EECS 598-008 & EECS 498-008: Intelligent Programming Systems

Lecture 8

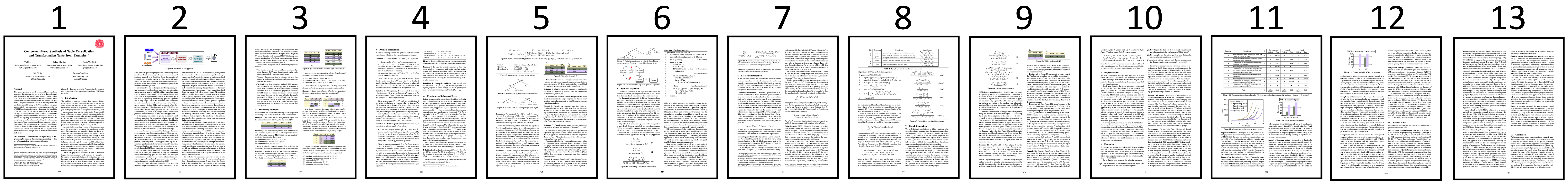
Announcements

- **Live**, remote discussion 3-4pm Friday (tomorrow)
  - Zoom link on course website
  - Discuss A2 (due next Monday)
- CFPP due **midnight Tuesday, September 28**
  - Submit your paper presentation preferences
  - Assignment will be released on Wednesday, after which you can start prep
- Course survey: https://forms.gle/XVQ3uMPwNomP1onn7
- More papers added to HotCRP

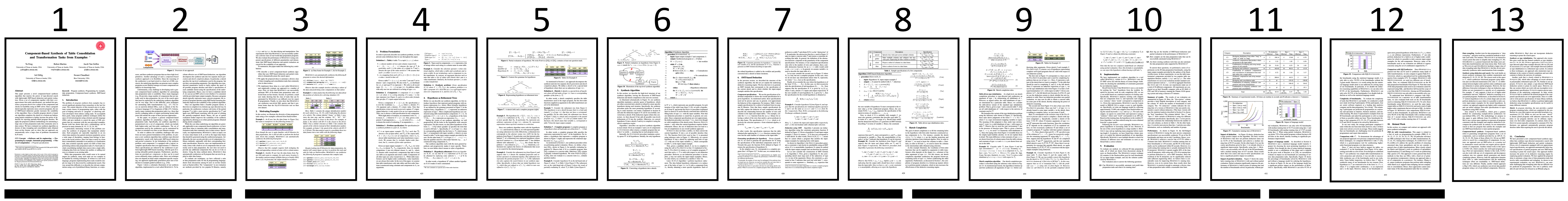
Today’s Agenda

- Present Morpheus paper
  - Talk about Morpheus
  - Talk about how to present a (PL) research paper in general

What Does A (PL) Research Paper Look Like?

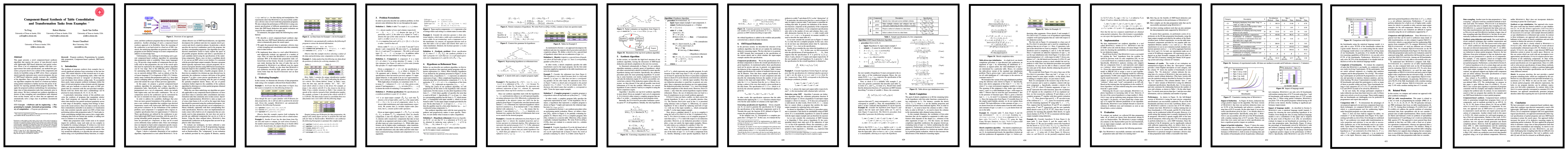


What Does A (PL) Research Paper Look Like?



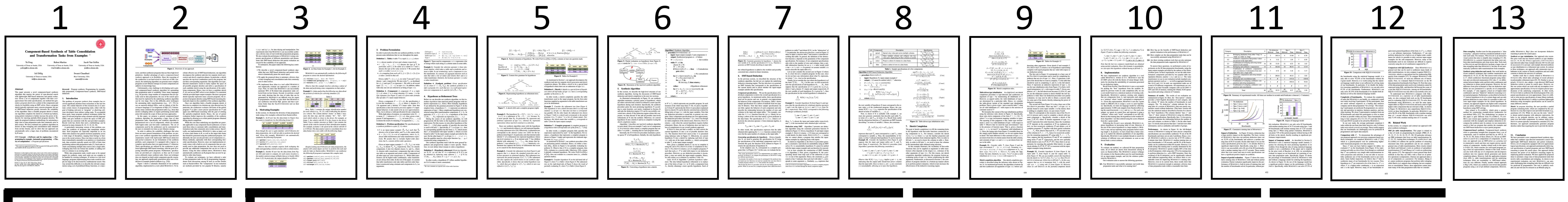
What Does A (PL) Research Paper Look Like?

1 2 3 4 5 6 7 8 9 10 11 12 13



Introduction: problem, idea, solution, evaluation, at high-level, 2 pages

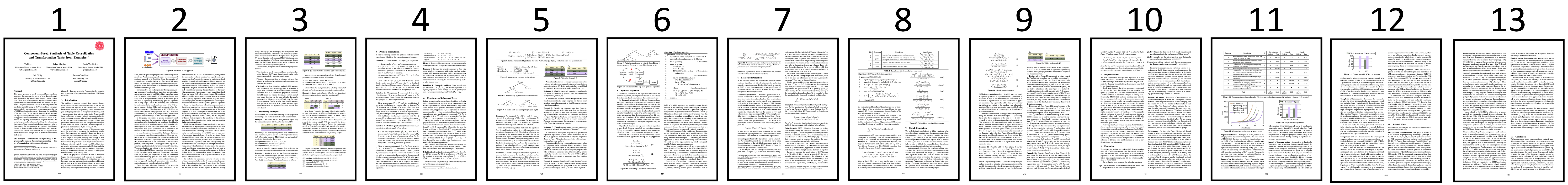
What Does A (PL) Research Paper Look Like?



Introduction: problem, idea, solution, evaluation, at high-level, 2 pages

Overview: illustrate problem, idea, solution, using examples, in more detail, 1-2 pages

What Does A (PL) Research Paper Look Like?

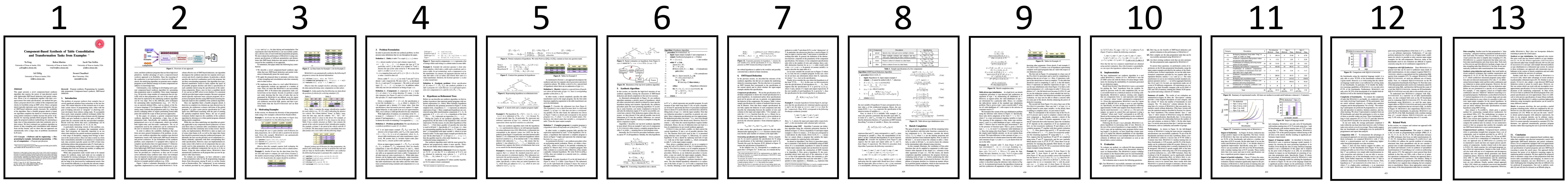


Introduction: problem, idea, solution, evaluation, at high-level, 2 pages

Overview: illustrate problem, idea, solution, using examples, in more detail, 1-2 pages

Technical sections: problem formulation, algorithms, with examples, in great detail, 4-5 pages

What Does A (PL) Research Paper Look Like?



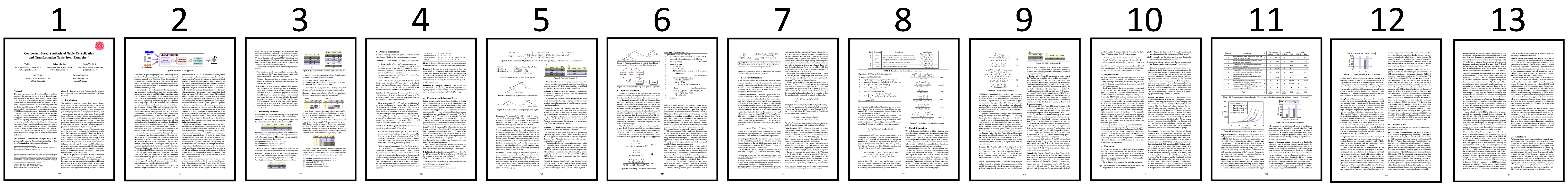
Introduction: problem, idea, solution, evaluation, at high-level, 2 pages

Overview: illustrate problem, idea, solution, using examples, in more detail, 1-2 pages

Technical sections: problem formulation, algorithms, with examples, in great detail, 4-5 pages

Implementation details, < 1 page

What Does A (PL) Research Paper Look Like?



Introduction: problem, idea, solution, evaluation, at high-level, 2 pages

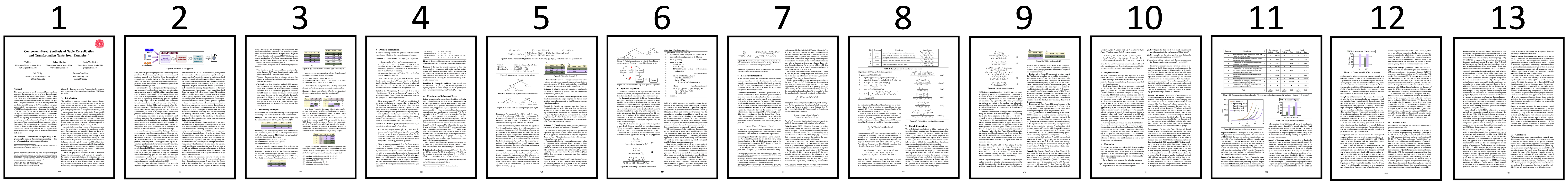
Overview: illustrate problem, idea, solution, using examples, in more detail, 1-2 pages

Technical sections: problem formulation, algorithms, with examples, in great detail, 4-5 pages

Implementation details, < 1 page

Evaluation: benchmarks, experimental setup, results, analysis, 2 pages

What Does A (PL) Research Paper Look Like?



Introduction: problem, idea, solution, evaluation, at high-level, 2 pages

Overview: illustrate problem, idea, solution, using examples, in more detail, 1-2 pages

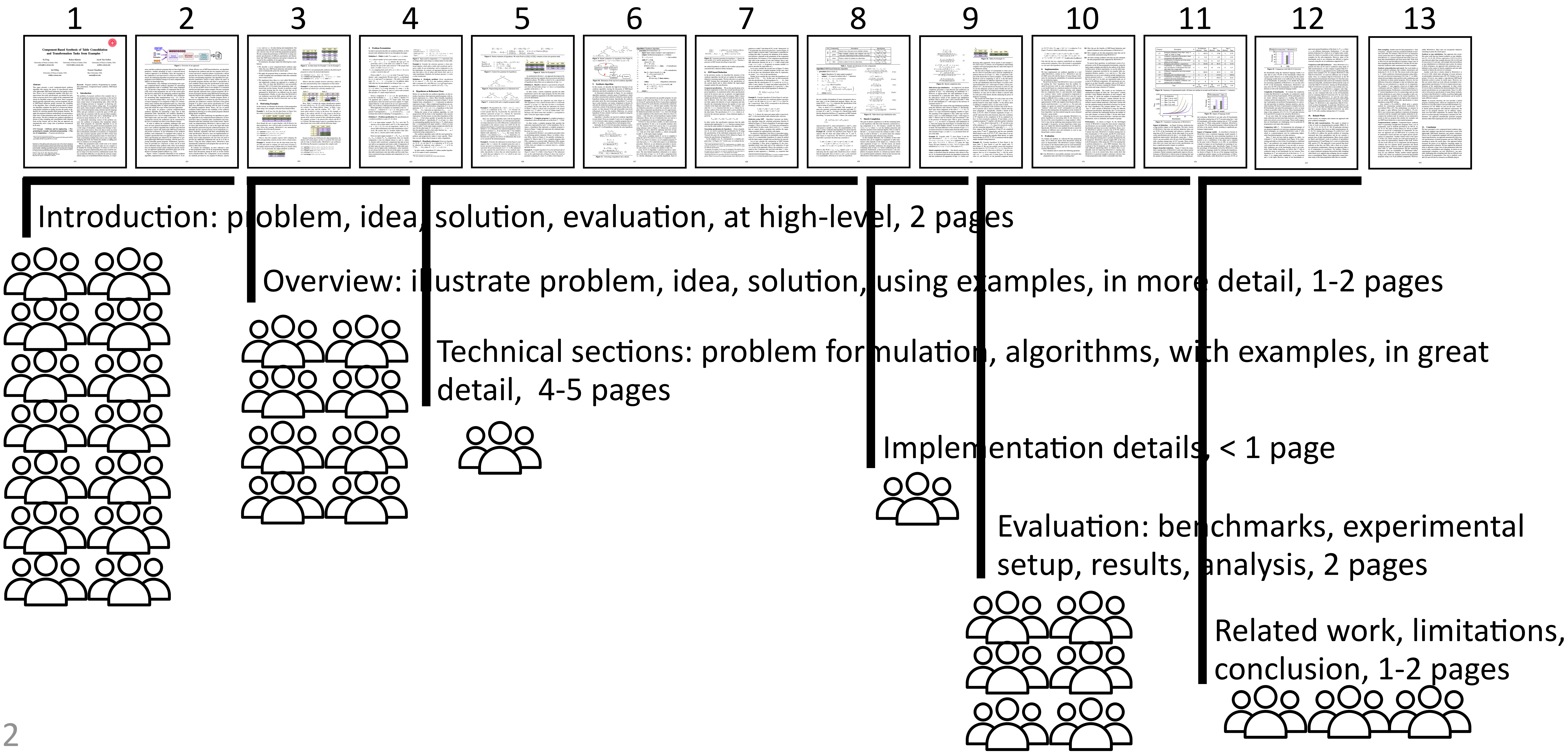
Technical sections: problem formulation, algorithms, with examples, in great detail, 4-5 pages

Implementation details, < 1 page

Evaluation: benchmarks, experimental setup, results, analysis, 2 pages

Related work, limitations, conclusion, 1-2 pages

What Does A (PL) Research Paper Look Like?



How To Present A Research Paper?

- **What's the problem? Why is it important?**
- Why is the problem challenging?
- How does the paper solve the problem? What's the key idea?
- Explain technique in more detail
- Evaluation
- Related work

Explain the Problem at a High-Level

- Data preparation
  - Data prep is tedious involving consolidating data sources, cleaning, reshaping, etc.
  - Especially important in the "big data" era

Component-Based Synthesis of Table Consolidation and Transformation Tasks from Examples *

Yu Feng
University of Texas at Austin, USA
yufeng@cs.utexas.edu

Ruben Martins
University of Texas at Austin, USA
rmartins@cs.utexas.edu

Jacob Van Geffen
University of Texas at Austin, USA
jsv@cs.utexas.edu

Isil Dillig
University of Texas at Austin, USA
isil@cs.utexas.edu

Swarat Chaudhuri
Rice University, USA
swarat@rice.edu

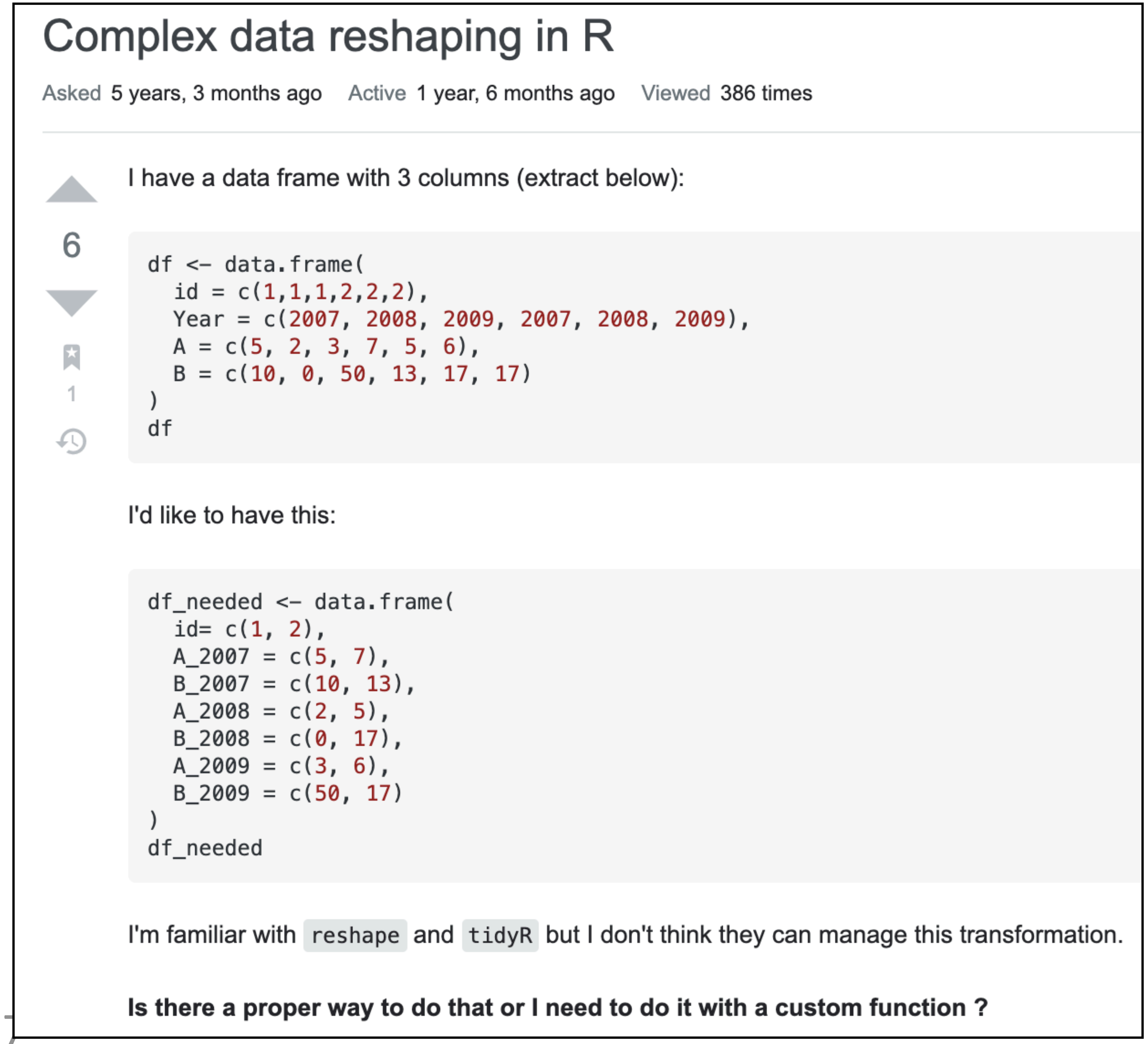
Explain the Problem at a High-Level

- Data preparation
  - Data prep is tedious involving consolidating data sources, cleaning, reshaping, etc.
  - Especially important in the "big data" era
- How to automatically synthesize table transformation programs?
  - Given a library of functions for table transformation and a set of input-output examples, how to find a program?

Explain the Problem at a High-Level

- Data preparation
  - Data prep is tedious involving consolidating data sources, cleaning, reshaping, etc.
  - Especially important in the "big data" era
- How to automatically synthesize table transformation programs?
  - Given a library of functions for table transformation and a set of input-output examples, how to find a program?
- Useful because with this technique, non-experts can also "write" programs

Use An Example to Illustrate the Problem



Complex data reshaping in R

Asked 5 years, 3 months ago Active 1 year, 6 months ago Viewed 386 times

6

1

I have a data frame with 3 columns (extract below):

```
df <- data.frame(
  id = c(1,1,1,2,2,2),
  Year = c(2007, 2008, 2009, 2007, 2008, 2009),
  A = c(5, 2, 3, 7, 5, 6),
  B = c(10, 0, 50, 13, 17, 17)
)
df
```

I'd like to have this:

```
df_needed <- data.frame(
  id= c(1, 2),
  A_2007 = c(5, 7),
  B_2007 = c(10, 13),
  A_2008 = c(2, 5),
  B_2008 = c(0, 17),
  A_2009 = c(3, 6),
  B_2009 = c(50, 17)
)
df_needed
```

I'm familiar with `reshape` and `tidyR` but I don't think they can manage this transformation.

Is there a proper way to do that or I need to do it with a custom function ?

Use An Example to Illustrate the Problem

Complex data reshaping in R

Asked 5 years, 3 months ago   Active 1 year, 6 months ago   Viewed 386 times

6

1

I have a data frame with 3 columns (extract below):

```
df <- data.frame(
  id = c(1,1,1,2,2,2),
  Year = c(2007, 2008, 2009, 2007, 2008, 2009),
  A = c(5, 2, 3, 7, 5, 6),
  B = c(10, 0, 50, 13, 17, 17)
)
df
```

I'd like to have this:

```
df_needed <- data.frame(
  id= c(1, 2),
  A_2007 = c(5, 7),
  B_2007 = c(10, 13),
  A_2008 = c(2, 5),
  B_2008 = c(0, 17),
  A_2009 = c(3, 6),
  B_2009 = c(50, 17)
)
df_needed
```

I'm familiar with `reshape` and `tidyR` but I don't think they can manage this transformation.

Is there a proper way to do that or I need to do it with a custom function ?

id	year	A	B
1	2007	5	10
2	2009	3	50
1	2007	5	17
2	2009	6	17

Input Example

```
df1=gather(input,var,val,id,A,B)
df2=unite(df1,yearvar,var,year)
df3=spread(df2,yearvar,val)
```

Output Example

id	*A_2007*	*B_2007*	*A_2009*	*B_2009*
1	*5*	*10*	*5*	*17*
2	*3*	*50*	*6*	*17*

Use More Examples to Illustrate the Problem

flight	origin	dest
11	EWR	SEA
725	JFK	BQN
495	JFK	SEA
461	LGA	ATL
1696	EWR	ORD
1670	EWR	SEA

Input Example

origin	*n*	*prop*
EWR	*2*	*0.6666667*
JFK	*1*	*0.3333333*

Output Example

"find out proportions of flights to destination(Seattle)"

```
df1=filter(input, dest == "SEA")
df2=summarize(group_by(df1, origin), n = n())
df3=mutate(df2, prop = n / sum(n))
```

Use More Examples to Illustrate the Problem

"I want to combine these 2 data frames to create a new one which looks like this"

Table 1:

frame	X1	X2	X3
1	0	0	0
2	10	15	0
3	15	10	0

Table 2:

frame	X1	X2	X3
1	0	0	0
2	14.53	12.57	0
3	13.90	14.65	0

Input Example

frame	pos	carid	speed
2	X1	10	14.53
3	X2	10	14.65
2	X2	15	12.57
3	X1	15	13.90

Output Example

```
df1=gather(table1,pos,carid,X1,X2,X3)
df2=gather(table2,pos,speed,X1,X2,X3)
df3=inner_join(df1,df2)
df4=filter(df3,carid != 0)
df5=arrange(df4,carid,frame)
```

How To Present A Research Paper?

- What’s the problem? Why is it important?
- **Why is the problem challenging?**
- How does the paper solve the problem? What’s the key idea?
- Explain technique in more detail
- Evaluation
- Related work

What are the Challenges?

- Problem: given a library of functions and a set of examples, find a program using functions in the library that satisfies the provided examples.

What are the Challenges?

- Problem: given a library of functions and a set of examples, find a program using functions in the library that satisfies the provided examples.

- **Key challenge: scalability**
  - Large number of functions in library (e.g., R)
  - Previous approaches consider very small languages

How To Present A Research Paper?

- What's the problem? Why is it important?
- Why is the problem challenging?
- **How does the paper solve the problem? What's the key idea?**
- Explain technique in more detail
- Evaluation
- Related work

Key idea

- Lightweight SMT-based deduction for pruning

How To Present A Research Paper?

- What’s the problem? Why is it important?
- Why is the problem challenging?
- How does the paper solve the problem? What’s the key idea?
- **Explain technique in more detail**
- Evaluation
- Related work

Problem Formulation

- Given an input-output example E and a library of components Λ, find a program $\lambda \overrightarrow{x} . e$ over Λ such that (1) e is well-typed over Λ and (2) $(\lambda \overrightarrow{x} . e)E_{in} = E_{out}$

Problem Formulation

- Given an input-output example E and a library of components Λ, find a program $\lambda \overrightarrow{x} \, . \, e$ over Λ such that (1) e is well-typed over Λ and (2) $(\lambda \overrightarrow{x} \, . \, e)E_{in} = E_{out}$

- Also known as "component-based program synthesis"
  - A program is a loop-free composition of components from a given library

Problem Formulation

- Given an input-output example E and a library of components Λ, find a program $\lambda \overrightarrow{x} \,.\, e$ over Λ such that (1) e is well-typed over Λ and (2) $(\lambda \overrightarrow{x} \,.\, e)E_{in} = E_{out}$

- Also known as "component-based program synthesis"
  - A program is a loop-free composition of components from a given library
  - Component-based vs. DSL-based
    - Any type-safe composition is okay vs. syntactic restrictions imposed by grammar

Important Concepts

- Hypothesis: “partial program”

$$
\begin{array}{lll}
\text{Term } t & := & \text{const} \mid y_i \mid \mathcal{X}(t_1, ..., t_n)\ (\mathcal{X} \in \Lambda_v) \\
\text{Qualifier } \mathcal{Q} & := & (x, \mathsf{T}) \mid \lambda y_1, \ldots y_n.\, t \\
\text{Hypothesis } \mathcal{H} & := & (?_i : \tau) \mid (?_i : \tau)@\mathcal{Q} \\
 & & \mid\, ?_i^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)\ \ (\mathcal{X} \in \Lambda_{\mathsf{T}})
\end{array}
$$

Figure 5. Context-free grammar for hypotheses

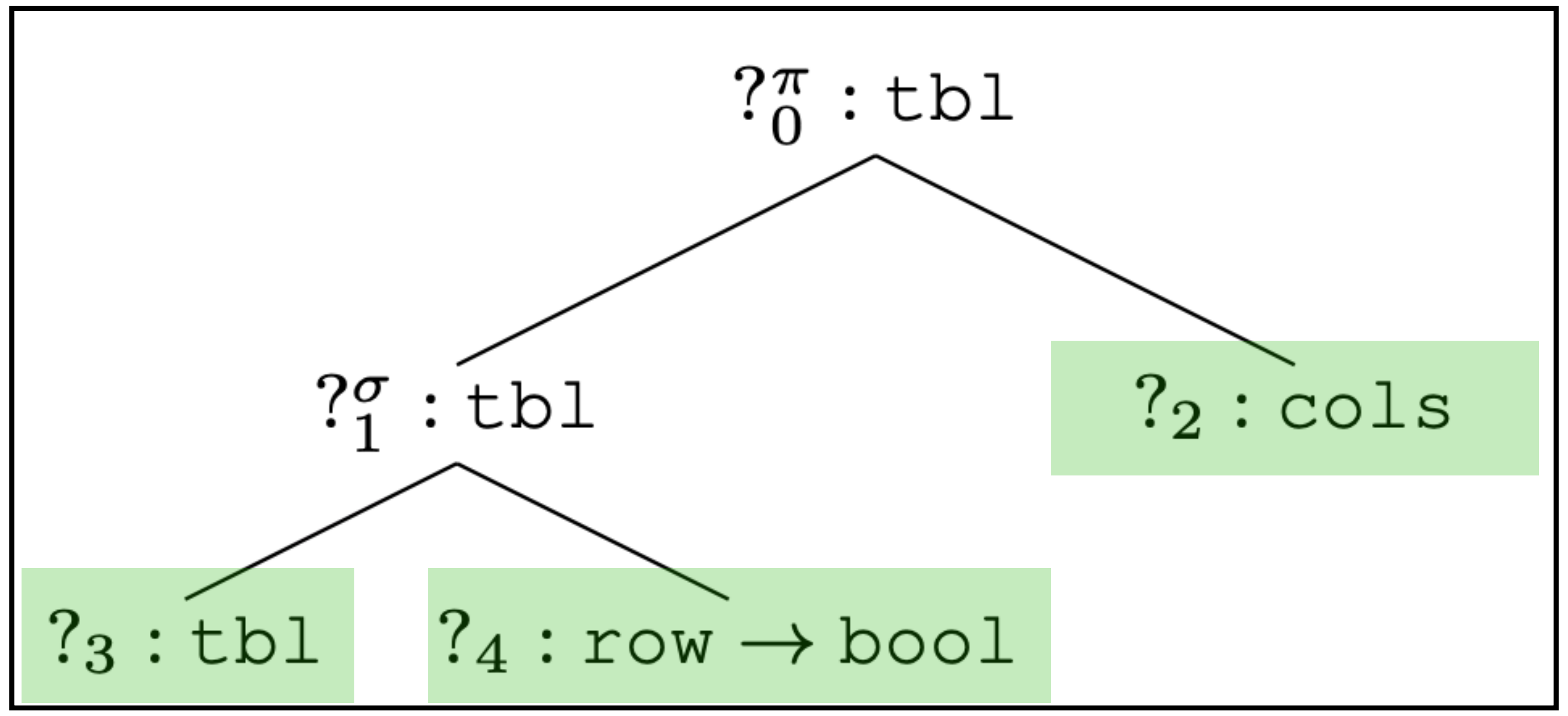
Important Concepts

- Hypothesis: “partial program”

$$
\begin{array}{lll}
\text{Term } t & := & \text{const} \mid y_i \mid \mathcal{X}(t_1, ..., t_n)\ (\mathcal{X} \in \Lambda_v) \\
\text{Qualifier } \mathcal{Q} & := & (x, \mathsf{T}) \mid \lambda y_1, \ldots y_n.\, t \\
\text{Hypothesis } \mathcal{H} & := & (?_i : \tau) \mid (?_i : \tau)@\mathcal{Q} \\
 & & \mid\ ?_i^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)\ \ (\mathcal{X} \in \Lambda_{\mathsf{T}})
\end{array}
$$

Figure 5. Context-free grammar for hypotheses

Leaf node is hole (base case)



Important Concepts

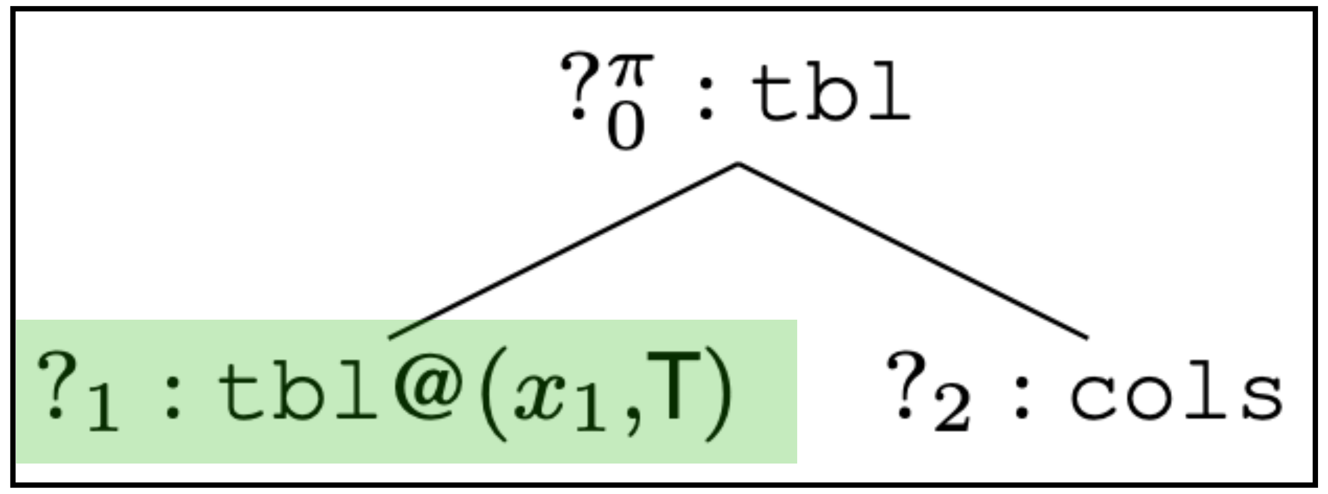
- Hypothesis: “partial program”

Term t	$:=$	$\text{const} \mid y_i \mid \mathcal{X}(t_1, ..., t_n)\ (\mathcal{X} \in \Lambda_v)$
Qualifier $\mathcal{Q}$	$:=$	$(x, \mathsf{T}) \mid \lambda y_1, \ldots y_n.\, t$
Hypothesis $\mathcal{H}$	$:=$	$(?_i : \tau) \mid (?_i : \tau)@\mathcal{Q}$
		$\mid\ ?_i^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)\ \ (\mathcal{X} \in \Lambda_{\mathsf{T}})$

Figure 5. Context-free grammar for hypotheses

Leaf node is hole with qualifier (base case)

A qualifier expresses additional information about the hole, i.e., how to fill the hole



$?_1$ must be replaced with variable x_1 which binds to table T, i.e., this leaf node is concrete

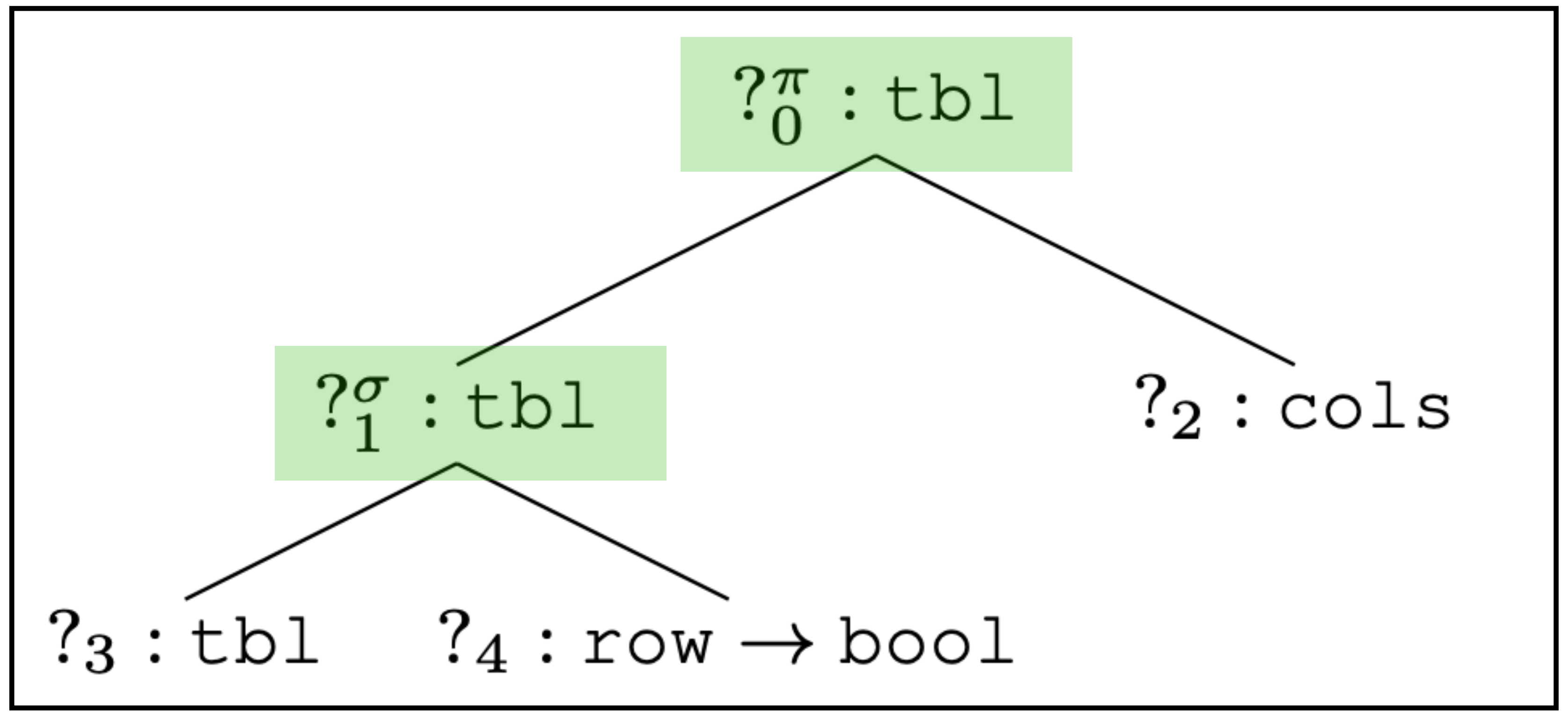
Important Concepts

- Hypothesis: "partial program"

$$
\begin{array}{lll}
\text{Term } t & := & \text{const} \mid y_i \mid \mathcal{X}(t_1, ..., t_n)\ (\mathcal{X} \in \Lambda_v) \\
\text{Qualifier } \mathcal{Q} & := & (x, \mathsf{T}) \mid \lambda y_1, \ldots y_n.\, t \\
\text{Hypothesis } \mathcal{H} & := & (?_i : \tau) \mid (?_i : \tau)@\mathcal{Q} \\
 & & \mid ?_i^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)\ \ (\mathcal{X} \in \Lambda_{\mathsf{T}})
\end{array}
$$

Figure 5. Context-free grammar for hypotheses

Non-leaf node (recursive case)



Important Concepts

- Hypothesis: "partial program"

Term t	$:=$	const$\mid y_i \mid \mathcal{X}(t_1, ..., t_n)\ (\mathcal{X} \in \Lambda_v)$
Qualifier $\mathcal{Q}$	$:=$	$(x, \mathsf{T}) \mid \lambda y_1, \ldots y_n. t$
Hypothesis $\mathcal{H}$	$:=$	$(?_i : \tau) \mid (?_i : \tau)@\mathcal{Q}$
		$\mid ?_i^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)\ (\mathcal{X} \in \Lambda_{\mathsf{T}})$

Figure 5. Context-free grammar for hypotheses

Table transformers

Functions that transform tables to tables

```
df1=filter(input, dest == "SEA")
df2=summarize(group_by(df1, origin), n = n())
df3=mutate(df2, prop = n / sum(n))
```

Important Concepts

- Hypothesis: “partial program”

Term t	:=	const$\mid y_i \mid \mathcal{X}(t_1, ..., t_n)$ $(\mathcal{X} \in \Lambda_v)$
Qualifier $\mathcal{Q}$	:=	$(x, \mathsf{T}) \mid \lambda y_1, \ldots y_n.\, t$
Hypothesis $\mathcal{H}$	:=	$(?_i : \tau) \mid (?_i : \tau)@\mathcal{Q}$
		$\mid ?_i^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)$ $(\mathcal{X} \in \Lambda_{\mathsf{T}})$

Figure 5. Context-free grammar for hypotheses

Value transformers

Functions that don’t transform tables; they transform values.
Constants are special value transformers.

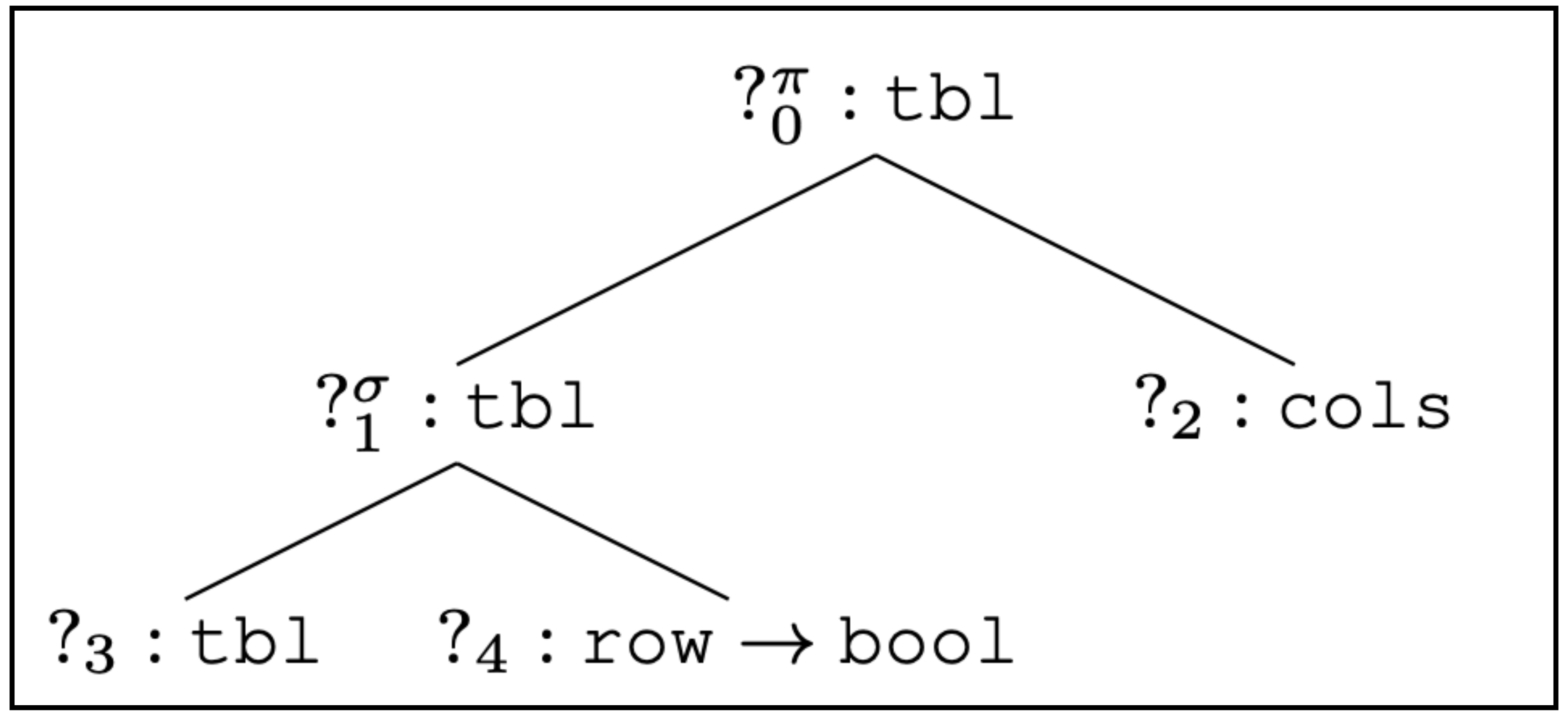
```
df1=filter(input, dest == "SEA")
df2=summarize(group_by(df1, origin), n = n())
df3=mutate(df2, prop = n / sum(n))
```

Important Concepts

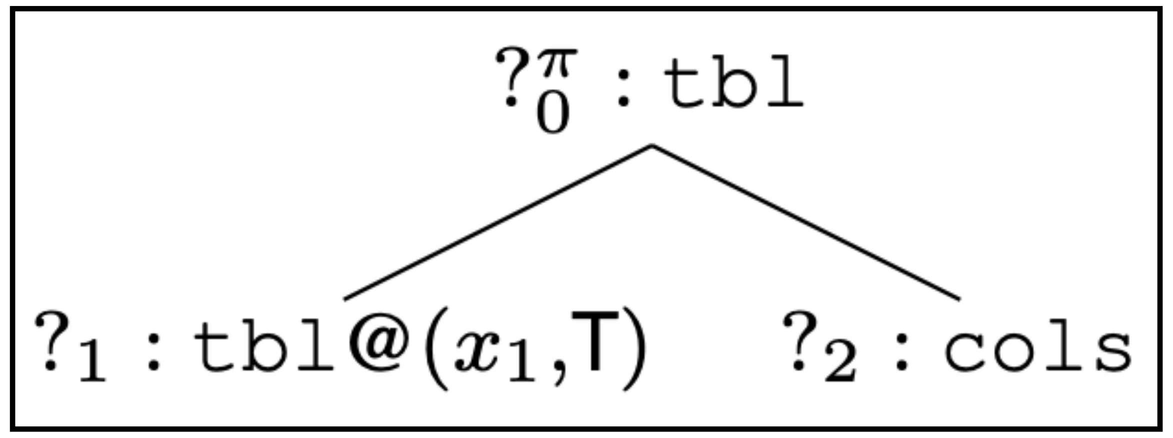
- Sketch: a special form of hypothesis, where all table-typed leaf nodes are concrete

Important Concepts

- Sketch: a special form of hypothesis, where all table-typed leaf nodes are concrete



Hypothesis, not sketch

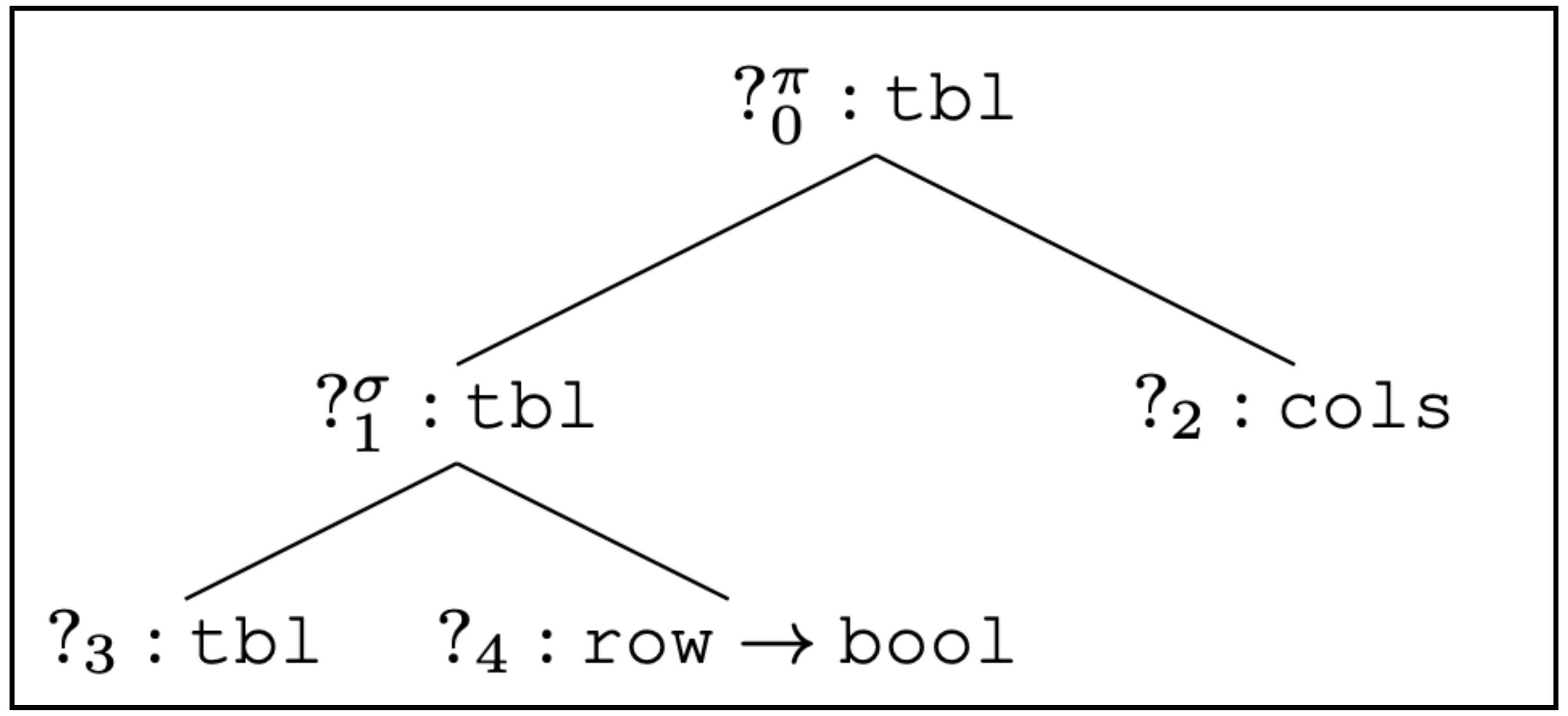


Hypothesis, and sketch

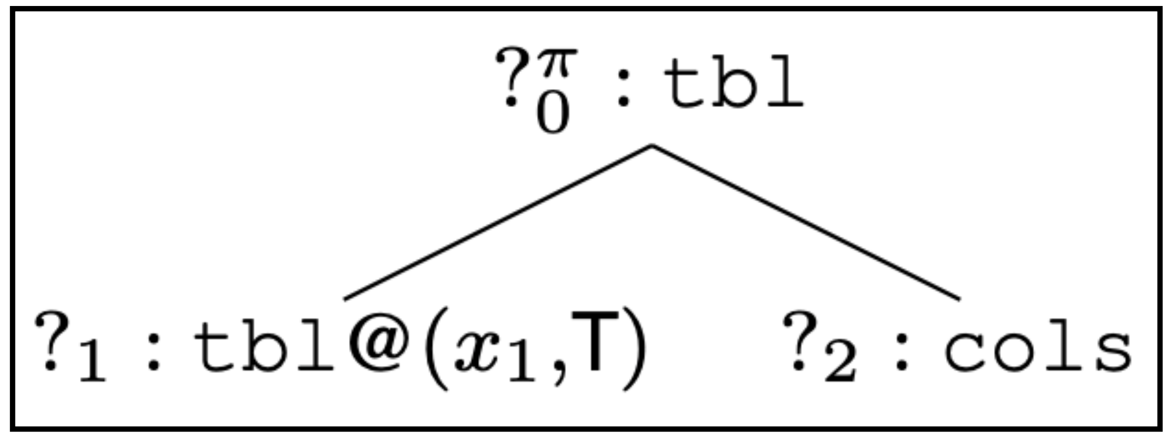
Important Concepts

- Sketch: a special form of hypothesis, where all table-typed leaf nodes are concrete

- Essentially, in sketch, all table-typed holes are concrete
- In other words, sketch represents a "smaller space" of concrete programs



Hypothesis, not sketch



Hypothesis, and sketch

Synthesis Algorithm

procedure SYNTHESIZE($\mathcal{E}, \Lambda$)

  input: Input-output example $\mathcal{E}$ and components Λ

  output: Synthesized program or $\perp$ if failure

  $W := \{?_0\text{:}\texttt{tbl}\}$ ▷ Init worklist

  while $W \neq \emptyset$ **do**

    choose $\mathcal{H} \in W$;

    $W := W \setminus \{\mathcal{H}\}$

    if DEDUCE($\mathcal{H}, \mathcal{E}$) $= \perp$ **then** ▷ Contradiction

      goto refine;

  ▷ No contradiction

    for $\mathcal{S} \in$ SKETCHES($\mathcal{H}, \mathcal{E}_{in}$) **do**

      $\mathcal{P}$:= FILLSKETCH($\mathcal{S}, \mathcal{E}$)

      for $p \in \mathcal{P}$ **do**

        if CHECK($p, \mathcal{E}$) **then return** p

    refine: ▷Hypothesis refinement

    for $\mathcal{X} \in \Lambda_{\mathsf{T}}$, $(?_i\text{: }\texttt{tbl}) \in$ LEAVES($\mathcal{H}$) **do**

      $\mathcal{H}' := \mathcal{H}[?_j^{\mathcal{X}}(?_j : \vec{\tau})/?_i]$

      $W := W \cup \mathcal{H}'$

  return $\perp$

Synthesis Algorithm

```
procedure SYNTHESIZE(ℰ, Λ)
    input: Input-output example ℰ and components Λ
    output: Synthesized program or ⊥ if failure
    W := {?₀:tbl}                                   ▷ Init worklist
    while W ≠ ∅ do
        choose ℋ ∈ W;
        W := W\{ℋ}
        if DEDUCE(ℋ, ℰ) = ⊥ then                    ▷ Contradiction
            goto refine;
                                                    ▷ No contradiction
        for 𝒮 ∈ SKETCHES(ℋ, ℰ_in) do
            𝒫 := FILLSKETCH(𝒮, ℰ)
            for p ∈ 𝒫 do
                if CHECK(p, ℰ) then return p

        refine:                                     ▷ Hypothesis refinement
        for 𝒳 ∈ Λ_T, (?_i: tbl) ∈ LEAVES(ℋ) do
            ℋ' := ℋ[?_j^𝒳(?_j : τ⃗)/?_i]
            W := W ∪ ℋ'
    return ⊥
```

Explain algorithm in terms of its input/output

Synthesis Algorithm

procedure SYNTHESIZE($\mathcal{E}, \Lambda$)
  input: Input-output example $\mathcal{E}$ and components Λ
  output: Synthesized program or $\bot$ if failure
  $W := \{?_0\mathtt{:tbl}\}$ ▷ Init worklist
  while $W \neq \emptyset$ **do**
    choose $\mathcal{H} \in W$;
    $W := W \backslash \{\mathcal{H}\}$
    if DEDUCE($\mathcal{H}, \mathcal{E}$) $= \bot$ **then** ▷ Contradiction
      goto refine;
  ▷ No contradiction
    for $\mathcal{S} \in$ SKETCHES($\mathcal{H}, \mathcal{E}_{in}$) **do**
      $\mathcal{P}$:= FILLSKETCH($\mathcal{S}, \mathcal{E}$)
      for $p \in \mathcal{P}$ **do**
        if CHECK($p, \mathcal{E}$) **then return** p
    refine: ▷Hypothesis refinement
    for $\mathcal{X} \in \Lambda_{\mathsf{T}}$, $(?_i\mathtt{: tbl}) \in$ LEAVES($\mathcal{H}$) **do**
      $\mathcal{H}' := \mathcal{H}[?_j^{\mathcal{X}}(?_j : \vec{\tau})/?_i]$
      $W := W \cup \mathcal{H}'$
  return $\bot$

Explain each step in an organized way

Synthesis Algorithm

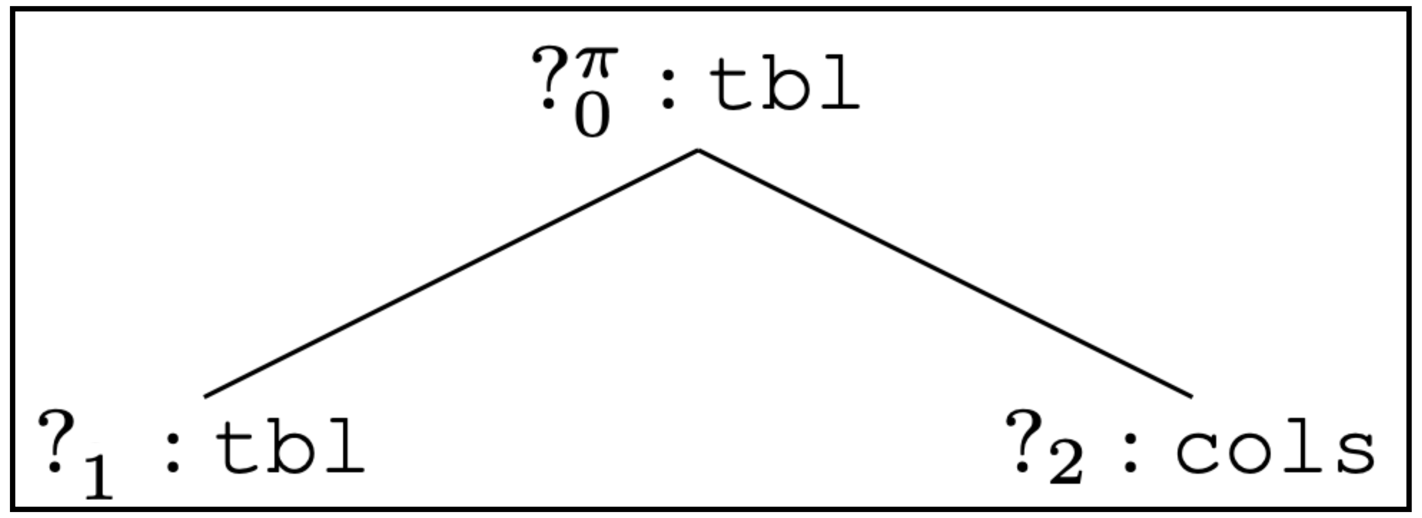
procedure $\textsc{Synthesize}(\mathcal{E}, \Lambda)$
    input: Input-output example $\mathcal{E}$ and components Λ
    output: Synthesized program or $\perp$ if failure
    $W := \{?_0{:}\texttt{tbl}\}$ ▷ Init worklist
    while $W \neq \emptyset$ **do**
        choose $\mathcal{H} \in W$;
        $W := W \backslash \{\mathcal{H}\}$
        if $\textsc{Deduce}(\mathcal{H}, \mathcal{E}) = \perp$ **then** ▷ Contradiction
            goto refine;
        ▷ No contradiction
        for $\mathcal{S} \in \textsc{Sketches}(\mathcal{H}, \mathcal{E}_{in})$ **do**
            $\mathcal{P} := \textsc{FillSketch}(\mathcal{S}, \mathcal{E})$
            for $p \in \mathcal{P}$ **do**
                if $\textsc{check}(p, \mathcal{E})$ **then return** p
        refine: ▷Hypothesis refinement
        for $\mathcal{X} \in \Lambda_{\mathsf{T}}$, $(?_i{:}\ \texttt{tbl}) \in \textsc{Leaves}(\mathcal{H})$ **do**
            $\mathcal{H}' := \mathcal{H}[?_j^{\mathcal{X}}(?_j : \vec{\tau})/?_i]$
            $W := W \cup \mathcal{H}'$
    return $\perp$

A worklist algorithm. Initialization.

Synthesis Algorithm

procedure $\textsc{Synthesize}(\mathcal{E}, \Lambda)$
    input: Input-output example $\mathcal{E}$ and components Λ
    output: Synthesized program or $\perp$ if failure
    $W := \{?_0\text{:}\texttt{tbl}\}$ ▷ Init worklist
    while $W \neq \emptyset$ **do**
        choose $\mathcal{H} \in W$;
        $W := W \backslash \{\mathcal{H}\}$
        if $\textsc{Deduce}(\mathcal{H}, \mathcal{E}) = \perp$ **then** ▷ Contradiction
            goto refine;
        ▷ No contradiction
        for $\mathcal{S} \in \textsc{Sketches}(\mathcal{H}, \mathcal{E}_{in})$ **do**
            $\mathcal{P} := \textsc{FillSketch}(\mathcal{S}, \mathcal{E})$
            for $p \in \mathcal{P}$ **do**
                if $\textsc{check}(p, \mathcal{E})$ **then return** p
        refine: ▷Hypothesis refinement
        for $\mathcal{X} \in \Lambda_{\mathsf{T}}, (?_i\text{: }\texttt{tbl}) \in \textsc{Leaves}(\mathcal{H})$ **do**
            $\mathcal{H}' := \mathcal{H}[?_j^{\mathcal{X}}(?_j : \vec{\tau})/?_i]$
            $W := W \cup \mathcal{H}'$
    return $\perp$

Remove one hypothesis from worklist



Synthesis Algorithm

procedure $\textsc{Synthesize}(\mathcal{E}, \Lambda)$

  input: Input-output example $\mathcal{E}$ and components Λ

  output: Synthesized program or $\bot$ if failure

  $W := \{?_0{:}\texttt{tbl}\}$ ▷ Init worklist

  while $W \neq \emptyset$ **do**

    choose $\mathcal{H} \in W$;

    $W := W \backslash \{\mathcal{H}\}$

    if $\textsc{Deduce}(\mathcal{H}, \mathcal{E}) = \bot$ **then** ▷ Contradiction

      goto refine;

  ▷ No contradiction

    for $\mathcal{S} \in \textsc{Sketches}(\mathcal{H}, \mathcal{E}_{in})$ **do**

      $\mathcal{P} := \textsc{FillSketch}(\mathcal{S}, \mathcal{E})$

      for $p \in \mathcal{P}$ **do**

        if $\textsc{check}(p, \mathcal{E})$ **then return** p

    refine: ▷Hypothesis refinement

    for $\mathcal{X} \in \Lambda_{\mathsf{T}}, (?_i{:}\ \texttt{tbl}) \in \textsc{Leaves}(\mathcal{H})$ **do**

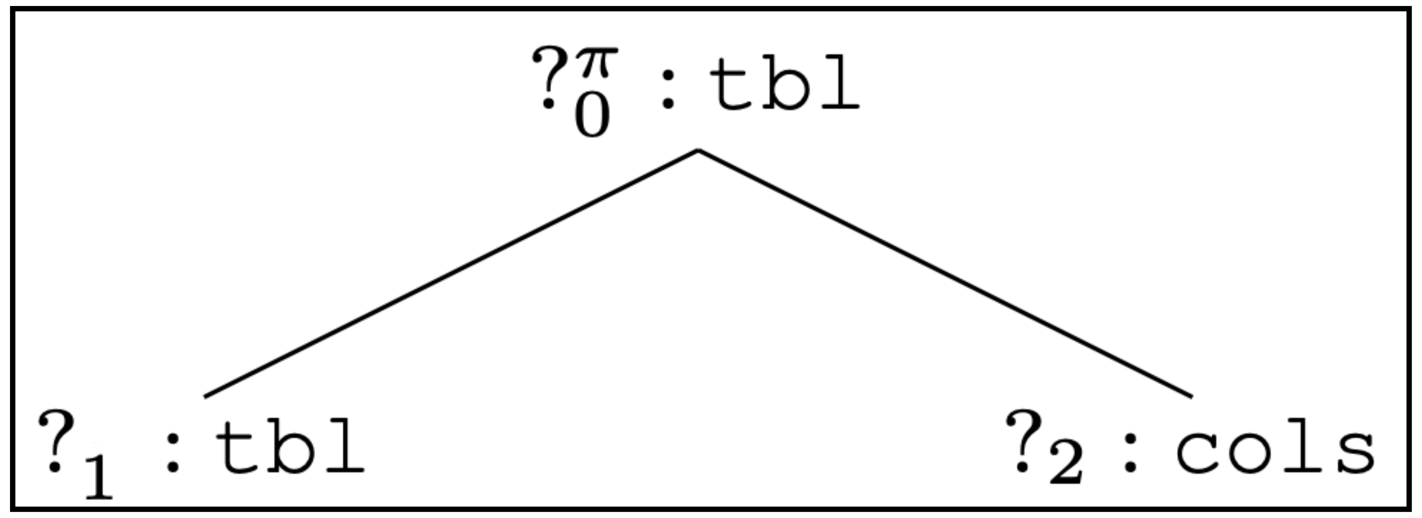
      $\mathcal{H}' := \mathcal{H}[?_j^{\mathcal{X}}(?_j : \vec{\tau})/?_i]$

      $W := W \cup \mathcal{H}'$

  return $\bot$

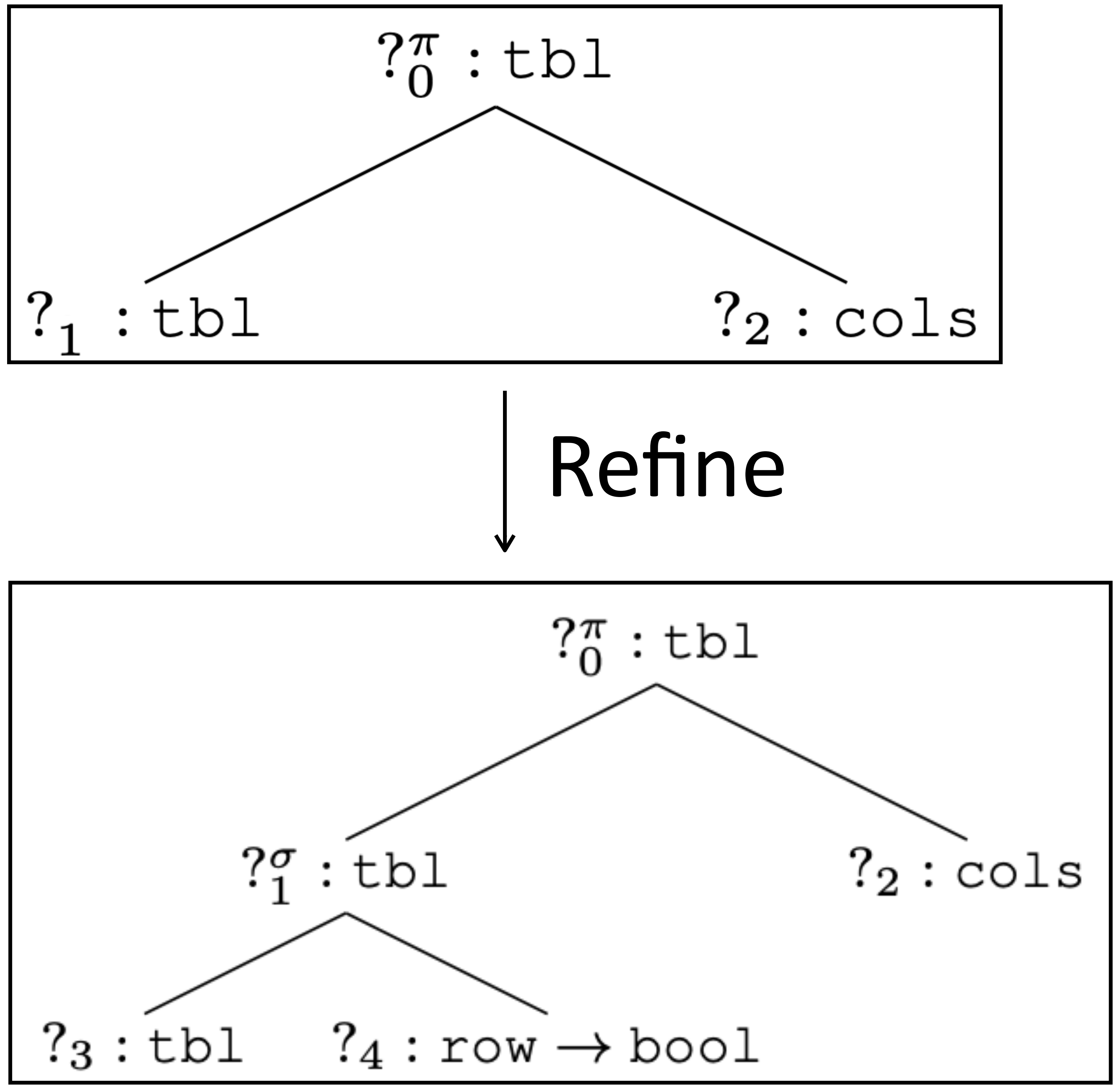
Prune using deduction (discuss later)

In particular, "Deduce" procedure checks whether we can prune **sketches** corresponding to the hypothesis (but not the entire hypothesis)



Synthesis Algorithm

procedure $\textsc{Synthesize}(\mathcal{E}, \Lambda)$
    input: Input-output example $\mathcal{E}$ and components Λ
    output: Synthesized program or $\bot$ if failure
    $W := \{?_0\texttt{:tbl}\}$ ▷ Init worklist
    while $W \neq \emptyset$ **do**
        choose $\mathcal{H} \in W$;
        $W := W \backslash \{\mathcal{H}\}$
        if $\textsc{Deduce}(\mathcal{H}, \mathcal{E}) = \bot$ **then**   ▷ Contradiction
            goto refine;
                                ▷ No contradiction
        for $\mathcal{S} \in \textsc{Sketches}(\mathcal{H}, \mathcal{E}_{in})$ **do**
            $\mathcal{P} := \textsc{FillSketch}(\mathcal{S}, \mathcal{E})$
            for $p \in \mathcal{P}$ **do**
                if $\textsc{Check}(p, \mathcal{E})$ **then return** p
        refine: ▷Hypothesis refinement
        for $\mathcal{X} \in \Lambda_\mathsf{T}, (?_i\texttt{: tbl}) \in \textsc{Leaves}(\mathcal{H})$ **do**
            $\mathcal{H}' := \mathcal{H}[?_j^{\mathcal{X}}(?_j : \vec{\tau})/?_i]$
            $W := W \cup \mathcal{H}'$
    return $\bot$

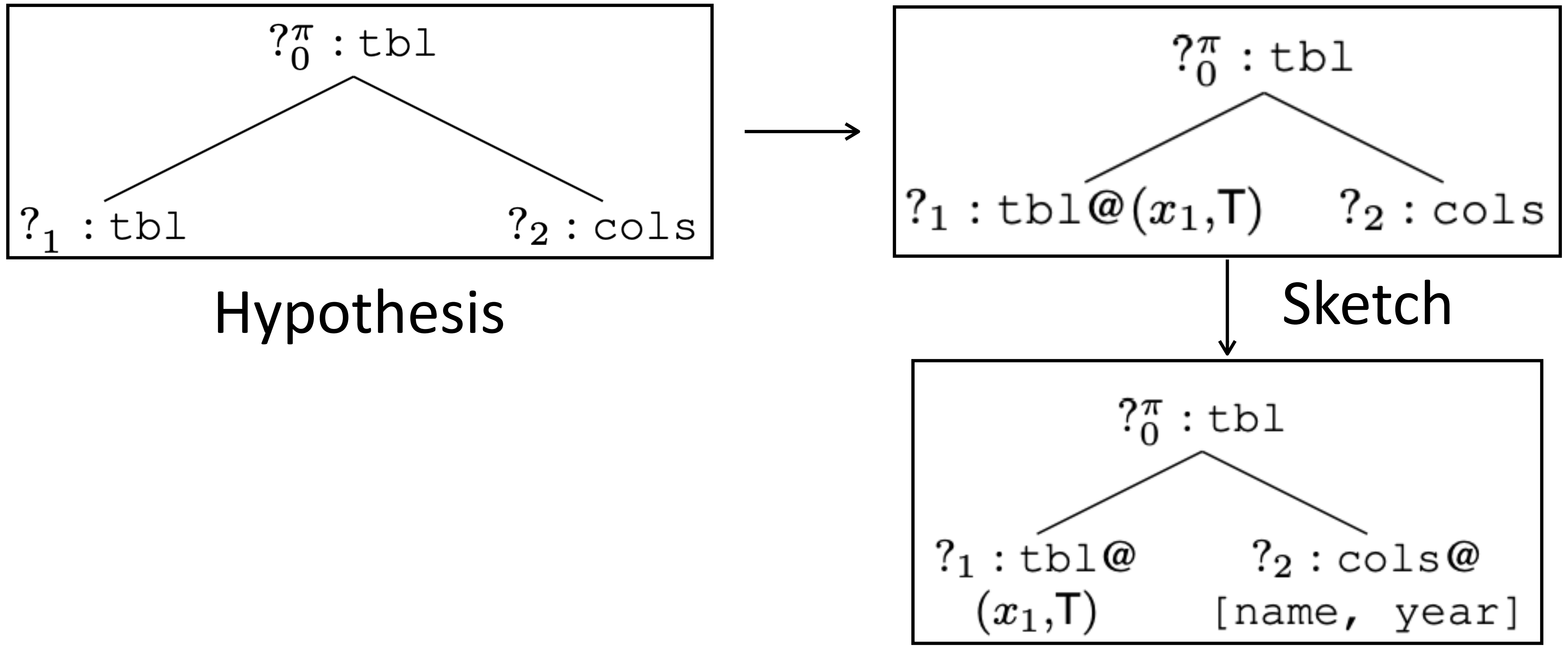


If can prune ("contradiction" means "can prune")

In particular, replace each table-typed leaf node in H with **table transformation operators** (not variables) in Λ_T

Synthesis Algorithm

procedure $\textsc{Synthesize}(\mathcal{E}, \Lambda)$
  input: Input-output example $\mathcal{E}$ and components Λ
  output: Synthesized program or $\bot$ if failure
  $W := \{?_0\text{:}\texttt{tbl}\}$ ▷ Init worklist
  while $W \neq \emptyset$ **do**
    choose $\mathcal{H} \in W$;
    $W := W \backslash \{\mathcal{H}\}$
    if $\textsc{Deduce}(\mathcal{H}, \mathcal{E}) = \bot$ **then** ▷ Contradiction
      goto refine;
    ▷ No contradiction
    for $\mathcal{S} \in \textsc{Sketches}(\mathcal{H}, \mathcal{E}_{in})$ **do**
      $\mathcal{P} := \textsc{FillSketch}(\mathcal{S}, \mathcal{E})$
      for $p \in \mathcal{P}$ **do**
        if $\textsc{Check}(p, \mathcal{E})$ **then return** p
    refine: ▷Hypothesis refinement
    for $\mathcal{X} \in \Lambda_{\mathsf{T}}, (?_i\text{: }\texttt{tbl}) \in \textsc{Leaves}(\mathcal{H})$ **do**
      $\mathcal{H}' := \mathcal{H}[?_j^{\mathcal{X}}(?_j : \vec{\tau})/?_i]$
      $W := W \cup \mathcal{H}'$
  return $\bot$



Hypothesis

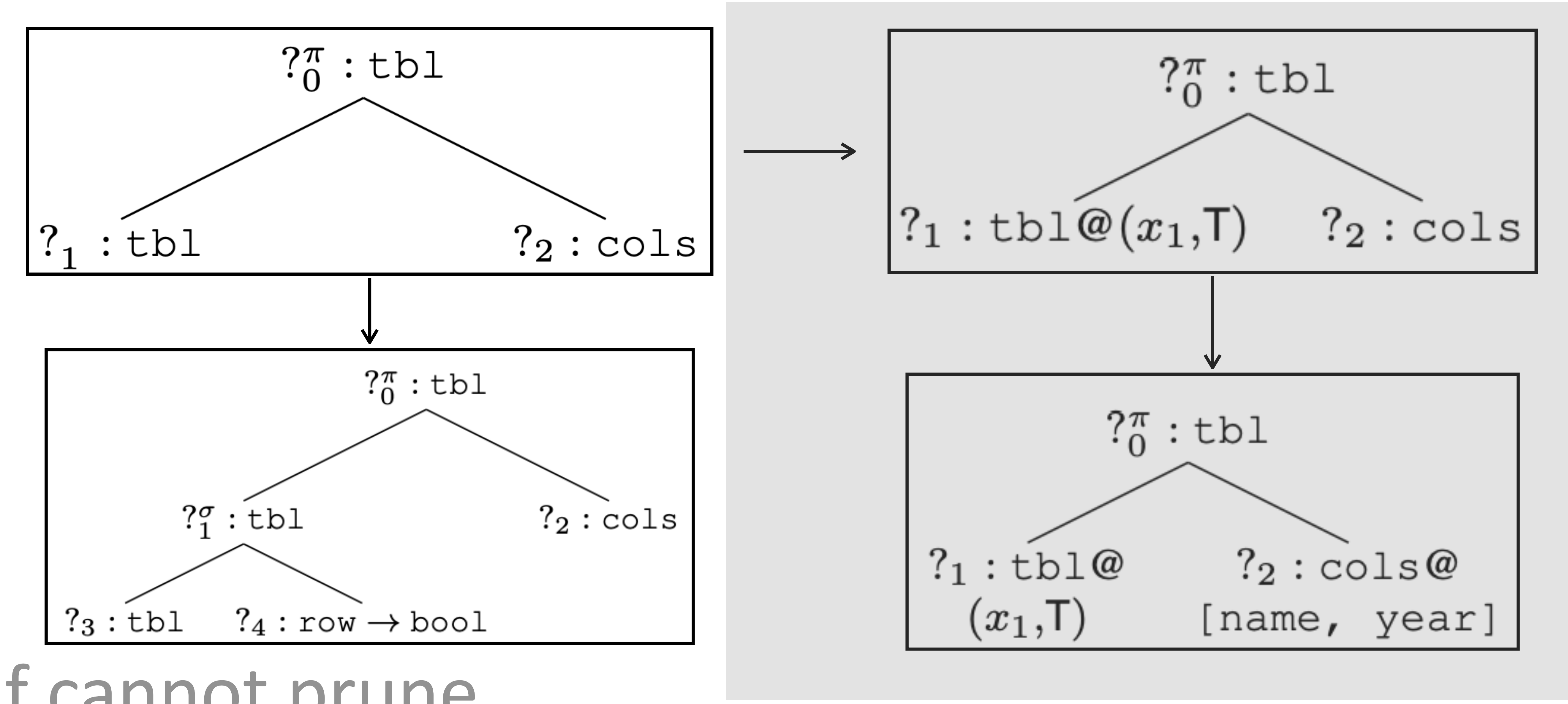
Concrete Program

If cannot prune

In particular, convert H to a set of sketches, fill each sketch, check each concrete program against spec

Synthesis Algorithm

```
procedure SYNTHESIZE(ℰ, Λ)
    input: Input-output example ℰ and components Λ
    output: Synthesized program or ⊥ if failure
    W := {?0:tbl}                                  ▷ Init worklist
    while W ≠ ∅ do
        choose ℋ ∈ W;
        W := W\{ℋ}
        if DEDUCE(ℋ, ℰ) = ⊥ then                  ▷ Contradiction
            goto refine;
                                                   ▷ No contradiction
        for 𝒮 ∈ SKETCHES(ℋ, ℰ_in) do
            𝒫 := FILLSKETCH(𝒮, ℰ)
            for p ∈ 𝒫 do
                if CHECK(p, ℰ) then return p
        refine:                                    ▷Hypothesis refinement
        for 𝒳 ∈ Λ_T, (?i: tbl) ∈ LEAVES(ℋ) do
            ℋ' := ℋ[?j^𝒳(?j : τ⃗)/?i]
            W := W ∪ ℋ'
    return ⊥
```



If cannot prune

In particular, convert H to a set of sketches, fill each sketch, check each concrete program against spec

Still need to refine

Deduction

procedure SYNTHESIZE($\mathcal{E}, \Lambda$)
  input: Input-output example $\mathcal{E}$ and components Λ
  output: Synthesized program or $\bot$ if failure
  $W := \{?_0\text{:}\texttt{tbl}\}$ ▷ Init worklist
  while $W \neq \emptyset$ **do**
    choose $\mathcal{H} \in W$;
    $W := W \backslash \{\mathcal{H}\}$
    if DEDUCE($\mathcal{H}, \mathcal{E}$) $= \bot$ **then** ▷ Contradiction
      goto refine;
    ▷ No contradiction
    for $\mathcal{S} \in$ SKETCHES($\mathcal{H}, \mathcal{E}_{in}$) **do**
      $\mathcal{P}$:= FILLSKETCH($\mathcal{S}, \mathcal{E}$)
      for $p \in \mathcal{P}$ **do**
        if CHECK($p, \mathcal{E}$) **then return** p
    refine: ▷Hypothesis refinement
    for $\mathcal{X} \in \Lambda_{\mathsf{T}}$, $(?_i\text{: }\texttt{tbl}) \in$ LEAVES($\mathcal{H}$) **do**
      $\mathcal{H}' := \mathcal{H}[?_j^{\mathcal{X}}(?_j : \vec{\tau})/?_i]$
      $W := W \cup \mathcal{H}'$
  return $\bot$

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)

  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$

  output: $\perp$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise

  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$

  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \leq i \leq |\mathcal{E}_{in}|} (?_j = x_i)$

  $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$

  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$

  return SAT(ψ)

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)

  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$

  output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise

  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$

  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$

  $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$

  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \land \varphi_{in} \land \varphi_{out} \land \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \land \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$

  return SAT(ψ)

Explain algorithm in terms of its input/output

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

Algorithm 2 SMT-based Deduction Algorithm

procedure $\textsc{Deduce}(\mathcal{H}, \mathcal{E})$
  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
  output: $\perp$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \textsc{Leaves}(\mathcal{H})\}$
  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \leq i \leq |\mathcal{E}_{in}|} (?_j = x_i)$
  $\varphi_{out} := (y = \textsc{RootVar}(\mathcal{H}))$
  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
  return $\textsc{Sat}(\psi)$

Explain each step in an organized way

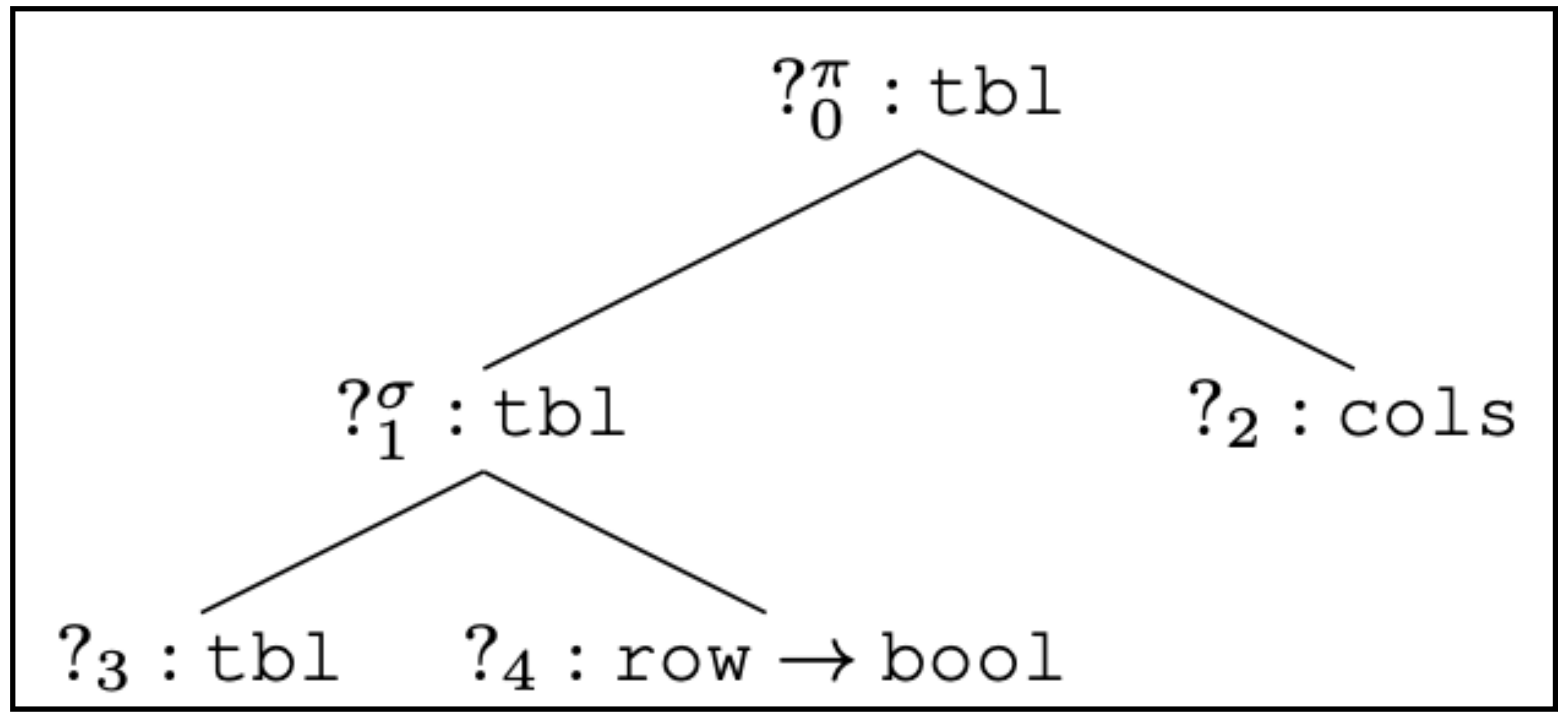
Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
  output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$
  $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \land \varphi_{in} \land \varphi_{out} \land \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \land \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
  return SAT(ψ)

S is set of table-typed leaf nodes in H



Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)

input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$

output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise

$\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$

$\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$

$\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$

$$\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$$

return SAT(ψ)

x_i is the ith table in input example

E_{in} is all input tables in input example

φ_{in} essentially encodes all possible sketches

(recall: table-typed leaf nodes in sketch must be concrete)

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

---

Algorithm 2 SMT-based Deduction Algorithm

---

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)

  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$

  output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise

  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$

  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$

  $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$

  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \land \varphi_{in} \land \varphi_{out} \land \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \land \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$

  return SAT(ψ)

---

y is the output of entire program

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
  output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$
  $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \land \varphi_{in} \land \varphi_{out} \land \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \land \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
  return SAT(ψ)

Compose constraints to form constraint of entire program

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

Algorithm 2 SMT-based Deduction Algorithm

procedure $\text{DEDUCE}(\mathcal{H}, \mathcal{E})$

input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$

output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise

$\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$

$\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \leq i \leq |\mathcal{E}_{in}|} (?_j = x_i)$

$\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$

$\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$

return $\text{SAT}(\psi)$

Constraint for table-typed leaf nodes

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
  output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \leq i \leq |\mathcal{E}_{in}|} (?_j = x_i)$
  $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \land \varphi_{in} \land \varphi_{out} \land \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \land \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
  return SAT(ψ)

Constraint for output of entire program

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)

input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$

output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise

$\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$

$\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$

$\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$

$\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \land \varphi_{in} \land \varphi_{out} \land \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \land \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$

return SAT(ψ)

Input-output example

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)

input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$

output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise

$\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$

$\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \leq i \leq |\mathcal{E}_{in}|} (?_j = x_i)$

$\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$

$\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \land \varphi_{in} \land \varphi_{out} \land \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \land \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$

return SAT(ψ)

Constraint for hypothesis

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

Algorithm 2 SMT-based Deduction Algorithm

```
procedure DEDUCE(𝓗, 𝓔)
    input: Hypothesis 𝓗, input-output example 𝓔
    output: ⊥ if cannot be unified with 𝓔; ⊤ otherwise
    𝓢 := {?_j | ?_j : tbl ∈ LEAVES(𝓗)}
    φ_in := ⋀_{?_j ∈ 𝓢} ⋁_{1 ≤ i ≤ |𝓔_in|} (?_j = x_i)
    φ_out := (y = ROOTVAR(𝓗))
    ψ := ( Φ(𝓗) ∧ φ_in ∧ φ_out ∧ ⋀_{T_i ∈ 𝓔_in} (α(T_i)[x_i/x]) ∧ α(T_out)[y/x] )
    return SAT(ψ)
```

$$
\begin{aligned}
\Phi(\mathcal{H}_i) &= \alpha(\llbracket \mathcal{H}_i \rrbracket_\partial)[?_i/x] \text{ if } \neg\textsc{Partial}(\llbracket \mathcal{H}_i \rrbracket_\partial)\\
\Phi(\mathcal{H}_i) &= \top \qquad \text{else if } \textsc{IsLeaf}(\mathcal{H}_i)\\
\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) &= \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \land \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}]
\end{aligned}
$$

Constraint for hypothesis

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

$$\Phi(\mathcal{H}_i) = \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] \text{ if } \neg\text{PARTIAL}([\![\mathcal{H}_i]\!]_\partial)$$
$$\Phi(\mathcal{H}_i) = \top \quad \text{else if IsLEAF}(\mathcal{H}_i)$$
$$\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) = \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}]$$

Leaf nodes (base case)

Algorithm 2 SMT-based Deduction Algorithm

1: **procedure** DEDUCE($\mathcal{H}, \mathcal{E}$)
2: **input:** Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
3: **output:** $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
4: $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
5: $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$
6: $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
7: $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
8: **return** SAT(ψ)

Constraint for hypothesis

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

Algorithm 2 SMT-based Deduction Algorithm

```
procedure DEDUCE(H, E)
    input: Hypothesis H, input-output example E
    output: ⊥ if cannot be unified with E; ⊤ otherwise
    S := {?_j | ?_j : tbl ∈ LEAVES(H)}
    φ_in := ⋀_{?_j∈S} ⋁_{1≤i≤|E_in|} (?_j = x_i)
    φ_out := (y = ROOTVAR(H))
    ψ := ( Φ(H) ∧ φ_in ∧ φ_out ∧ ⋀_{T_i∈E_in} (α(T_i)[x_i/x]) ∧ α(T_out)[y/x] )
    return SAT(ψ)
```

$$\Phi(\mathcal{H}_i) = \alpha(\llbracket \mathcal{H}_i \rrbracket_\partial)[?_i/x] \text{ if } \neg\text{PARTIAL}(\llbracket \mathcal{H}_i \rrbracket_\partial)$$

$$\Phi(\mathcal{H}_i) = \top \quad \text{else if } \text{ISLEAF}(\mathcal{H}_i)$$

$$\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) = \bigwedge_{1\le i\le n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}]$$

Concrete program (base case)

Execute, produce a concrete output table, abstract output table using abstraction function α

Constraint for hypothesis

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

$$
\begin{array}{rll}
\Phi(\mathcal{H}_i) & = \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] & \text{if } \neg\textsc{Partial}([\![\mathcal{H}_i]\!]_\partial) \\
\Phi(\mathcal{H}_i) & = \top & \text{else if } \textsc{IsLeaf}(\mathcal{H}_i) \\
\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) & = \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}] &
\end{array}
$$

Make use of "partial evaluation"

Algorithm 2 SMT-based Deduction Algorithm

procedure $\textsc{Deduce}(\mathcal{H}, \mathcal{E})$

input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$

output: $\perp$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise

$\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \textsc{Leaves}(\mathcal{H})\}$

$\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$

$\varphi_{out} := (y = \textsc{RootVar}(\mathcal{H}))$

$\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$

return $\textsc{SAT}(\psi)$

Constraint for hypothesis

Deduction

$$\begin{aligned}
\Phi(\mathcal{H}_i) &= \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] \text{ if } \neg\textsc{Partial}([\![\mathcal{H}_i]\!]_\partial) \\
\Phi(\mathcal{H}_i) &= \top \quad \text{else if } \textsc{IsLeaf}(\mathcal{H}_i) \\
\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) &= \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_\chi[?_0/y, \vec{?_i}/\vec{x_i}]
\end{aligned}$$

Partial evaluation of H_i

Idea: if some sub-program in H_i is already concrete, evaluate it to a concrete table

Deduction

$$
\begin{array}{lll}
\Phi(\mathcal{H}_i) & = \alpha(\llbracket \mathcal{H}_i \rrbracket_\partial)[?_i/x] \text{ if } \neg\textsc{Partial}(\llbracket \mathcal{H}_i \rrbracket_\partial) \\
\Phi(\mathcal{H}_i) & = \top \qquad \text{else if } \textsc{IsLeaf}(\mathcal{H}_i) \\
\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) & = \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}]
\end{array}
$$

Partial evaluation of H_i

Idea: if some sub-program in H_i is already concrete, evaluate it to a concrete table

$$
\llbracket (?_i : \tau) \rrbracket_\partial = ?_i \qquad \llbracket (?_i : \tau)@(x, \mathsf{T}) \rrbracket_\partial = \mathsf{T} \qquad \llbracket (?_i : \tau)@t \rrbracket_\partial = t
$$

$$
\llbracket ?_i^{\mathcal{X}}(\mathcal{H}_1, \ldots, \mathcal{H}_n) \rrbracket_\partial = \begin{cases} \mathcal{X}(\llbracket \mathcal{H}_1 \rrbracket_\partial, \ldots, \llbracket \mathcal{H}_n \rrbracket_\partial) & \text{if } \exists i \in [1, n].\ \textsc{Partial}(\llbracket \mathcal{H}_i \rrbracket_\partial) \\ \llbracket \mathcal{X}(\llbracket \mathcal{H}_1 \rrbracket_\partial, \ldots, \llbracket \mathcal{H}_n \rrbracket_\partial) \rrbracket & \text{otherwise} \end{cases}
$$

Deduction

$$\begin{aligned}
\Phi(\mathcal{H}_i) &= \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] \text{ if } \neg\textsc{Partial}([\![\mathcal{H}_i]\!]_\partial)\\
\Phi(\mathcal{H}_i) &= \top \quad\quad \text{else if } \textsc{IsLeaf}(\mathcal{H}_i)\\
\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) &= \bigwedge_{1\le i\le n} \Phi(\mathcal{H}_i) \land \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}]
\end{aligned}$$

Partial evaluation of H_i

Idea: if some sub-program in H_i is already concrete, evaluate it to a concrete table

If can be concrete, be concrete (base case)

$$[\![(?_i : \tau)]\!]_\partial = ?_i \qquad [\![(?_i : \tau)@(x, \mathsf{T})]\!]_\partial = \mathsf{T} \qquad [\![(?_i : \tau)@t]\!]_\partial = t$$

$$[\![?_i^{\mathcal{X}}(\mathcal{H}_1, \ldots, \mathcal{H}_n)]\!]_\partial = \begin{cases} \mathcal{X}([\![\mathcal{H}_1]\!]_\partial, \ldots, [\![\mathcal{H}_n]\!]_\partial) & \text{if } \exists i \in [1, n].\ \textsc{Partial}([\![\mathcal{H}_i]\!]_\partial)\\ [\![\mathcal{X}([\![\mathcal{H}_1]\!]_\partial, \ldots, [\![\mathcal{H}_n]\!]_\partial)]\!] & \text{otherwise} \end{cases}$$

Deduction

$$\begin{aligned}
\Phi(\mathcal{H}_i) &= \alpha(\llbracket\mathcal{H}_i\rrbracket_\partial)[?_i/x] \text{ if } \neg\textsc{Partial}(\llbracket\mathcal{H}_i\rrbracket_\partial) \\
\Phi(\mathcal{H}_i) &= \top \quad \text{else if } \textsc{IsLeaf}(\mathcal{H}_i) \\
\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) &= \bigwedge_{1\leq i\leq n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}]
\end{aligned}$$

Partial evaluation of H_i

Idea: if some sub-program in H_i is already concrete, evaluate it to a concrete table

If cannot be concrete, keep holes (base case)

$$\llbracket(?_i : \tau)\rrbracket_\partial = ?_i \qquad \llbracket(?_i : \tau)@(x, \mathsf{T})\rrbracket_\partial = \mathsf{T} \qquad \llbracket(?_i : \tau)@t\rrbracket_\partial = t$$

$$\llbracket ?_i^{\mathcal{X}}(\mathcal{H}_1, \ldots, \mathcal{H}_n)\rrbracket_\partial = \begin{cases} \mathcal{X}(\llbracket\mathcal{H}_1\rrbracket_\partial, \ldots, \llbracket\mathcal{H}_n\rrbracket_\partial) & \text{if } \exists i \in [1, n].\ \textsc{Partial}(\llbracket\mathcal{H}_i\rrbracket_\partial) \\ \llbracket\mathcal{X}(\llbracket\mathcal{H}_1\rrbracket_\partial, \ldots, \llbracket\mathcal{H}_n\rrbracket_\partial)\rrbracket & \text{otherwise} \end{cases}$$

Deduction

$$\Phi(\mathcal{H}_i) = \alpha(\llbracket \mathcal{H}_i \rrbracket_\partial)[?_i/x] \text{ if } \neg\textsc{Partial}(\llbracket \mathcal{H}_i \rrbracket_\partial)$$
$$\Phi(\mathcal{H}_i) = \top \quad \text{else if } \textsc{IsLeaf}(\mathcal{H}_i)$$
$$\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) = \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?}_i/\vec{x}_i]$$

Partial evaluation of H_i

Idea: if some sub-program in H_i is already concrete, evaluate it to a concrete table

$$\llbracket (?_i : \tau) \rrbracket_\partial = ?_i \qquad \llbracket (?_i : \tau)@(x, \mathsf{T}) \rrbracket_\partial = \mathsf{T} \qquad \llbracket (?_i : \tau)@t \rrbracket_\partial = t$$

$$\llbracket ?_i^{\mathcal{X}}(\mathcal{H}_1, \ldots, \mathcal{H}_n) \rrbracket_\partial = \begin{cases} \mathcal{X}(\llbracket \mathcal{H}_1 \rrbracket_\partial, \ldots, \llbracket \mathcal{H}_n \rrbracket_\partial) & \text{if } \exists i \in [1, n].\ \textsc{Partial}(\llbracket \mathcal{H}_i \rrbracket_\partial) \\ \llbracket \mathcal{X}(\llbracket \mathcal{H}_1 \rrbracket_\partial, \ldots, \llbracket \mathcal{H}_n \rrbracket_\partial) \rrbracket & \text{otherwise} \end{cases}$$

Recursive case

Deduction

- Given hypothesis H, generate SMT formula that corresponds to sketches of H, and check against example

Algorithm 2 SMT-based Deduction Algorithm

1: **procedure** DEDUCE($\mathcal{H}, \mathcal{E}$)
2: **input:** Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
3: **output:** $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
4: $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
5: $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \leq i \leq |\mathcal{E}_{in}|} (?_j = x_i)$
6: $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
7: $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \land \varphi_{in} \land \varphi_{out} \land \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \land \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
8: **return** SAT(ψ)

$$\Phi(\mathcal{H}_i) = \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] \text{ if } \neg\text{PARTIAL}([\![\mathcal{H}_i]\!]_\partial)$$
$$\Phi(\mathcal{H}_i) = \top \quad \text{else if } \text{ISLEAF}(\mathcal{H}_i)$$
$$\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) = \bigwedge_{1 \leq i \leq n} \Phi(\mathcal{H}_i) \land \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}]$$

Subtree (recursive case)

Use specification for operator, need renaming

Constraint for hypothesis

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
    input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
    output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
    $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
    $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \leq i \leq |\mathcal{E}_{in}|} (?_j = x_i)$
    $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
    $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
    return SAT(ψ)

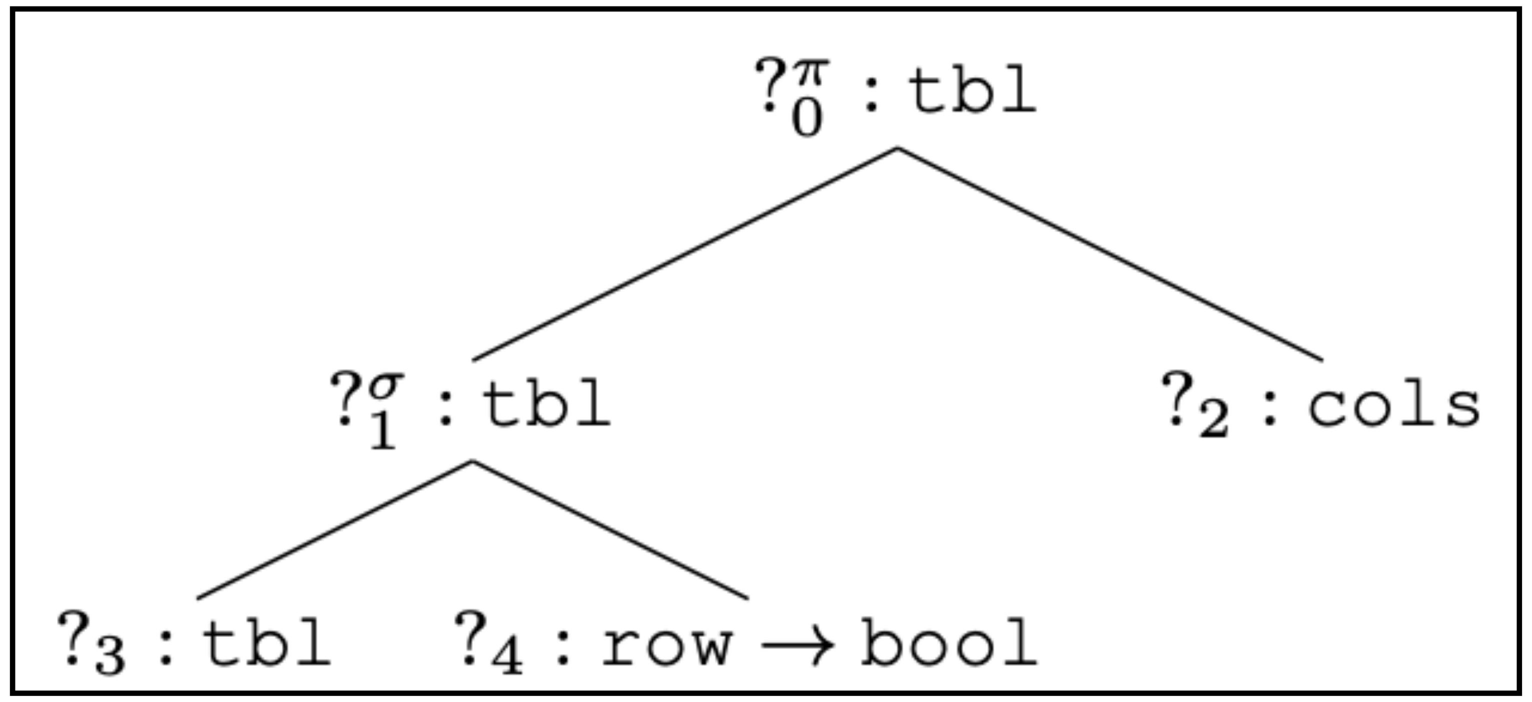
$$
\begin{aligned}
\Phi(\mathcal{H}_i) &= \alpha(\llbracket \mathcal{H}_i \rrbracket_\partial)[?_i/x] \text{ if } \neg\text{PARTIAL}(\llbracket \mathcal{H}_i \rrbracket_\partial) \\
\Phi(\mathcal{H}_i) &= \top \qquad \text{else if ISLEAF}(\mathcal{H}_i) \\
\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, \ldots, \mathcal{H}_n)) &= \bigwedge_{1 \leq i \leq n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}]
\end{aligned}
$$

id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

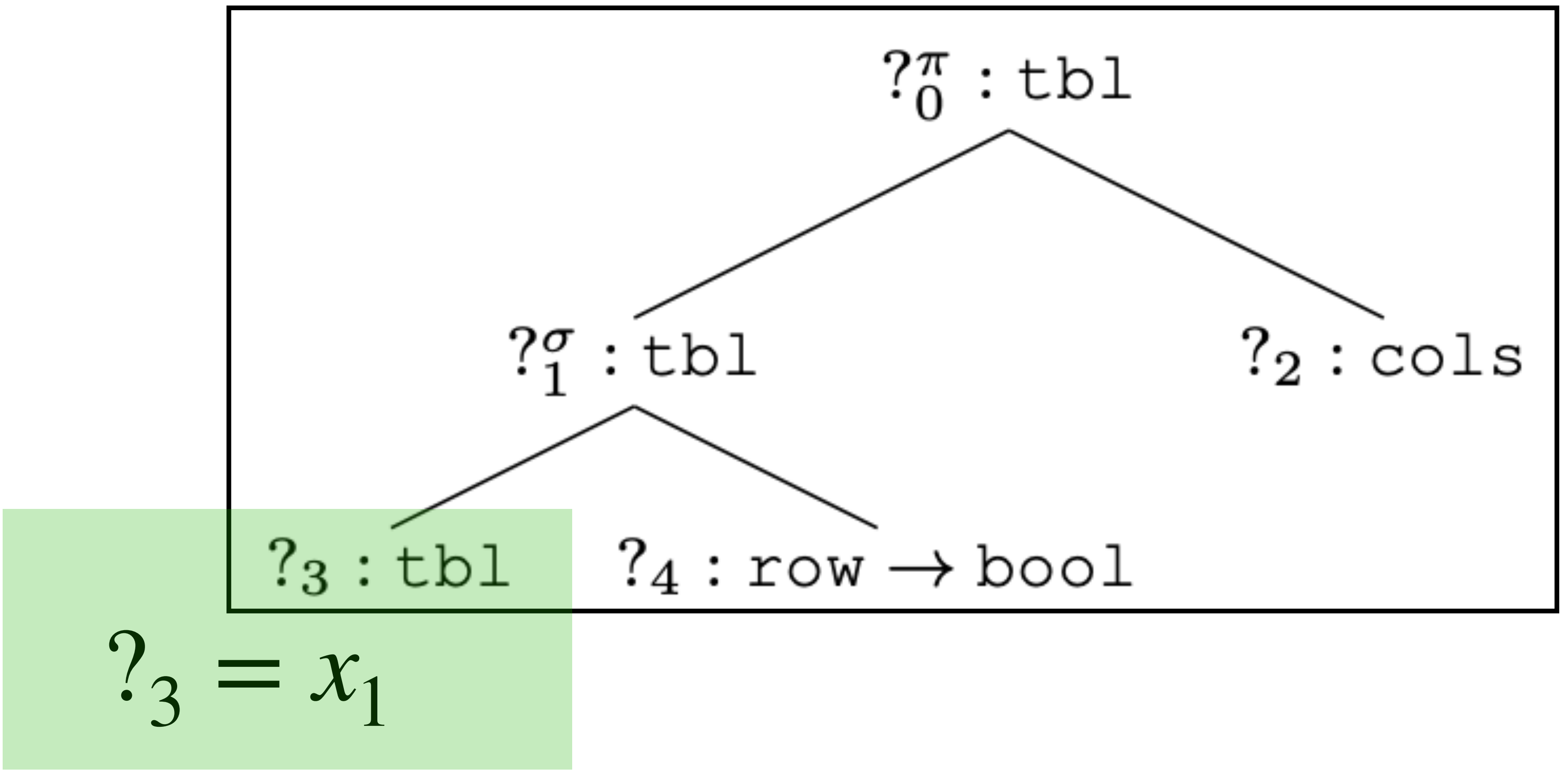


Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
    input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
    output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
    $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
    $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$
    $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
    $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
    return SAT(ψ)

$$\begin{aligned} \Phi(\mathcal{H}_i) &= \alpha(⟦\mathcal{H}_i⟧_\partial)[?_i/x] \text{ if } \neg\text{PARTIAL}(⟦\mathcal{H}_i⟧_\partial) \\ \Phi(\mathcal{H}_i) &= \top \qquad \text{else if ISLEAF}(\mathcal{H}_i) \\ \Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) &= \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}] \end{aligned}$$



id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
- **input:** Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
- **output:** $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
- $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \textsc{leaves}(\mathcal{H})\}$
- $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$
- $\varphi_{out} := (y = \textsc{RootVar}(\mathcal{H}))$
- $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
- **return** SAT(ψ)

$$\Phi(\mathcal{H}_i) = \alpha(\llbracket \mathcal{H}_i \rrbracket_\partial)[?_i/x] \text{ if } \neg\textsc{Partial}(\llbracket \mathcal{H}_i \rrbracket_\partial)$$
$$\Phi(\mathcal{H}_i) = \top \quad \text{else if } \textsc{IsLeaf}(\mathcal{H}_i)$$
$$\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) = \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}]$$

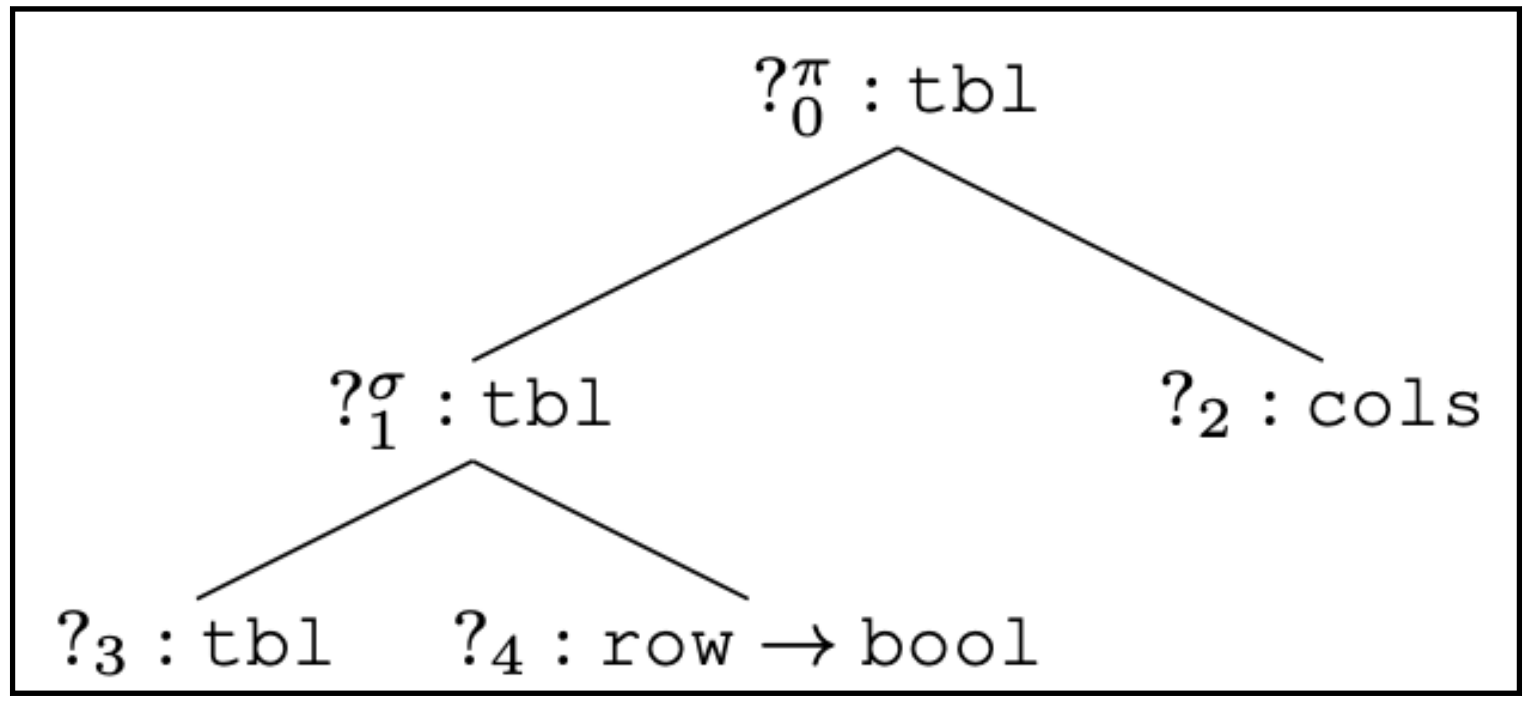
id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

$y = ?_0$



$?_3 = x_1$

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)

input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
output: $\perp$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise

$\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$

$\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$

$\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$

$\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$

return SAT(ψ)

$$\begin{aligned} \Phi(\mathcal{H}_i) &= \alpha(\llbracket \mathcal{H}_i \rrbracket_\partial)[?_i/x] \text{ if } \neg\text{PARTIAL}(\llbracket \mathcal{H}_i \rrbracket_\partial) \\ \Phi(\mathcal{H}_i) &= \top \quad \text{else if ISLEAF}(\mathcal{H}_i) \\ \Phi(?_0^{\chi}(\mathcal{H}_1, \ldots, \mathcal{H}_n)) &= \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_\chi[?_0/y, \vec{?_i}/\vec{x_i}] \end{aligned}$$

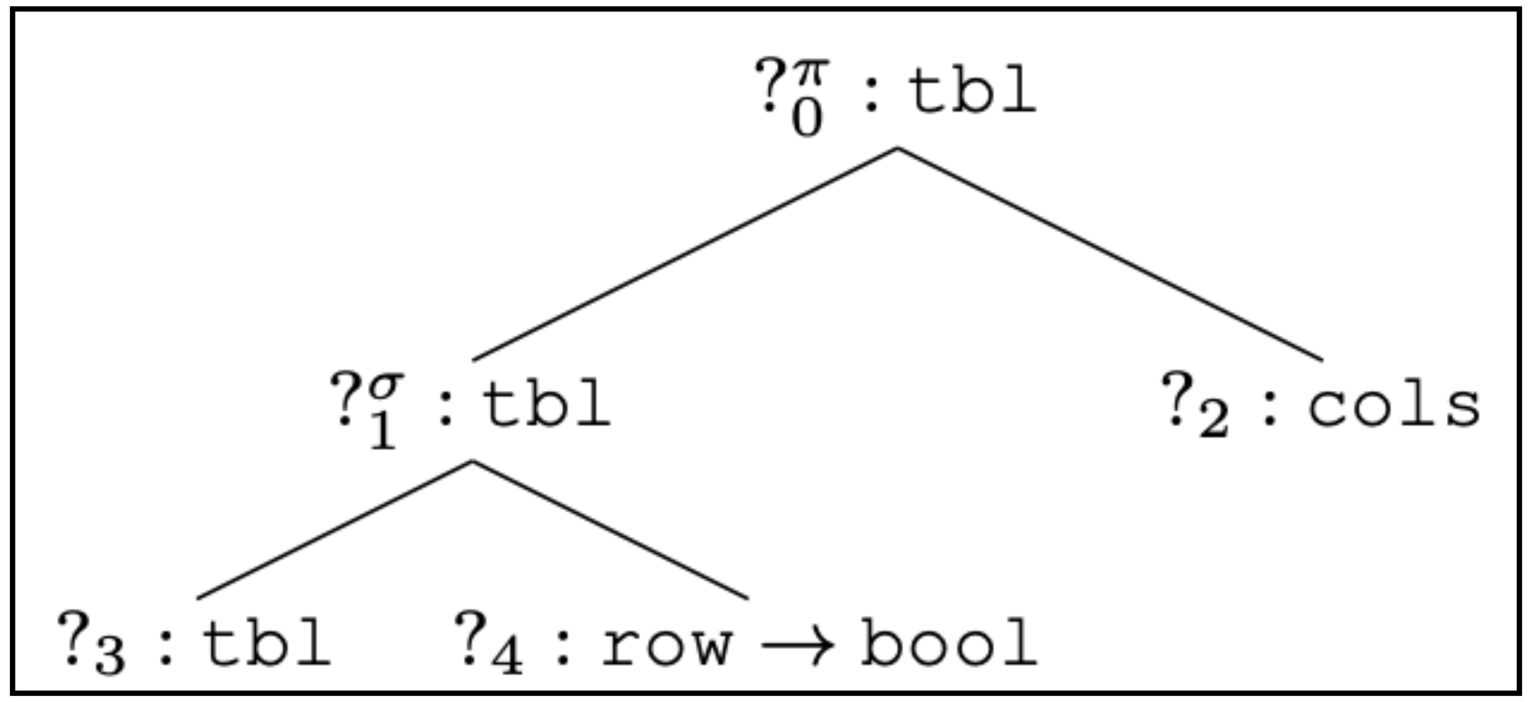
id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

$y = ?_0$



$?_3 = x_1$

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
  output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$
  $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
  return SAT(ψ)

$$\begin{array}{lll} \Phi(\mathcal{H}_i) & = \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] & \text{if } \neg\text{PARTIAL}([\![\mathcal{H}_i]\!]_\partial) \\ \Phi(\mathcal{H}_i) & = \top & \text{else if ISLEAF}(\mathcal{H}_i) \\ \Phi(?_0^{\chi}(\mathcal{H}_1, ..., \mathcal{H}_n)) & = \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_\chi[?_0/y, \vec{?_i}/\vec{x_i}] & \end{array}$$

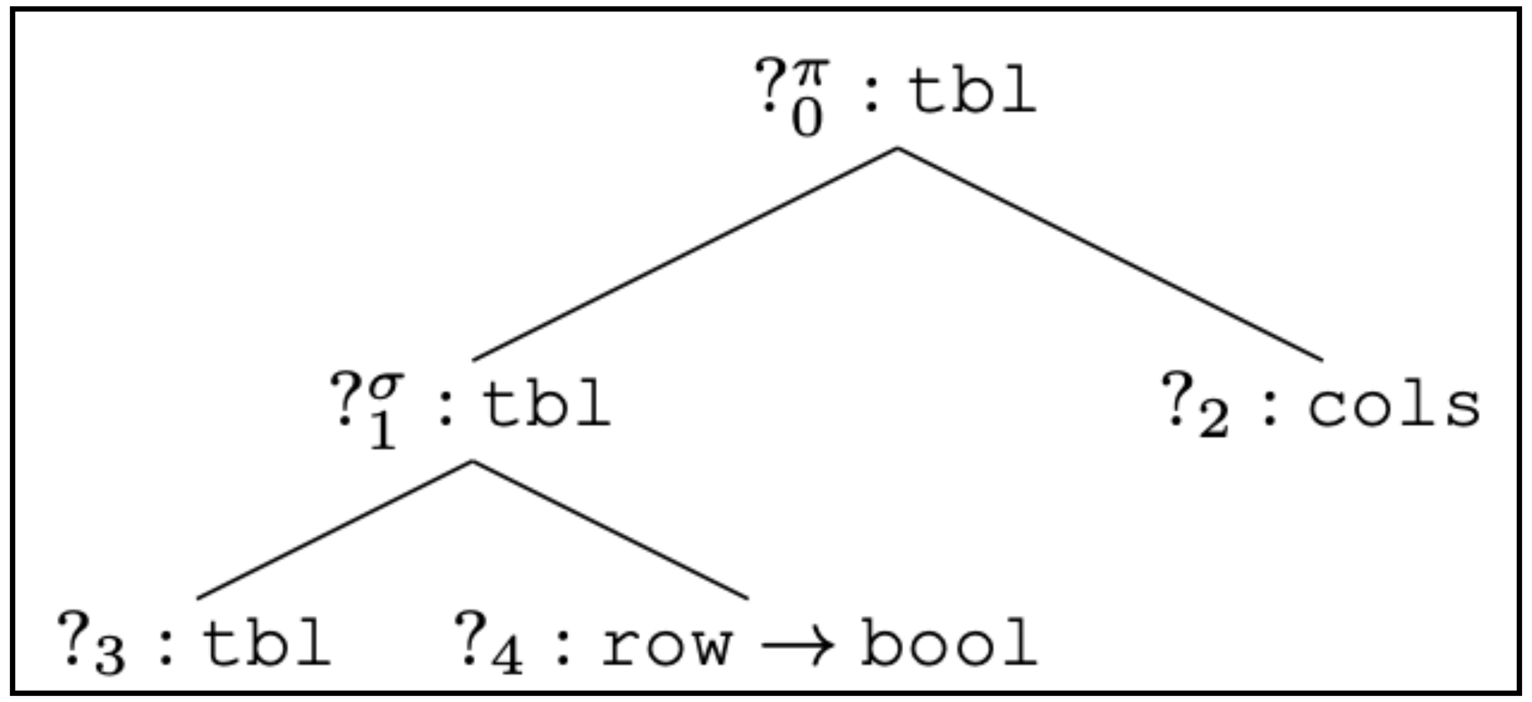
id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

$y = ?_0$



$?_3 = x_1$

$x_1 . row = 3 \wedge x_1 . col = 4 \bigwedge y . row = 2 \wedge y . col = 4$

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)

    input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
    output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise

    $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$

    $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$

    $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$

    $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$

    return SAT(ψ)

$$
\begin{aligned}
\Phi(\mathcal{H}_i) &= \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] \text{ if } \neg\text{PARTIAL}([\![\mathcal{H}_i]\!]_\partial) \\
\Phi(\mathcal{H}_i) &= \top \qquad \text{else if } \text{ISLEAF}(\mathcal{H}_i) \\
\Phi(?_0^{\chi}(\mathcal{H}_1, ..., \mathcal{H}_n)) &= \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_\chi[?_0/y, \vec{?_i}/\vec{x_i}]
\end{aligned}
$$

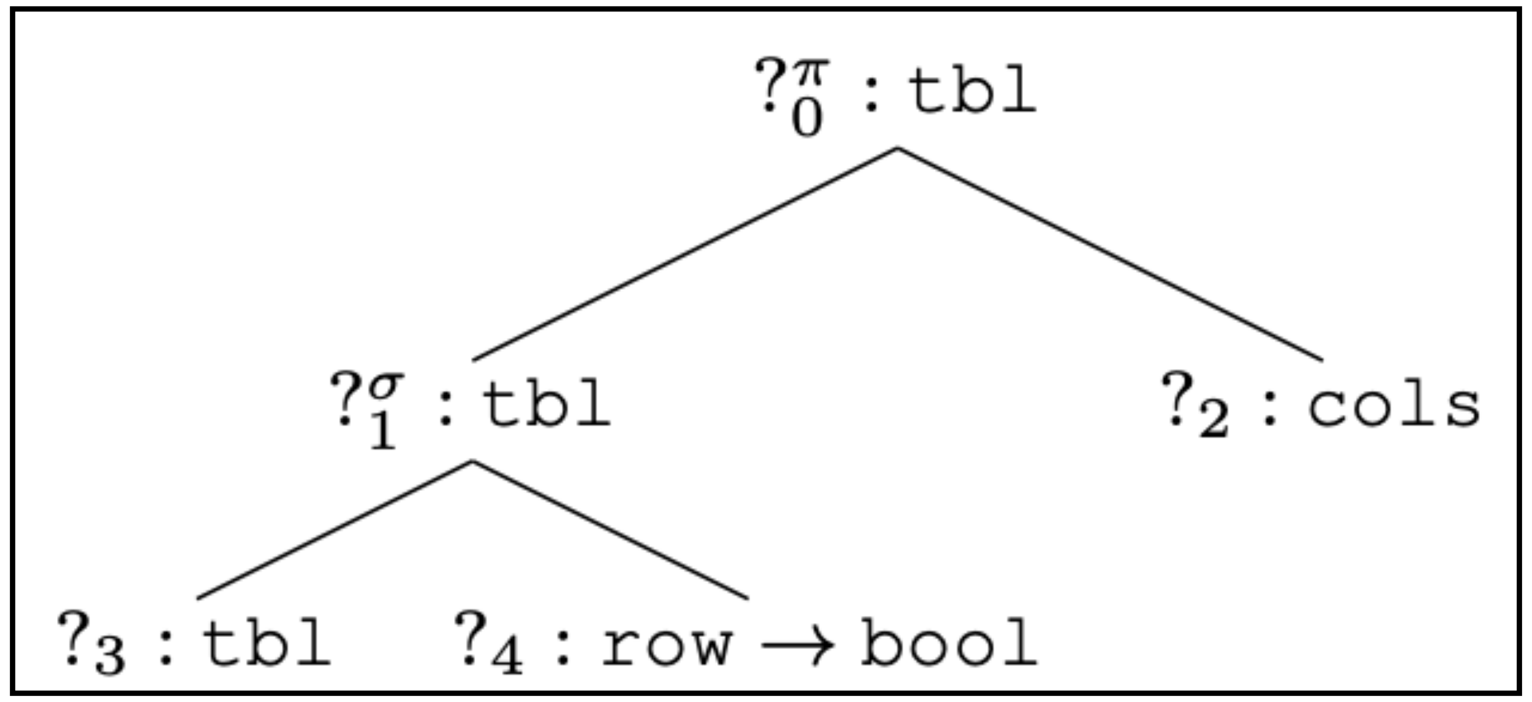
id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

$y = ?_0$



$?_3 = x_1$

$x_1.row = 3 \wedge x_1.col = 4 \bigwedge y.row = 2 \wedge y.col = 4$

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

```
procedure DEDUCE(H, E)
    input: Hypothesis H, input-output example E
    output: ⊥ if cannot be unified with E; ⊤ otherwise
    S := {?_j | ?_j : tbl ∈ LEAVES(H)}
    φ_in := ⋀_{?_j∈S} ⋁_{1≤i≤|E_in|} (?_j = x_i)
    φ_out := (y = ROOTVAR(H))
    ψ := ( Φ(H) ∧ φ_in ∧ φ_out ∧ ⋀_{T_i∈E_in} (α(T_i)[x_i/x]) ∧ α(T_out)[y/x] )
    return SAT(ψ)
```

$$\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$$

$$\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$$

$$\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$$

$$\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$$

$$\begin{aligned} \Phi(\mathcal{H}_i) &= \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] \text{ if } \neg\text{PARTIAL}([\![\mathcal{H}_i]\!]_\partial) \\ \Phi(\mathcal{H}_i) &= \top \quad \text{else if ISLEAF}(\mathcal{H}_i) \\ \Phi(?_0^{\chi}(\mathcal{H}_1, ..., \mathcal{H}_n)) &= \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_\chi[?_0/y, \vec{?_i}/\vec{x_i}] \end{aligned}$$

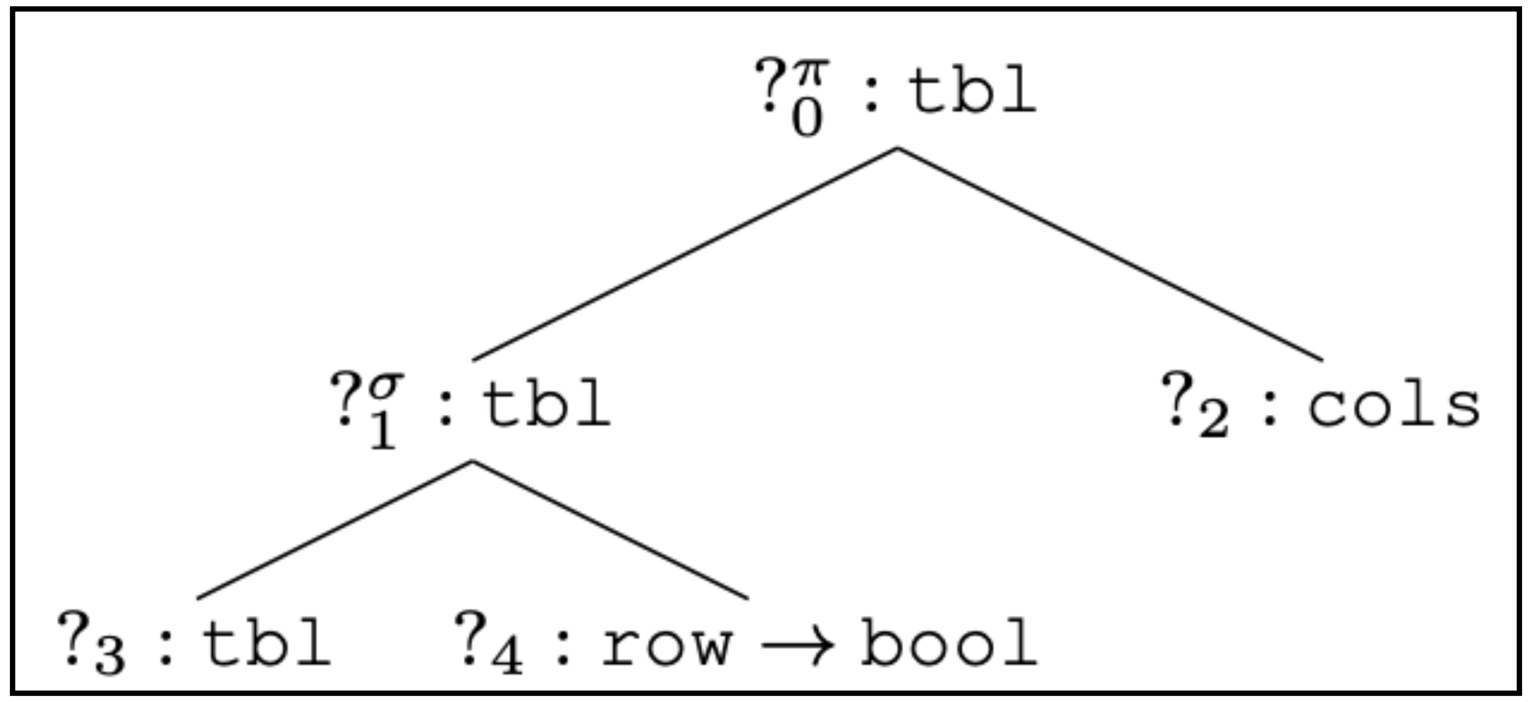
id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

$$y = ?_0$$



$$?_3 = x_1$$

$$x_1.row = 3 \wedge x_1.col = 4 \bigwedge y.row = 2 \wedge y.col = 4$$

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)

input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$

output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise

$\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$

$\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$

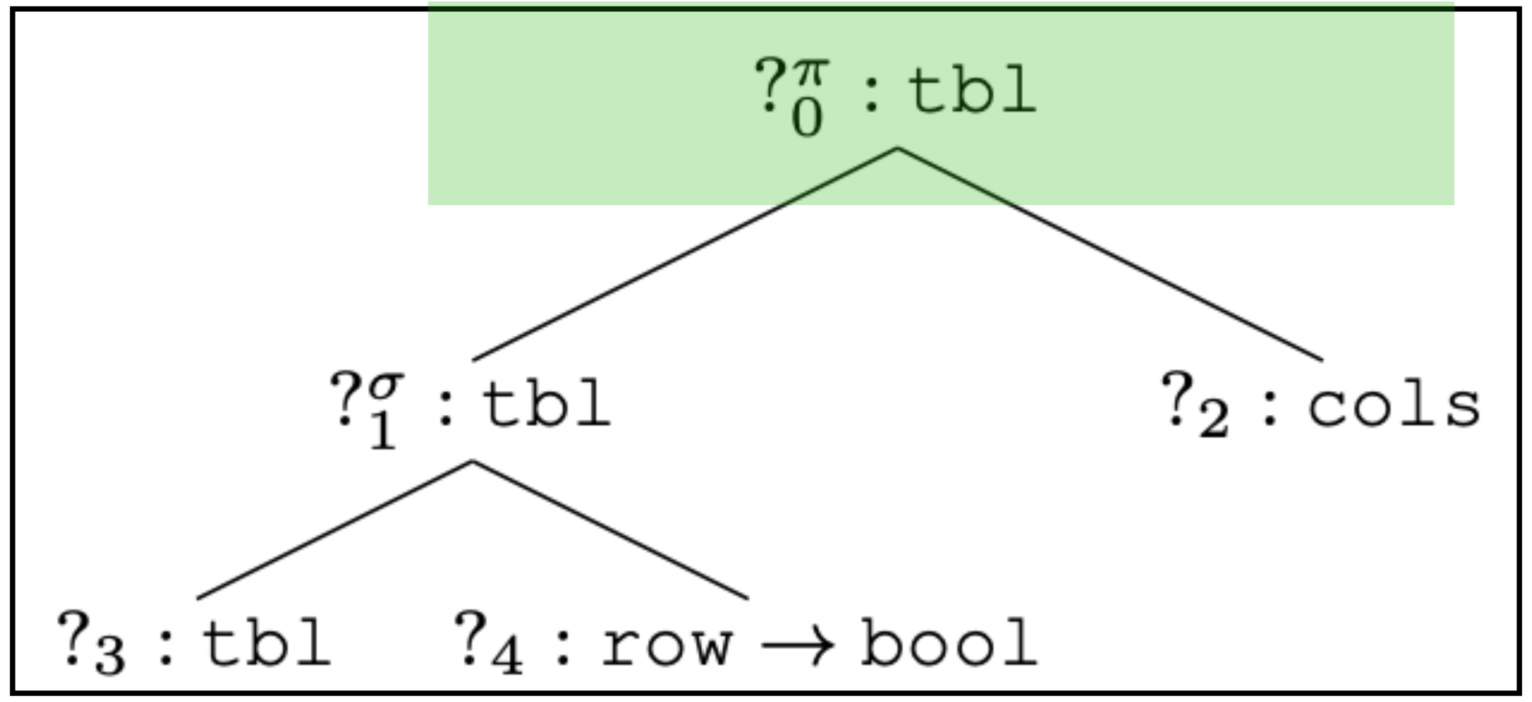
$\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$

$\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$

return SAT(ψ)

$y = ?_0$

$?_0 . row = ?_1 . row \wedge ?_0 . col < ?_1 . col$



$?_3 = x_1$

$$\begin{array}{lll} \Phi(\mathcal{H}_i) & = \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] & \text{if } \neg\text{PARTIAL}([\![\mathcal{H}_i]\!]_\partial) \\ \Phi(\mathcal{H}_i) & = \top & \text{else if ISLEAF}(\mathcal{H}_i) \\ \Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, \dots, \mathcal{H}_n)) & = \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}] & \end{array}$$

$x_1 . row = 3 \wedge x_1 . col = 4 \bigwedge y . row = 2 \wedge y . col = 4$

id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
  output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$
  $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \land \varphi_{in} \land \varphi_{out} \land \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \land \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
  return SAT(ψ)

$$\begin{aligned} \Phi(\mathcal{H}_i) &= \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] \text{ if } \neg\text{PARTIAL}([\![\mathcal{H}_i]\!]_\partial) \\ \Phi(\mathcal{H}_i) &= \top \quad \text{else if ISLEAF}(\mathcal{H}_i) \\ \Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, \ldots, \mathcal{H}_n)) &= \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \land \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}] \end{aligned}$$

id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

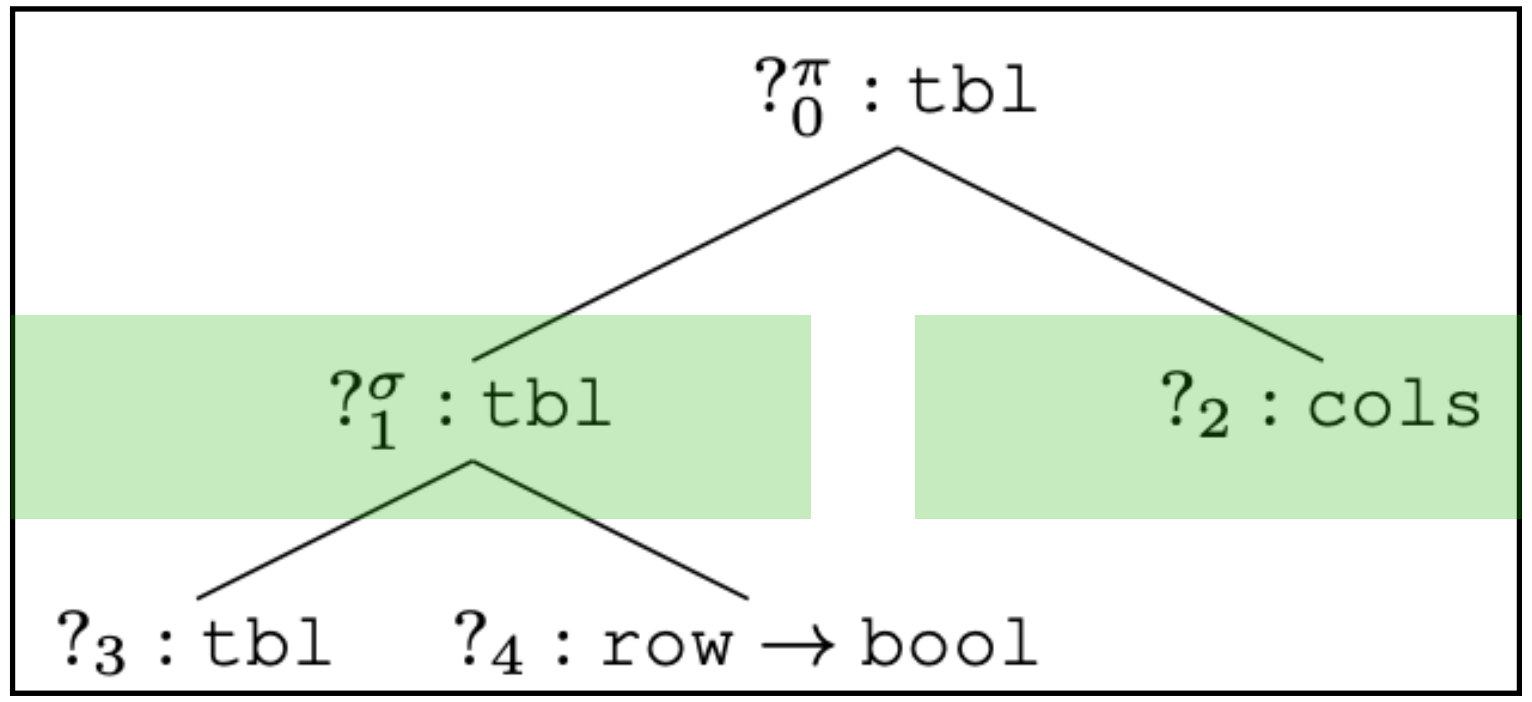
Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

$$y = ?_0$$

$$?_0.row = ?_1.row \land ?_0.col < ?_1.col$$



$$?_3 = x_1$$

$$x_1.row = 3 \land x_1.col = 4 \bigwedge y.row = 2 \land y.col = 4$$

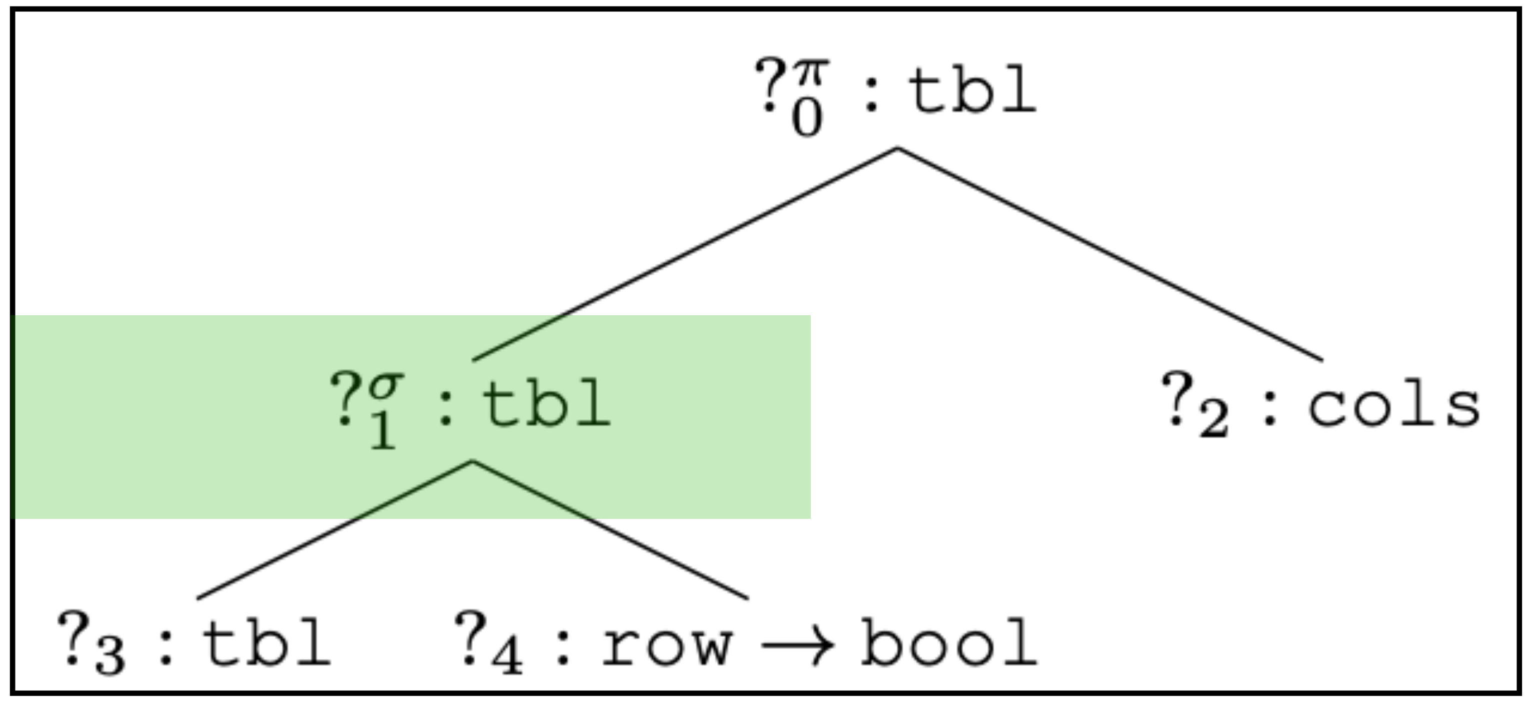
Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

```
procedure DEDUCE(H, E)
    input: Hypothesis H, input-output example E
    output: ⊥ if cannot be unified with E; ⊤ otherwise
    S := {?_j | ?_j : tbl ∈ LEAVES(H)}
    φ_in := ⋀_{?_j∈S} ⋁_{1≤i≤|E_in|} (?_j = x_i)
    φ_out := (y = ROOTVAR(H))
    ψ := ( Φ(H) ∧ φ_in ∧ φ_out ∧ ⋀_{T_i∈E_in} (α(T_i)[x_i/x]) ∧ α(T_out)[y/x] )
    return SAT(ψ)
```

$$\Phi(\mathcal{H}_i) = \alpha(\llbracket \mathcal{H}_i \rrbracket_\partial)[?_i/x] \text{ if } \neg\textsc{Partial}(\llbracket \mathcal{H}_i \rrbracket_\partial)$$
$$\Phi(\mathcal{H}_i) = \top \quad \text{else if } \textsc{IsLeaf}(\mathcal{H}_i)$$
$$\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) = \bigwedge_{1\le i\le n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}]$$

$$y = ?_0$$
$$?_0 . row = ?_1 . row \wedge ?_0 . col < ?_1 . col$$



$$?_3 = x_1$$

$$x_1 . row = 3 \wedge x_1 . col = 4 \bigwedge y . row = 2 \wedge y . col = 4$$

id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

```
procedure DEDUCE(H, E)
    input: Hypothesis H, input-output example E
    output: ⊥ if cannot be unified with E; ⊤ otherwise
    S := {?_j | ?_j : tbl ∈ LEAVES(H)}
    φ_in := ⋀_{?_j ∈ S} ⋁_{1≤i≤|E_in|} (?_j = x_i)
    φ_out := (y = ROOTVAR(H))
    ψ := ( Φ(H) ∧ φ_in ∧ φ_out ∧ ⋀_{T_i ∈ E_in} (α(T_i)[x_i/x]) ∧ α(T_out)[y/x] )
    return SAT(ψ)
```

$$\Phi(\mathcal{H}_i) = \alpha(\llbracket \mathcal{H}_i \rrbracket_\partial)[?_i/x] \text{ if } \neg\textsc{Partial}(\llbracket \mathcal{H}_i \rrbracket_\partial)$$

$$\Phi(\mathcal{H}_i) = \top \quad \text{else if } \textsc{IsLeaf}(\mathcal{H}_i)$$

$$\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) = \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}]$$

id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

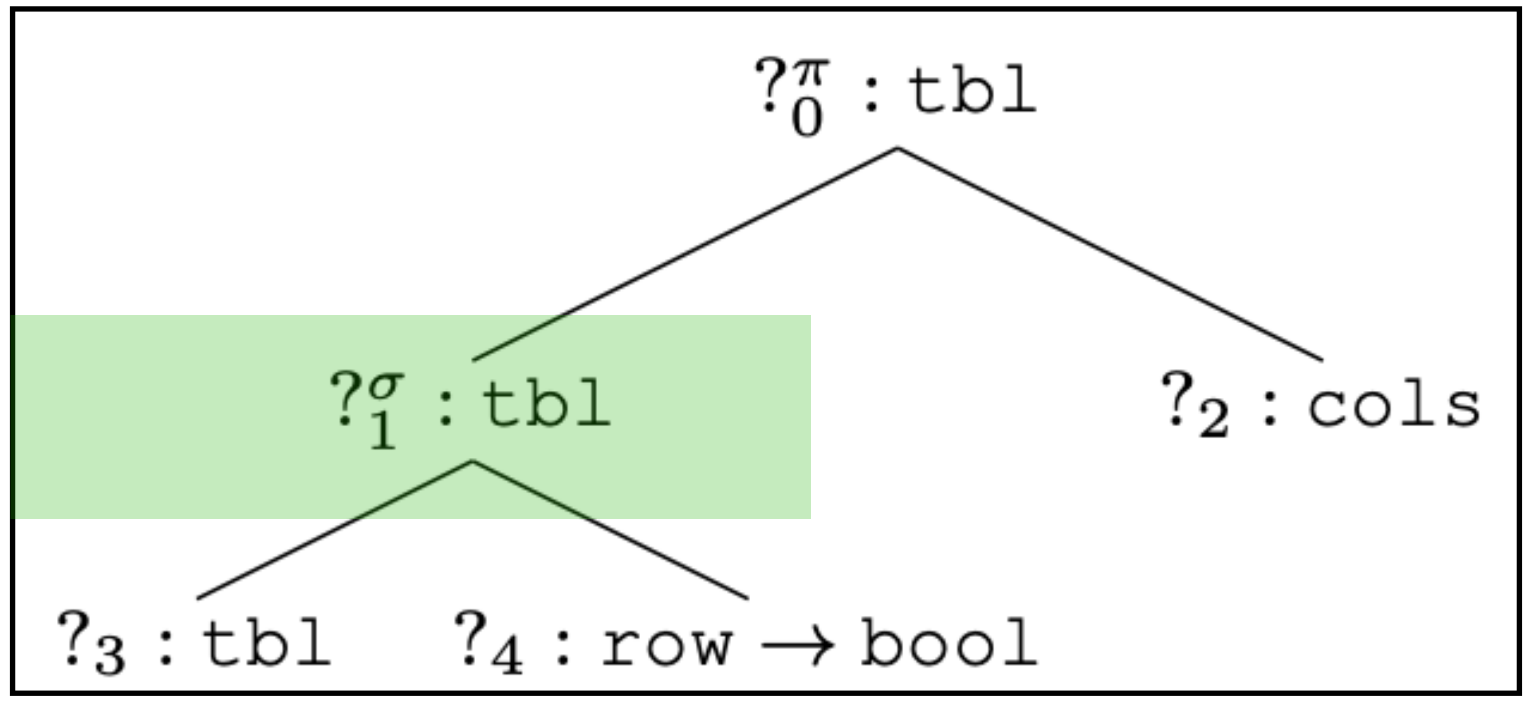
Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

$$y = ?_0$$

$$?_0 . row = ?_1 . row \wedge ?_0 . col < ?_1 . col$$



$$?_3 = x_1$$

$$x_1 . row = 3 \wedge x_1 . col = 4 \bigwedge y . row = 2 \wedge y . col = 4$$

Use an Example to Explain Deduction

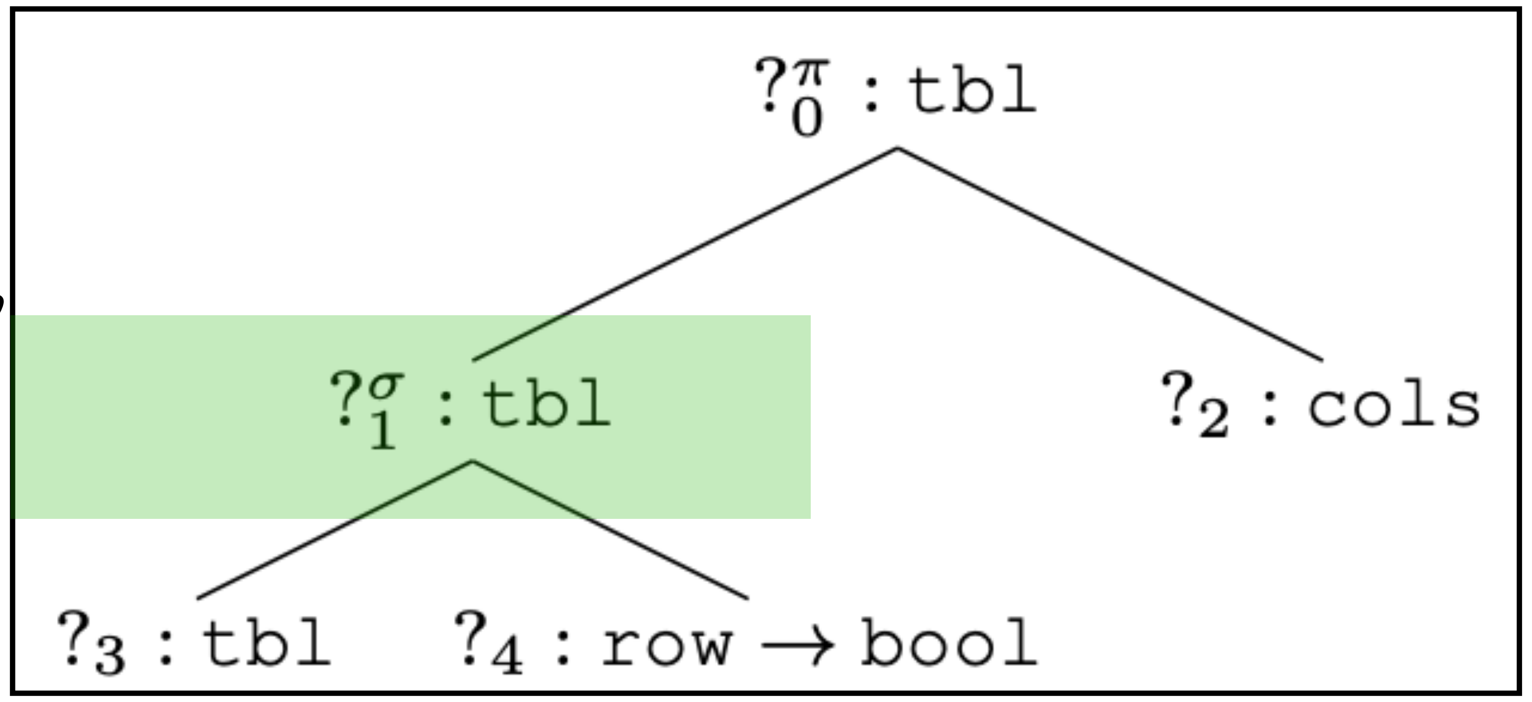
Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
  output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$
  $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \land \varphi_{in} \land \varphi_{out} \land \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \land \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
  return SAT(ψ)

$$y = ?_0$$

$$?_0.row = ?_1.row \land ?_0.col < ?_1.col$$

$$?_1.row < ?_3.row \land ?_1.col = ?_3.col$$



$$?_3 = x_1$$

$$\begin{array}{lll} \Phi(\mathcal{H}_i) & = \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] & \text{if } \neg\text{PARTIAL}([\![\mathcal{H}_i]\!]_\partial) \\ \Phi(\mathcal{H}_i) & = \top & \text{else if ISLEAF}(\mathcal{H}_i) \\ \Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) & = \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \land \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}] & \end{array}$$

$$x_1.row = 3 \land x_1.col = 4 \bigwedge y.row = 2 \land y.col = 4$$

id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
  output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$
  $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
  return SAT(ψ)

$$\begin{aligned} \Phi(\mathcal{H}_i) &= \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] \text{ if } \neg\text{PARTIAL}([\![\mathcal{H}_i]\!]_\partial) \\ \Phi(\mathcal{H}_i) &= \top \quad \text{else if ISLEAF}(\mathcal{H}_i) \\ \Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) &= \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}] \end{aligned}$$

id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

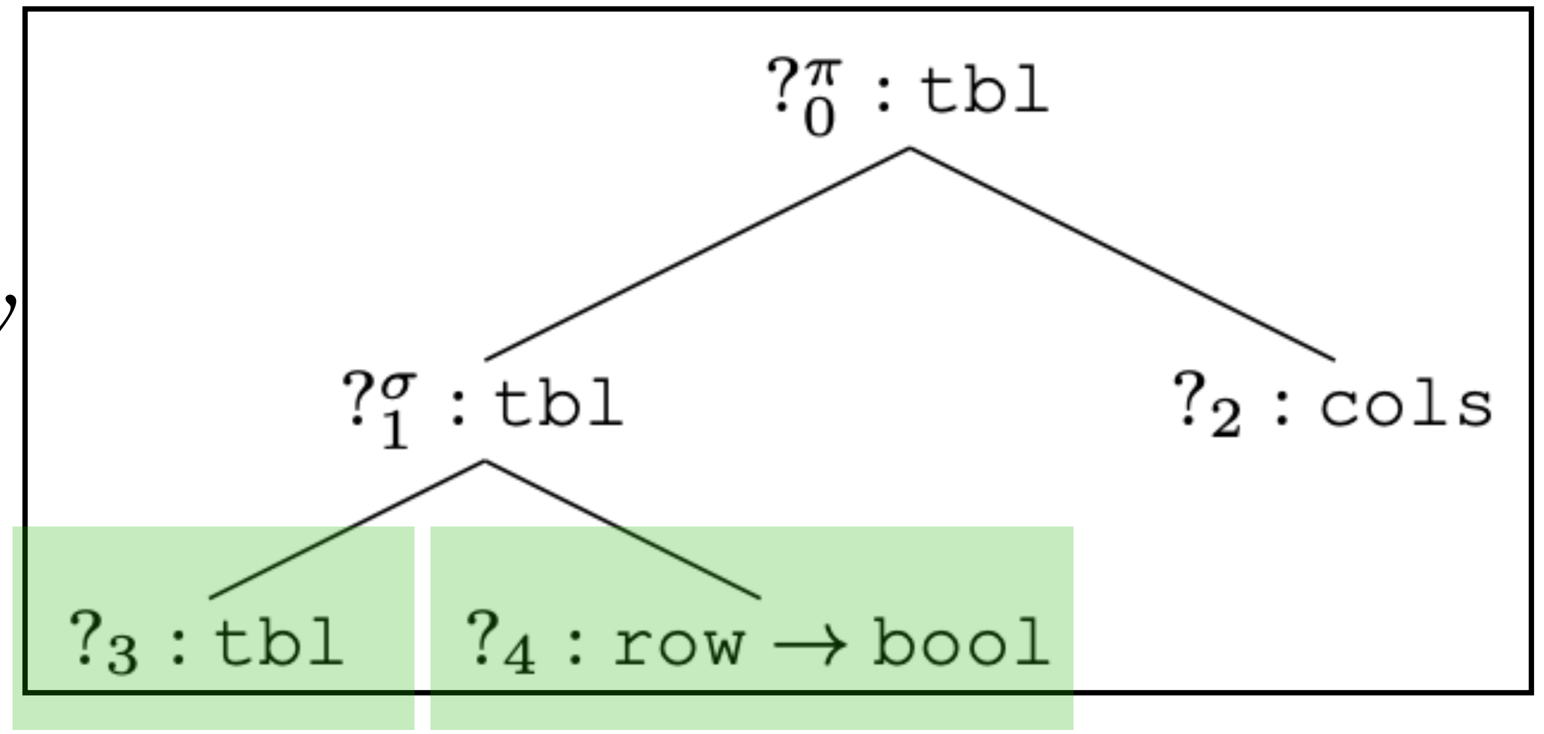
id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

$$y = ?_0$$

$$?_0.row = ?_1.row \wedge ?_0.col < ?_1.col$$

$$?_1.row < ?_3.row \wedge ?_1.col = ?_3.col$$



$$?_3 = x_1$$

$$x_1.row = 3 \wedge x_1.col = 4 \bigwedge y.row = 2 \wedge y.col = 4$$

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
  output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$
  $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
  return SAT(ψ)

$$\begin{array}{lll} \Phi(\mathcal{H}_i) & = \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] & \text{if } \neg\text{PARTIAL}([\![\mathcal{H}_i]\!]_\partial) \\ \Phi(\mathcal{H}_i) & = \top & \text{else if ISLEAF}(\mathcal{H}_i) \\ \Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) & = \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}] & \end{array}$$

id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

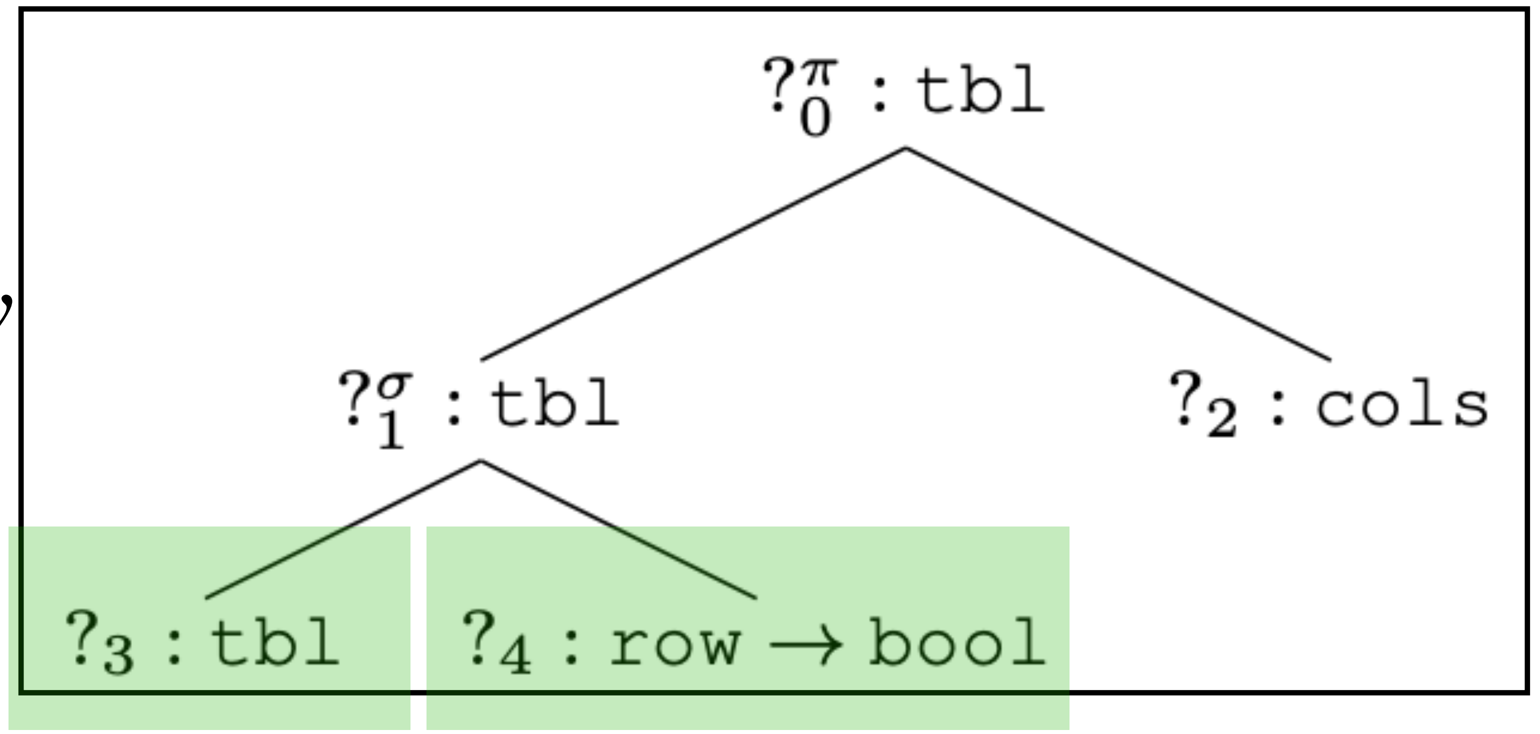
id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

$$y = ?_0$$

$$?_0 . row = ?_1 . row \wedge ?_0 . col < ?_1 . col$$

$$?_1 . row < ?_3 . row \wedge ?_1 . col = ?_3 . col$$



$$?_3 = x_1 \quad \top \quad \top$$

$$x_1 . row = 3 \wedge x_1 . col = 4 \bigwedge y . row = 2 \wedge y . col = 4$$

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
  output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$
  $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
  return SAT(ψ)

$$\Phi(\mathcal{H}_i) = \alpha(\llbracket \mathcal{H}_i \rrbracket_\partial)[?_i/x] \text{ if } \neg\text{PARTIAL}(\llbracket \mathcal{H}_i \rrbracket_\partial)$$
$$\Phi(\mathcal{H}_i) = \top \quad \text{else if ISLEAF}(\mathcal{H}_i)$$
$$\Phi(?_0^{\chi}(\mathcal{H}_1, \ldots, \mathcal{H}_n)) = \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_\chi[?_0/y, \vec{?_i}/\vec{x_i}]$$

id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

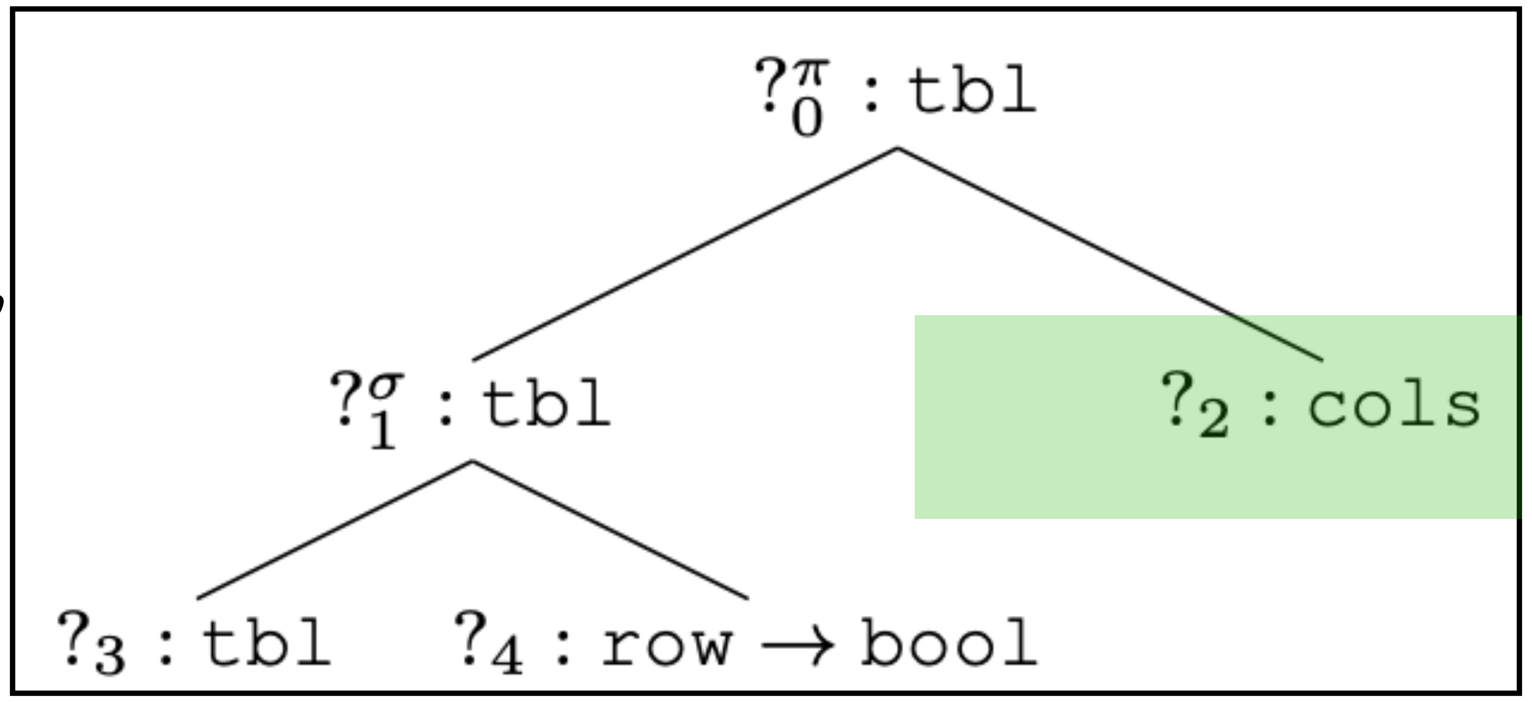
Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

$$y = ?_0$$
$$?_0.row = ?_1.row \wedge ?_0.col < ?_1.col$$

$$?_1.row < ?_3.row \wedge ?_1.col = ?_3.col$$



$\top$

$?_3 = x_1$ $\top$ $\top$

$$x_1.row = 3 \wedge x_1.col = 4 \bigwedge y.row = 2 \wedge y.col = 4$$

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

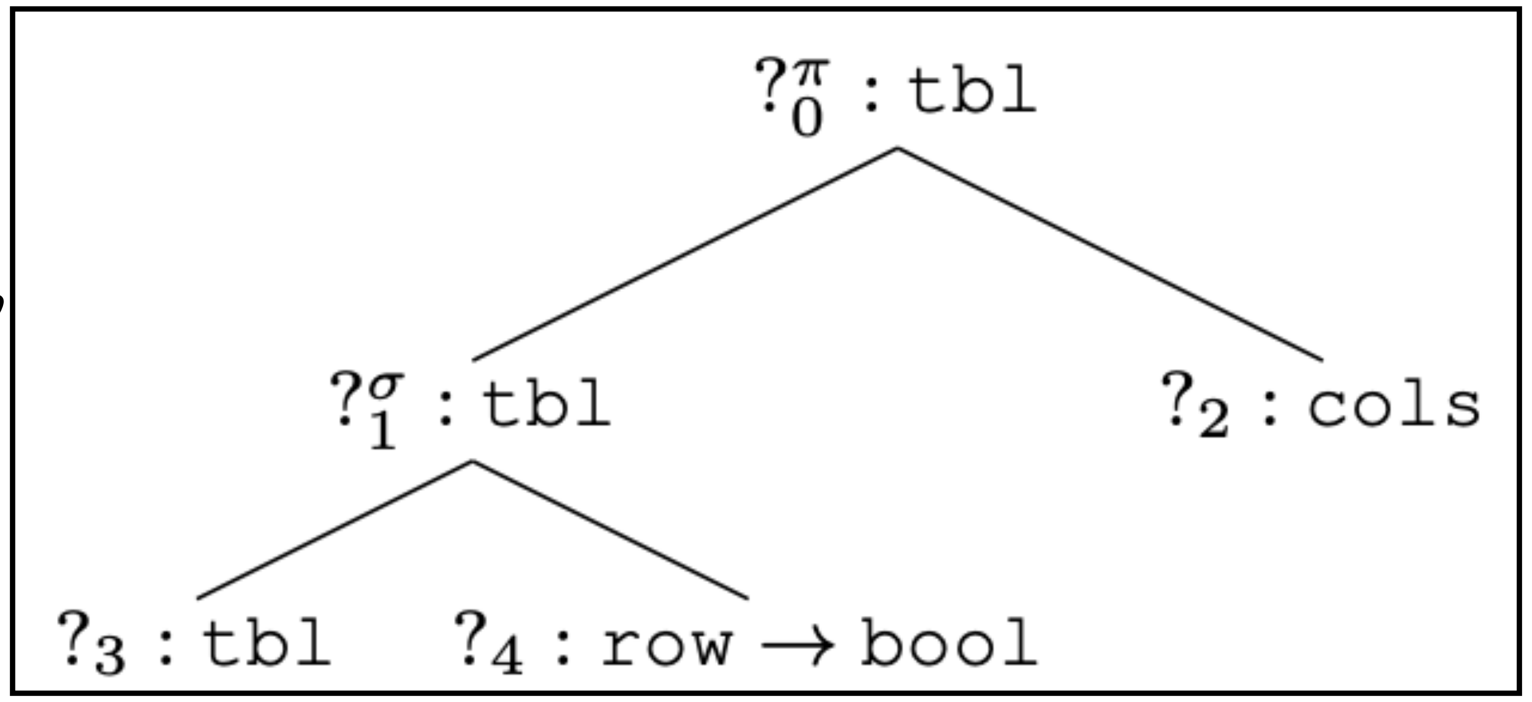
procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
    input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
    output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
    $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
    $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$
    $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
    $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \wedge \varphi_{in} \wedge \varphi_{out} \wedge \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \wedge \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
    return SAT(ψ)

$$\begin{aligned} \Phi(\mathcal{H}_i) &= \alpha(\llbracket \mathcal{H}_i \rrbracket_\partial)[?_i/x] \text{ if } \neg\text{PARTIAL}(\llbracket \mathcal{H}_i \rrbracket_\partial) \\ \Phi(\mathcal{H}_i) &= \top \quad \text{else if ISLEAF}(\mathcal{H}_i) \\ \Phi(?_0^{\chi}(\mathcal{H}_1, ..., \mathcal{H}_n)) &= \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_\chi[?_0/y, \vec{?_i}/\vec{x_i}] \end{aligned}$$

$y = ?_0$

$?_0 . row = ?_1 . row \wedge ?_0 . col < ?_1 . col$

$?_1 . row < ?_3 . row$
$\wedge ?_1 . col = ?_3 . col$



$\top$

$?_3 = x_1$ $\top$ $\top$

$x_1 . row = 3 \wedge x_1 . col = 4 \bigwedge y . row = 2 \wedge y . col = 4$

id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

Use an Example to Explain Deduction

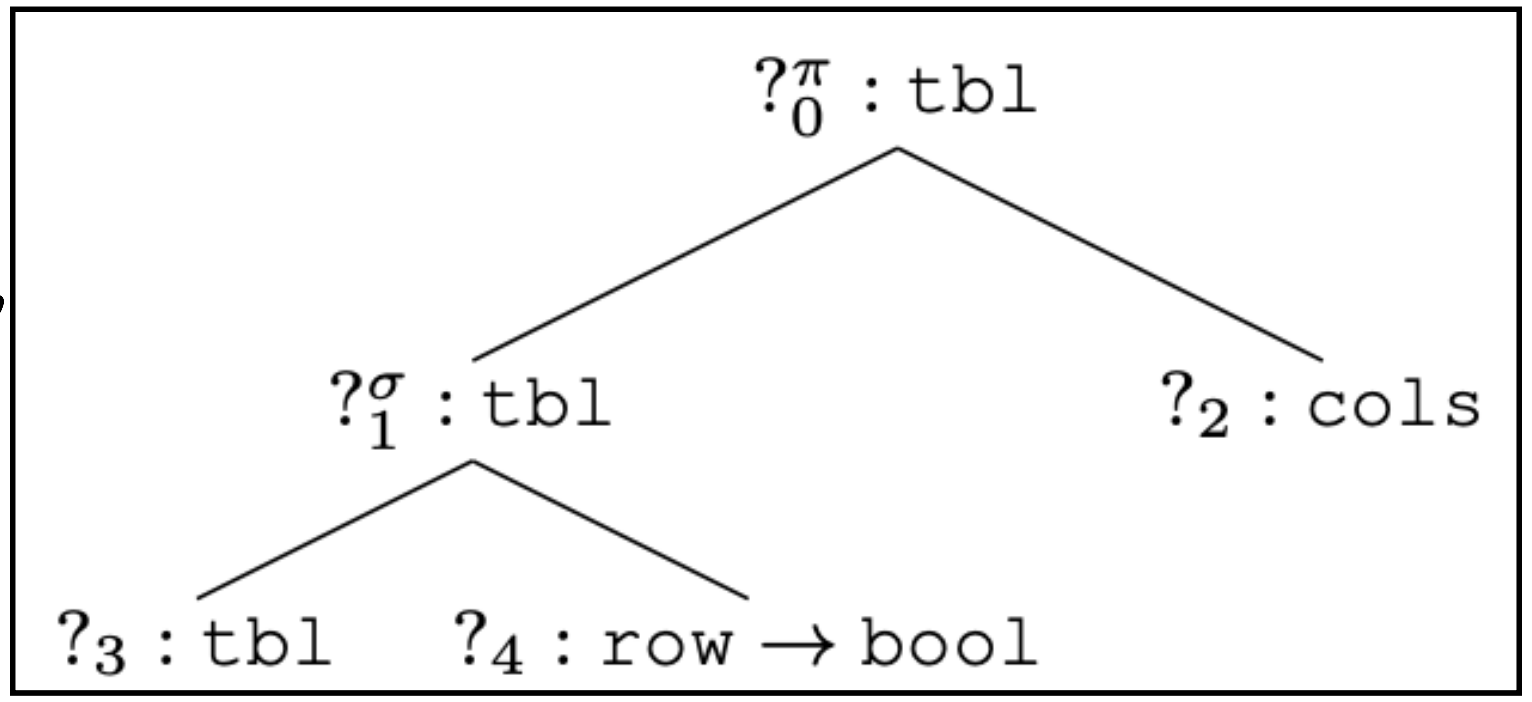
Algorithm 2 SMT-based Deduction Algorithm

```
procedure DEDUCE(H, E)
    input: Hypothesis H, input-output example E
    output: ⊥ if cannot be unified with E; ⊤ otherwise
    S := {?_j | ?_j : tbl ∈ LEAVES(H)}
    φ_in := ⋀_{?_j ∈ S} ⋁_{1≤i≤|E_in|} (?_j = x_i)
    φ_out := (y = ROOTVAR(H))
    ψ := ( Φ(H) ∧ φ_in ∧ φ_out ∧ ⋀_{T_i ∈ E_in} (α(T_i)[x_i/x]) ∧ α(T_out)[y/x] )
    return SAT(ψ)
```

$$y = ?_0$$

$$?_0 . row = ?_1 . row \wedge ?_0 . col < ?_1 . col$$

$$?_1 . row < ?_3 . row$$
$$\wedge\ ?_1 . col = ?_3 . col$$



$\top$

$?_3 = x_1$ $\top$ $\top$

$$\begin{aligned}
\Phi(\mathcal{H}_i) &= \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] \text{ if } \neg\textsc{Partial}([\![\mathcal{H}_i]\!]_\partial) \\
\Phi(\mathcal{H}_i) &= \top \quad \text{else if } \textsc{IsLeaf}(\mathcal{H}_i) \\
\Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) &= \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \wedge \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}]
\end{aligned}$$

$$x_1 . row = 3 \wedge x_1 . col = 4 \bigwedge y . row = 2 \wedge y . col = 4$$

id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

Where do we use partial evaluation?

Use an Example to Explain Deduction

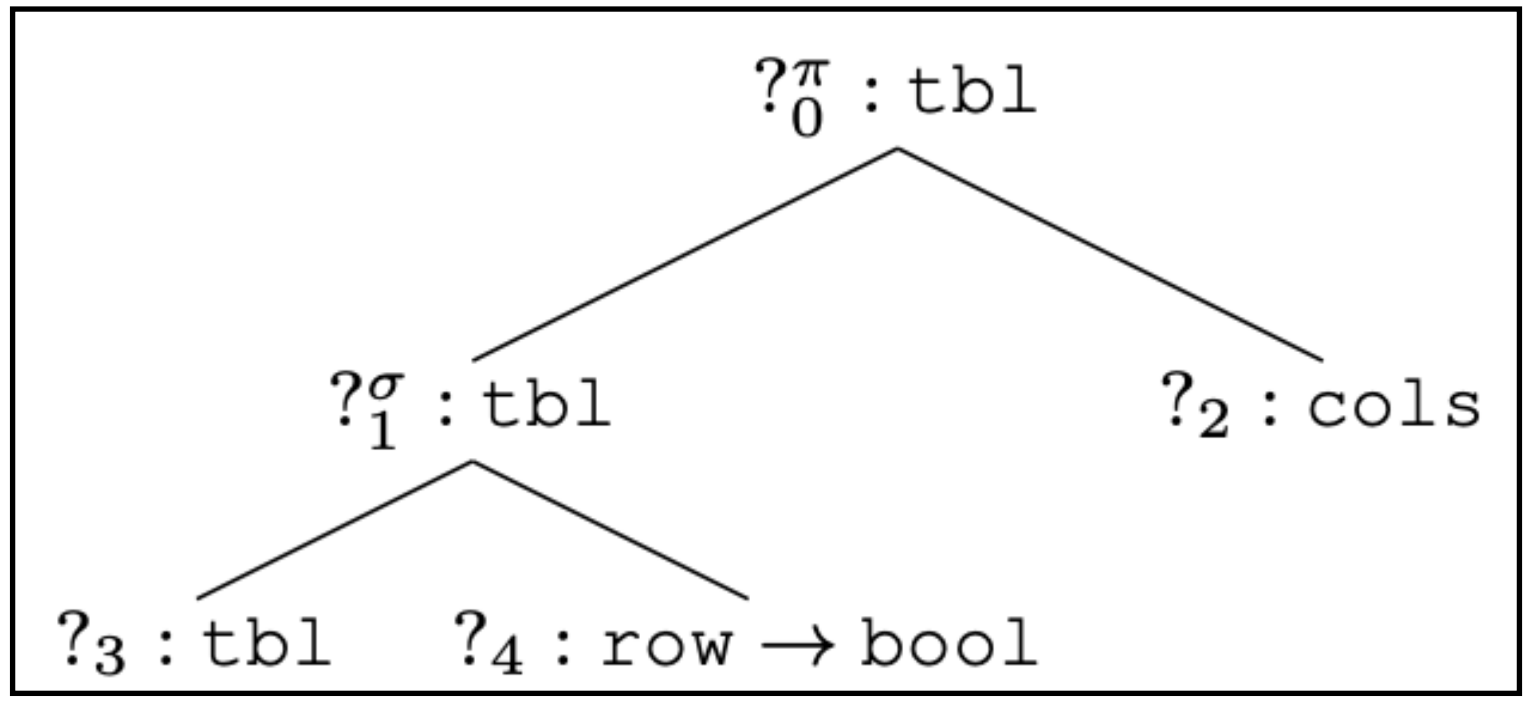
Algorithm 2 SMT-based Deduction Algorithm

```
procedure DEDUCE(H, E)
    input: Hypothesis H, input-output example E
    output: ⊥ if cannot be unified with E; ⊤ otherwise
    S := {?_j | ?_j : tbl ∈ LEAVES(H)}
    φ_in := ⋀_{?_j ∈ S} ⋁_{1 ≤ i ≤ |E_in|} (?_j = x_i)
    φ_out := (y = ROOTVAR(H))
    ψ := ( Φ(H) ∧ φ_in ∧ φ_out ∧ ⋀_{T_i ∈ E_in} (α(T_i)[x_i/x]) ∧ α(T_out)[y/x] )
    return SAT(ψ)
```

$$?_1.row < ?_3.row \land ?_1.col = ?_3.col \land ?_0.row = ?_1.row \land ?_0.col < ?_1.col \bigwedge ?_3 = x_1 \bigwedge y = ?_0$$

$$\bigwedge x_1.row = 3 \land x_1.col = 4 \bigwedge y.row = 2 \land y.col = 4$$

$$\begin{aligned} \Phi(\mathcal{H}_i) &= \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] \text{ if } \neg\textsc{Partial}([\![\mathcal{H}_i]\!]_\partial) \\ \Phi(\mathcal{H}_i) &= \top \quad \text{else if } \textsc{IsLeaf}(\mathcal{H}_i) \\ \Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, \ldots, \mathcal{H}_n)) &= \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \land \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}] \end{aligned}$$



id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

Use an Example to Explain Deduction

Algorithm 2 SMT-based Deduction Algorithm

procedure DEDUCE($\mathcal{H}, \mathcal{E}$)
  input: Hypothesis $\mathcal{H}$, input-output example $\mathcal{E}$
  output: $\bot$ if cannot be unified with $\mathcal{E}$; $\top$ otherwise
  $\mathcal{S} := \{?_j \mid ?_j : \texttt{tbl} \in \text{LEAVES}(\mathcal{H})\}$
  $\varphi_{in} := \bigwedge_{?_j \in \mathcal{S}} \bigvee_{1 \le i \le |\mathcal{E}_{in}|} (?_j = x_i)$
  $\varphi_{out} := (y = \text{ROOTVAR}(\mathcal{H}))$
  $\psi := \left(\begin{array}{c} \Phi(\mathcal{H}) \land \varphi_{in} \land \varphi_{out} \land \\ \bigwedge_{\mathsf{T}_i \in \mathcal{E}_{in}} (\alpha(\mathsf{T}_i)[x_i/x]) \land \alpha(\mathsf{T}_{out})[y/x] \end{array} \right)$
  return SAT(ψ)

$$\begin{aligned} \Phi(\mathcal{H}_i) &= \alpha([\![\mathcal{H}_i]\!]_\partial)[?_i/x] \text{ if } \neg\text{PARTIAL}([\![\mathcal{H}_i]\!]_\partial) \\ \Phi(\mathcal{H}_i) &= \top \qquad \text{else if ISLEAF}(\mathcal{H}_i) \\ \Phi(?_0^{\mathcal{X}}(\mathcal{H}_1, ..., \mathcal{H}_n)) &= \bigwedge_{1 \le i \le n} \Phi(\mathcal{H}_i) \land \phi_{\mathcal{X}}[?_0/y, \vec{?_i}/\vec{x_i}] \end{aligned}$$

id	name	age	GPA
1	Alice	8	4.0
2	Bob	18	3.2
3	Tom	12	3.0

Input Example

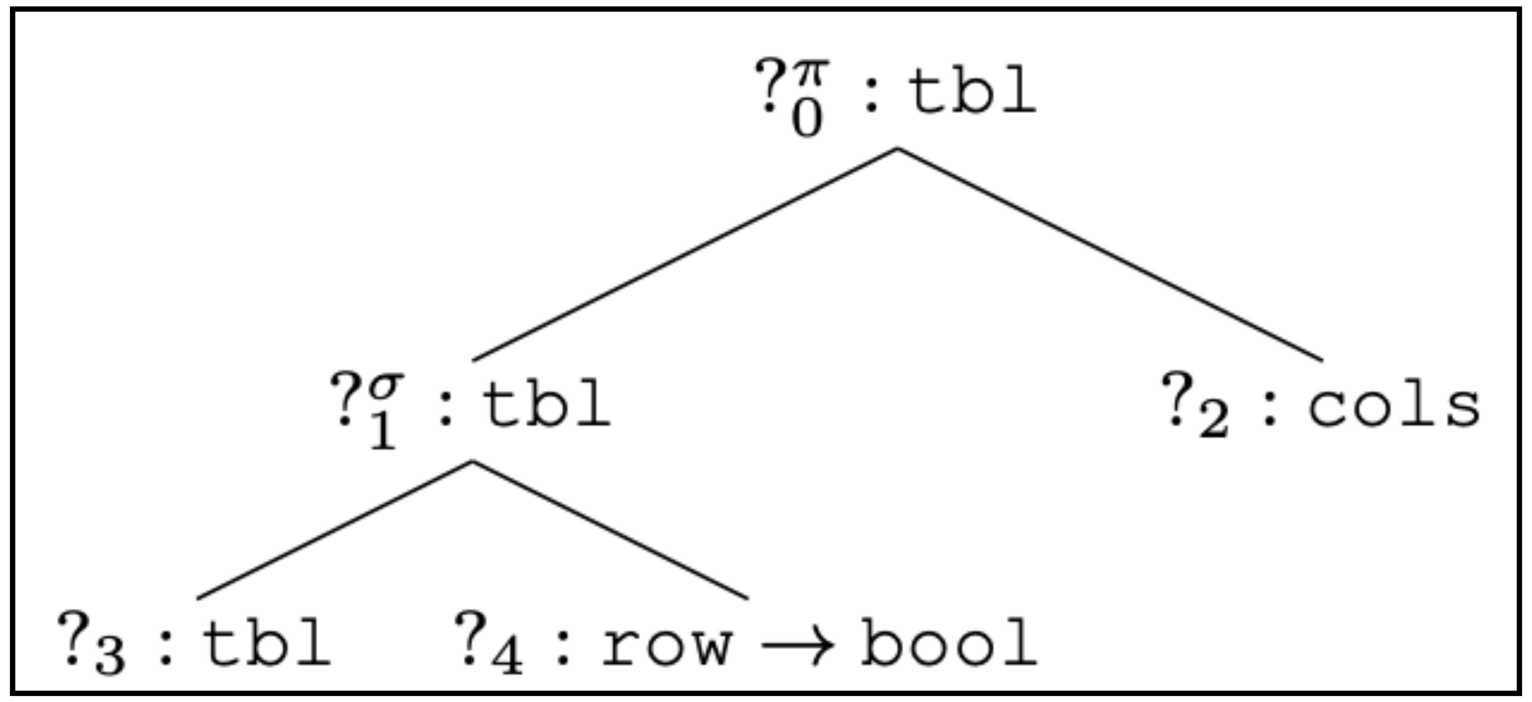
id	name	age	GPA
2	Bob	18	3.2
3	Tom	12	3.0

Output Example

UNSAT

$$?_1.row < ?_3.row \land ?_1.col = ?_3.col \land ?_0.row = ?_1.row \land ?_0.col < ?_1.col \land ?_3 = x_1 \land y = ?_0$$

$$\land\ x_1.row = 3 \land x_1.col = 4 \land y.row = 2 \land y.col = 4$$



Sketch Completion

procedure SYNTHESIZE($\mathcal{E}, \Lambda$)
  input: Input-output example $\mathcal{E}$ and components Λ
  output: Synthesized program or $\perp$ if failure
  $W := \{?_0\texttt{:tbl}\}$ ▷ Init worklist
  while $W \neq \emptyset$ **do**
    choose $\mathcal{H} \in W$;
    $W := W \backslash \{\mathcal{H}\}$
    if DEDUCE($\mathcal{H}, \mathcal{E}$) $= \perp$ **then** ▷ Contradiction
      goto refine;
    ▷ No contradiction
    for $\mathcal{S} \in$ SKETCHES($\mathcal{H}, \mathcal{E}_{in}$) **do**
      $\mathcal{P}$:= FILLSKETCH($\mathcal{S}, \mathcal{E}$)
      for $p \in \mathcal{P}$ **do**
        if CHECK($p, \mathcal{E}$) **then return** p
    refine: ▷Hypothesis refinement
    for $\mathcal{X} \in \Lambda_{\mathsf{T}}$, $(?_i\texttt{: tbl}) \in$ LEAVES($\mathcal{H}$) **do**
      $\mathcal{H}' := \mathcal{H}[?_j^{\mathcal{X}}(?_j : \vec{\tau})/?_i]$
      $W := W \cup \mathcal{H}'$
  return $\perp$

Sketch Completion

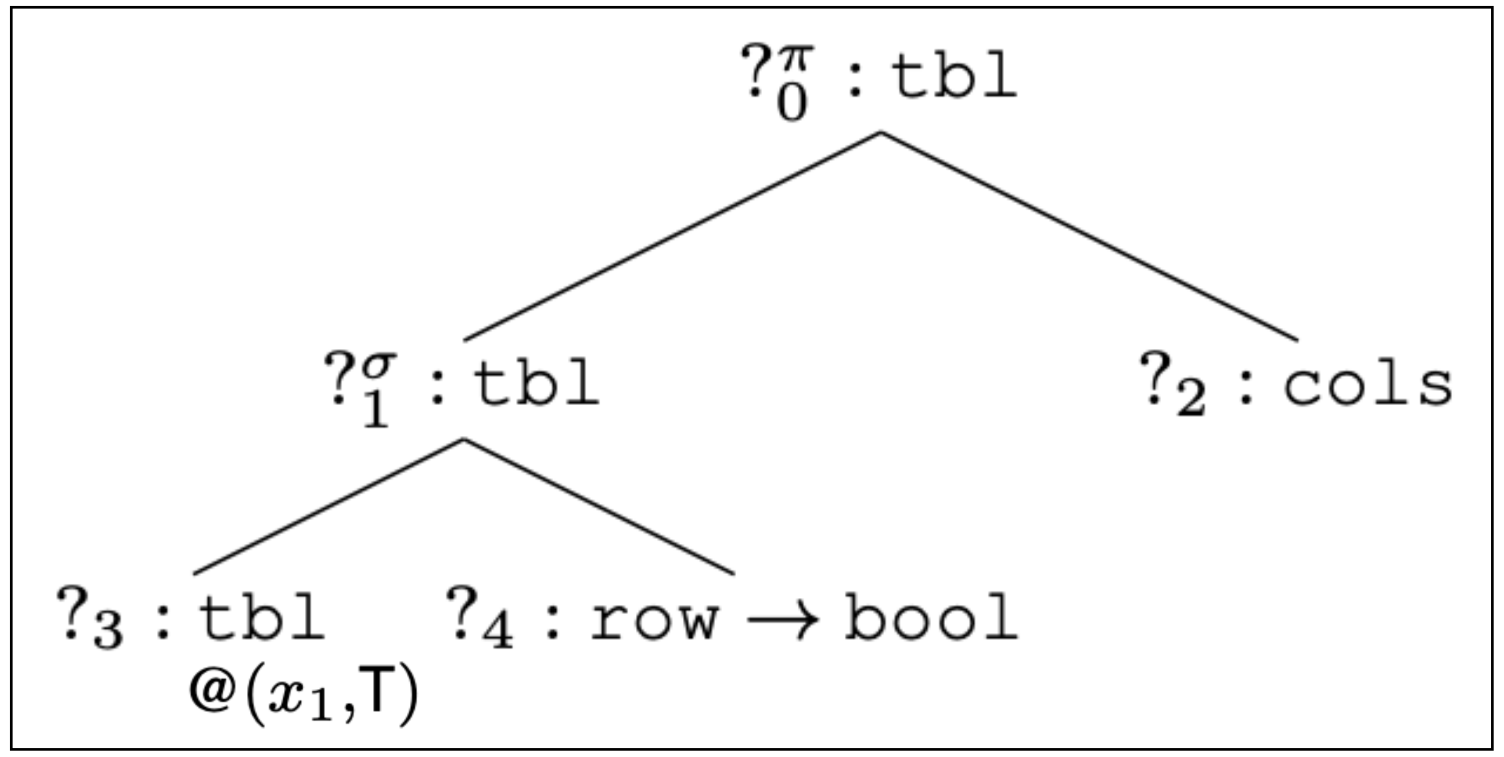
- Given a sketch, fill holes with value transformers

Sketch Completion

- Given a sketch, fill holes with value transformers
- Table-driven type inhabitation
  - Make sure enumerate only (sub-)programs that are well-typed

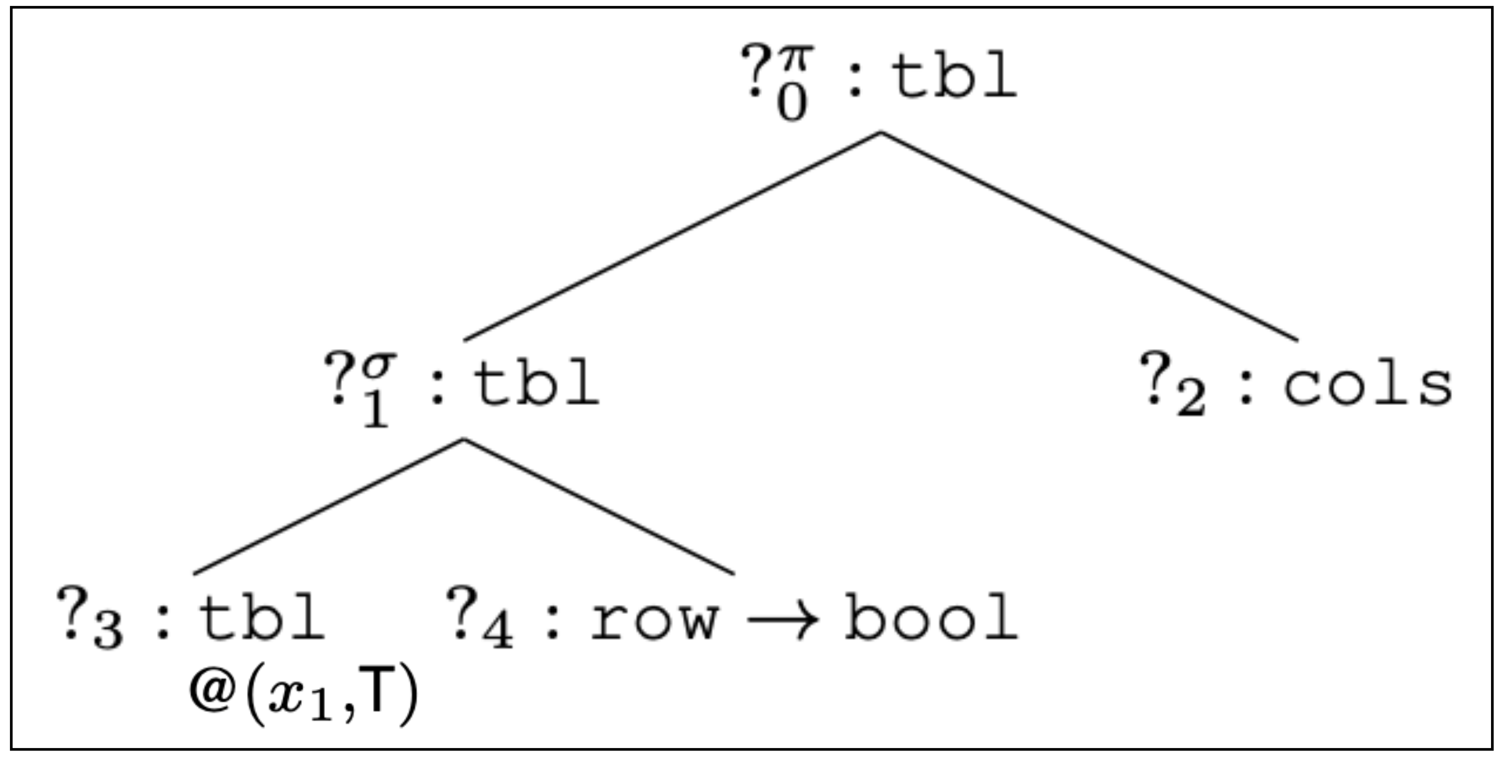
Sketch Completion

- Given a sketch, fill holes with value transformers
- Table-driven type inhabitation
  - Make sure enumerate only (sub-)programs that are well-typed



Sketch Completion

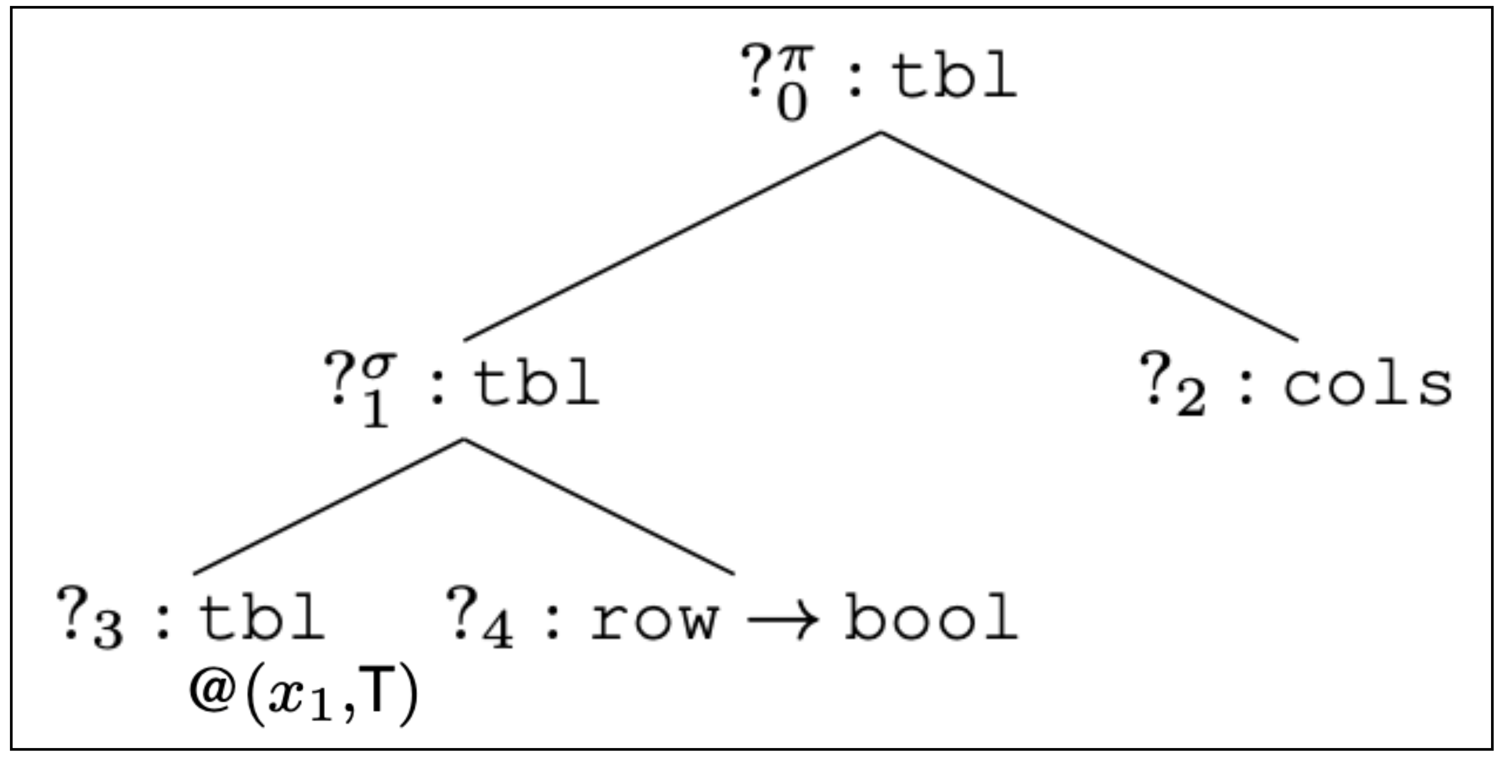
- Given a sketch, fill holes with value transformers
- Table-driven type inhabitation
  - Make sure enumerate only (sub-)programs that are well-typed



- To fill $?_4$, need to know table $?_3$

Sketch Completion

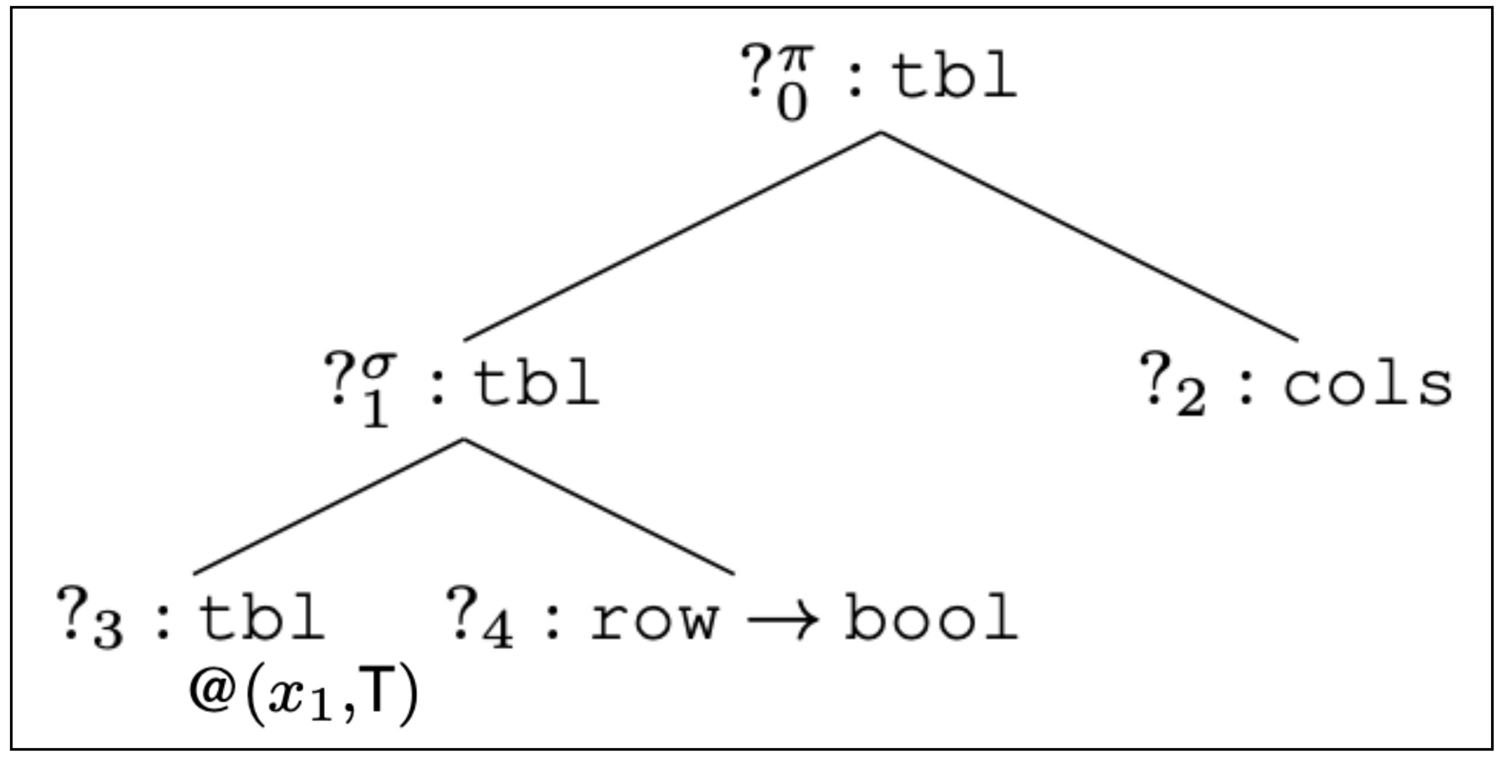
- Given a sketch, fill holes with value transformers
- Table-driven type inhabitation
  - Make sure enumerate only (sub-)programs that are well-typed



- To fill $?_4$, need to know table $?_3$
- To fill $?_2$, need to know intermediate table at $?_1$

Sketch Completion

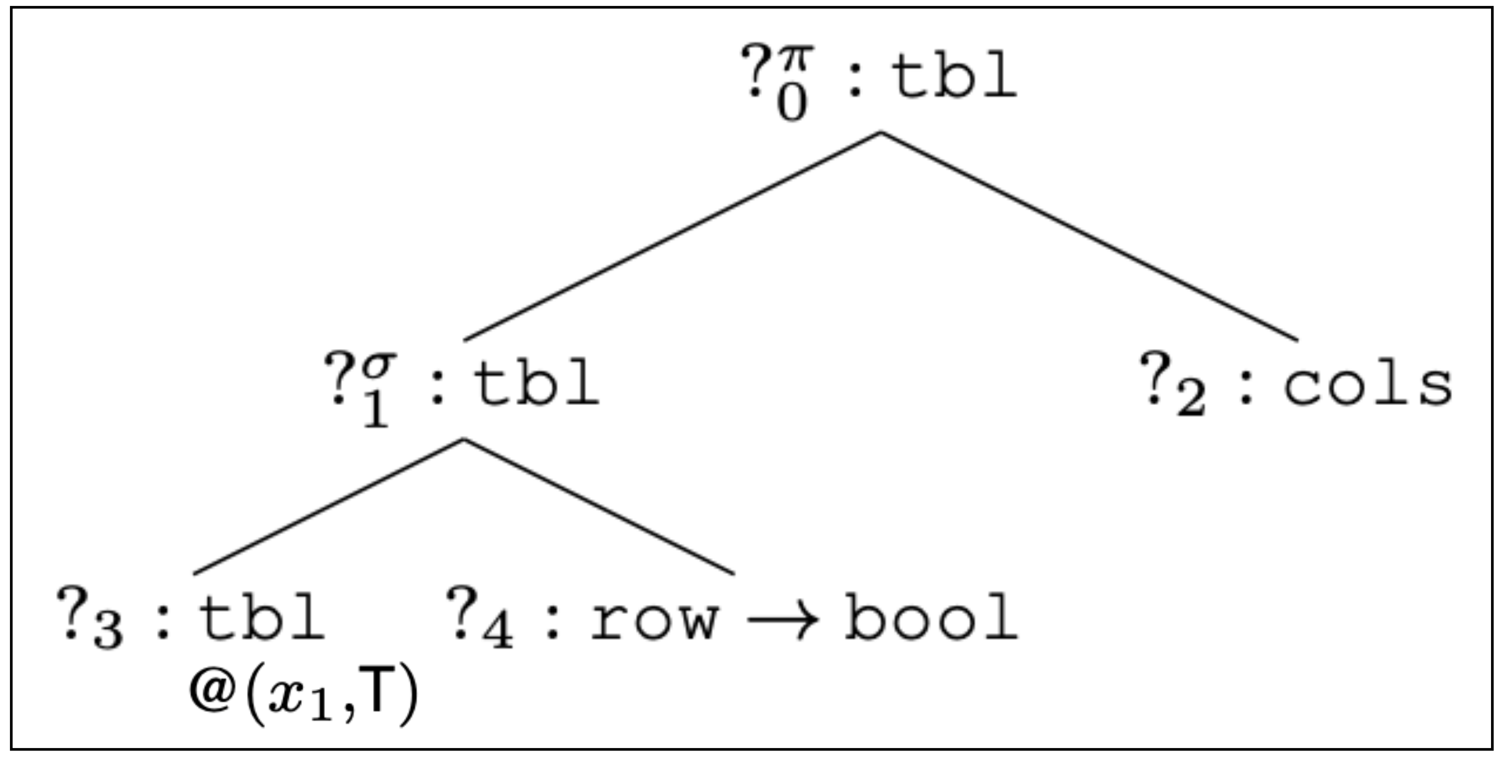
- Given a sketch, fill holes with value transformers
- Table-driven type inhabitation
  - Make sure enumerate only (sub-)programs that are well-typed



- To fill $?_4$, need to know table $?_3$
- To fill $?_2$, need to know intermediate table at $?_1$
- We want: fill $?_4$ first, then $?_2$

Sketch Completion

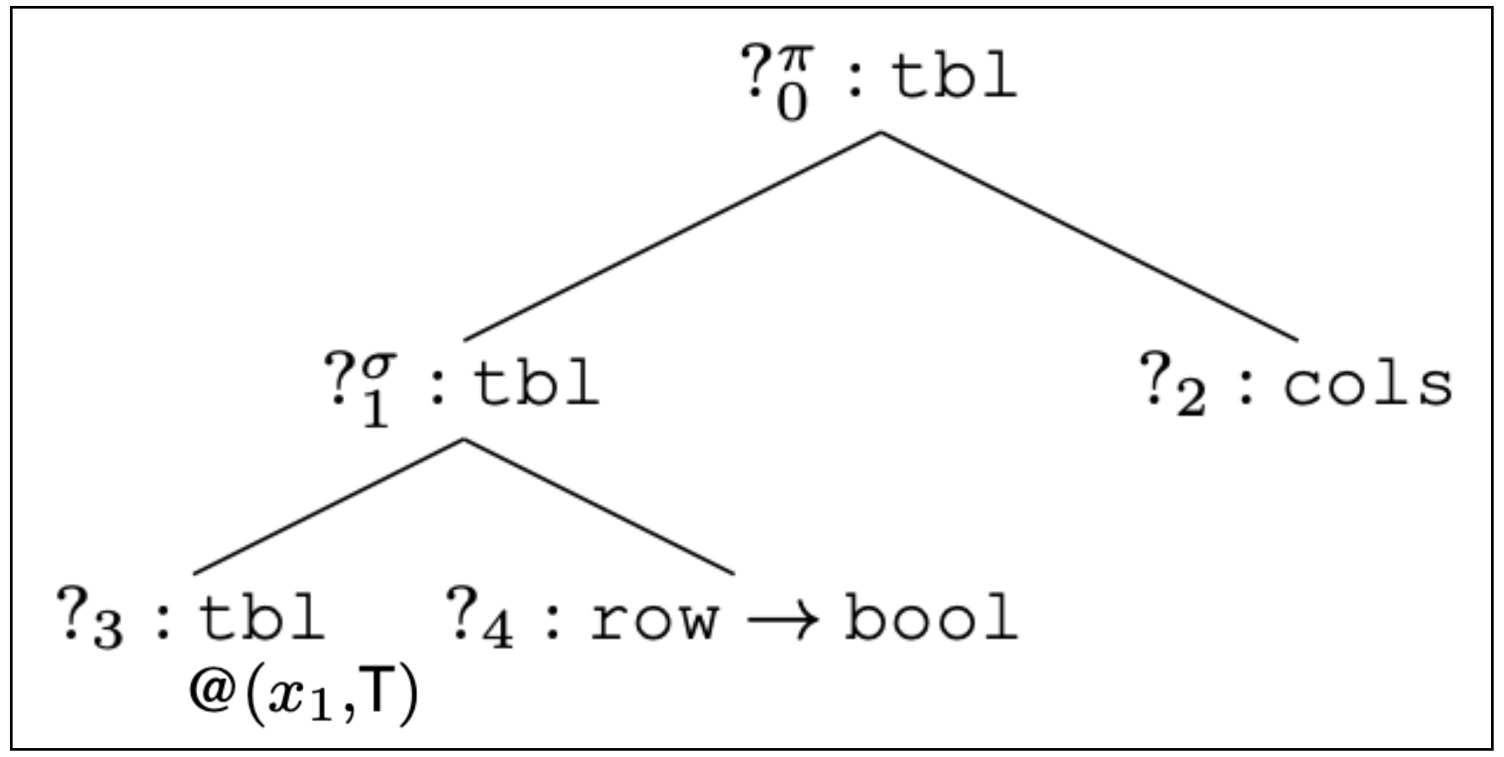
- Given a sketch, fill holes with value transformers
- Table-driven type inhabitation
  - Make sure enumerate only (sub-)programs that are well-typed



- To fill $?_4$, need to know table $?_3$
- To fill $?_2$, need to know intermediate table at $?_1$
- We want: fill $?_4$ first, then $?_2$
- Be "bottom-up" to leverage partial evaluation

Sketch Completion

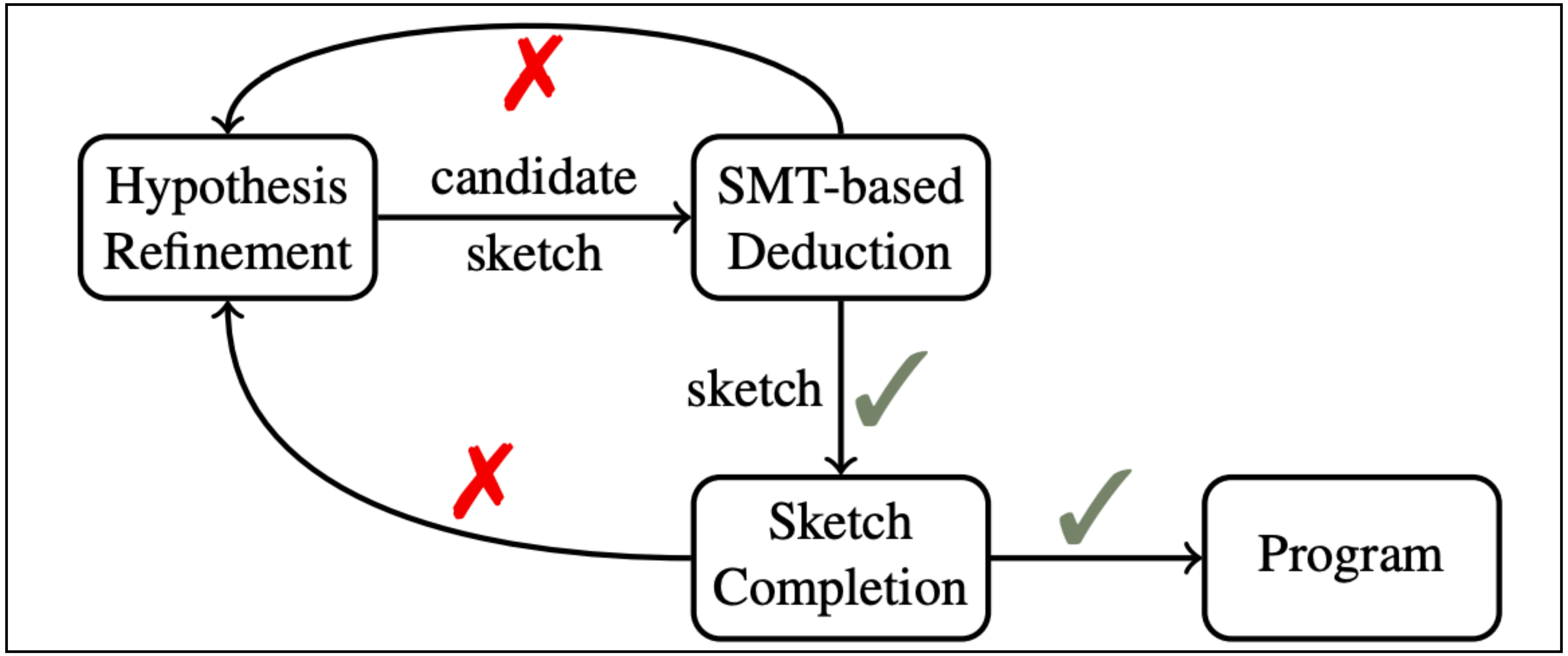
- Given a sketch, fill holes with value transformers
- Table-driven type inhabitation
  - Make sure enumerate only (sub-)programs that are well-typed



- To fill $?_4$, need to know table $?_3$
- To fill $?_2$, need to know intermediate table at $?_1$
- We want: fill $?_4$ first, then $?_2$
- Be "bottom-up" to leverage partial evaluation

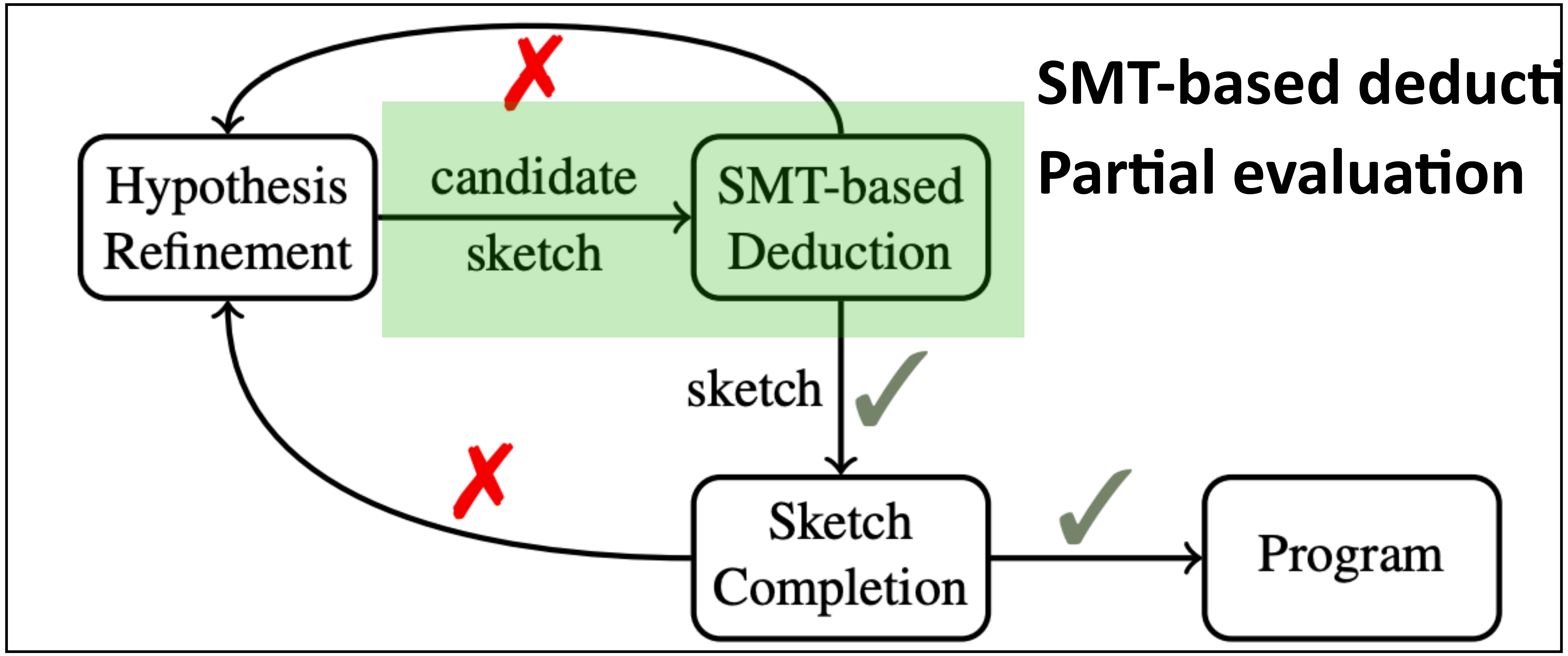
- Skip details

Synthesis Algorithm Recap



- Initial hypothesis is a hole

Synthesis Algorithm Recap

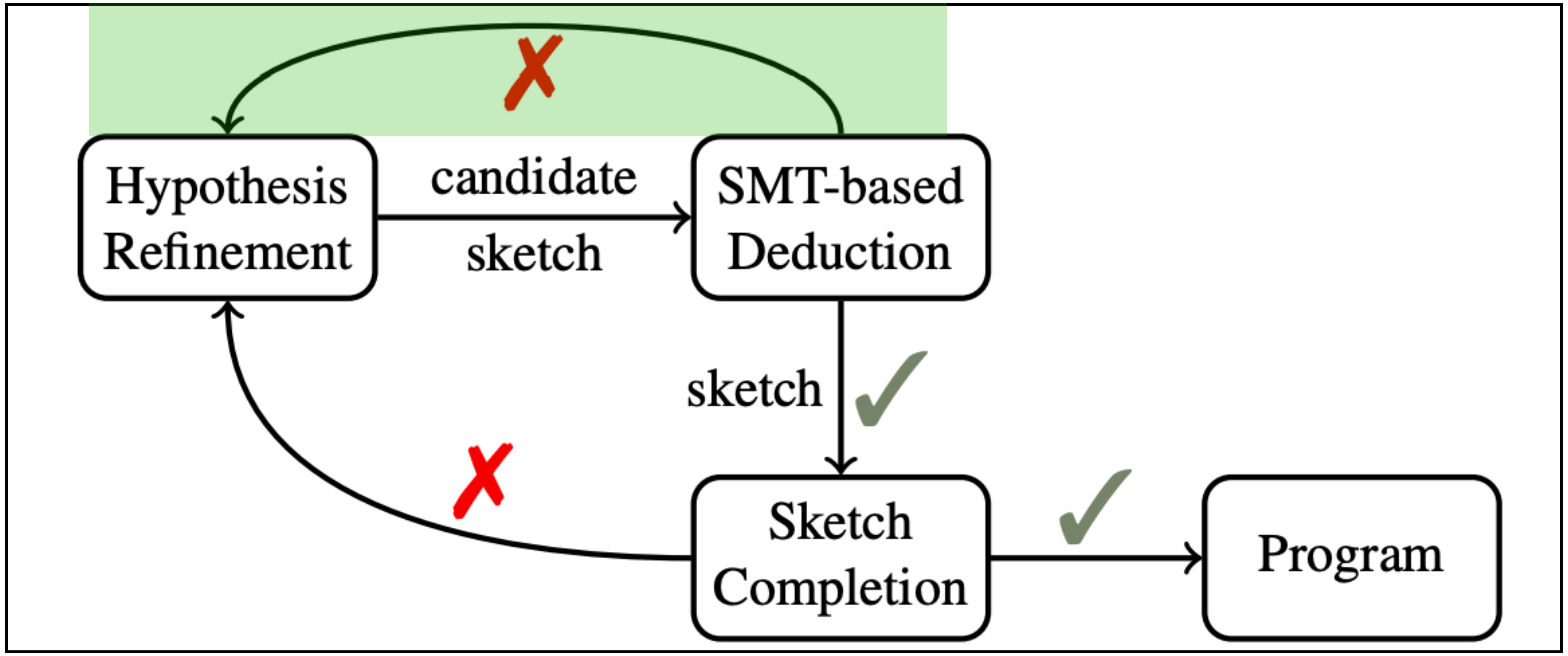


SMT-based deduction

Partial evaluation

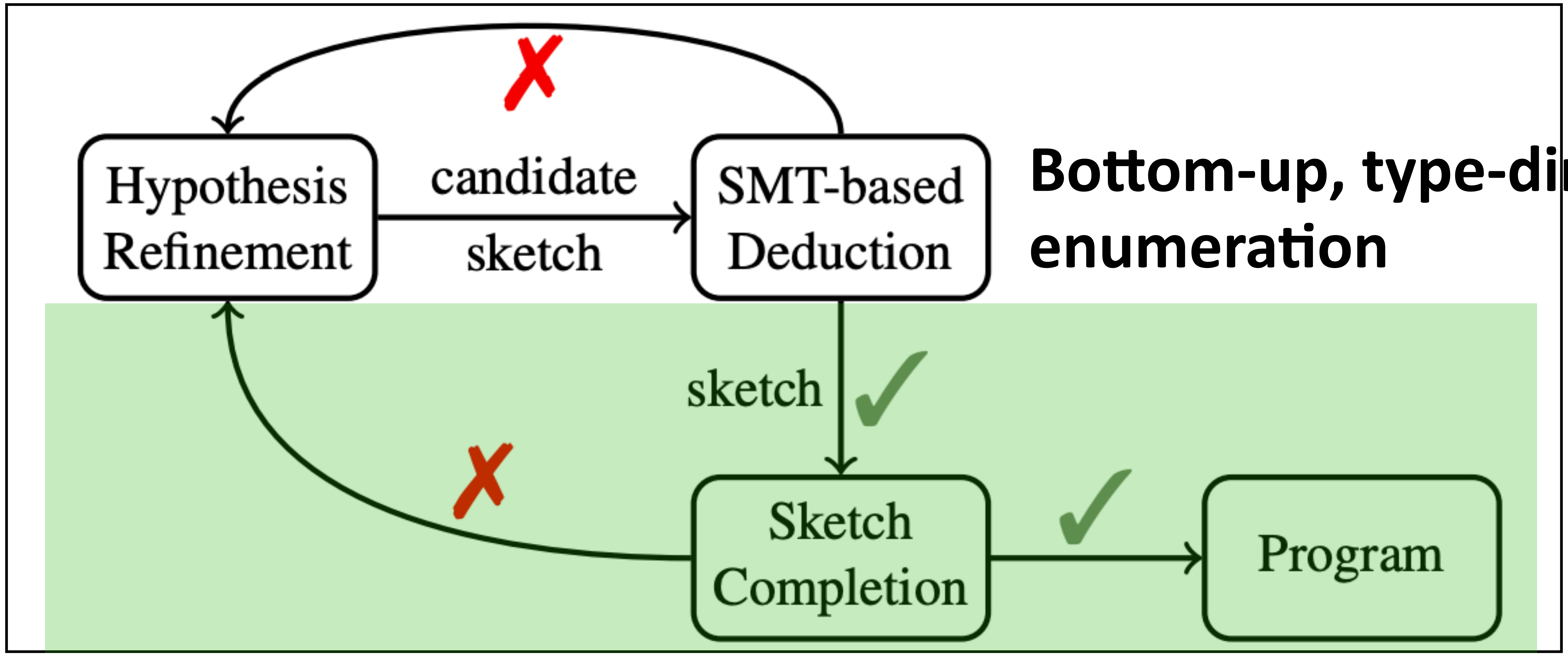
- Initial hypothesis is a hole
- Pruning: consider sketches corresponding to hypothesis

Synthesis Algorithm Recap



- Initial hypothesis is a hole
- Pruning: consider sketches corresponding to hypothesis
  - Can prune: refine hypothesis

Synthesis Algorithm Recap



Bottom-up, type-directed enumeration

- Initial hypothesis is a hole
- Pruning: consider sketches corresponding to hypothesis
  - Can prune: refine hypothesis
  - Can't prune: convert to sketches, complete sketches, if program found, return; otherwise, refine hypothesis

Use N-gram Models for Search Prioritization

- Not a major contribution of this paper: application of standard technique
- Implementation section

> Recall from Section 5 that MORPHEUS uses a cost model for picking the "best" hypothesis from the worklist. Inspired by previous work on code completion [28], we use a cost model based on a statistical analysis of existing code. Specifically, MORPHEUS analyzes existing code snippets that use components from Λ_T and represents each snippet as a 'sentence' where 'words' correspond to components in Λ_T. Given this representation, MORPHEUS uses the 2-gram model in SRILM [34] to assign a score to each hypothesis. Specifically, we train our language model by collecting approximately 15,000 code snippets from Stackoverflow using the search keywords `tidyr` and `dplyr`. For each code snippet, we ignore its control flow and represent it using a "sentence" where each "word" corresponds to an API call. Based on this training data, the hypotheses in the worklist W from Algorithm 1 are then ordered using the scores obtained from the n-gram model.

How To Present A Research Paper?

- What’s the problem? Why is it important?
- Why is the problem challenging?
- How does the paper solve the problem? What’s the key idea?
- Explain technique in more detail
- **Evaluation**
- Related work

Evaluation

- Research questions
  - How well does Morpheus work on real-world table transformation tasks?

Evaluation

- Research questions
  - How well does Morpheus work on real-world table transformation tasks?
  - Ablation study
    - How much does SMT-based deduction help?
    - How much does partial evaluation help?
    - How much does n-gram model help?

Evaluation

- Research questions
  - How well does Morpheus work on real-world table transformation tasks?
  - Ablation study
    - How much does SMT-based deduction help?
    - How much does partial evaluation help?
    - How much does n-gram model help?
  - Comparison against baselines
    - Comparison against λ^2 [1]
    - Comparison against SQLSynthesizer [2]

[1] Synthesizing data structure transformations from input-output examples. Feser et al. 2015.
[2] Automatically synthesizing sql queries from input-output examples. Zhang et al. 2013.

Evaluation

- Benchmarks
  - 80 data preparation tasks in R from StackOverflow
  - 20 components from `tidyr` and `dplyr` packages

Evaluation

- Research questions
  - **How well does Morpheus work on real-world table transformation tasks?**
  - Ablation study
    - How much does SMT-based deduction help?
    - How much does partial evaluation help?
    - How much does n-gram model help?
  - Comparison against baselines
    - Comparison against λ^2 [1]
    - Comparison against SQLSynthesizer [2]

[1] Synthesizing data structure transformations from input-output examples. Feser et al. 2015.

[2] Automatically synthesizing sql queries from input-output examples. Zhang et al. 2013.

Evaluate Morpheus

Category	Description	#	Spec 2 #Solved	Spec 2 Time
C1	*Reshaping* dataframes from either "long" to "wide" or "wide" to "long"	4	4	6.70
C2	*Arithmetic computations* that produce values not present in the input tables	7	7	0.59
C3	Combination of *reshaping* and *string manipulation* of cell contents	34	34	1.63
C4	*Reshaping* and *arithmetic computations*	14	12	15.35
C5	Combination of *arithmetic computations* and *consolidation* of information from multiple tables into a single table	11	11	3.17
C6	*Arithmetic computations* and *string manipulation* tasks	2	2	3.03
C7	*Reshaping* and *consolidation* tasks	1	1	130.92
C8	Combination of *reshaping*, *arithmetic computations* and *string manipulation*	6	6	38.42
C9	Combination of *reshaping*, *arithmetic computations* and *consolidation*	1	1	97.3
Total		80	78 (97.5%)	3.59

Evaluate Morpheus

Category	Description	#	Spec 2 #Solved	Spec 2 Time
C1	*Reshaping* dataframes from either “long” to “wide” or “wide” to “long”	4	4	6.70
C2	*Arithmetic computations* that produce values not present in the input tables	7	7	0.59
C3	Combination of *reshaping* and *string manipulation* of cell contents	34	34	1.63
C4	*Reshaping* and *arithmetic computations*	14	12	15.35
C5	Combination of *arithmetic computations* and *consolidation* of information from multiple tables into a single table	11	11	3.17
C6	*Arithmetic computations* and *string manipulation* tasks	2		
C7	*Reshaping* and *consolidation* tasks	1		
C8	Combination of *reshaping*, *arithmetic computations* and *string manipulation*	6		
C9	Combination of *reshaping*, *arithmetic computations* and *consolidation*	1		
Total		80	78 (97.5%)	3.59

Take-away: Morpheus can solve almost all benchmarks within seconds

Evaluation

- Research questions
  - How well does Morpheus work on real-world table transformation tasks?
  - **Ablation study**
    - How much does SMT-based deduction help?
    - How much does partial evaluation help?
    - How much does n-gram model help?
  - Comparison against baselines
    - Comparison against λ^2 [1]
    - Comparison against SQLSynthesizer [2]

[1] Synthesizing data structure transformations from input-output examples. Feser et al. 2015.

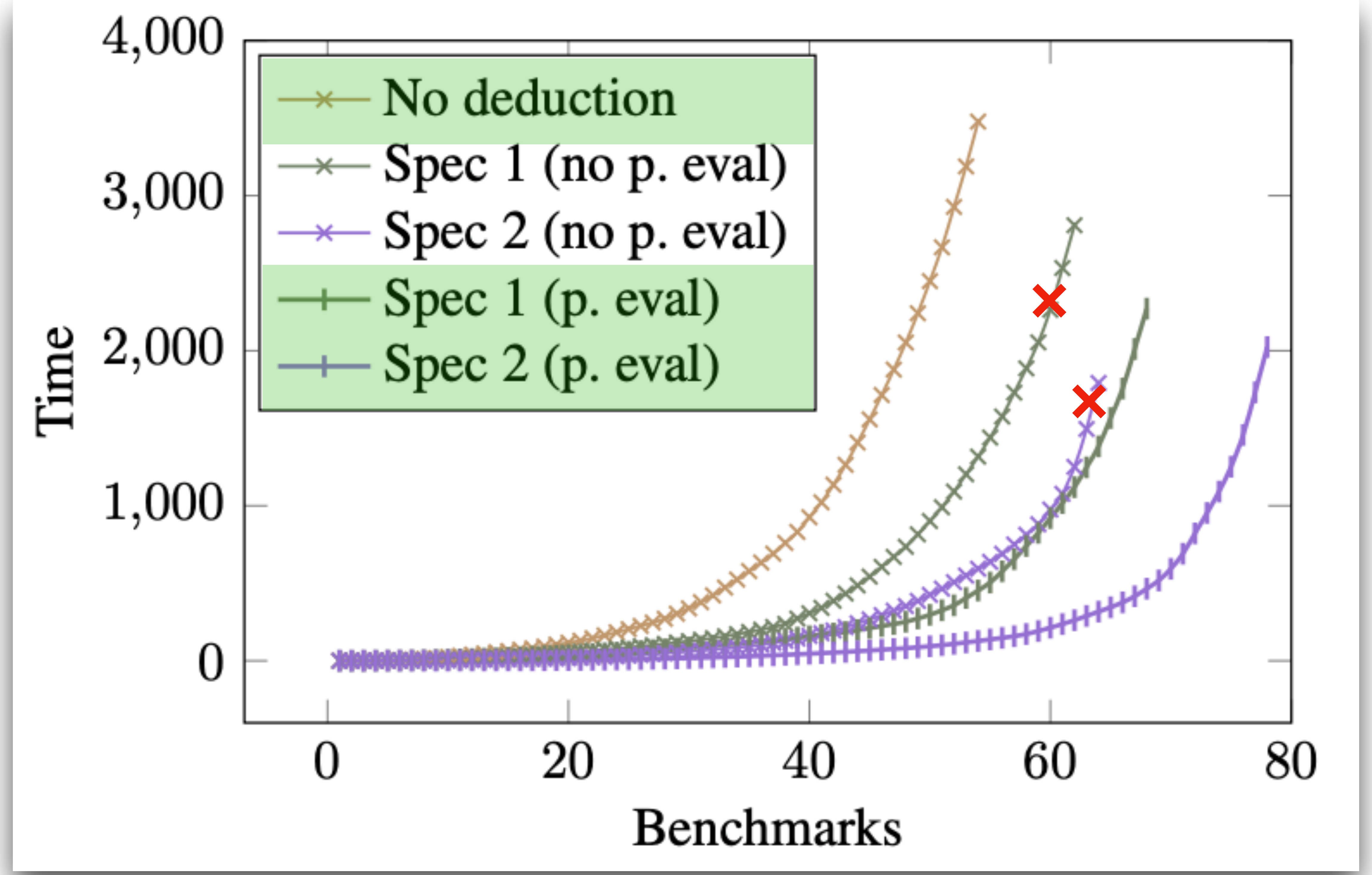
[2] Automatically synthesizing sql queries from input-output examples. Zhang et al. 2013.

Evaluate Usefulness of SMT-based Deduction

- Evaluate impact of different specifications on performance
  - No spec
  - Spec 1: less precise
  - Spec 2: more precise

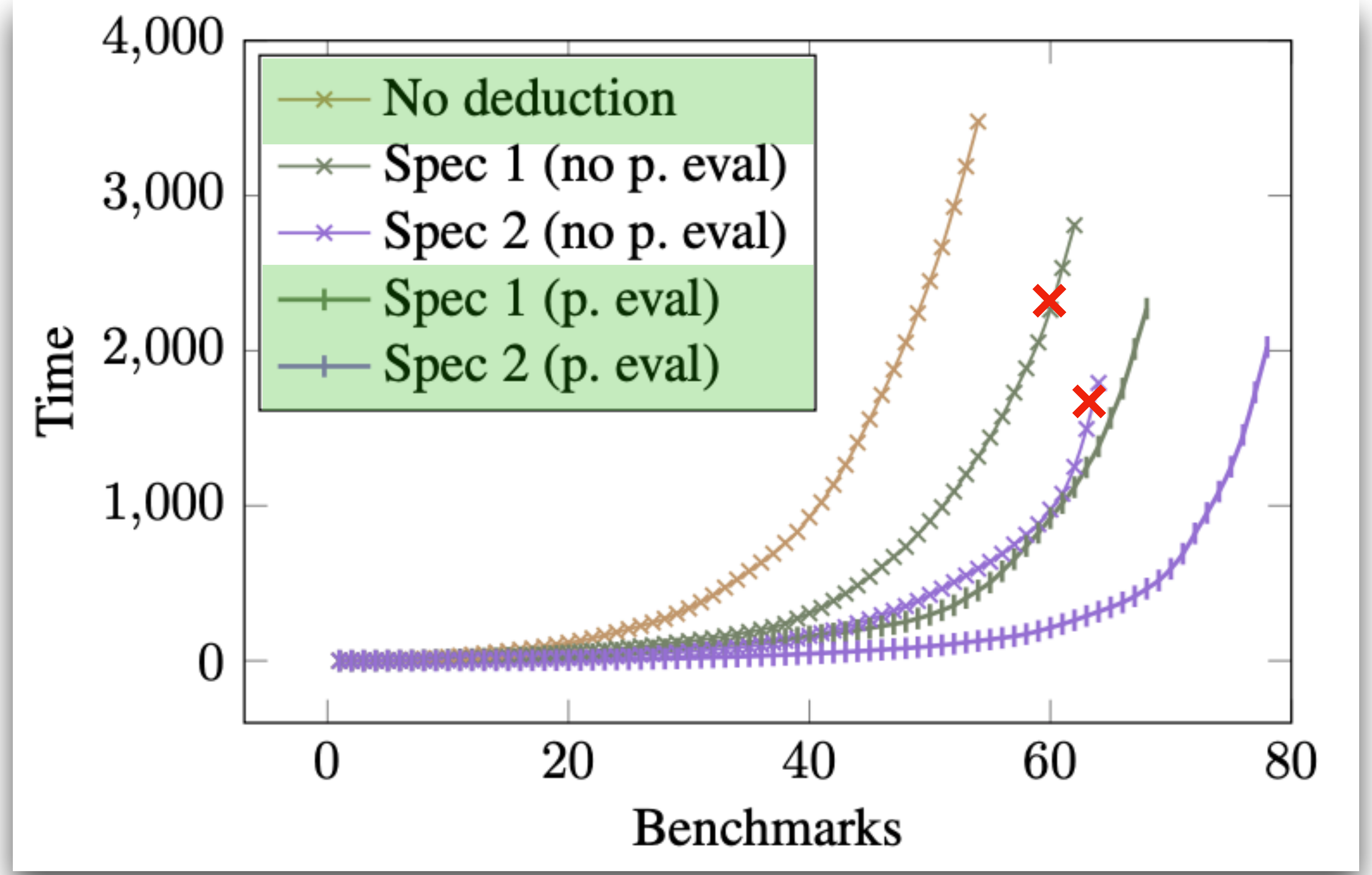
Evaluate Usefulness of SMT-based Deduction

- Evaluate impact of different specifications on performance
  - No spec
  - Spec 1: less precise
  - Spec 2: more precise



Evaluate Usefulness of SMT-based Deduction

- Evaluate impact of different specifications on performance
  - No spec
  - Spec 1: less precise
  - Spec 2: more precise



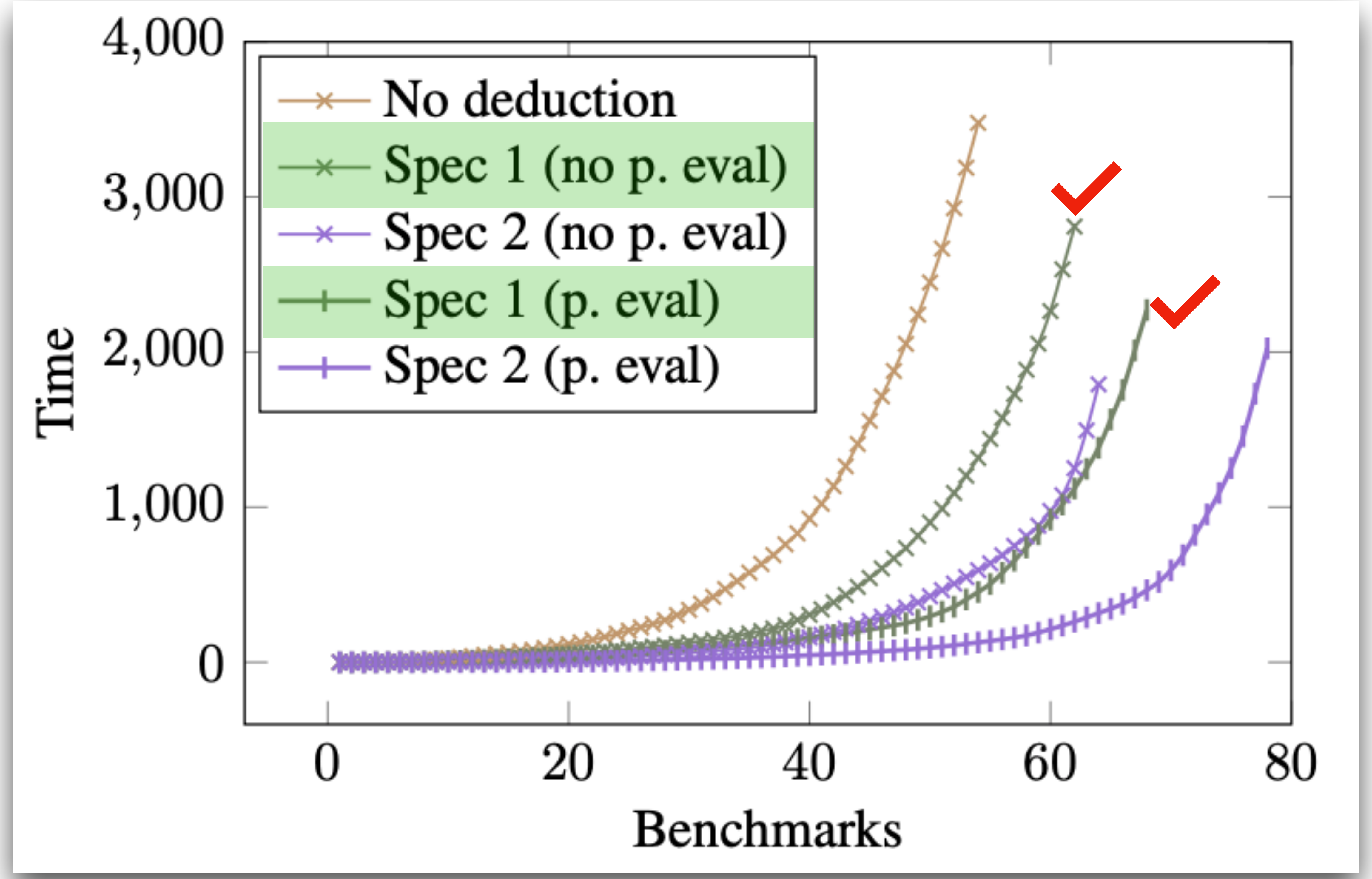
Take-away: more precise, better performance

Evaluate Usefulness of Partial Evaluation

- Evaluate impact of partial evaluation
  - Spec 1: w/ and w/o PE
  - Spec 2: w/ and w/o PE

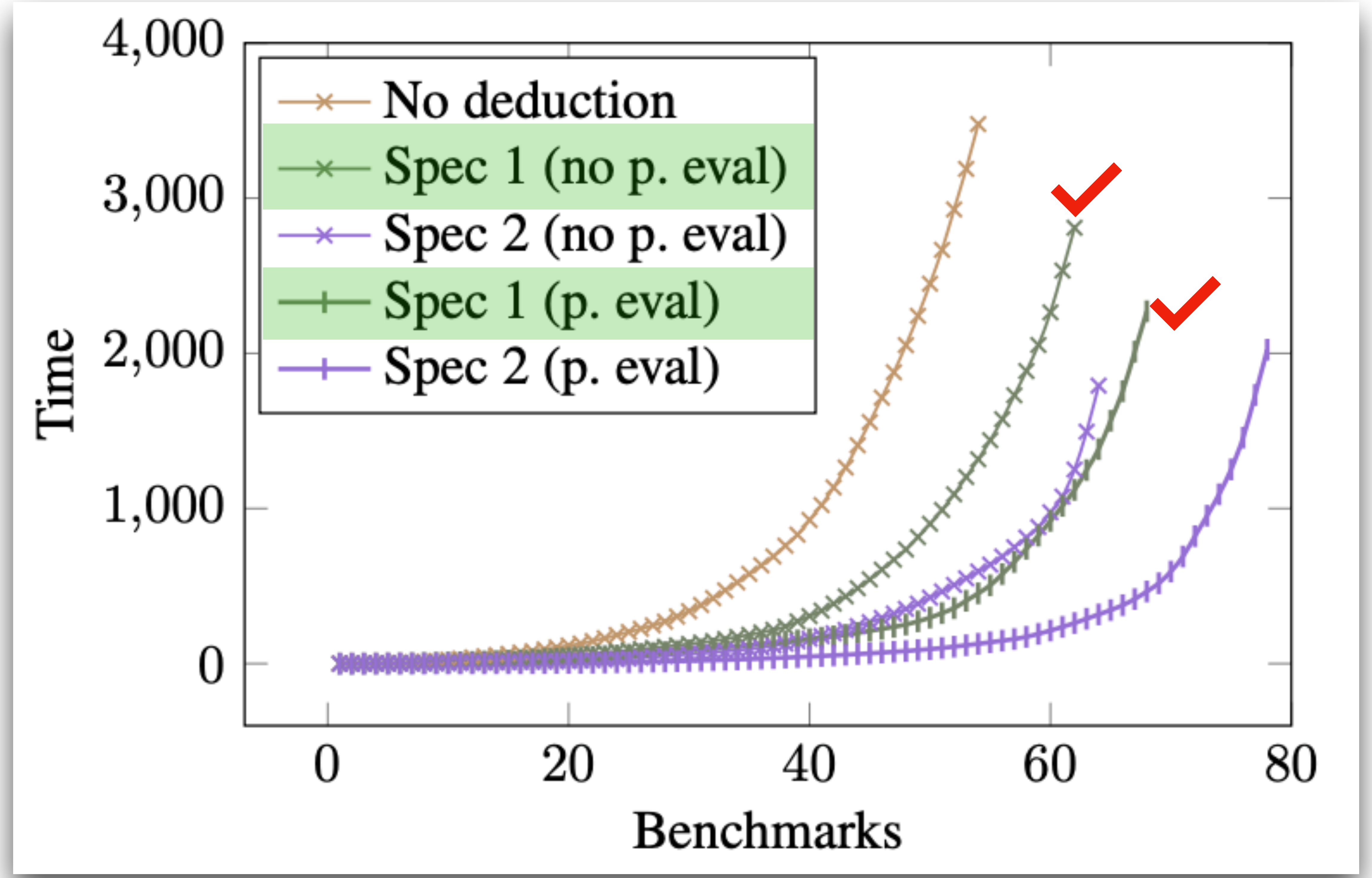
Evaluate Usefulness of Partial Evaluation

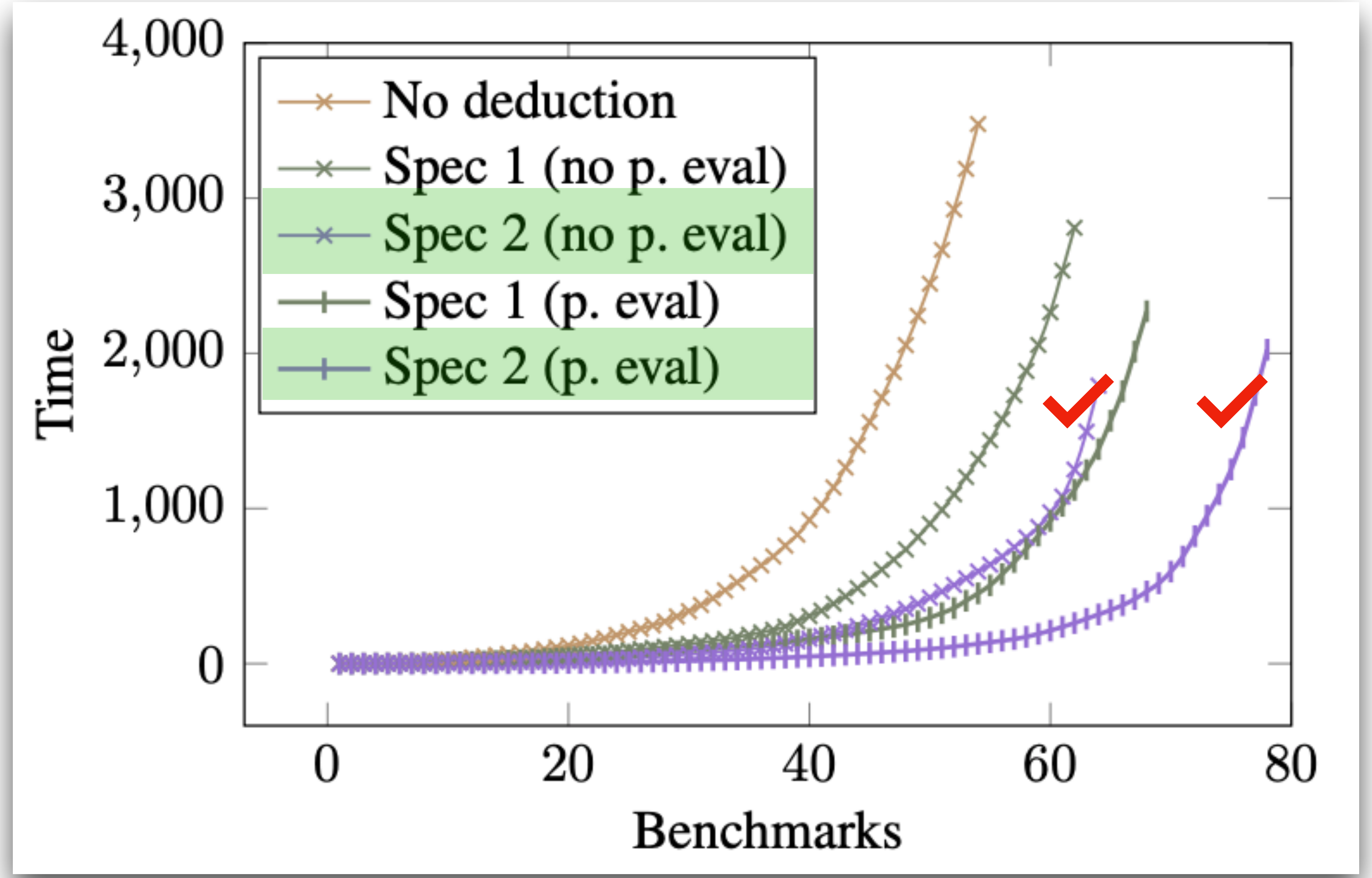
- Evaluate impact of partial evaluation
  - Spec 1: w/ and w/o PE
  - Spec 2: w/ and w/o PE



Evaluate Usefulness of Partial Evaluation

- Evaluate impact of partial evaluation
  - Spec 1: w/ and w/o PE
  - Spec 2: w/ and w/o PE

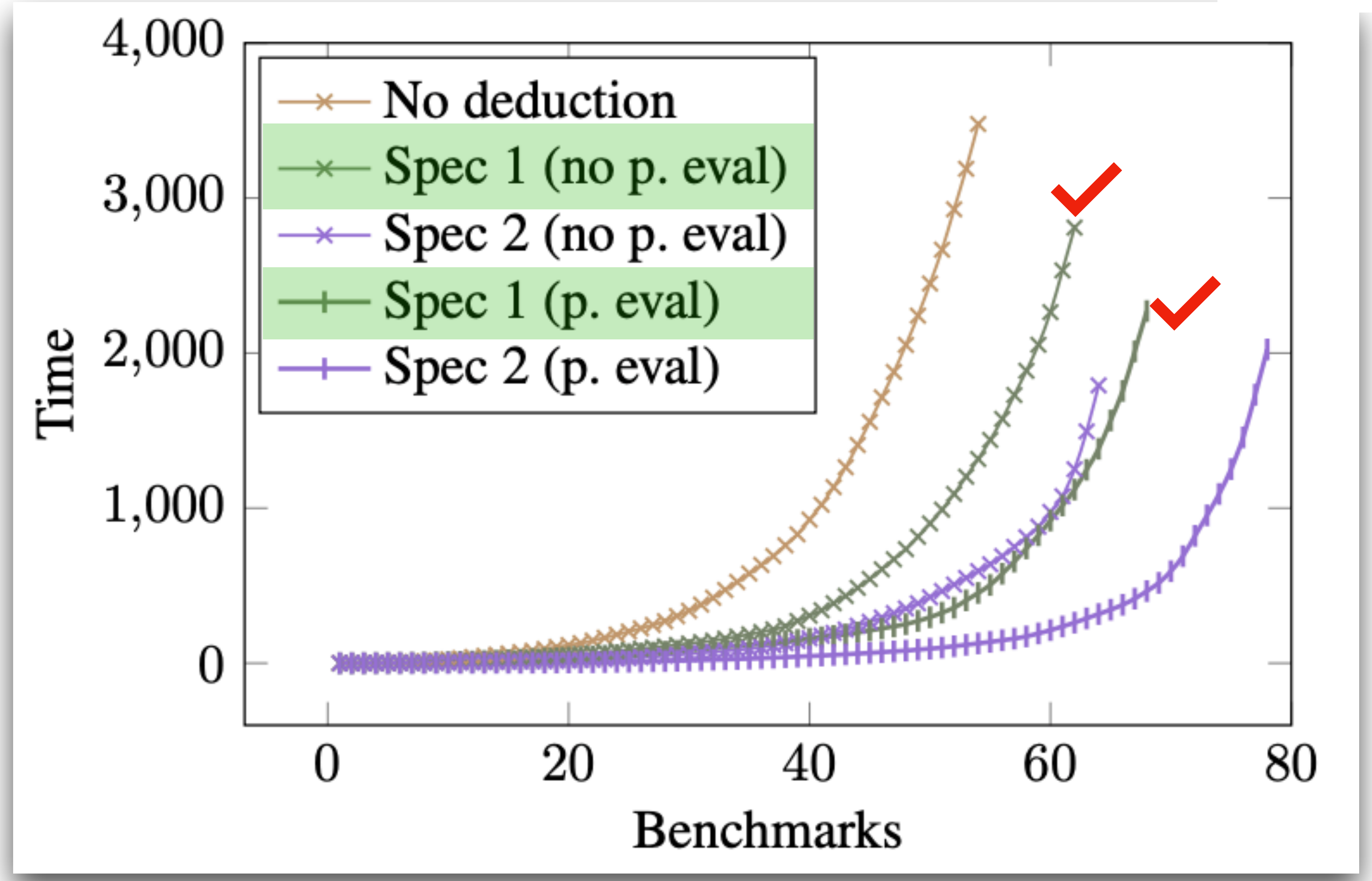


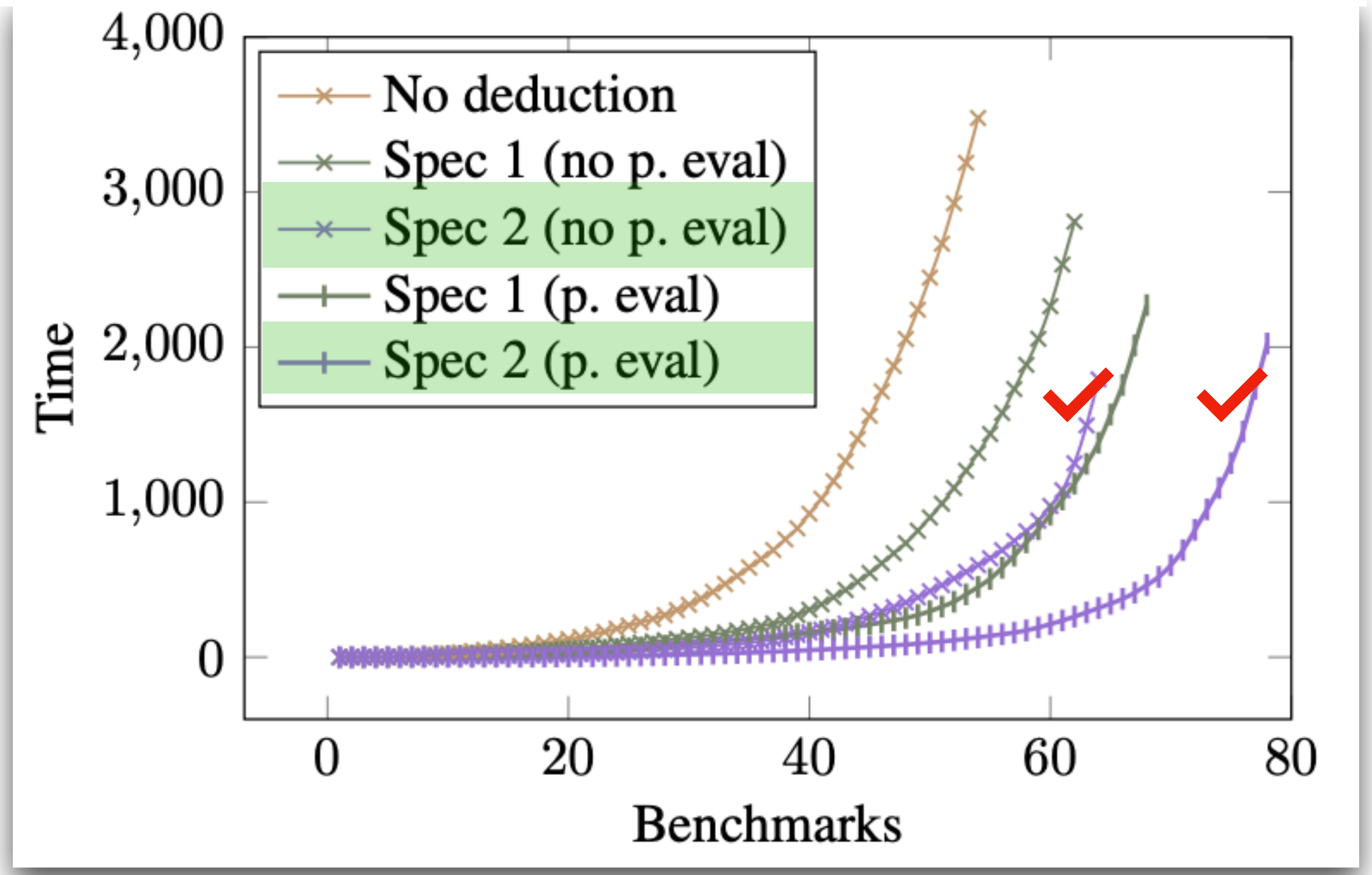


Evaluate Usefulness of Partial Evaluation

- Evaluate impact of partial evaluation
  - Spec 1: w/ and w/o PE
  - Spec 2: w/ and w/o PE

Take-away: PE helps speed up search



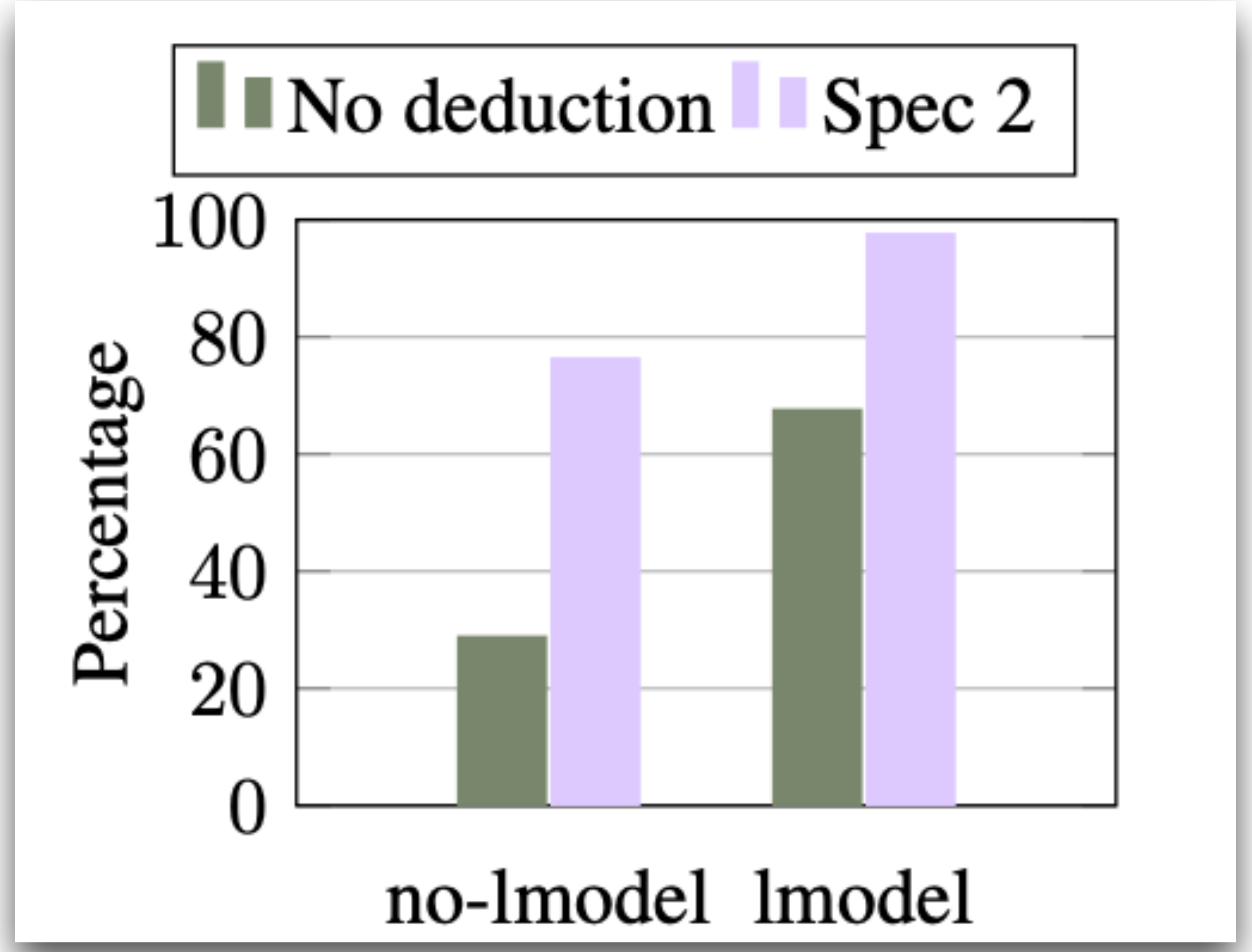


Evaluate Usefulness of N-gram Model

- Evaluate impact of n-gram model
  - No deduction, w/ and w/o n-gram model
  - Spec 2, w/ and w/o n-gram model

Evaluate Usefulness of N-gram Model

- Evaluate impact of n-gram model
  - No deduction, w/ and w/o n-gram model
  - Spec 2, w/ and w/o n-gram model



Take-away: n-gram model helps speed up search

Evaluation

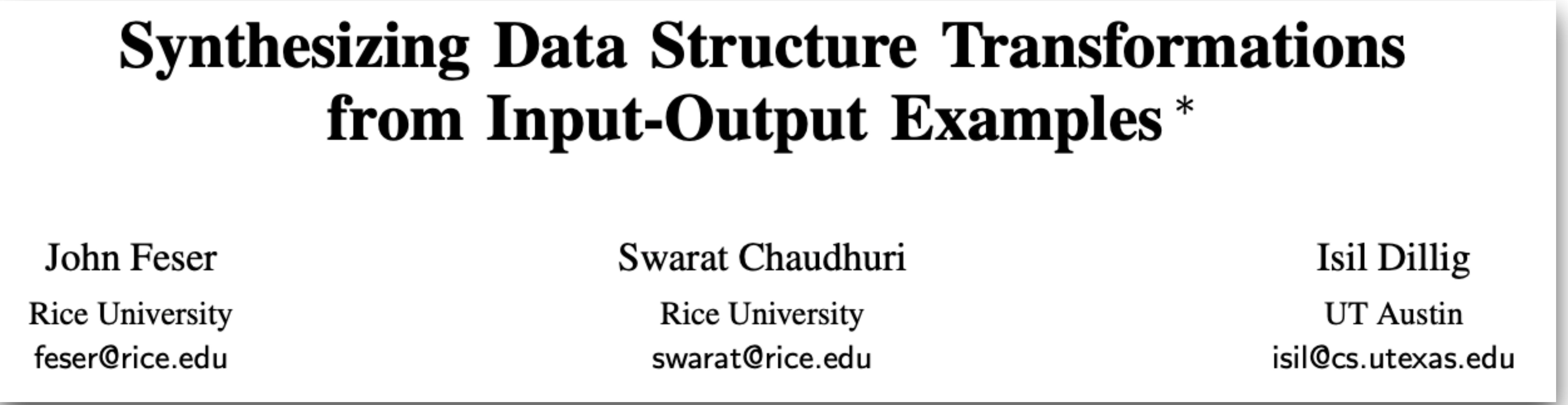
- Research questions
  - How well does Morpheus work on real-world table transformation tasks?
  - Ablation study
    - How much does SMT-based deduction help?
    - How much does partial evaluation help?
    - How much does n-gram model help?
  - **Comparison against baselines**
    - Comparison against λ^2 [1]
    - Comparison against SQLSynthesizer [2]

[1] Synthesizing data structure transformations from input-output examples. Feser et al. 2015.

[2] Automatically synthesizing sql queries from input-output examples. Zhang et al. 2013.

Morpheus vs. λ^2

- λ^2 solves 0 out of 80 benchmarks
  - Because λ^2 uses a DSL that's not tailored towards table transformations in R
  - **Take-away: having the right DSL (abstraction) is very important for synthesis!**



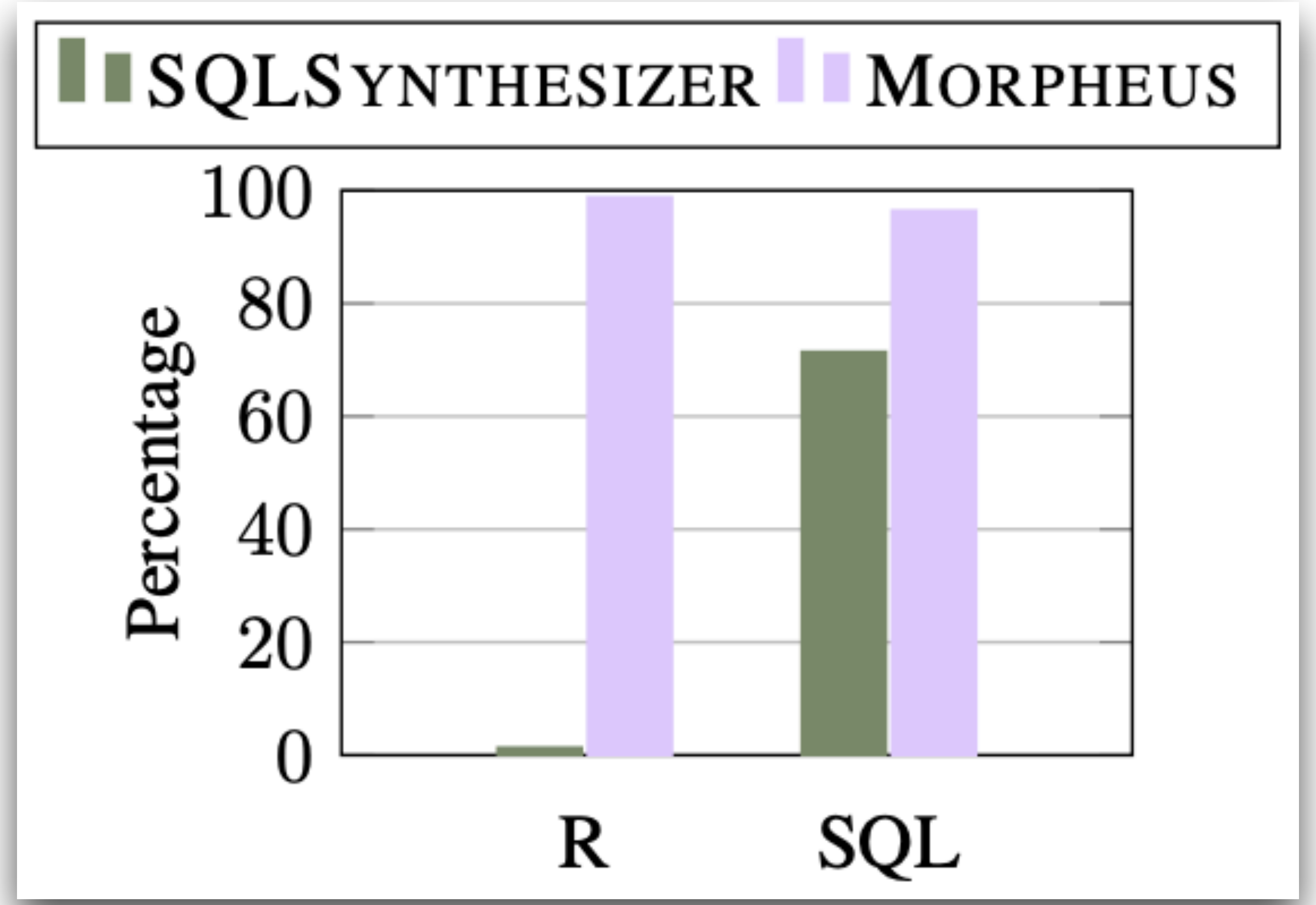
Synthesizing Data Structure Transformations from Input-Output Examples *

John Feser
Rice University
feser@rice.edu

Swarat Chaudhuri
Rice University
swarat@rice.edu

Isil Dillig
UT Austin
isil@cs.utexas.edu

Morpheus vs. SQLSynthesizer



- On 80 R benchmarks, 1 (SQLSynthesizer) vs. 78 (Morpheus)
- On 28 SQLSynthesizer benchmarks, 20 (SQLSynthesizer) vs. 27 (Morpheus)
- **Morpheus technique is better than prior techniques**

Summary

- What's the problem? Why is it important?
  - High-level, use examples
- Why is the problem challenging?
  - High-level, use examples
- How does the paper solve the problem? What's the key idea?
  - One single key idea
  - More detail, still relatively high-level, use examples
- Explain technique in more detail
  - Great detail, organized, use examples
- Evaluation
  - Summarize results and take-aways