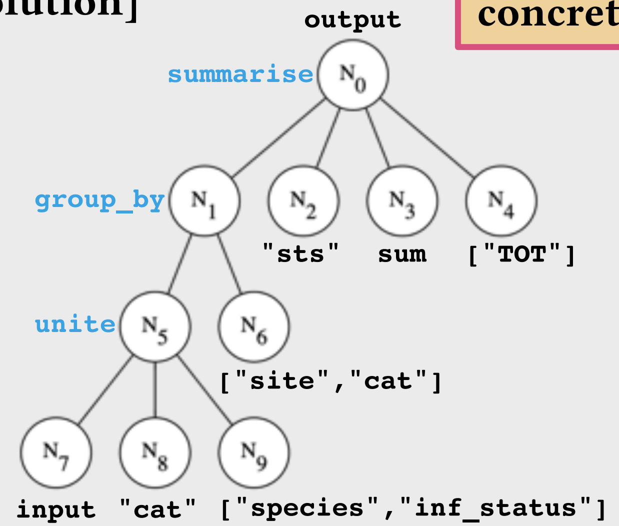
sample_ID	site	coll_date	cat	TOT
382870	site1	27/10/2007	SpeciesB_positive	1
382872	site2	27/10/2007	SpeciesB_negative	1
487405	site3	28/10/2007	SpeciesA_positive	1
487405	site3	28/10/2007	SpeciesA_positive	1

group_by → summarise

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

[Solution]

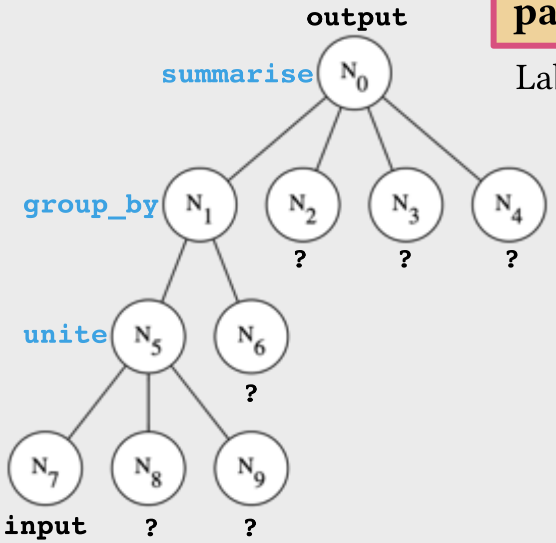
concrete program




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

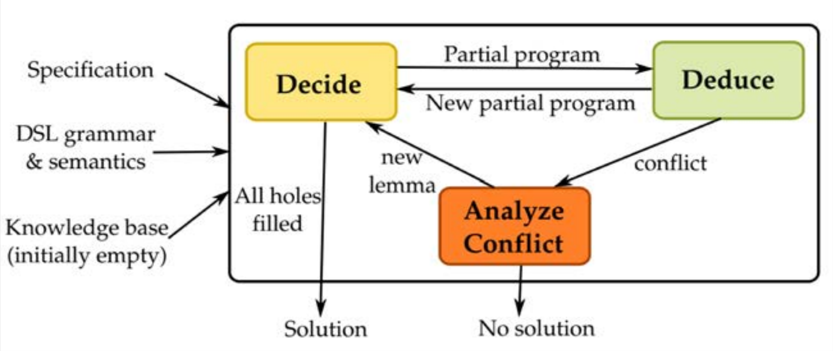
partial program / sketch

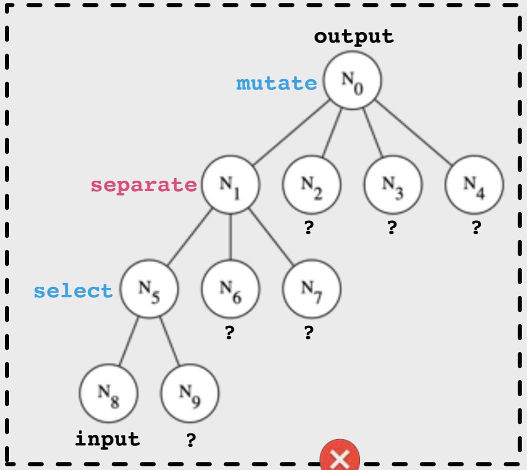
Labels of some AST nodes are yet to be determined.




```
T0 = unite( input, ?, ? )
T1 = group_by( T0, ? )
output = summarise( T1, ?, ?, ? )
```

NEO[1]: A Brief Overview



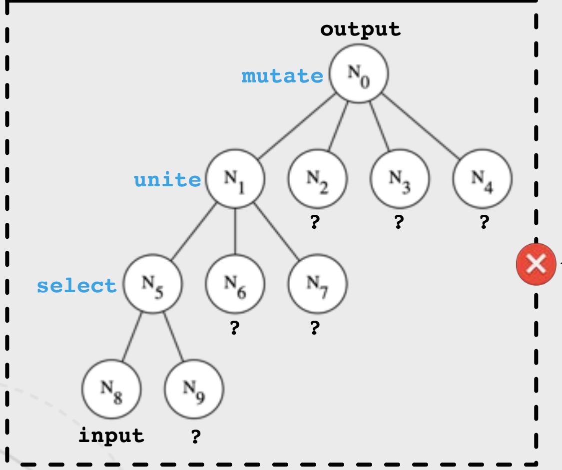


Equivalent Modulo Conflict (EMC)[1]

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-1 ∧ out.col==in.col

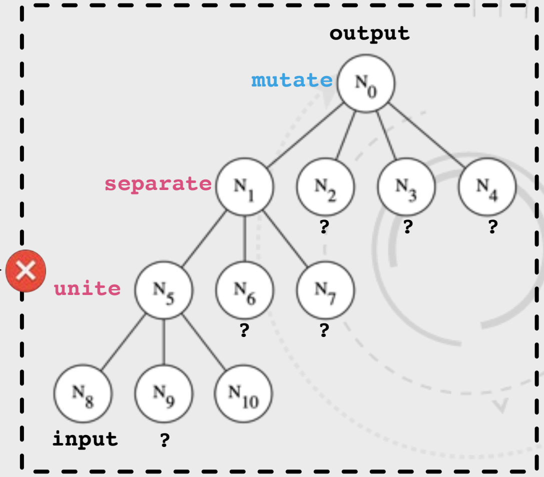
Component-Based Specifications[2] for Data Wrangling DSL

proposed sketch



generated constraints

input.row==4 ∧ input.col==6 ∧	(input example)
N8.row==input.row ∧ N8.col==input.col ∧	(input alignment)
N5.row==N8.row ∧ N5.col<=N8.col-1 ∧	(select semantics)
N1.row==N5.row ∧ N1.col==N5.col-1 ∧	(unite semantics)
N0.row==N1.row ∧ N0.col==N1.col+1 ∧	(mutate semantics)
output.row==N0.row ∧ output.col==N0.col ∧	(output alignment)
output.row==3 ∧ output.col==3	(output example)



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

[1] Feng, Y. et al. Program Synthesis using Conflict-Driven Learning. PLDI'18
[2] Feng, Y. et al. Component-based Synthesis of Table Consolidation and Transformation Tasks from Examples. PLDI'17

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

Intertitles

Bridging the Logical and Statistical Lands

- Observations & Motivations
  - Existing tools do have logical and statistical components combined
    - Example: NEO[1] / TRINITY[2]
    - But they are no more than "wired" together: still talk in different languages, act independently
- Two Bridging Directions
  - MARS[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[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
[2] Martins, R. et al. Trinity: An Extensible Synthesis Framework for Data Science. VLDB'19
[3] Chen, Y. et al. Maximal Multi-layer Specification Synthesis. FSE'19
[4] Chen, Y. et al. Program Synthesis Using Deduction-Guided Reinforcement Learning. CAV'20

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

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]

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

➔

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

[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]

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

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

[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]

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

➜

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

[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

summarise

- Motivations
  - Exa group_by mprecise
  - Multi-modal specifications contain more useful information

[Title] r script to count columns within dataset

unite

[Example]

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

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

[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!

sum

Natural language provides hints to problem solutions.

Formalization

- Maximal Multi-Layer Specification Synthesis

hard constraints/specifications: examples

DSL construct

Given specification $(\mathcal{E}, \Psi, \Sigma)$ where $\mathcal{E} = (T_{in}, T_{out})$, $\Psi = \bigcup(\chi_i, \omega_i)$, and Σ 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 Σ,
- $\mathcal{P}(T_{in}) = T_{out}$, and
- $\sum \omega_i$ is maximized.

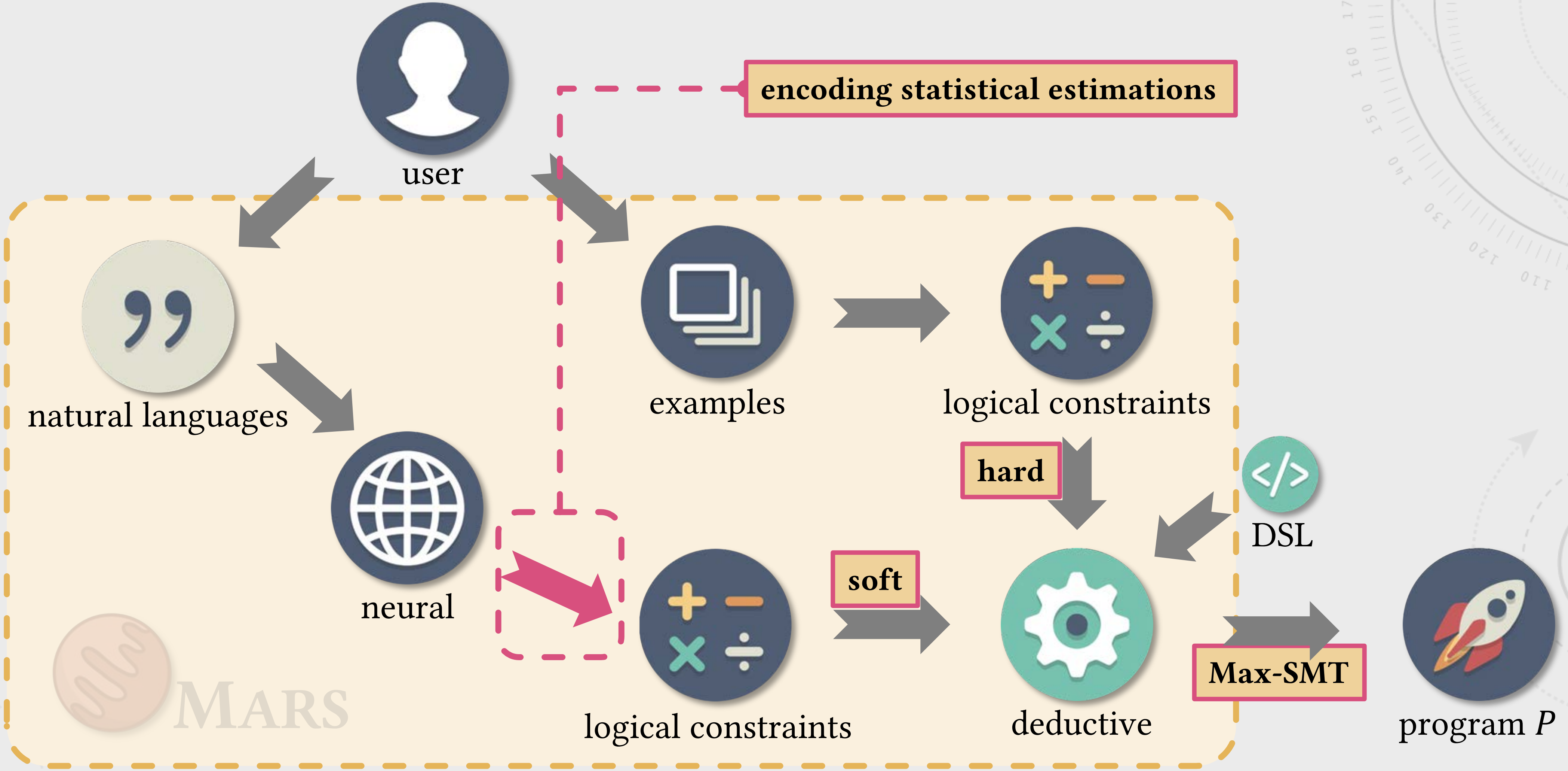
soft constraints/specifications: natural languages

preference/confidence

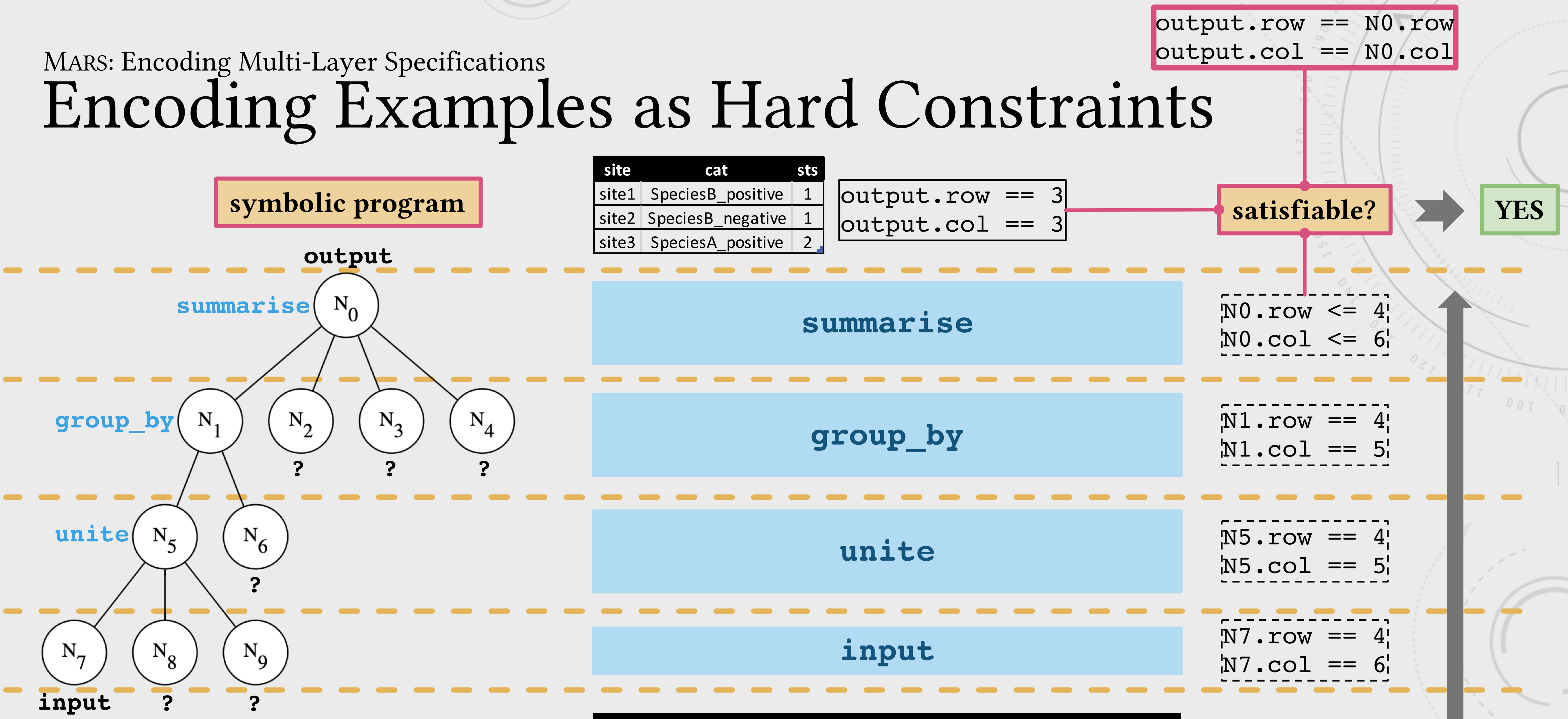
- We model the problem using *maximum satisfiability modulo theory (Max-SMT)* and solve it with an off-the-shelf SMT solver.

Hard constraints should be satisfied;
Soft constraints should be maximized.

Framework Overview



Encoding Examples as Hard Constraints

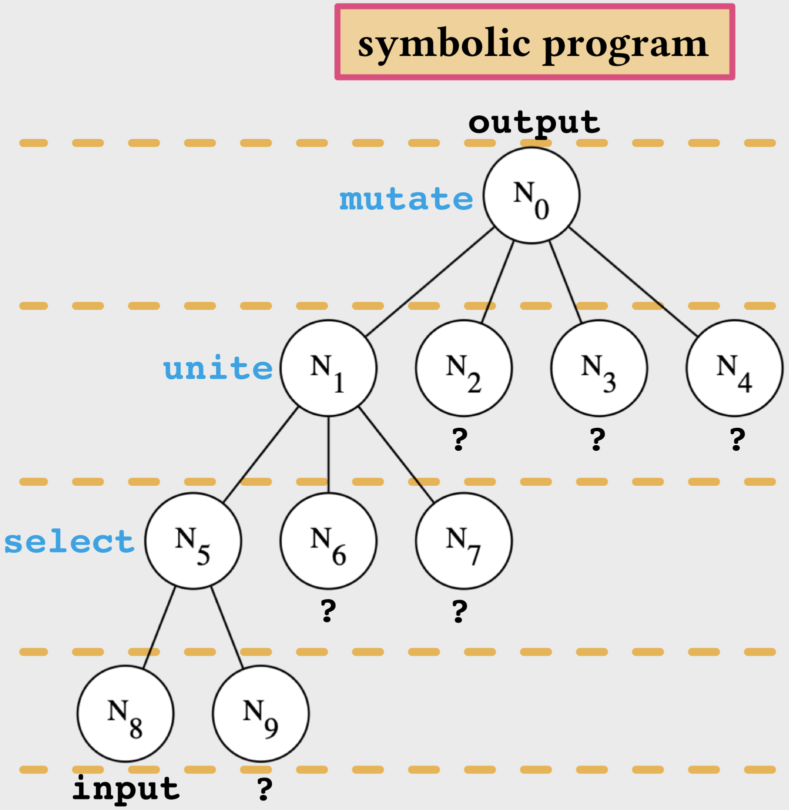


site	cat	sts
site1	SpeciesB_positive	1
site2	SpeciesB_negative	1
site3	SpeciesA_positive	2

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

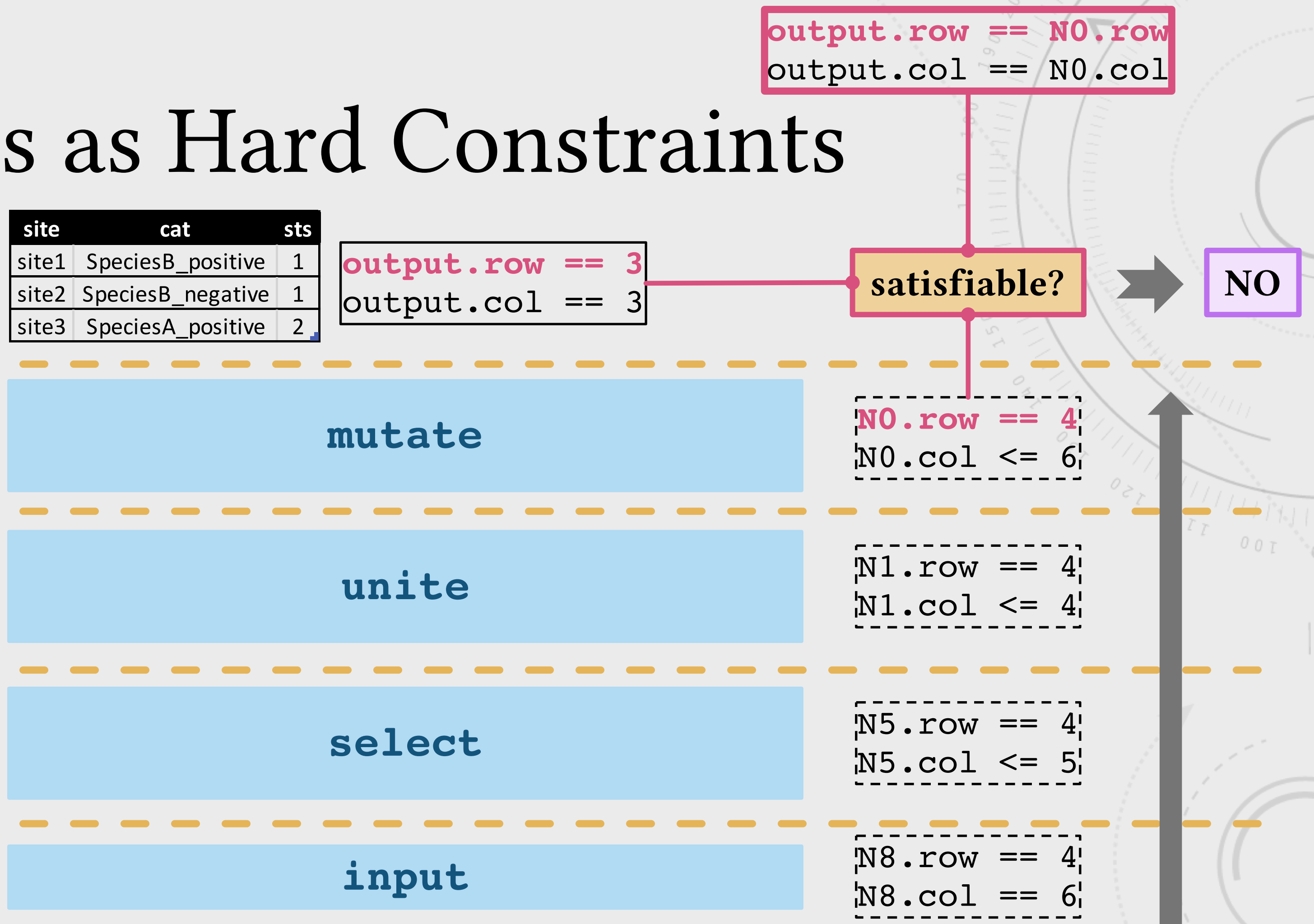
```
N5 = unite( ?, ?, ? )
N1 = group_by( N5, ? )
N0 = summarise( N1, ?, ?, ? )
```

Encoding Examples as Hard Constraints




```
N5 = select( ?, ? )
N1 = unite( N5, ?, ? )
N0 = mutate( N1, ?, ?, ? )
```

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

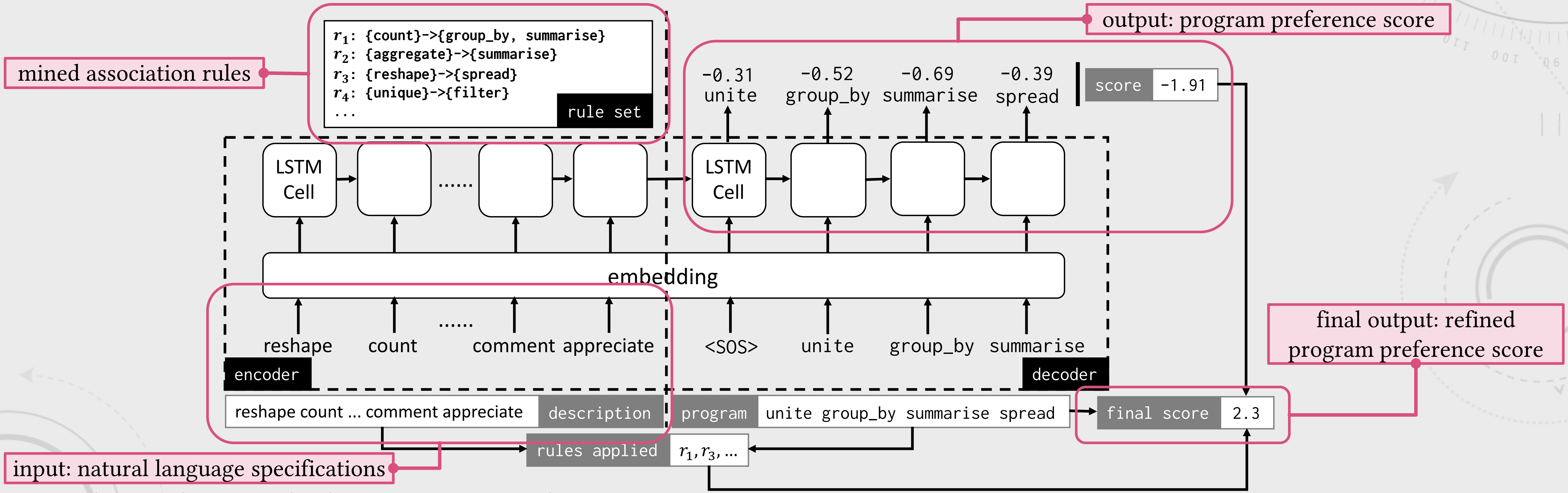


sample_ID	site	coll_date	species	TOT	inf_status
382870	site1	27/10/2007	SpeciesB	1	positive
382872	site2	27/10/2007	SpeciesB	1	negative
487405	site3	28/10/2007	SpeciesA	1	positive
487405	site3	28/10/2007	SpeciesA	1	positive

```
input.row == 4
input.col == 6
```

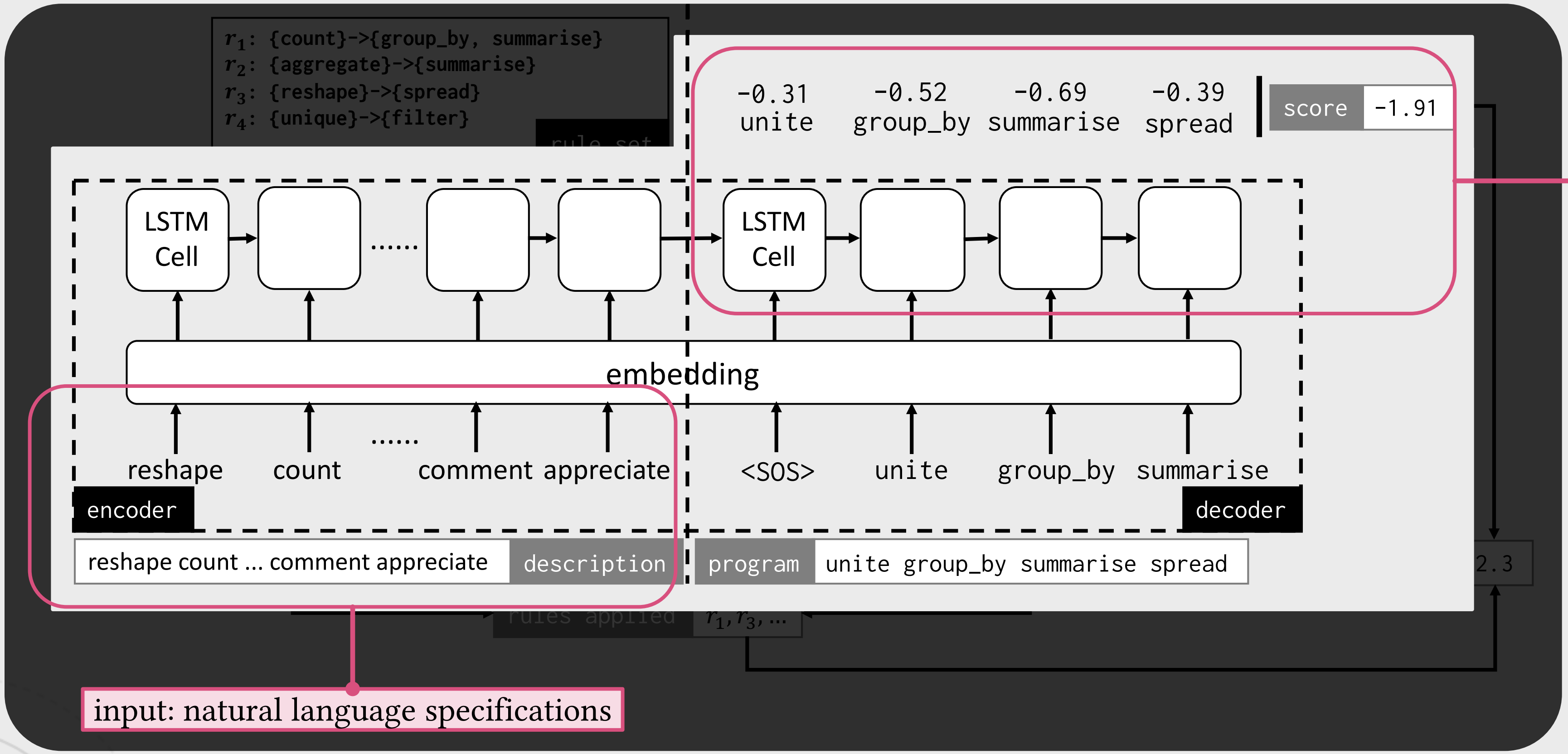
Encoding Natural Language Specifications

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



Encoding Natural Language Specifications

- The `seq2seq` model



output: program preference score

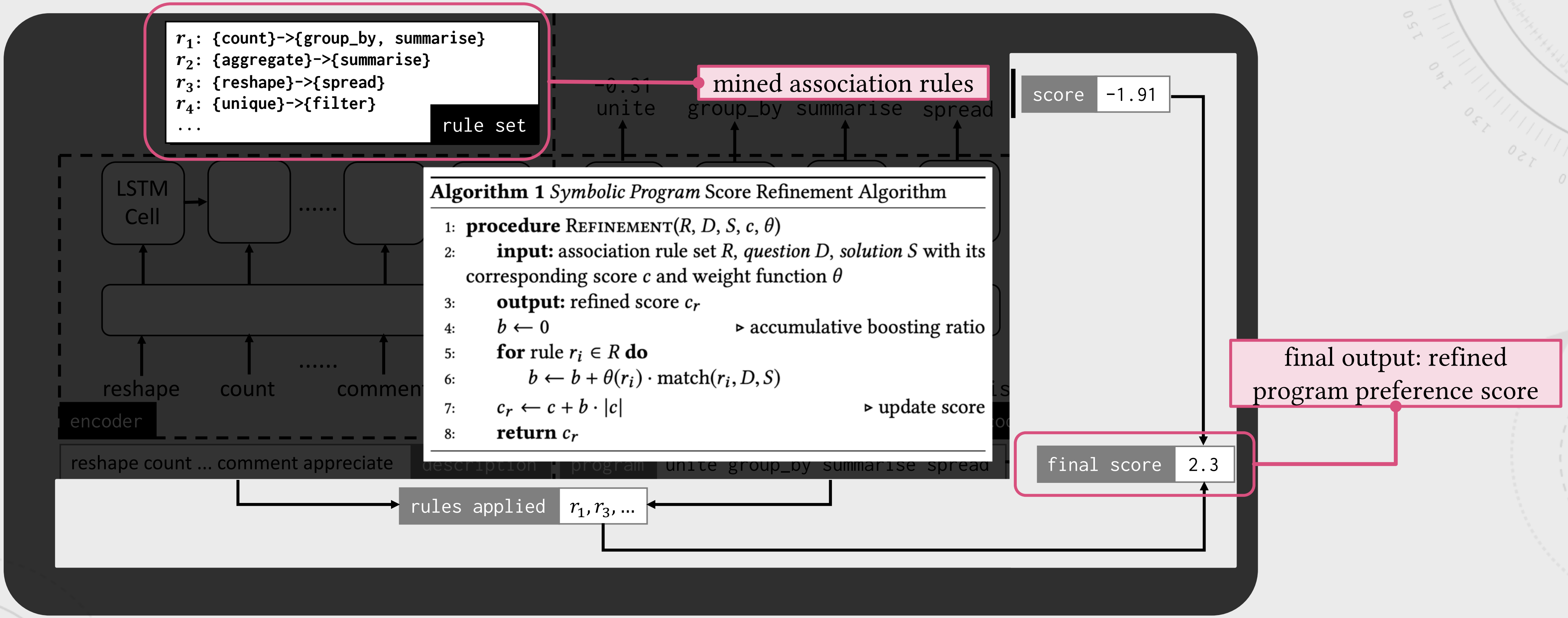
$$\ell(P|D) = \sum_i \log \pi(P_i|D)$$

*π is the seq2seq model

`seq2seq` model captures global user intents.

Encoding Natural Language Specifications

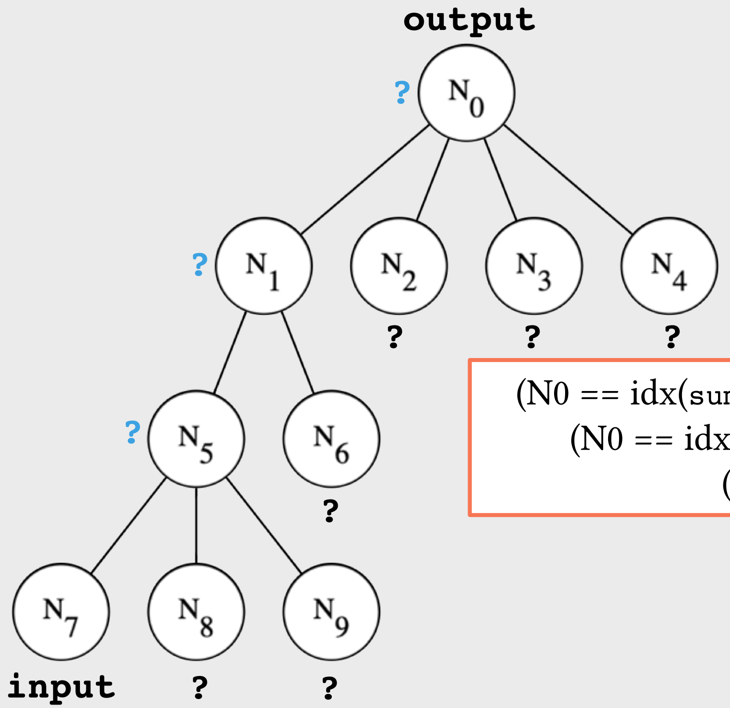
- The association rule module



Association rules captures local user intents.

Encoding Natural Language Specifications

- Encoding refined preference scores



2.3	`unite group_by summarise`

occurs predicates

occurs(`summarise`, 2.3) ∧
occurs(`group_by`, 2.3) ∧
occurs(`unite`, 2.3)

weight=2.3

(N0 == idx(`summarise`) ∨ N1 == idx(`summarise`) ∨ N5 == idx(`summarise`)) ∧
(N0 == idx(`group_by`) ∨ N1 == idx(`group_by`) ∨ N5 == idx(`group_by`)) ∧
(N0 == idx(`unite`) ∨ N1 == idx(`unite`) ∨ N5 == idx(`unite`))

hasChild predicates

hasChild(summarise, group_by, 2.3) ∧
hasChild(group_by, unite, 2.3)

weight=2.3

(N0 == idx(`summarise`) ⇒ N1 == idx(`group_by`)) ∧
(N0 == idx(`summarise`) ⇒ N5 == idx(`unite`)) ∧
(N1 == idx(`group_by`) ⇒ N5 == idx(`unite`))

encoding **occurs**(p_i, ω_i)

$$\bigwedge_{p_i \in \Lambda} \bigvee_{N_i \in N} N_i == idx(p_i)$$

encoding **hasChild**(p_i, p_j, ω_i)

$$\bigwedge_{p_i, p_j \in \Lambda, N_i \in N} N_i == idx(p_i) \Rightarrow \bigwedge_{N_j \in Ch(N_i)} N_j == idx(p_j)$$

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

Evaluation Results & Analysis

- Timeout: 5 mins
- Ablation Variants
  - *ngram*: built-in statistical model in MORPHEUS
  - *seq2seq*: MARS with `seq2seq` model
  - *hybrid*: MARS with `seq2seq` model and preference score refinement (association rules)

Table 1: Statistics for different model rankings.

model	*n-gram*	*seq2seq*	*hybrid*
average[*]	42	25	18
std.[1]	70	39	26

[1] standard deviation.
[*] computed based on the rankings of the correct solutions.

Table 2: Counts of top-1s and top-3s in different models.

model	*n-gram*	*seq2seq*	*hybrid*
Top-1 total[*]	0	8	11
Top-3 total[*]	2	18	29

[*] computed based on the rankings of the correct solutions.

Table 3: Statistics of running time.

model	avg. speedup[1]	#timeouts[*]
ngram	1x	11
seq2seq	6x	8
hybrid	15x	2

[1] average speedup on challenging solved benchmarks.
[*] number of timeouts on all benchmarks.

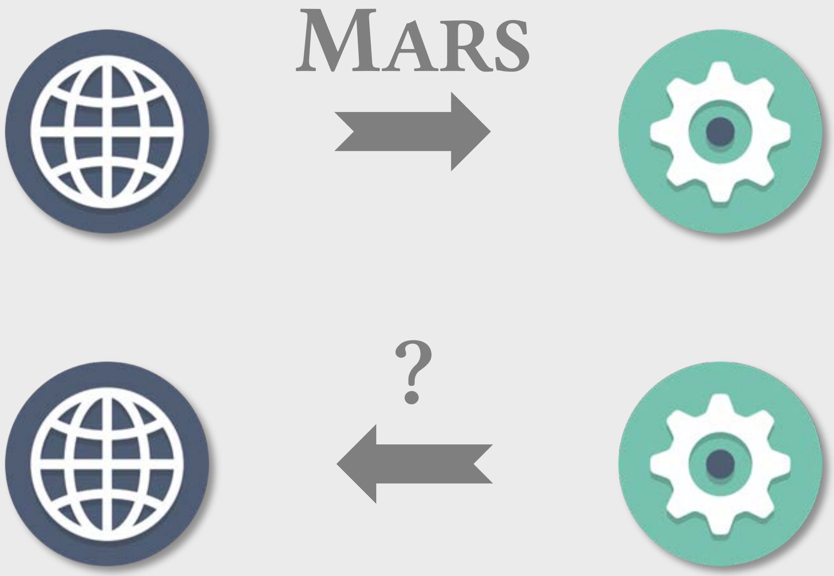
Discussions

- 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



CONCORD[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

[1] Chen, Y. et al. Program Synthesis Using Deduction-Guided Reinforcement Learning. CAV'20

Deduction-Guided Reinforcement Learning

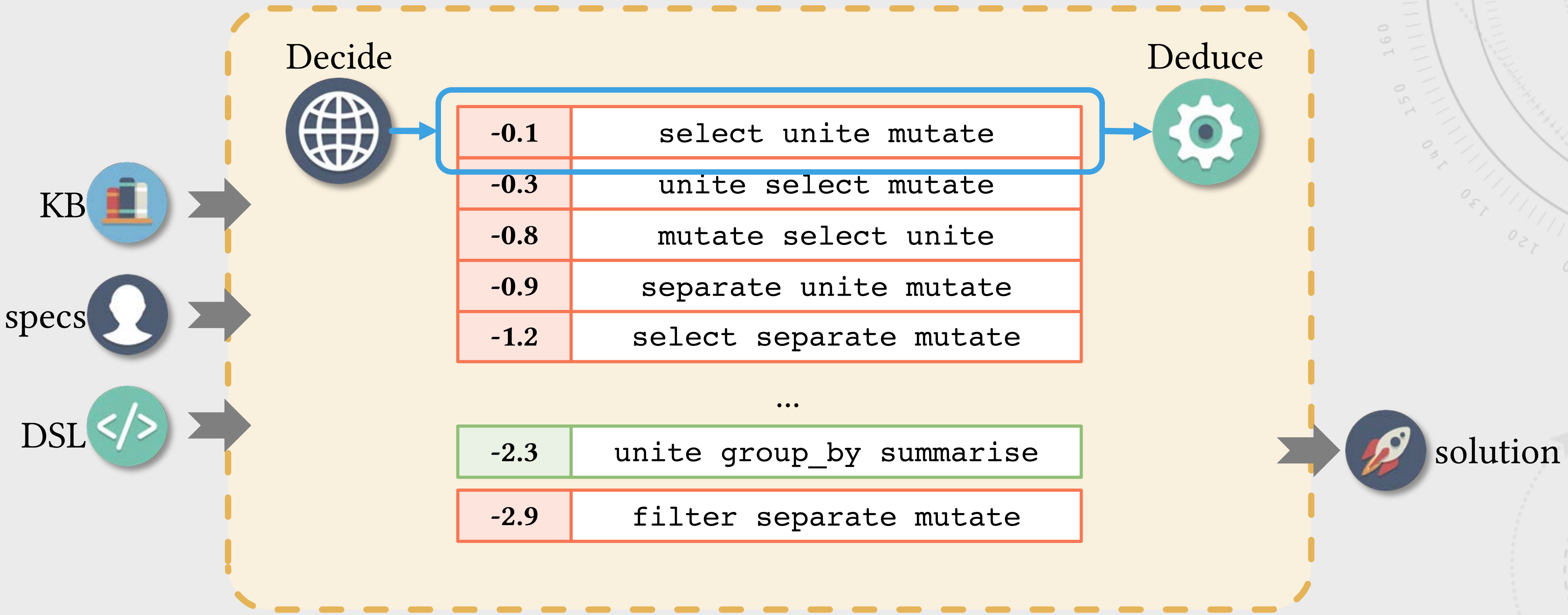
- Motivations
  - Feedback of deduction cannot be seamlessly used by statistical model

-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

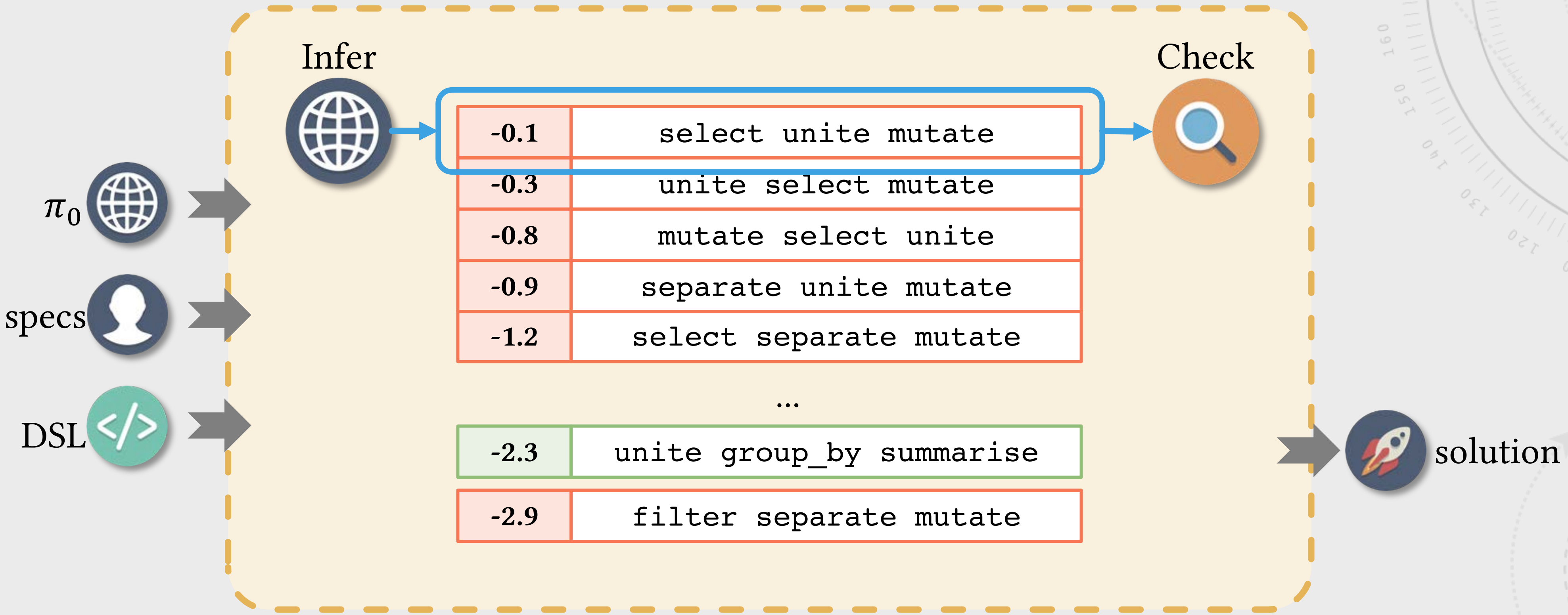


- rich and accurate feedback
- efficient search space pruning

- KB maintenance can be difficult
- no feedback incorporation

CONCORD: Deduction-Guided Reinforcement Learning

Statistical Approach

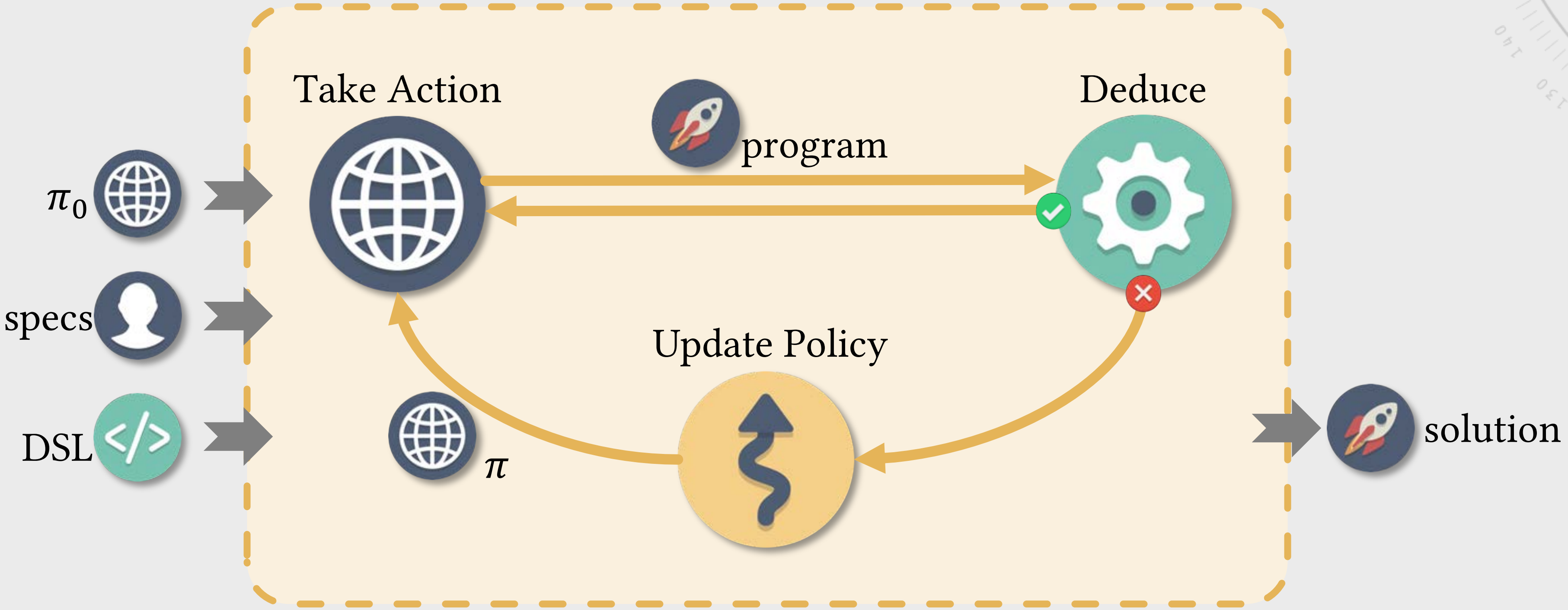


- data-driven candidate list
- can update policy seamlessly

- less informative feedback
- inefficient pruning

CONCORD: Deduction-Guided Reinforcement Learning

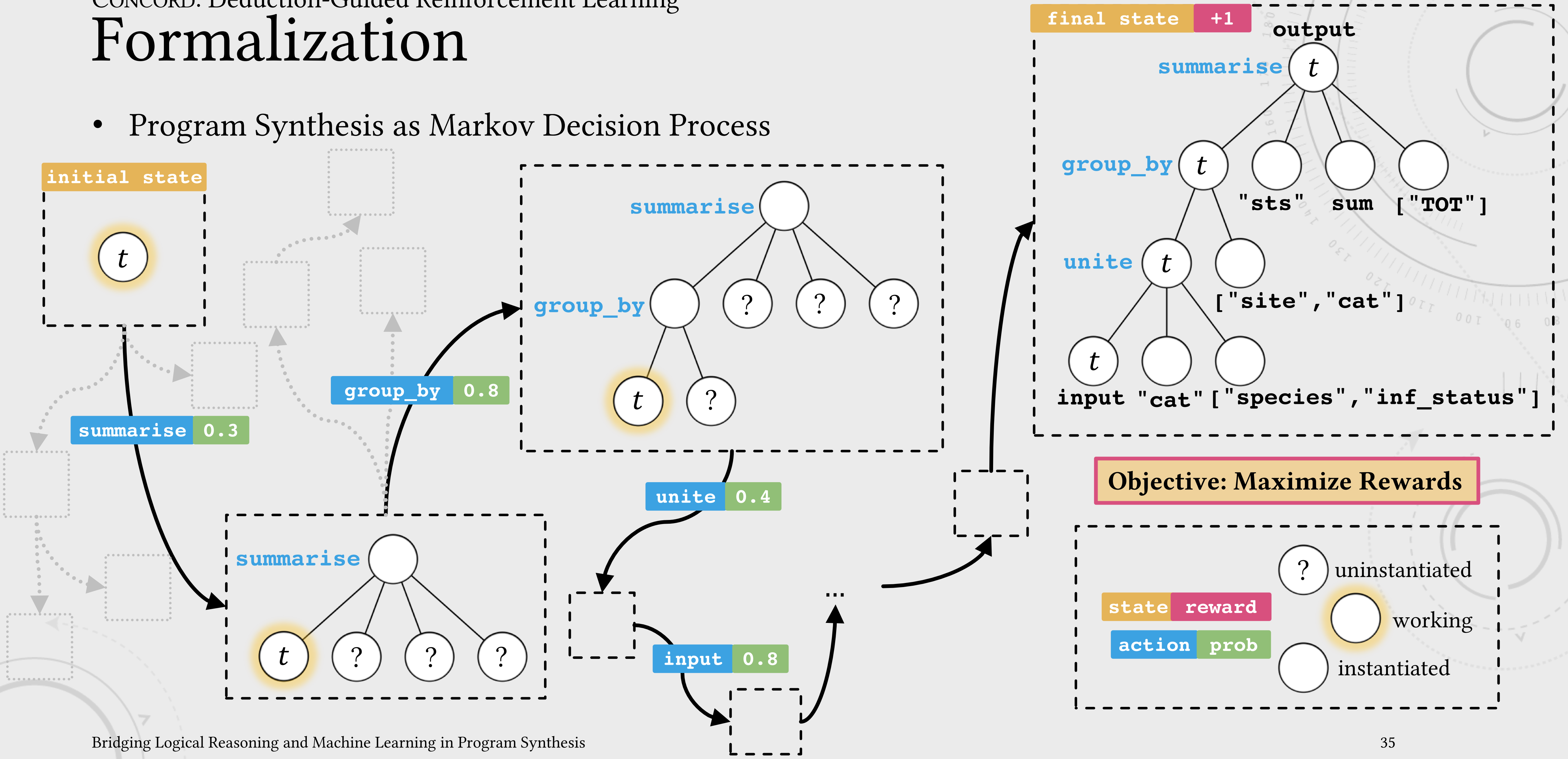
Framework Overview



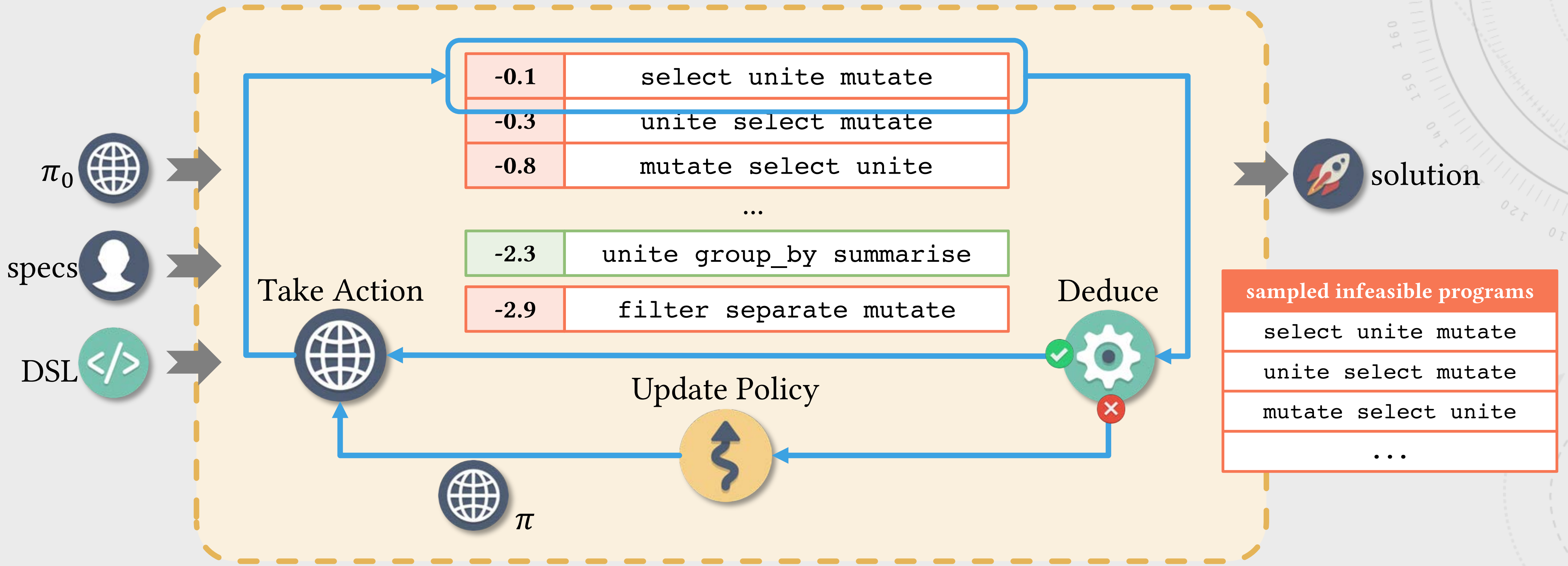
CONCORD: Deduction-Guided Reinforcement Learning

Formalization

- Program Synthesis as Markov Decision Process



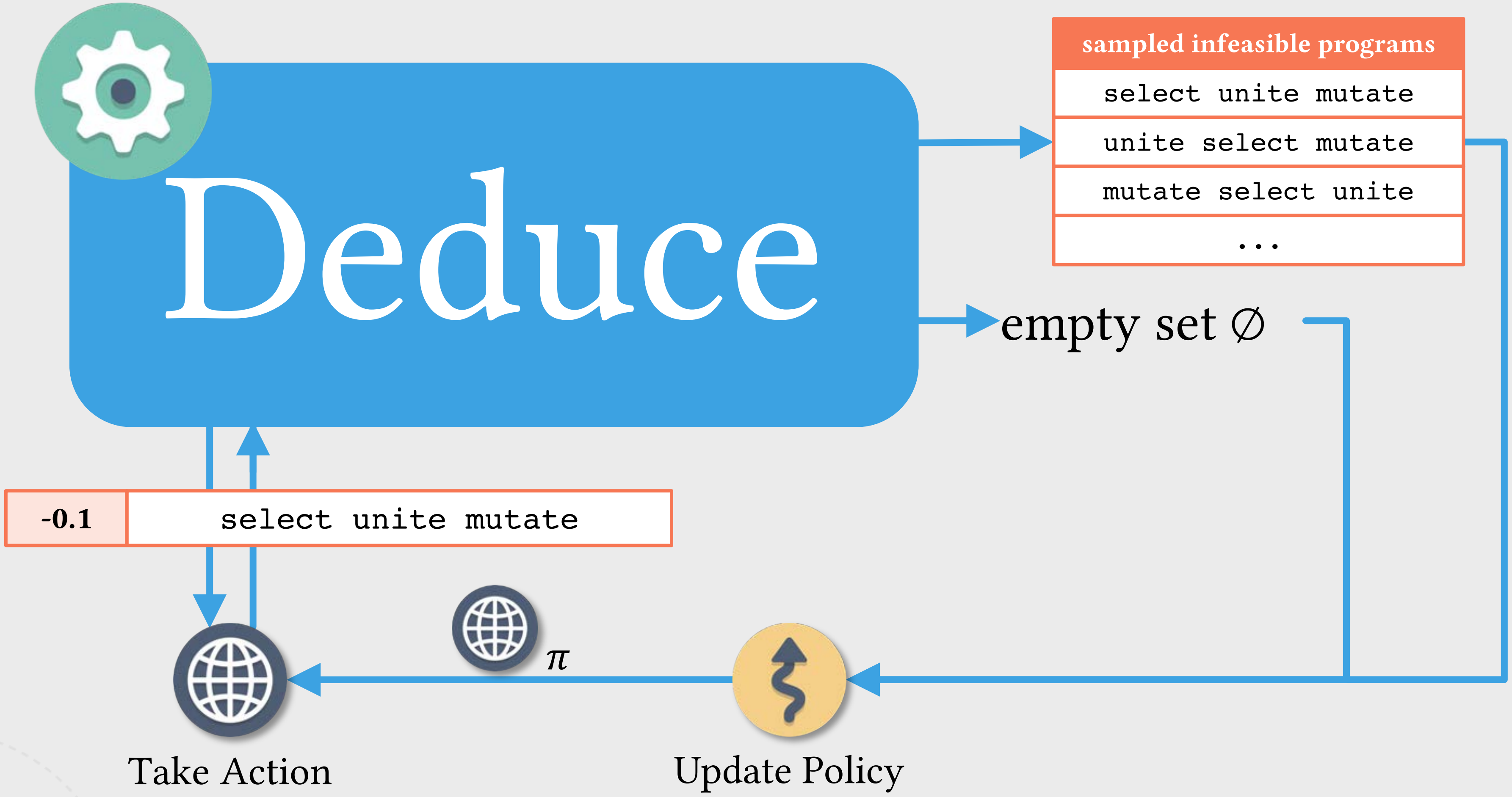
Running Example



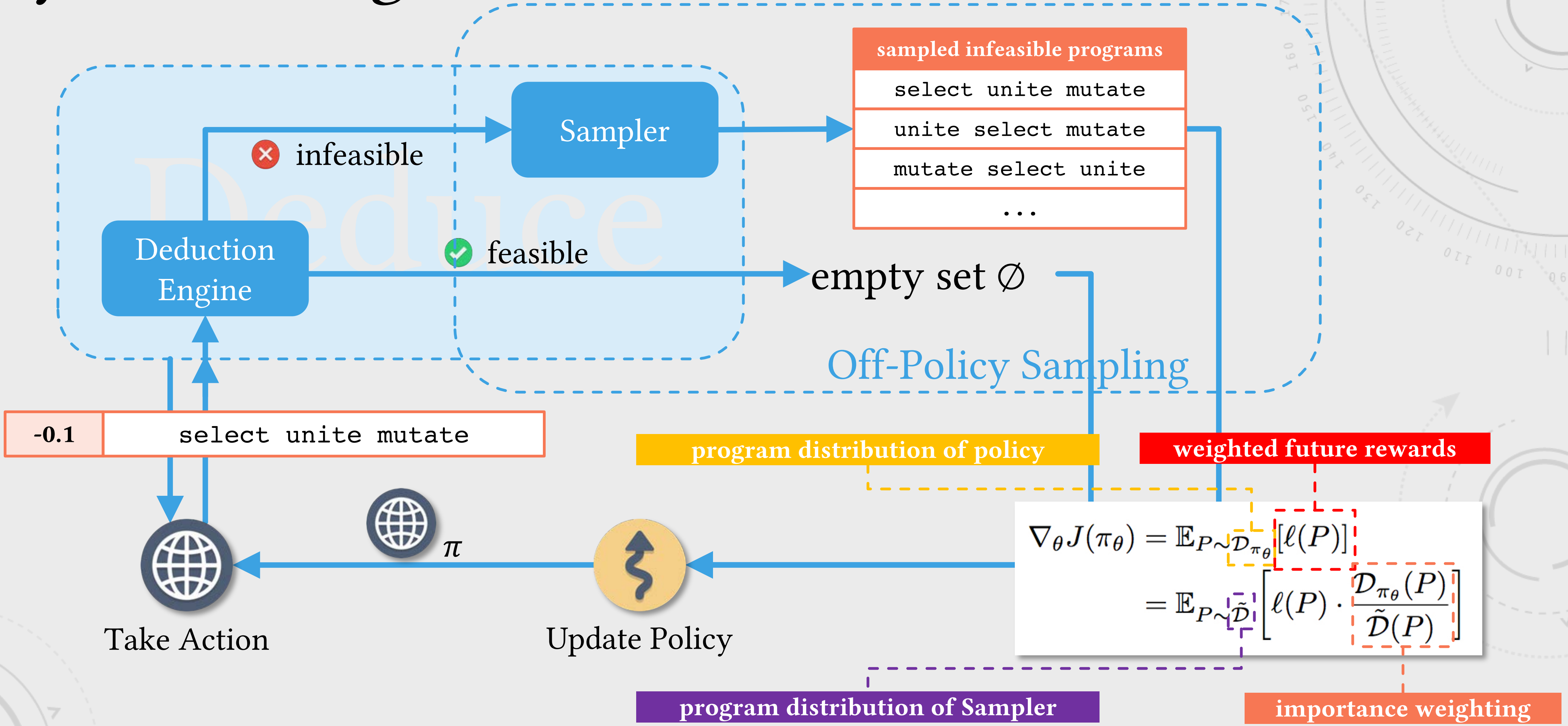
- feedback from deduction flows seamlessly to the policy update
- not only prune the search space, but also promote good candidates

CONCORD: Deduction-Guided Reinforcement Learning

Synthesis Algorithm

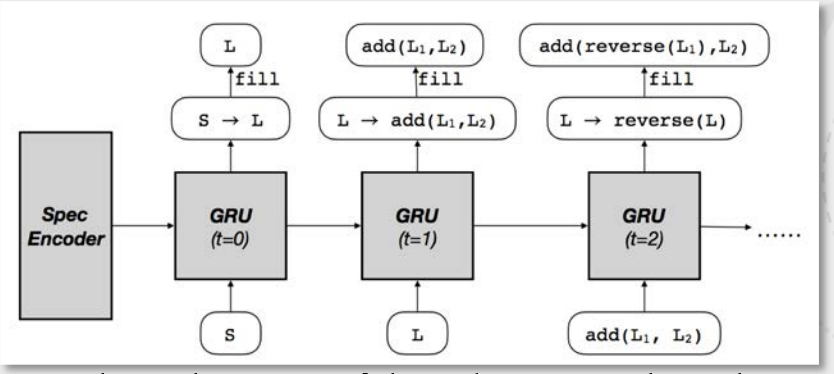


Synthesis Algorithm



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)



The architecture of the policy network used

Evaluation Results & Analysis

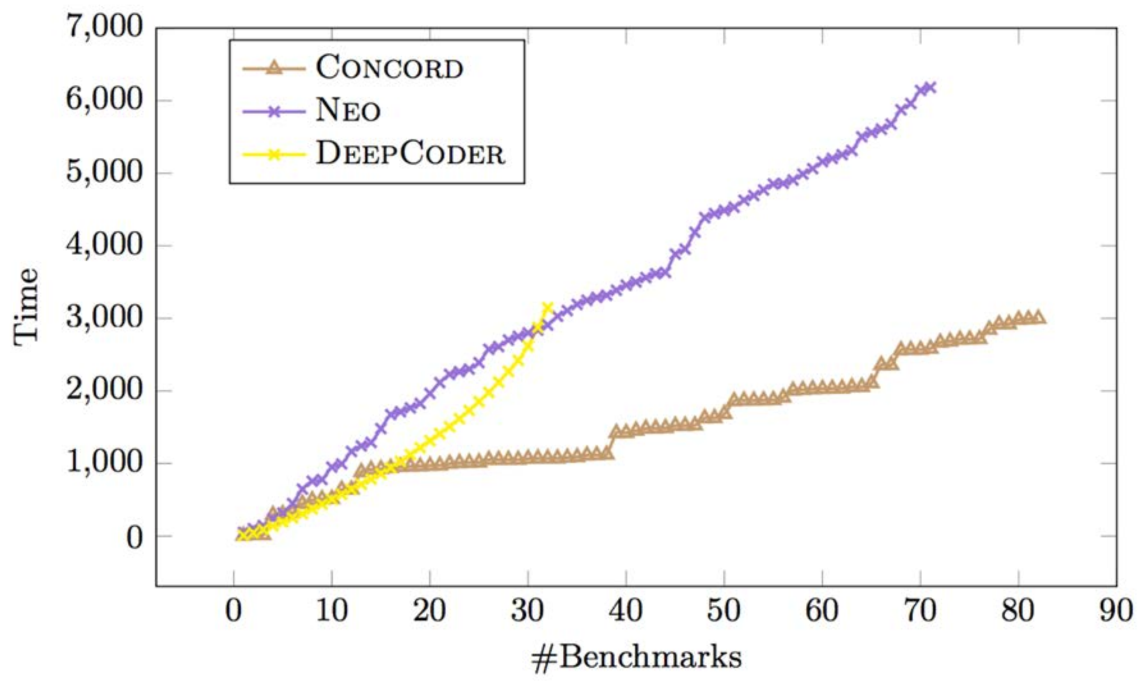


Fig. 5. Comparison between CONCORD, NEO, and DEEPCODER

tool	solved	time
CONCORD	82%	36s
NEO	71%	99s
DEEPCODER	32%	205s

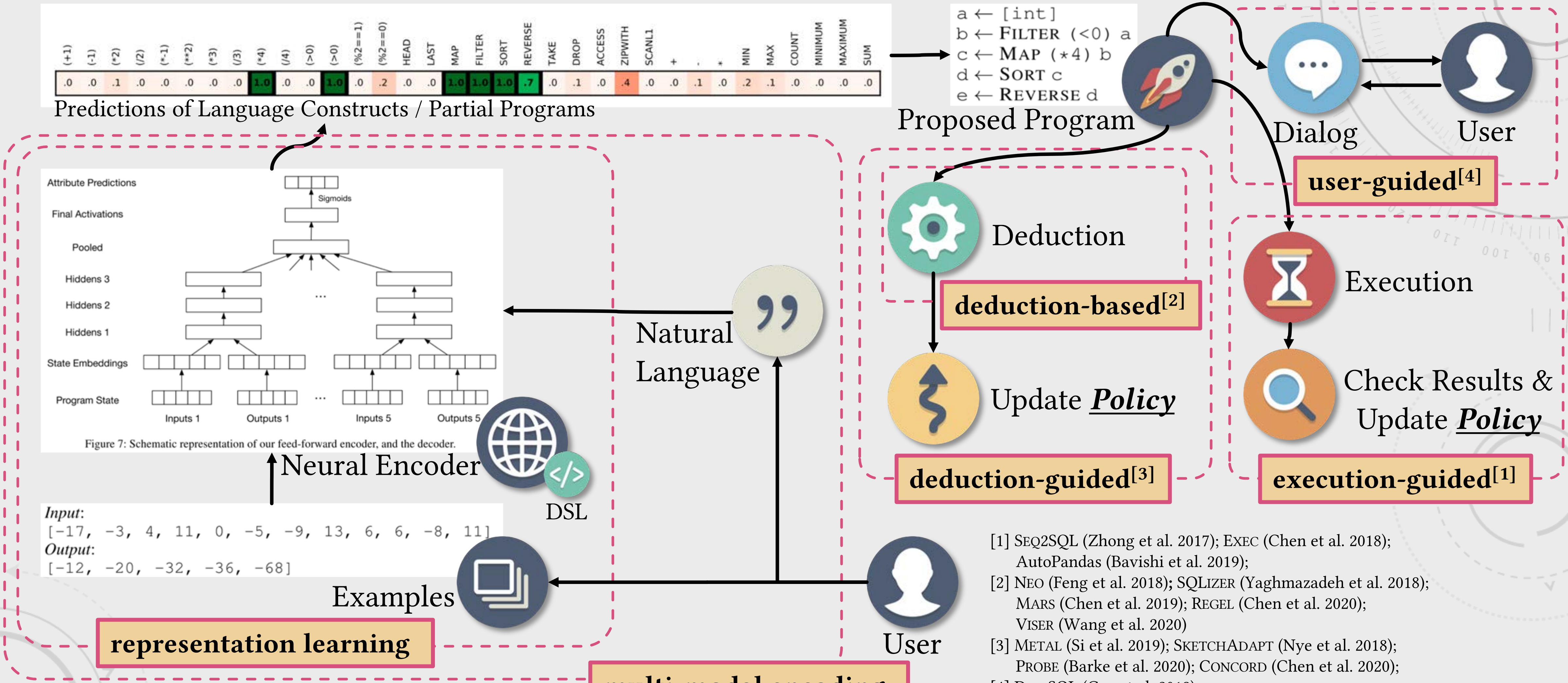
tool	solved	speedup over NEO
CONCORD	82%	8.71x

- Concord tightly couples statistical and deductive reasoning based on reinforcement learning.
- 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)



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

[2] NEO (Feng et al. 2018); SQLIZER (Yaghmazadeh et al. 2018); MARS (Chen et al. 2019); REGEL (Chen et al. 2020); VISER (Wang et al. 2020)

[3] METAL (Si et al. 2019); SKETCHADAPT (Nye et al. 2018); PROBE (Barke et al. 2020); CONCORD (Chen et al. 2020);

[4] DIALSQL (Gur et al. 2018);

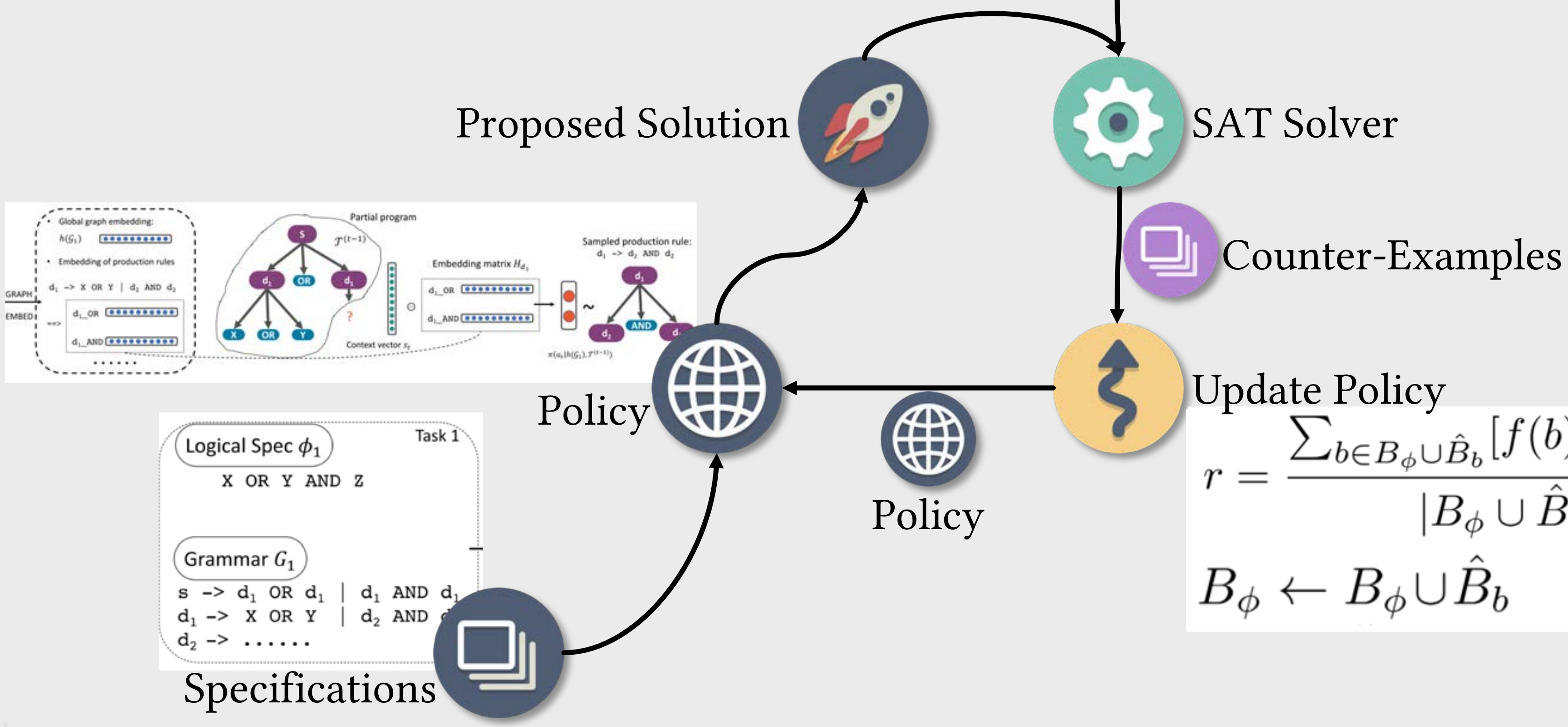
Related Works

- 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
[3] Dai, W.-Z. et al. Bridging Machine Learning and Logical Reasoning by Abductive Learning. NeurIPS'19

METAL[1] (The Reinforcement Learning Part)

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



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

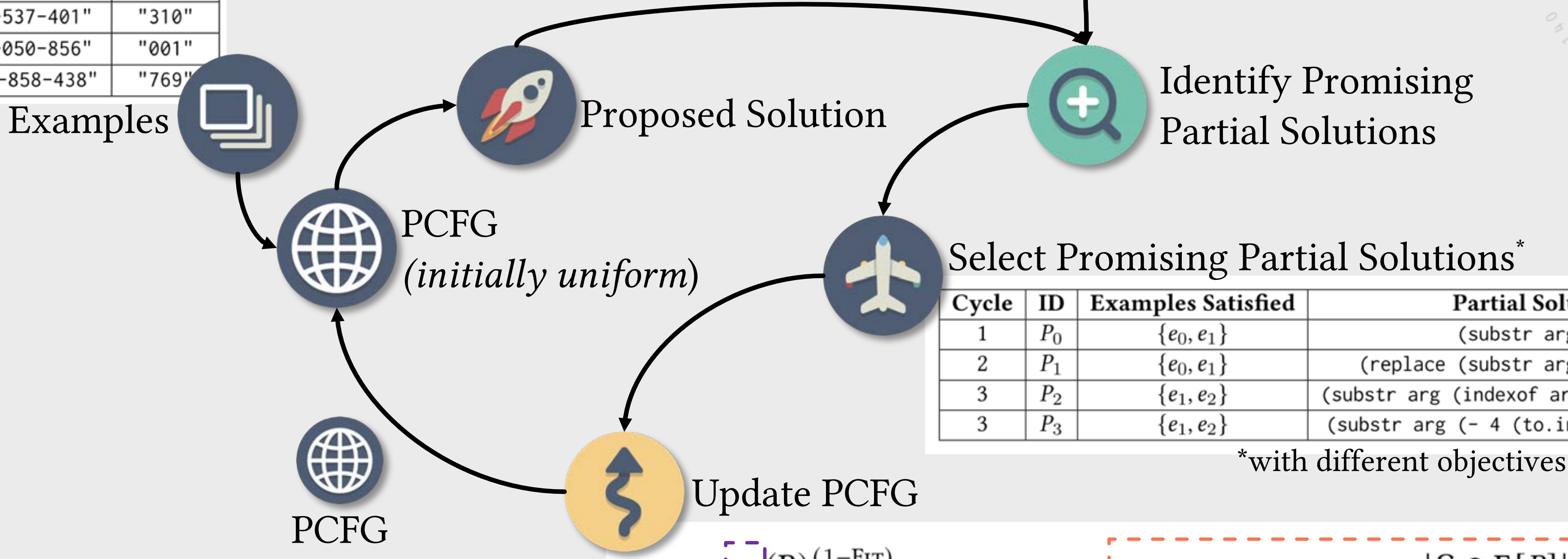
$$B_\phi \leftarrow B_\phi \cup \hat{B}_b$$

[1] Si, X. et al. Learning a Meta-Solver for Syntax-Guided Program Synthesis. ICLR'19

PROBE[1] (The Just-in-Time Learning Part)

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

ID	Input	Output
e_0	"+95 310-537-401"	"310"
e_1	"+72 001-050-856"	"001"
e_2	"+106 769-858-438"	"769"



Cycle	ID	Examples Satisfied	Partial Solutions	Cost
1	P_0	$\{e_0, e_1\}$	`(substr arg 4 3)`	20
2	P_1	$\{e_0, e_1\}$	`(replace (substr arg 4 3) " " arg)`	21
3	P_2	$\{e_1, e_2\}$	`(substr arg (indexof arg (at arg 5) 3) 3)`	37
3	P_3	$\{e_1, e_2\}$	`(substr arg (- 4 (to.int (at arg 4))) 3)`	37

*with different objectives and selection schemas

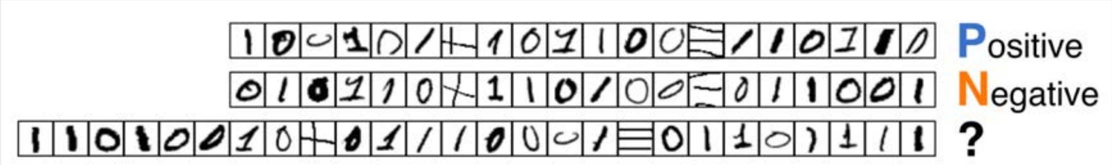
$$p(\mathrm{R}) = \frac{p_u(\mathrm{R})^{(1-\textsc{Fit})}}{Z} \quad \text{where} \quad \textsc{Fit} = \max_{\{P \in \mathrm{PSol} \mid \mathrm{R} \in \mathrm{tr}(P)\}} \frac{|\mathcal{E} \cap \mathrm{E}[P]|}{|\mathcal{E}|}$$

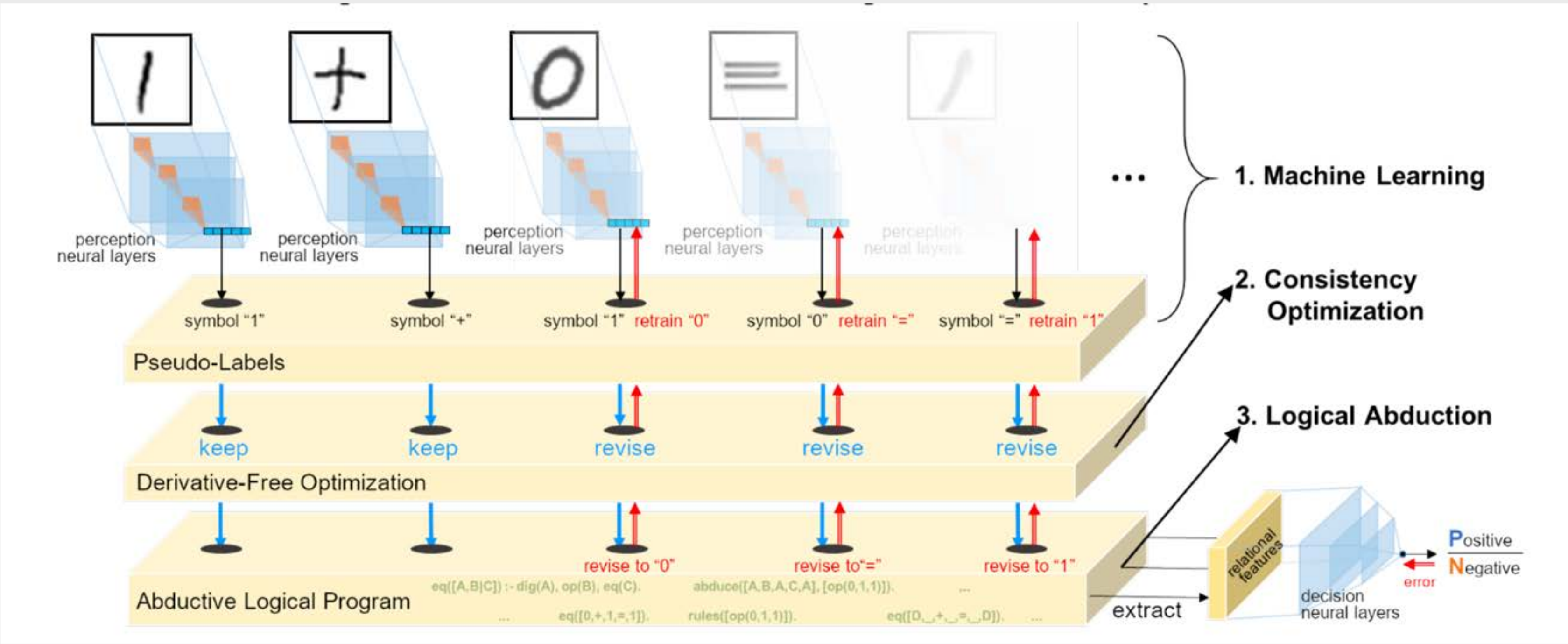
uniform distribution

highest proportion of IO satisfied

[1] Barke, S. et al. Just-in-Time Learning for Bottom-up Enumerative Synthesis. OOPSLA'20

ABL[1]: A Brief Overview





[1] Dai, W.-Z. et al. Bridging Machine Learning and Logical Reasoning by Abductive Learning. NeurIPS'19

Challenges, Conclusions & Future Works

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); ...

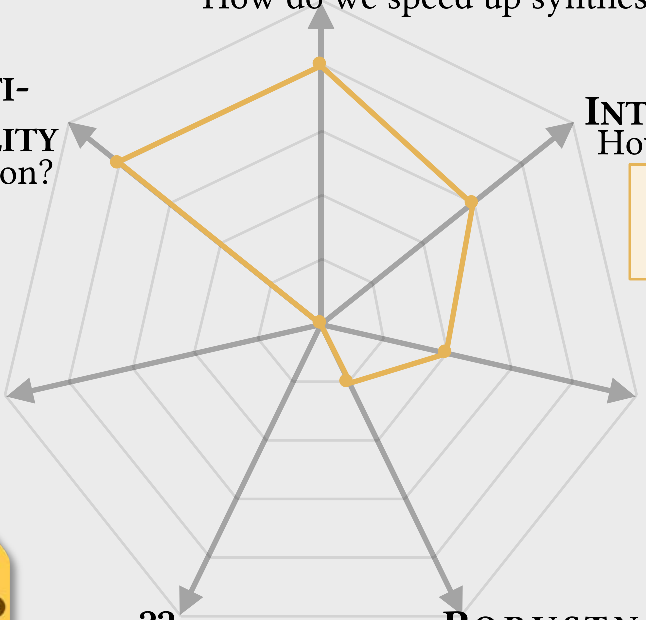
SCALABILITY
How do we speed up synthesis for given task?

MULTI-MODALITY
How do we process multi-modal information?

SEQ2SQL (Zhong et al. 2017); MARS (Chen et al. 2019); REGEL (Chen et al. 2020); VISER (Wang et al. 2020); ...

INTERACTIVITY
How do we access and utilize extra information from users?

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



??

??

CONTINUALITY
How do we distill useful knowledge across synthesis?

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

ROBUSTNESS
How do we tolerate specification mistakes/noises during synthesis?

FLASHFILL (Gulwani 2011); RULESYNTH (Singh 2017); BESTER (Peleg et al. 2020); ...

... and some more interesting dimensions?

References I

- 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
- Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Betteridge, J., Carlson, A., … Welling, J. Never-Ending Learning. In AAAI'15
- Chen, T., Goodfellow, I. J., & Shlens, J. Net2Net: Accelerating Learning via Knowledge Transfer. In ICLR'16
- Balog, M., Gaunt, A. L., Brockschmidt, M., Nowozin, S., & Tarlow, D. DeepCoder: Learning to Write Programs. In ICLR'17
- 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
- Yaghmazadeh, N., Wang, Y., Dillig, I., & Dillig, T. SQLizer: Query Synthesis from Natural Language. In OOPSLA'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
- Gur, I., Yavuz, S., Su, Y., & Yan, X. {D}ial{SQL}: Dialogue Based Structured Query Generation. In ACL'18
- Peleg, H., Shoham, S., & Yahav, E. Programming Not Only by Example. In ICSE'18
- Bavishi, R., Lemieux, C., Fox, R., Sen, K., & Stoica, I. AutoPandas: Neural-backed Generators for Program Synthesis. In OOPSLA'19

References II

- Chen, X., Liu, C., & Song, D. Execution-Guided Neural Program Synthesis. In ICLR'19
- **Chen, Y., Martins, R., & Feng, Y. Maximal Multi-layer Specification Synthesis. In FSE'19**
- Dai, W.-Z., Xu, Q., Yu, Y., & Zhou, Z.-H. Bridging Machine Learning and Logical Reasoning by Abductive Learning. In NeurIPS'19
- **Martins, R., Chen, J., Chen, Y., Feng, Y., & Dillig, I. Trinity: An Extensible Synthesis Framework for Data Science. In VLDB'19**
- Nye, M., Hewitt, L., Tenenbaum, J., & Solar-Lezama, A. Learning to Infer Program Sketches. In ICML'19
- Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. Continual lifelong learning with neural networks: A review. Neural Networks, 113, 54–71.
- Si, X., Yang, Y., Dai, H., Naik, M., & Song, L. Learning a Meta-Solver for Syntax-Guided Program Synthesis. In ICLR'19
- Barke, S., Peleg, H., & Polikarpova, N. Just-in-Time Learning for Bottom-up Enumerative Synthesis. In OOPSLA'20
- Chen, Q., Wang, X., Ye, X., Durrett, G., & Dillig, I. Multi-Modal Synthesis of Regular Expressions. In PLDI'20
- **Chen, Y., Wang, C., Bastani, O., Dillig, I., & Feng, Y. Program Synthesis Using Deduction-Guided Reinforcement Learning. In CAV'20**
- Ji, R., Liang, J., Xiong, Y., Zhang, L., & Hu, Z. Question Selection for Interactive Program Synthesis. In PLDI'20
- **Mariano, B., Chen, Y., Feng, Y., Lahiri, S., & Dillig, I. Demystifying Loops in Smart Contracts. In ASE'20**
- 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