Advanced Data Management (CSCI 490/680)

Data Transformation

Dr. David Koop

Wrangler

- Data cleaning takes a lot of **time** and **human effort**
- "Tedium is the message"
- Repeating this process on multiple data sets is even worse!
- Solution:
  - interactive interface (mixed-initiative)
  - transformation language with natural language "translations"
  - suggestions + "programming by demonstration"

Potter's Wheel: Example

		Stewart,Bob
Anna	Davis	
		Dole,Jerry
Joan	Marsh	

Format
'(.*), (.*)' to '\2 \1'

		Bob Stewart
Anna	Davis	
		Jerry Dole
Joan	Marsh	

Split at ' '

		Bob	Stewart
Anna	Davis		
		Jerry	Dole
Joan	Marsh		

2 Merges

Bob	Stewart
Anna	Davis
Jerry	Dole
Joan	Marsh

[V. Raman and J. Hellerstein, 2001]

Potter's Wheel: Transforms

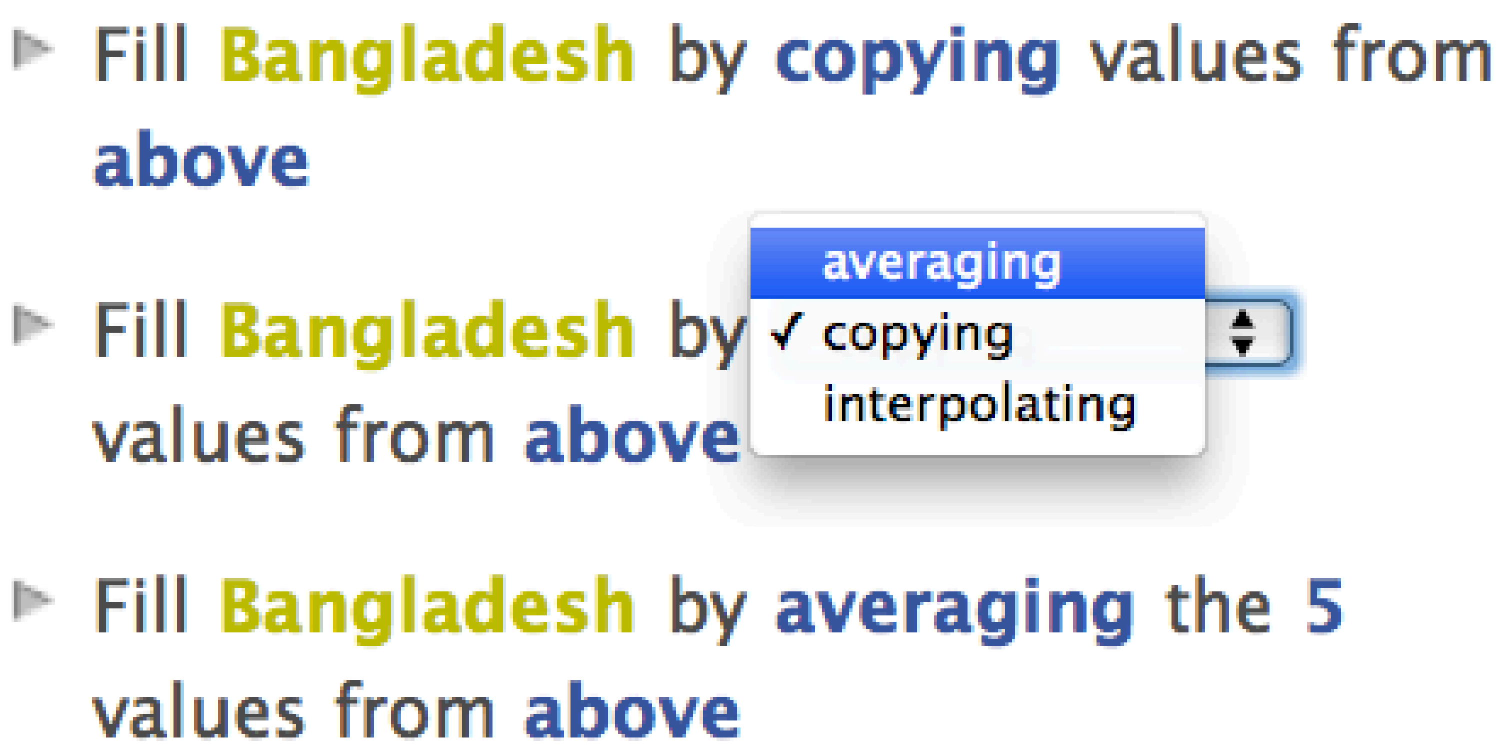
Transform	Definition		
Format	$\phi(R, i, f)$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, f(a_i)) \mid (a_1, \ldots, a_n) \in R\}$
Add	$\alpha(R, x)$	$=$	$\{(a_1, \ldots, a_n, x) \mid (a_1, \ldots, a_n) \in R\}$
Drop	$\pi(R, i)$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n) \mid (a_1, \ldots, a_n) \in R\}$
Copy	$\kappa((a_1, \ldots, a_n), i)$	$=$	$\{(a_1, \ldots, a_n, a_i) \mid (a_1, \ldots, a_n) \in R\}$
Merge	$\mu((a_1, \ldots, a_n), i, j, \text{glue})$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n, a_i \oplus \text{glue} \oplus a_j) \mid (a_1, \ldots, a_n) \in R\}$
Split	$\omega((a_1, \ldots, a_n), i, \text{splitter})$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, \text{left}(a_i, \text{splitter}), \text{right}(a_i, \text{splitter})) \mid (a_1, \ldots, a_n) \in R\}$
Divide	$\delta((a_1, \ldots, a_n), i, \text{pred})$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, a_i, \text{null}) \mid (a_1, \ldots, a_n) \in R \wedge \text{pred}(a_i)\} \cup$ $\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, \text{null}, a_i) \mid (a_1, \ldots, a_n) \in R \wedge \neg\text{pred}(a_i)\}$
Fold	$\lambda(R, i_1, i_2, \ldots i_k)$	$=$	$\{(a_1, \ldots, a_{i_1-1}, a_{i_1+1}, \ldots, a_{i_2-1}, a_{i_2+1}, \ldots, a_{i_k-1}, a_{i_k+1}, \ldots, a_n, a_{i_l}) \mid$ $(a_1, \ldots, a_n) \in R \wedge 1 \leq l \leq k\}$
Select	$\sigma(R, \text{pred})$	$=$	$\{(a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in R \wedge \text{pred}((a_1, \ldots, a_n))\}$

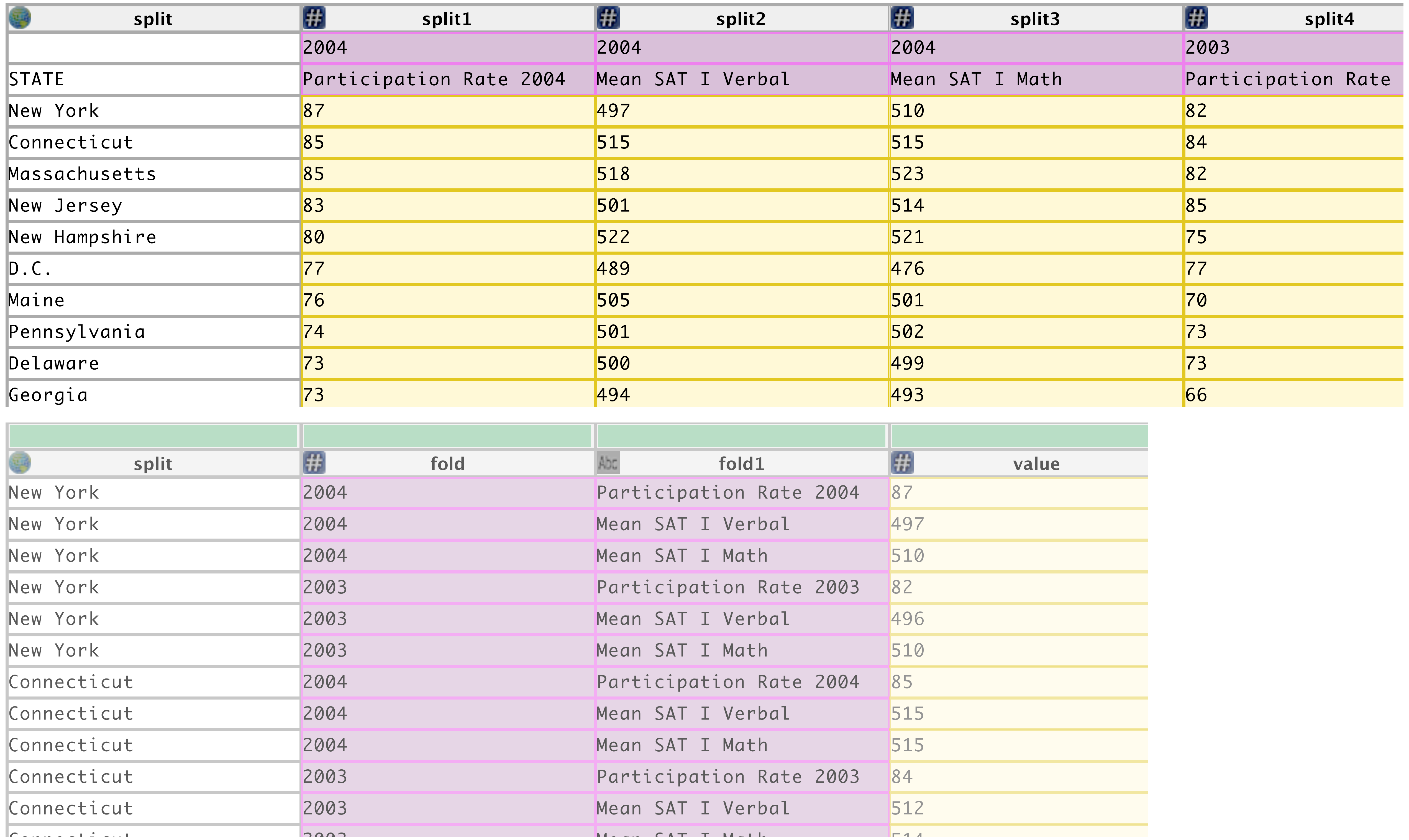
Notation: R is a relation with n columns. i, j are column indices and a_i represents the value of a column in a row. x and glue are values. f is a function mapping values to values. $x \oplus y$ concatenates x and y. splitter is a position in a string or a regular expression, left$(x, \text{splitter})$ is the left part of x after splitting by splitter. pred is a function returning a boolean.

[V. Raman and J. Hellerstein, 2001]

Interface

- Automated Transformation Suggestions
- Editable Natural Language Explanations





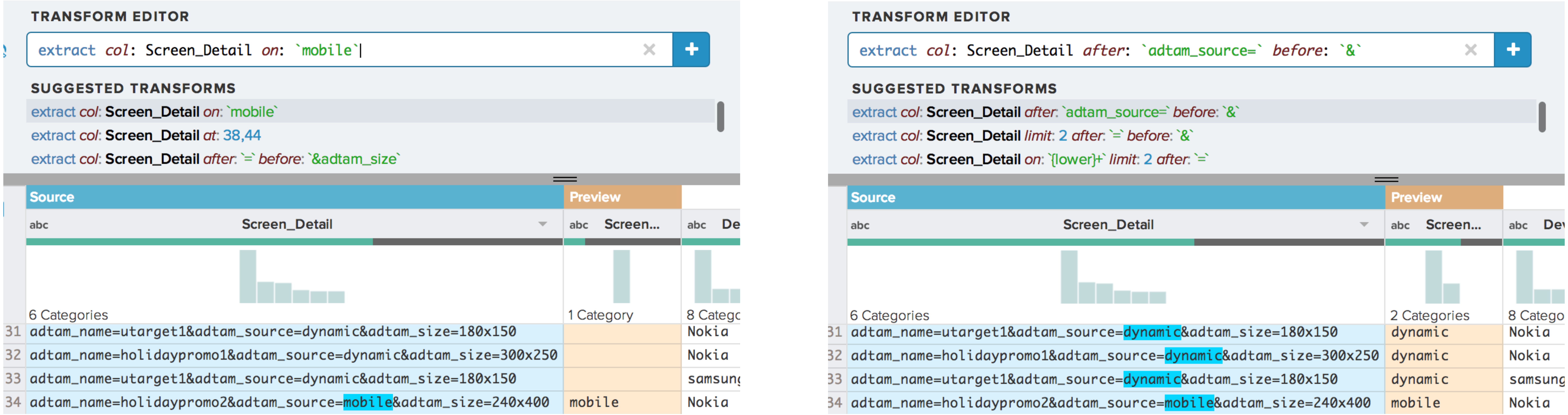
split	split1	split2	split3	split4
	2004	2004	2004	2003
STATE	Participation Rate 2004	Mean SAT I Verbal	Mean SAT I Math	Participation Rate
New York	87	497	510	82
Connecticut	85	515	515	84
Massachusetts	85	518	523	82
New Jersey	83	501	514	85
New Hampshire	80	522	521	75
D.C.	77	489	476	77
Maine	76	505	501	70
Pennsylvania	74	501	502	73
Delaware	73	500	499	73
Georgia	73	494	493	66

split	fold	fold1	value
New York	2004	Participation Rate 2004	87
New York	2004	Mean SAT I Verbal	497
New York	2004	Mean SAT I Math	510
New York	2003	Participation Rate 2003	82
New York	2003	Mean SAT I Verbal	496
New York	2003	Mean SAT I Math	510
Connecticut	2004	Participation Rate 2004	85
Connecticut	2004	Mean SAT I Verbal	515
Connecticut	2004	Mean SAT I Math	515
Connecticut	2003	Participation Rate 2003	84
Connecticut	2003	Mean SAT I Verbal	512

- Visual Transformation Previews
- Transformation History

[S. Kandel et al., 2011]

Improvements in Prediction



Update suggestions when given more information

[Heer et al., 2015]

Differences with Extract-Transform-Load (ETL)

- ETL:
  - Who: IT Professionals
  - Why: Create static data pipeline
  - What: Structured data
  - Where: Data centers
- "Modern Data Preparation":
  - Who: Analysts
  - Why: Solve problems by designing recipes to use data
  - What: Original, custom data blended with other data
  - Where: Cloud, desktop

[J. M. Hellerstein et al., 2018]

Handling Missing Data

- Filtering out missing data:
  - Can choose rows or columns
- Filling in missing data:
  - with a default value
  - with an interpolated value
- In pandas:

Argument	Description
`dropna`	Filter axis labels based on whether values for each label have missing data, with varying thresholds for how much missing data to tolerate.
`fillna`	Fill in missing data with some value or using an interpolation method such as `'ffill'` or `'bfill'`.
`isnull`	Return boolean values indicating which values are missing/NA.
`notnull`	Negation of `isnull`.

[W. McKinney, Python for Data Analysis]

Filtering and Cleaning Data

- Find duplicates
  - `duplicated`: returns boolean Series indicating whether row is a duplicate—first instance is **not marked** as a duplicate
- Remove duplicates:
  - `drop_duplicates`: drops all rows where `duplicated` is `True`
  - `keep`: which value to keep (first or last)
- Can pass specific columns to check for duplicates, e.g. check only key column

Replacing Values

- `fillna` is a special case
- What if `-999` in our dataset was identified as a missing value?

```
In [61]: data
Out[61]:
0       1.0
1    -999.0
2       2.0
3    -999.0
4   -1000.0
5       3.0
dtype: float64
```

```
In [62]: data.replace(-999, np.nan)
Out[62]:
0       1.0
1       NaN
2       2.0
3       NaN
4   -1000.0
5       3.0
dtype: float64
```

- Can pass list of values or dictionary to change different values

String Transformation

- One of the reasons for Python's popularity is string/text processing
- `split(<delimiter>)`: break a string into pieces:
  - ```
    s = "12,13, 14"
    slist = s.split(',') # ["12", "13", " 14"]
    ```
- `<delimiter>.join([<str>])`: join several strings by a delimiter
  - `":".join(slist) # "12:13: 14"`
- `strip()`: remove leading and trailing whitespace
  - `[p.strip() for p in slist] # ["12", "13", "14"]`
    ```

String Transformation

- `replace(<from>,<to>)`: change substrings to another substring
- `upper()`/`lower()`: casing
- `index(<str>)`: find where a substring first occurs (Error if not found)
- `find(<str>)`: same as index but `-1` if not found
- `startswith()`/`endswith()`: boolean checks for string occurrence

Assignment 2

- Due Friday
- Same data as A1, different version of the dataset
- Dealing with the raw data now
- Same questions as A1, but use pandas
- Potential answers sent via email
- CS680 students + some questions about problems with the data

Test 1

- Wednesday, February 17, 3:30pm-4:45pm Online (Blackboard)
- Includes much of the python content we have covered plus data, data cleaning, data transformation topics (content through today's lecture)
- Format:
  - Multiple Choice
  - Free Response (see web page for examples)
  - CS680 students will have additional questions
- Coding questions will focus on broad syntax not your memorization of every pandas function
- Concept questions can include discussions of the research papers

Regular Expressions in Python

- `import re`
- `re.search(<pattern>, <str_to_check>)`
  - Returns `None` if no match, information about the match otherwise
- Capturing information about what is in a string → **parentheses**
- `(\d+)/\d+/\d+` will **capture** information about the month
-
```
match = re.search('(\d+)/\d+/\d+','12/31/2016')
if match:
    match.group() # 12
```
- `re.findall(<pattern>, <str_to_check>)`
  - Finds all matches in the string, search only finds the first match
- Can pass in flags to alter methods: e.g. `re.IGNORECASE`

Pandas String Methods

- Any column or series can have the string methods (e.g. replace, split) applied to the entire series
- Fast (vectorized) on whole columns or datasets
- use `.str.<method_name>`
- `.str` is **important**!
  - ```
    data = pd.Series({'Dave': 'dave@google.com',
                      'Steve': 'steve@gmail.com',
                      'Rob': 'rob@gmail.com',
                      'Wes': np.nan})
    data.str.contains('gmail')
    data.str.split('@').str[1]
    data.str[-3:]
    ```
    ```

Regular Expression Methods

Argument	Description
`findall`	Return all non-overlapping matching patterns in a string as a list
`finditer`	Like `findall`, but returns an iterator
`match`	Match pattern at start of string and optionally segment pattern components into groups; if the pattern matches, returns a match object, and otherwise `None`
`search`	Scan string for match to pattern; returning a match object if so; unlike `match`, the match can be anywhere in the string as opposed to only at the beginning
`split`	Break string into pieces at each occurrence of pattern
`sub, subn`	Replace all (`sub`) or first `n` occurrences (`subn`) of pattern in string with replacement expression; use symbols `\1, \2, ...` to refer to match group elements in the replacement string

[W. McKinney, Python for Data Analysis]

Pandas String Methods with Regexs

```
In [172]: pattern
Out[172]: '([A-Z0-9._%+-]+)@([A-Z0-9.-]+)\\.([A-Z]{2,4})'

In [173]: data.str.findall(pattern, flags=re.IGNORECASE)
Out[173]:
Dave     [(dave, google, com)]
Rob        [(rob, gmail, com)]
Steve    [(steve, gmail, com)]
Wes                        NaN
dtype: object

In [174]: matches = data.str.match(pattern, flags=re.IGNORECASE)

In [175]: matches
Out[175]:
Dave     True
Rob      True
Steve    True
Wes       NaN
dtype: object
```

[W. McKinney, Python for Data Analysis]

Foofah: Transforming Data By Example

Z. Jin, M. R. Anderson, M. Cafarella, and
H. V. Jagadish

Foofah Discussion

Foofah Discussion

- What is the paper's contribution?

Foofah Discussion

- What is the paper's contribution?
- What questions do you have about what is going on?

Foofah Discussion

- What is the paper's contribution?
- What questions do you have about what is going on?
- What does the technique do well/have issues with?

Foofah Discussion

- What is the paper's contribution?
- What questions do you have about what is going on?
- What does the technique do well/have issues with?
- How does its approach compare with Trifacta?

Starting Point: Raw Data

split	split1	split2	split3	split4
	2004	2004	2004	2003
STATE	Participation Rate 2004	Mean SAT I Verbal	Mean SAT I Math	Participation Rate
New York	87	497	510	82
Connecticut	85	515	515	84
Massachusetts	85	518	523	82
New Jersey	83	501	514	85
New Hampshire	80	522	521	75
D.C.	77	489	476	77
Maine	76	505	501	70
Pennsylvania	74	501	502	73
Delaware	73	500	499	73
Georgia	73	494	493	66

split	fold	fold1	value
New York	2004	Participation Rate 2004	87
New York	2004	Mean SAT I Verbal	497
New York	2004	Mean SAT I Math	510
New York	2003	Participation Rate 2003	82
New York	2003	Mean SAT I Verbal	496
New York	2003	Mean SAT I Math	510
Connecticut	2004	Participation Rate 2004	85
Connecticut	2004	Mean SAT I Verbal	515
Connecticut	2004	Mean SAT I Math	515
Connecticut	2003	Participation Rate 2003	84
Connecticut	2003	Mean SAT I Verbal	512

[Guo et al., 2011]

Goal

- Focus on data transformation
- Data transformation tools suffer usability issues:
  - High Skill: familiarity with operations and the effect or their order
  - High Effort: user effort increases as the program becomes longer
- Repetitive and tedious
- Goal: minimize a user's effort and reduce the required background knowledge for data transformation tasks

Getting Lost in Transformations

Bureau of I.A.	
Regional Director	Numbers
Niles C.	Tel: (800)645-8397
	Fax: (907)586-7252
Jean H.	Tel: (918)781-4600
	Fax: (918)781-4604
Frank K.	Tel: (615)564-6500
	Fax: (615)564-6701

Original Table

Split+Delete

Niles C.	Tel	(800)645-8397
	Fax	(907)586-7252
Jean H.	Tel	(918)781-4600
	Fax	(918)781-4604
Frank K.	Tel	(615)564-6500
	Fax	(615)564-6701

Intermediate Table

Unfold

	Tel	Fax
Niles C.	(800)645-8397	
		(615)564-6701
Jean H.	(918)781-4600	
Frank K.	(615)564-6500	

Problem Table

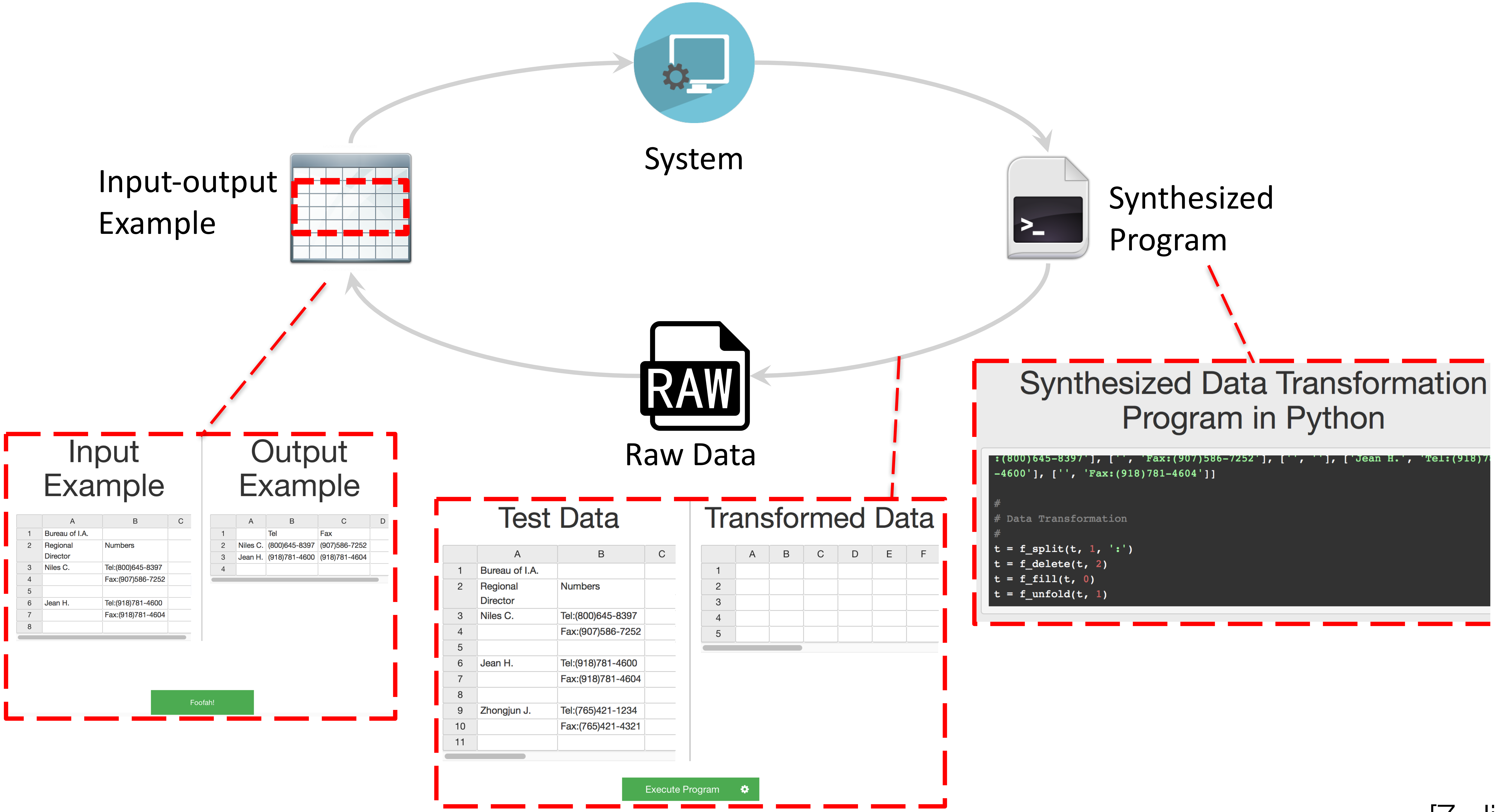
Fill+ Unfold

	Tel	Fax
Niles C.	(800)645-8397	(907)586-7252
Jean H.	(918)781-4600	(918)781-4604
Frank K.	(615)564-6500	(615)564-6701

Desired Solution

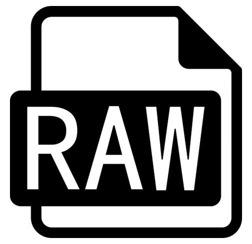
[Z. Jin et al., 2017]

Foofah Design: Programming by Example



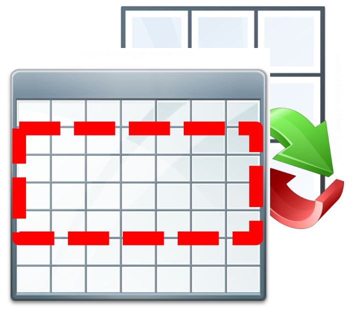
[Z. Jin et al., 2017]

Input, Output, and Transformations



Raw Data:

- A grid of values, i.e., spreadsheets
- "Somewhat" structured - must have some regular structure or is automatically generated.



User Input:

- Sample from raw data
- Transformed view of the sample

Program to synthesize:

- A loop-free Potter's Wheel [2] program

Transformations Targeted:

1. Layout transformation

2. String transformation

05-16-2017
05-17-2017
…

→

05/16/2017
05/17/2017
…

[Z. Jin et al., 2017]

Transformations

Operator	Description
Drop	Deletes a column in the table
Move	Relocates a column from one position to another in the table
Copy	Duplicates a column and append the copied column to the end of the table
Merge	Concatenates two columns and append the merged column to the end of the table
Split	Separates a column into two or more halves at the occurrences of the delimiter
Fold	Collapses all columns after a specific column into one column in the output table
Unfold	"Unflatten" tables and move information from data values to column names
Fill	Fill empty cells with the value from above
Divide	Divide is used to divide one column into two columns based on some predicate
Delete	Delete rows or columns that match a given predicate
Extract	Extract first match of a given regular expression each cell of a designated column
Transpose	Transpose the rows and columns of the table
Wrap (added)	Concatenate multiple rows conditionally

[Z. Jin et al., 2017]

Proposed Solution

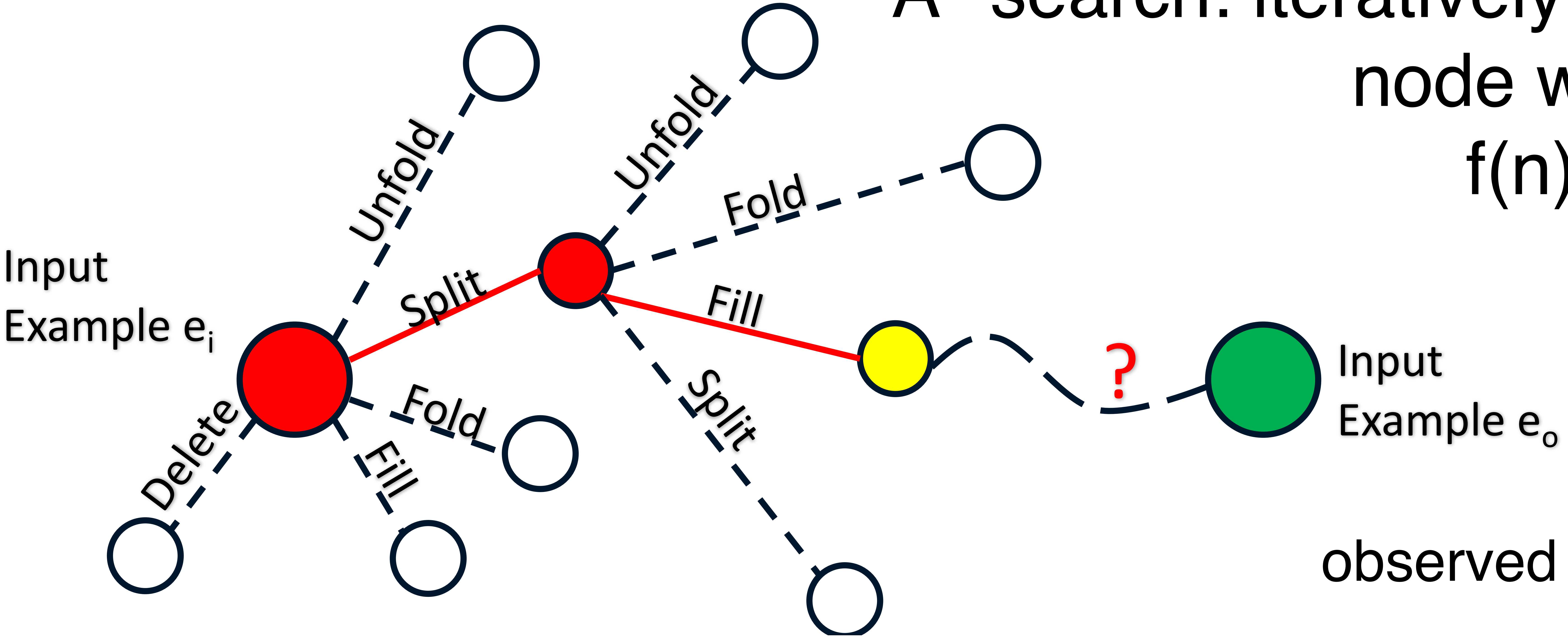
- Use a small, manually transformed portion of the data to infer a program (in Potter's Wheel syntax) based on the specified data transformation operations
- No loops
- Assumes relational tables
- … and perfect data?

Foofah Solution

A search problem solved by A* algorithm

edges: operation
nodes: different views of the data
A* search: iteratively explore the node with min f(n)

$f(n) = g(n) + h(n)$



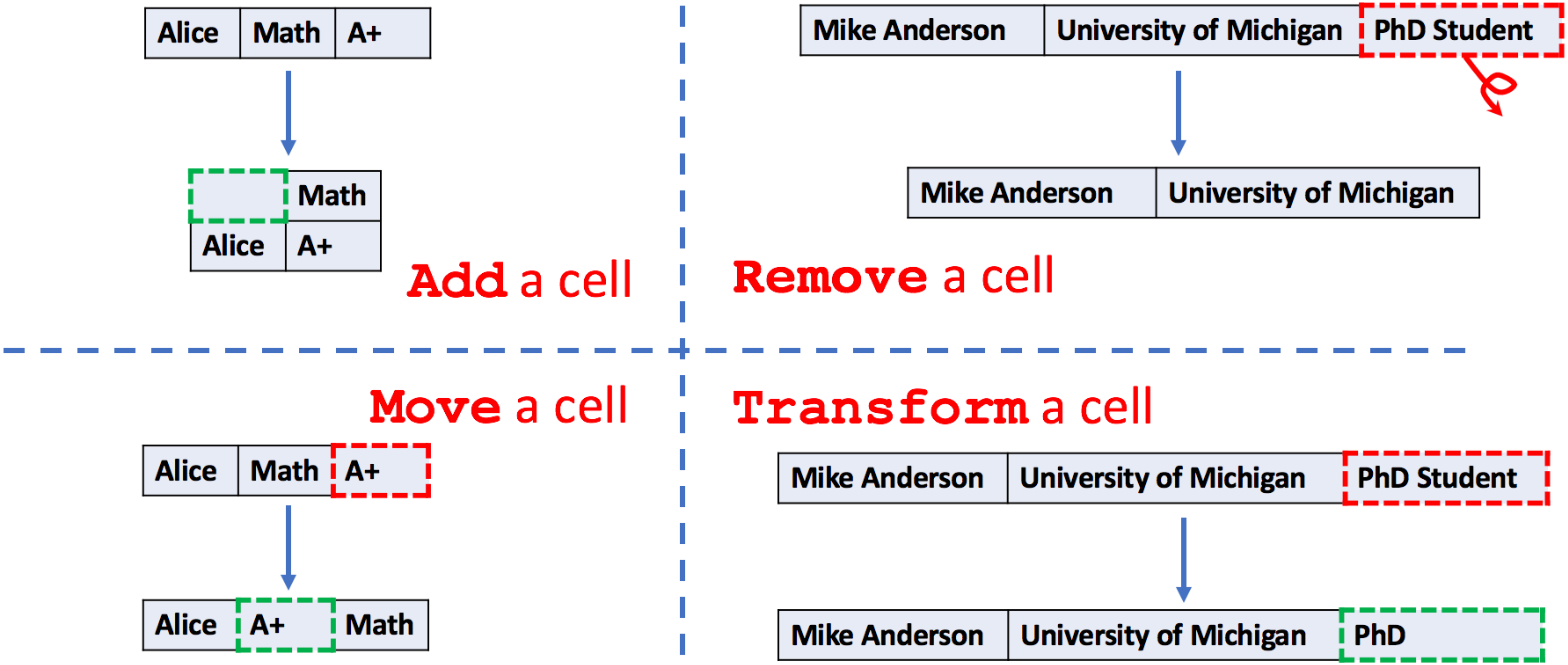
estimated distance

observed distance

[Z. Jin et al., 2017]

Need a Heuristic Function to Prune

Most transformations are composed of cell-based operations



[Z. Jin et al., 2017]

Table Edit Distance

- Akin to Graph Edit Distance
- Count the number of operations required to transform one table to another
- Use Add/Remove/Modify + Move

Table Edit Distance (TED) Definition:
The cost of transforming Table T_1 to Table T_2 using the cell-level operators Add/Remove/Move/Transform cell.

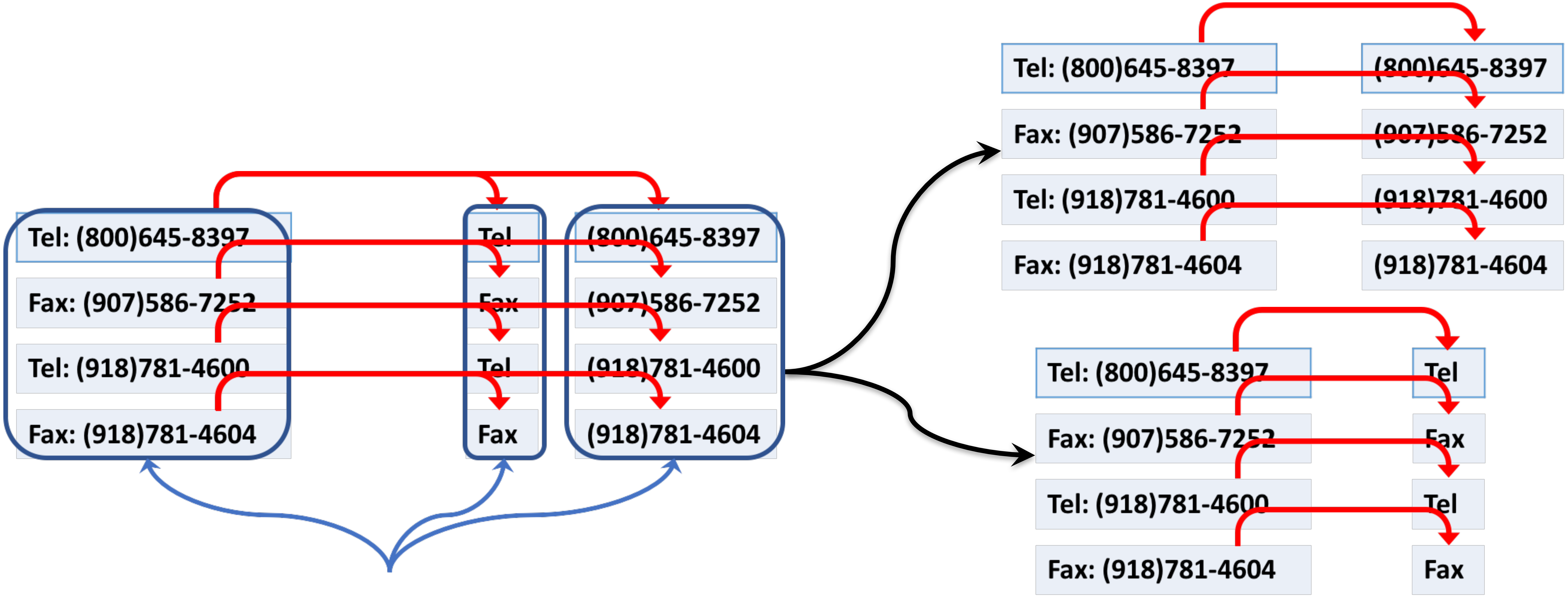
$$\mathrm{TED}(T_1, T_2) = \min_{(p_1, \ldots, p_k) \in P(T_1, T_2)} \sum_{i=0}^{k} cost(p_i)$$

- $P(T_1, T_2)$: Set of all "paths" transforming T_1 to T_2 using cell-level operators

[Z. Jin et al., 2017]

Table Edit Distance Batch

Batch the geometrically-adjacent cell-level operations of the same type



8 `Transform` operations

2 "batched" `Transform` operations

[Z. Jin et al., 2017]

Geometric Patterns Used to Batch

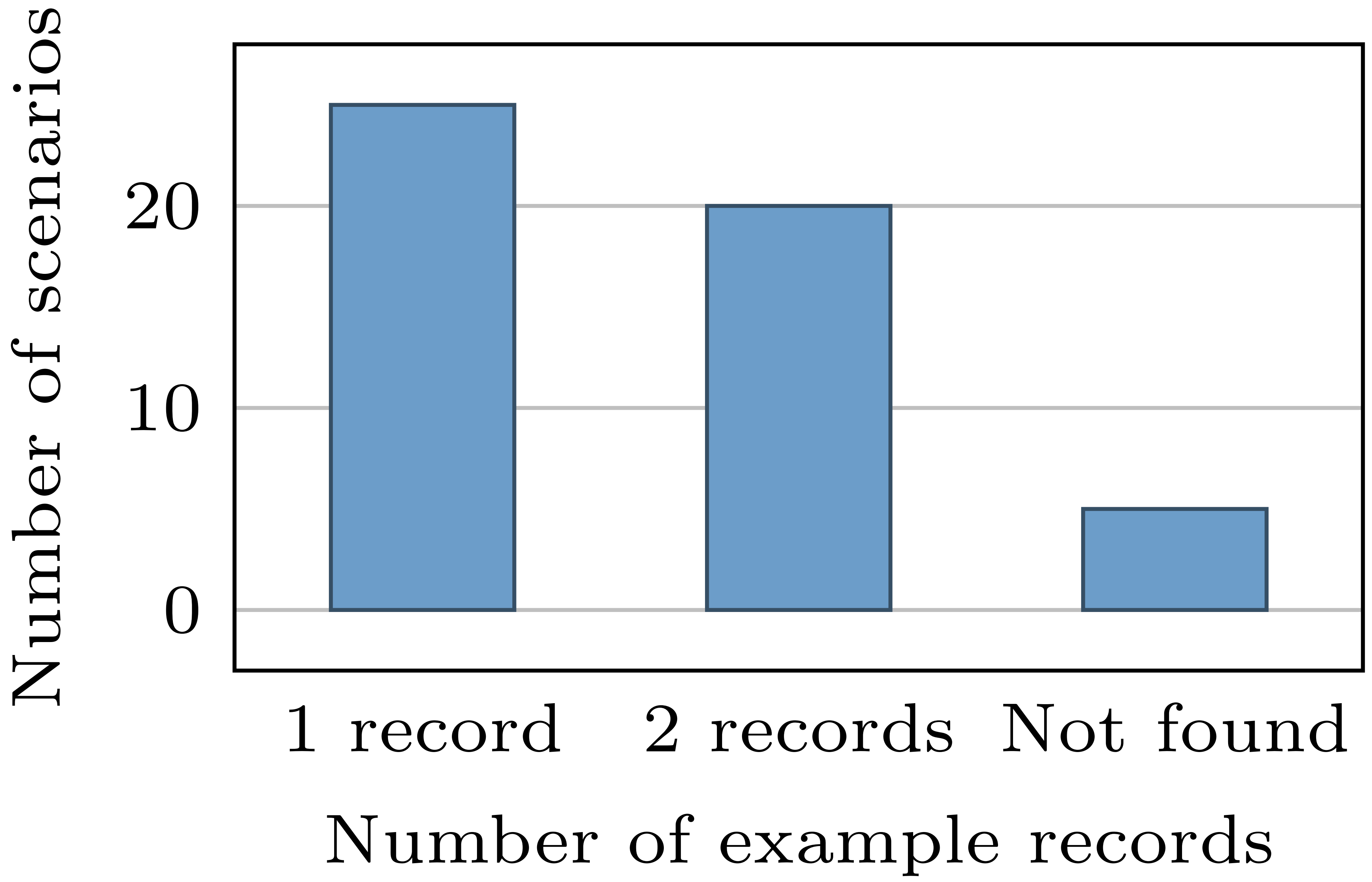
Pattern	Formulation (X is a table edit operator)	Related Operators
Horizontal to Horizontal	$\{X((x_i, y_i), (x_j, y_j)), X((x_i, y_i + 1), (x_j, y_j + 1)), \ldots\}$	Delete(Possibly)
Horizontal to Vertical	$\{X((x_i, y_i), (x_j, y_j)), X((x_i, y_i + 1), (x_j + 1, y_j)), \ldots\}$	Fold, Transpose
Vertical to Horizontal	$\{X((x_i, y_i), (x_j, y_j)), X((x_i + 1, y_i), (x_j, y_j + 1)), \ldots\}$	Unfold,Transpose
Vertical to Vertical	$\{X((x_i, y_i), (x_j, y_j)), X((x_i + 1, y_i), (x_j + 1, y_j)), \ldots\}$	Move, Copy, Merge, Split, Extract, Drop
One to Horizontal	$\{X((x_i, y_i), (x_j, y_j)), X((x_i, y_i), (x_j, y_j + 1)), \ldots\}$	Fold(Possibly), Fill(Possibly)
One to Vertical	$\{X((x_i, y_i), (x_j, y_j)), X((x_i, y_i), (x_j + 1, y_j)), \ldots\}$	Fold, Fill
Remove Horizontal	$\{X((x_i, y_i)), X((x_i, y_i + 1)), \ldots\}$	Delete
Remove Vertical	$\{X((x_i, y_i)), X((x_i + 1, y_i)), \ldots\}$	Drop, Unfold

[Z. Jin et al., 2017]

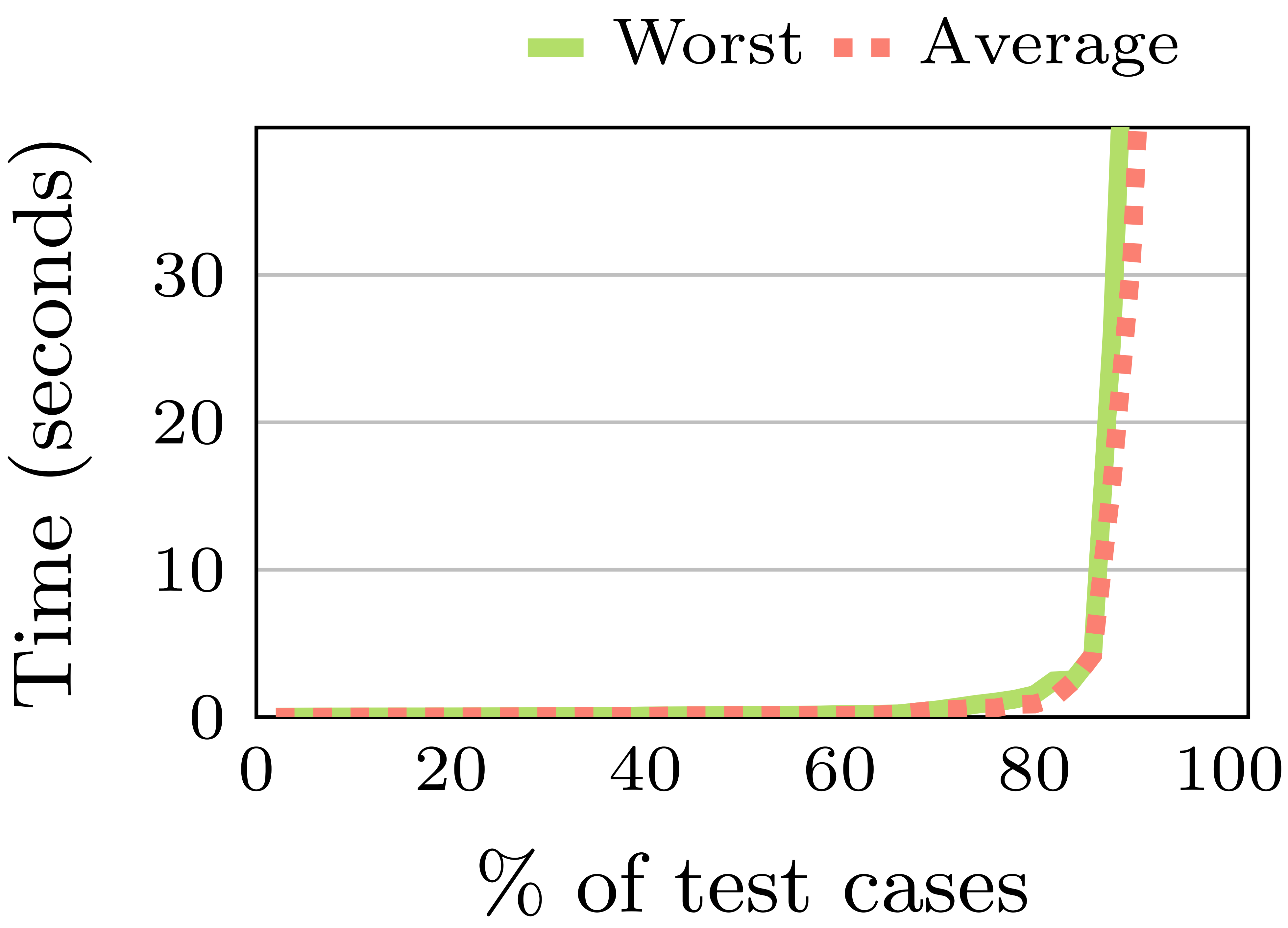
Other Pruning Rules

- Global:
  - Missing Alphanumerics: check that character maintained
  - No effect: meaningless operation
  - Introducing Novel Symbols: check that no new characters added
- Property-specific:
  - Generating Empty Columns
  - Null in Column

Evaluation Results: # Test Records & Time



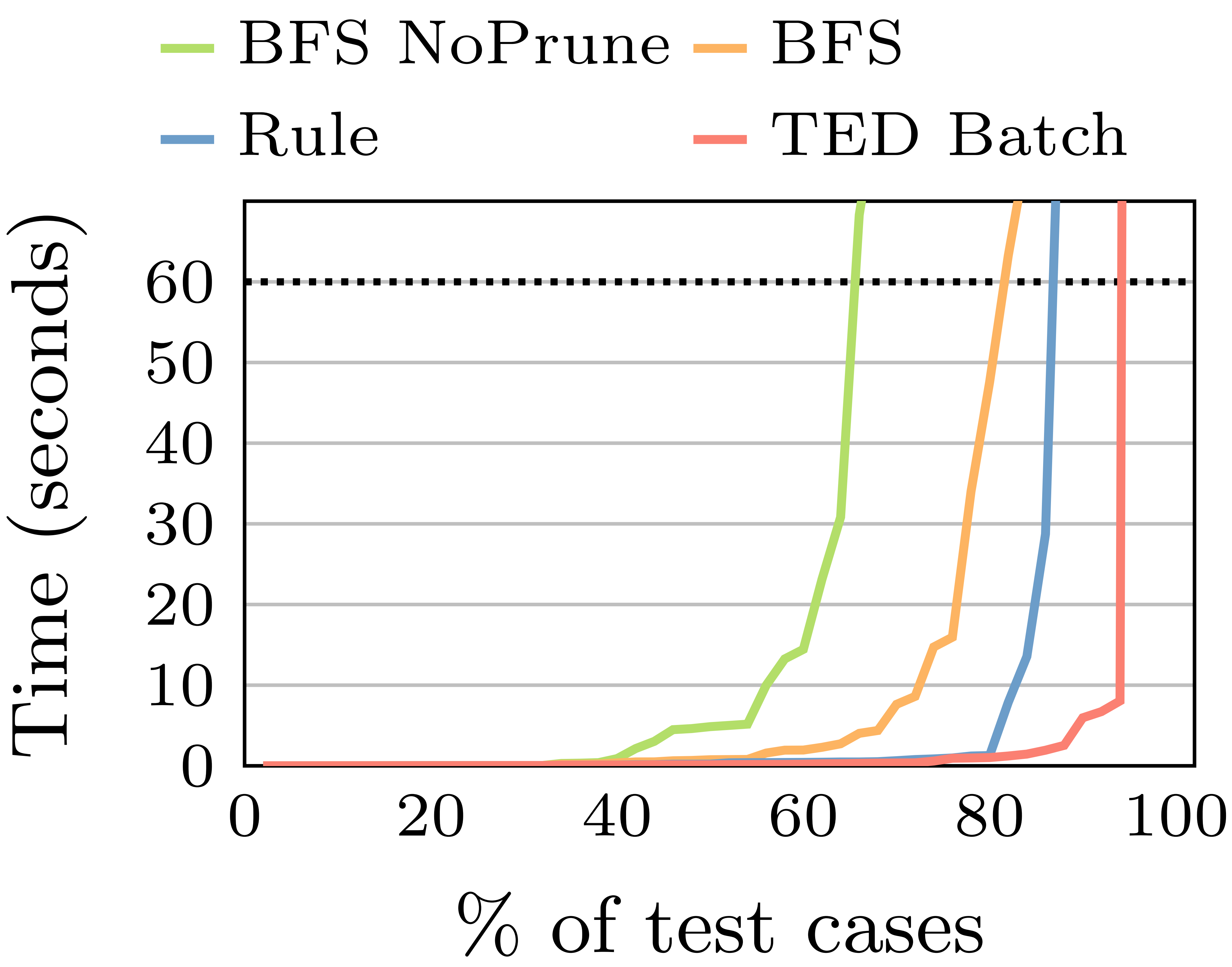
(a) Number of records required in test scenarios to infer *perfect* programs



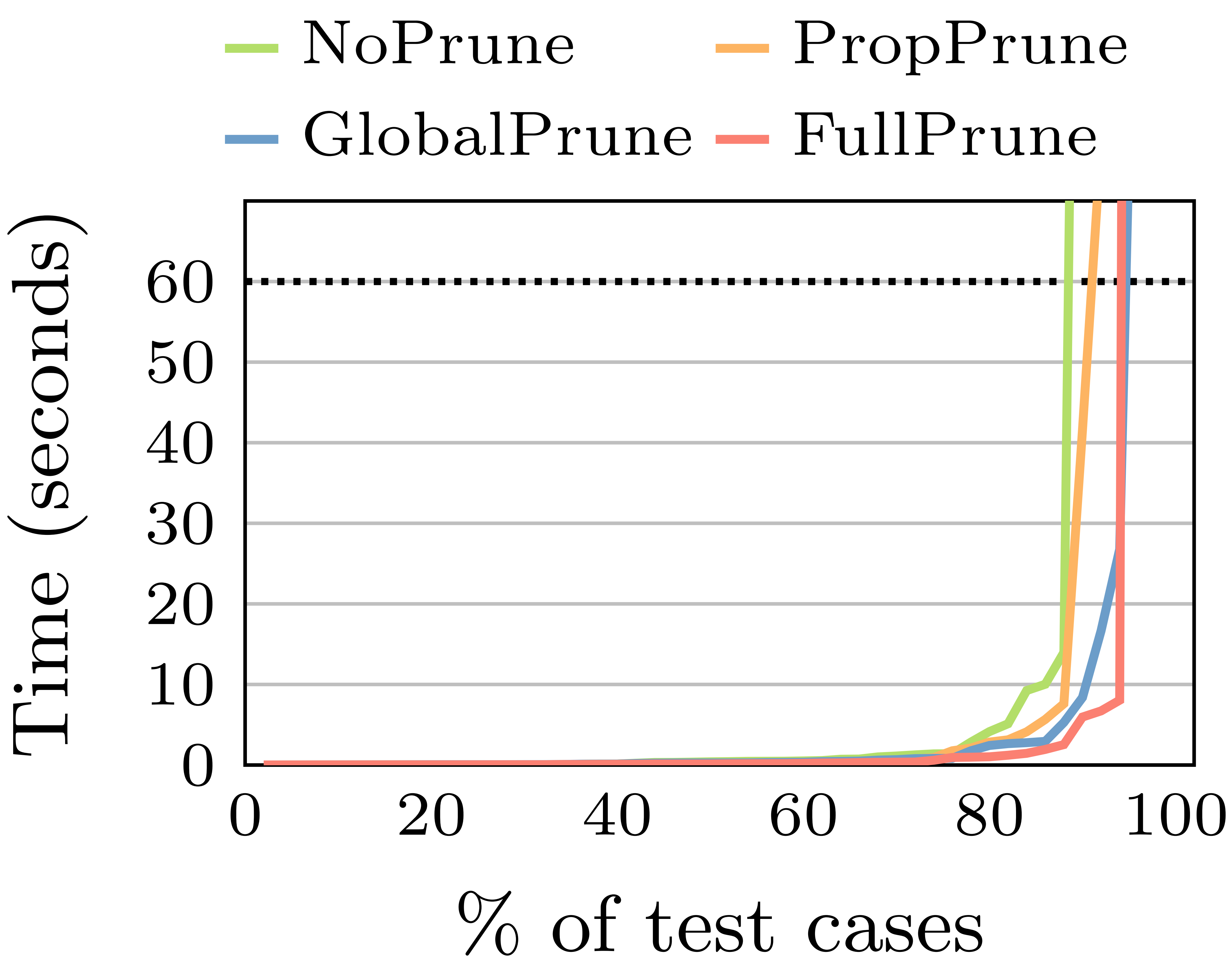
(b) Worst and average synthesis time in each interaction

[Z. Jin et al., 2017]

Search Strategies and Pruning Rules



(a) Compare search strategies



(b) Effectiveness of pruning rules

[Z. Jin et al., 2017]

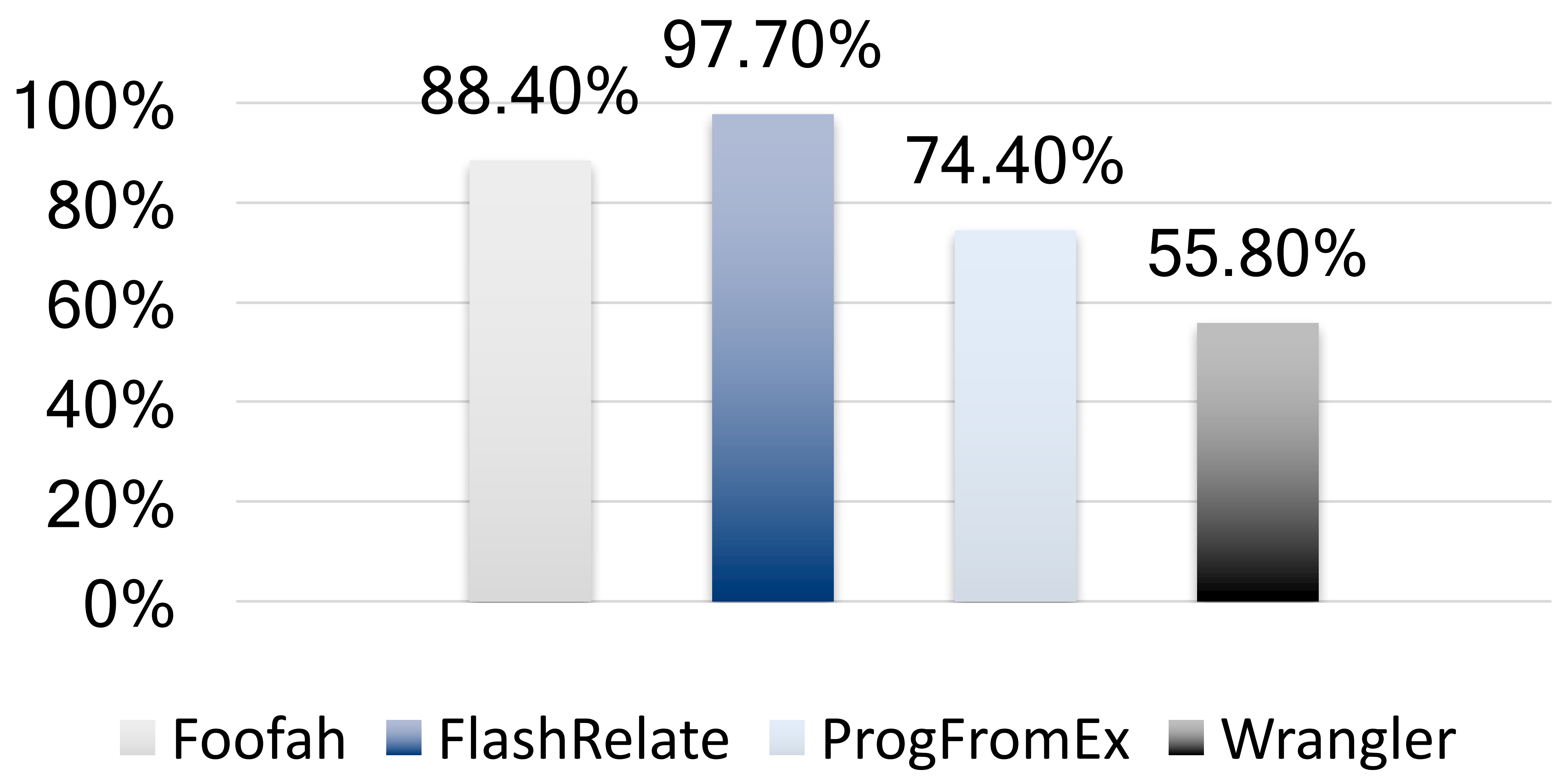
User Study Results

			Wrangler			Foofah		
Test	Complex	$\geq$ 4 Ops	Time	Mouse	Key	Time vs Wrangler	Mouse	Key
PW1	No	No	**104.2**	17.8	11.6	**49.4** ↘**52.6%**	20.8	22.6
PW3 (modified)	No	No	**96.4**	28.8	26.6	**38.6** ↘**60.0%**	14.2	23.6
ProgFromEx13	Yes	No	**263.6**	59.0	16.2	**145.8** ↘**44.7%**	43.6	78.4
PW5	Yes	No	**242.0**	52.0	15.2	**58.8** ↘**75.7%**	31.4	32.4
ProgFromEx17	No	Yes	**72.4**	18.8	11.6	**48.6** ↘**32.9%**	18.2	15.2
PW7	No	Yes	**141.0**	41.8	12.2	**44.4** ↘**68.5%**	19.6	35.8
Proactive1	Yes	Yes	**324.2**	60.0	13.8	**104.2** ↘**67.9%**	41.4	57.0
Wrangler3	Yes	Yes	**590.6**	133.2	29.6	**137.0** ↘**76.8%**	58.6	99.8

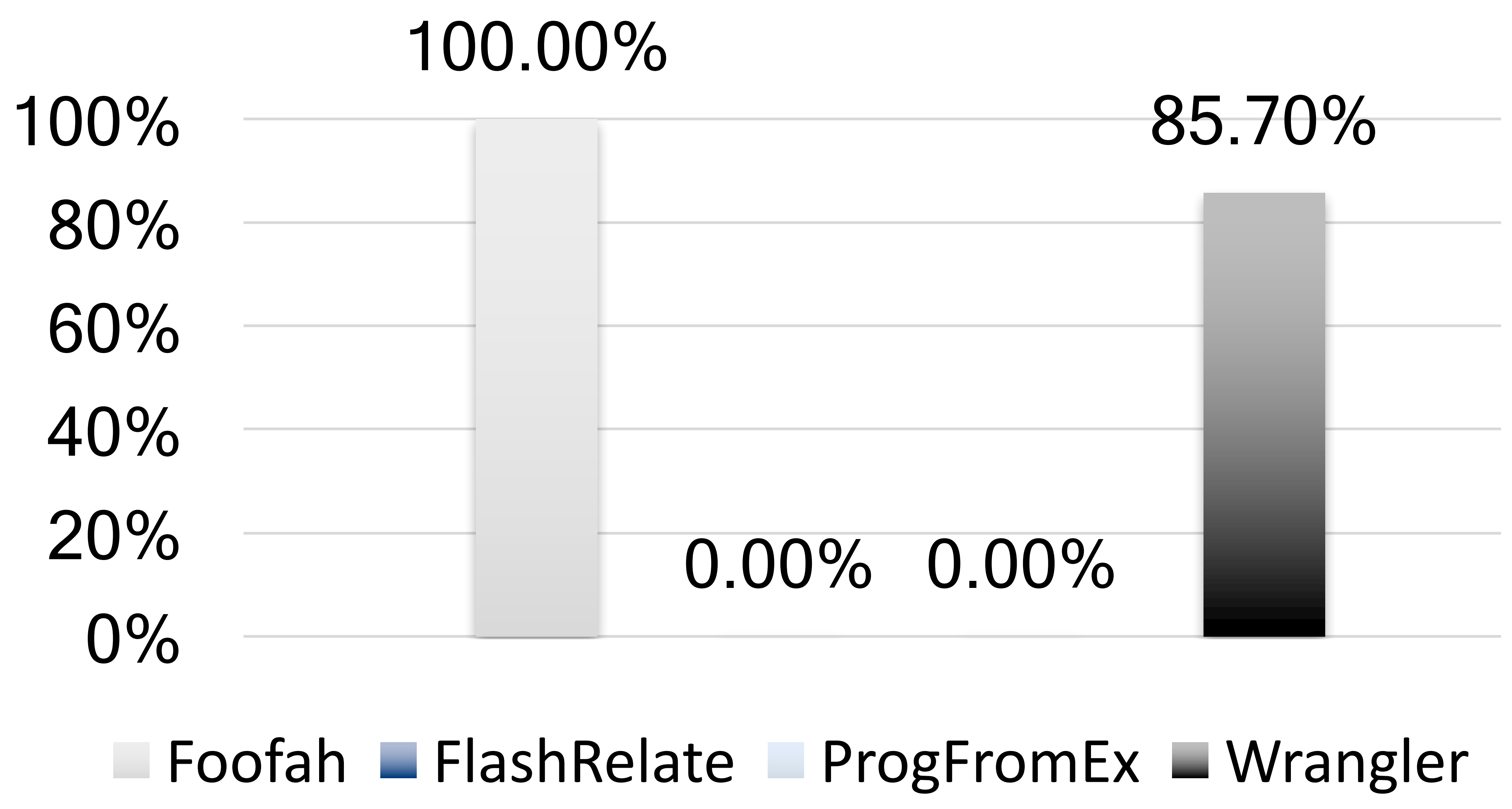
[Z. Jin et al., 2017]

Comparisons with other tools

Success rates on pure layout transformation benchmark tasks



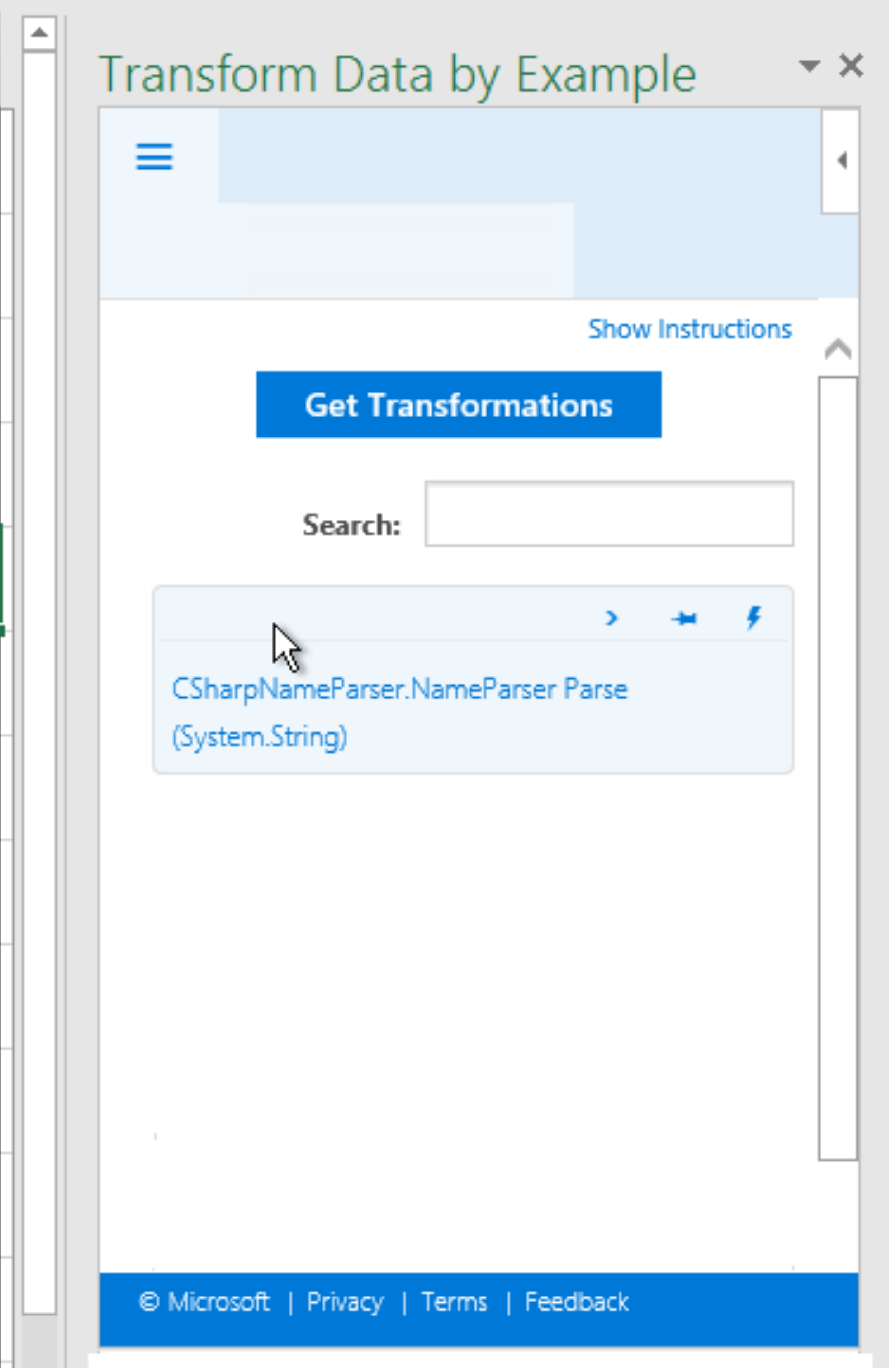
Success rates on benchmark tasks requiring syntactic transformations



[Z. Jin et al., 2017]

TDE: Transform Data by Example

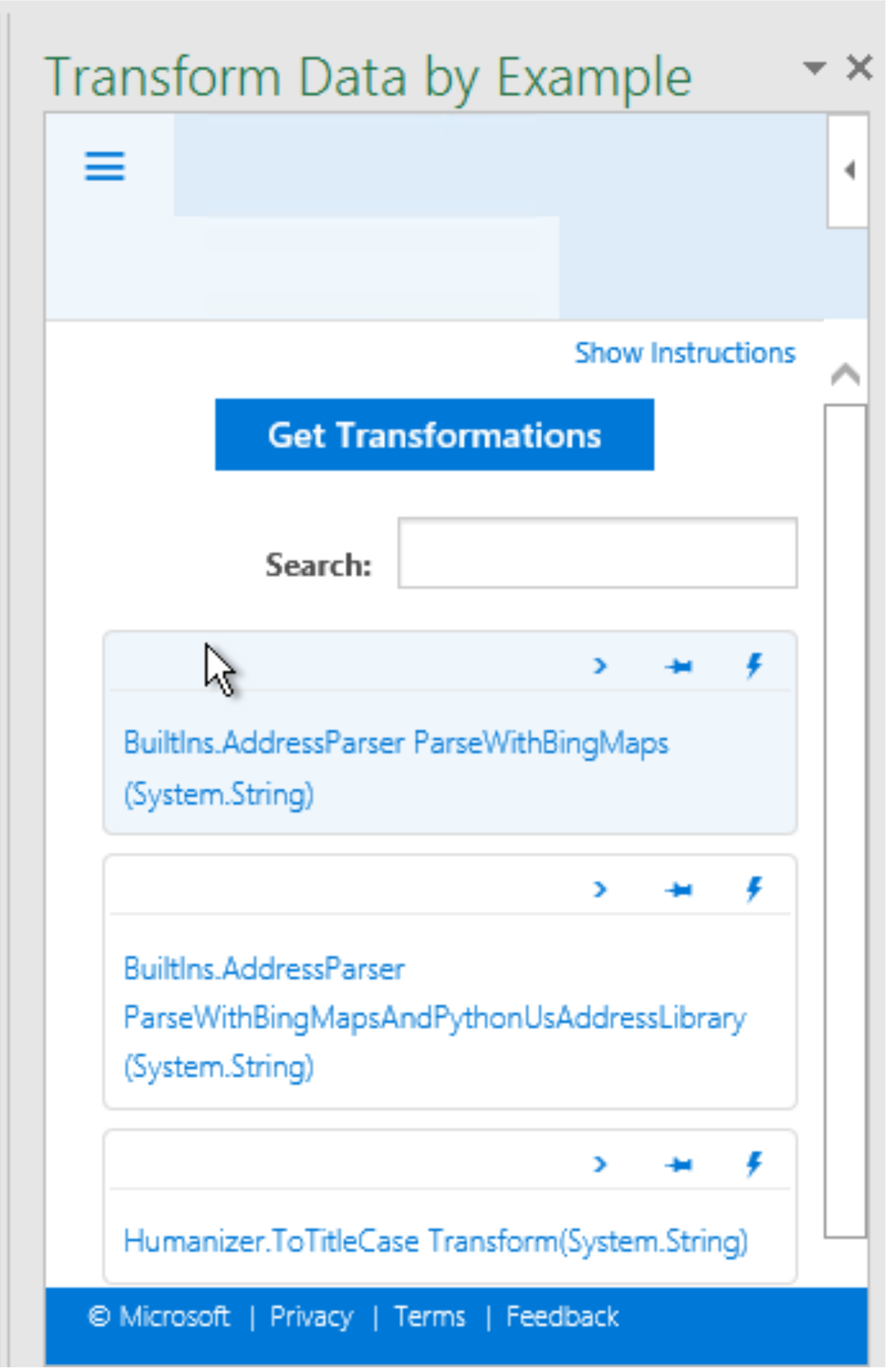
C	D
Customer Name	**Output**
John K. **Doe** Jr.	Doe, John
Mr. **Doe, John**	Doe, John
Jane A. **Smith**	Smith, Jane
MS. **Jane Smith**	Smith, Jane
Smith, Jane	Smith, Jane
Dr **Anthony** R **Von Fange** III	Von Fange, Anthony
Peter Tyson	Tyson, Peter
Dan E. **Williams**	Williams, Dan
James Davis Sr.	Davis, James
James J. **Davis**	Davis, James
Mr. **Donald** Edward **Miller**	Miller, Donald



[Y. He et al., 2018]

TDE: Transform Data by Example

C	D
Address	**Output**
4297 148th Avenue NE L105, Bellevue, WA 98007	Bellevue, WA, 98007
2720 N Mesa St, El Paso, 79902, USA	El Paso, TX, 79902
3524 W Shore Rd APT 1002, Warwick,02886	Warwick, RI, 02886
4740 N 132nd St, Omaha, 68164	Omaha, NE, 68164
10508 Prairie Ln, Oklahoma City	Oklahoma City, OK, 73162
525 1st St, Marysville, WA 95901	Marysville, CA, 95901
211 W Ridge Dr, Waukon,52172	Waukon, IA, 52172
1008 Whitlock Ave NW, Marietta, 30064	Marietta, GA, 30064
602 Highland Ave, Shinnston, 26431	Shinnston, WV, 26431
840 W Star St, Greenville, 27834	Greenville, NC, 27834



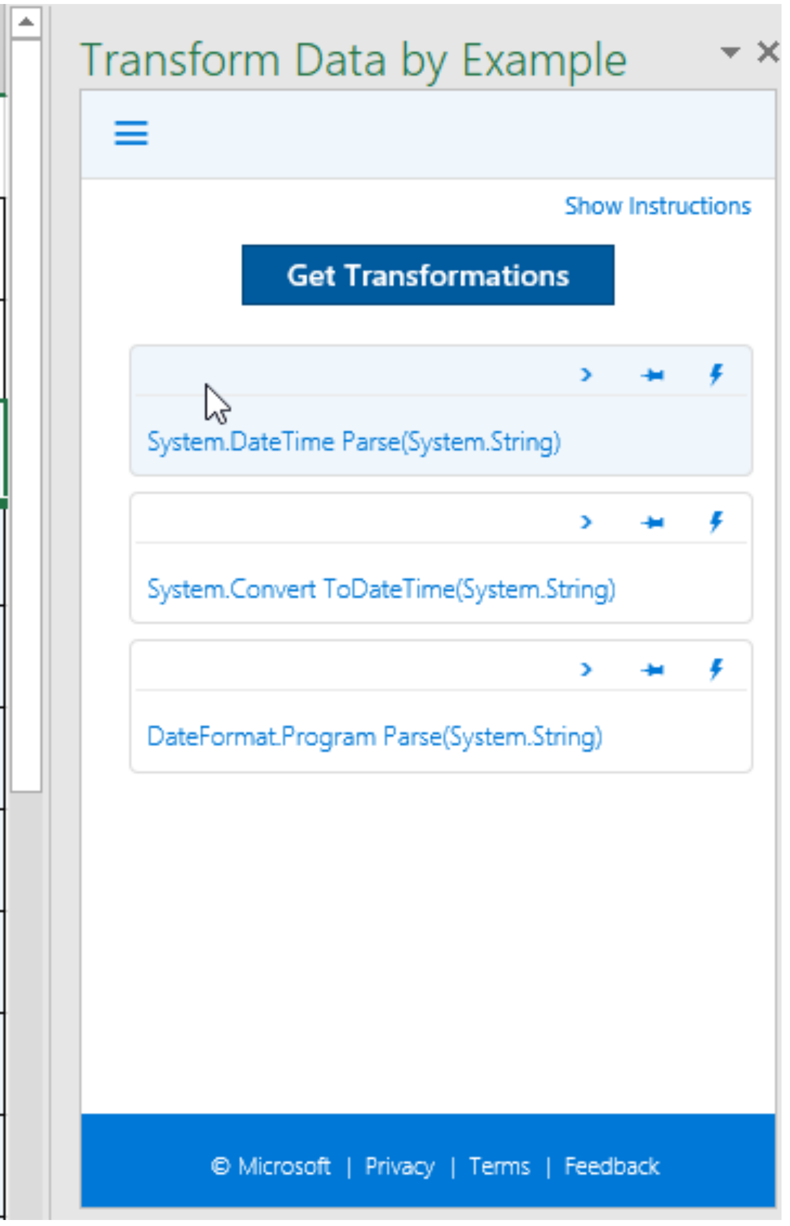
[Y. He et al., 2018]

TDE: Transform Data by Example

C	D
Transaction Date	**output**
Wed, 12 Jan 2011	2011-01-12-Wednesday
Thu, 15 Sep 2011	2011-09-15-Thursday
Mon, 17 Sep 2012	
2010-Nov-30 11:10:41	
2011-Jan-11 02:27:21	
2011-Jan-12	
2010-Dec-24	
9/22/2011	
7/11/2012	
2/12/2012	

→

C	D
Transaction Date	**output**
Wed, 12 Jan 2011	2011-01-12-Wednesday
Thu, 15 Sep 2011	2011-09-15-Thursday
Mon, 17 Sep 2012	2012-09-17-Monday
2010-Nov-30 11:10:41	2010-11-30-Tuesday
2011-Jan-11 02:27:21	2011-01-11-Tuesday
2011-Jan-12	2011-01-12-Wednesday
2010-Dec-24	2010-12-24-Friday
9/22/2011	2011-09-22-Thursday
7/11/2012	2012-07-11-Wednesday
2/12/2012	2012-02-12-Sunday



[Y. He et al., 2018]

TDE: Synthesized Function

Input Examples
Wed, 12 Jan 2011
Thu, 15 Sep 2011
Mon, 17 Sep 2012
2010-Nov-30 21:10:41
2011-Jan-11 02:27:21
2011-Jan-12

f →

Return Object Dump						Member method result dump				
Year	Month	Day	Day-of-week	Day-of-Year	...	ToLongDateString()	ToTimeStr()	ToUTC()	ToBinary()	...
2011	01	12	Wednesday	12	...	Wednesday, January 12, 2011	12:00:00 AM	...	...	...
2011	09	15	Thursday	258	...	Thursday, September 15, 2011	12:00:00 AM	...	...	...
2012	09	17	Monday	261	...	Monday, September 17, 2012	12:00:00 AM	...	...	...
2010	11	30	Tuesday	334	...	Tuesday, November 30, 2010	09:10:41 PM	...	...	...
2011	01	11	Tuesday	11	...	Tuesday, January 11, 2011	02:27:21 AM	...	...	...
2011	01	12	Wednesday	12	...	Wednesday, January 12, 2012	12:00:00 AM	...	...	...

Synthesize →

Desired Output
2011-01-12 (Wed)
2011-09-15 (Thu)

[Y. He et al., 2018]

TDE: Transform Data by Example

- Row-to-row translation only
- Search System, GitHub, and StackOverflow for functions
- Given dataset with examples
  - Use L1 from library
  - Compose synthesized programs (L2)
  - Rank best transformations

TDE Benchmarks

System	Total cases (239)	FF-GR-Trifacta (46)	Head cases (44)	StackOverflow (49)	BingQL-Unit (50)	BingQL-Other (50)
TDE	**72% (173)**	**91% (42)**	**82% (36)**	**63% (31)**	**96% (48)**	**32% (16)**
TDE-NF	53% (128)	87% (40)	41% (18)	35% (17)	96% (48)	10% (5)
FlashFill	23% (56)	57% (26)	34% (15)	31% (15)	0% (0)	0% (0)
Foofah	3% (7)	9% (4)	2% (1)	4% (2)	0% (0)	0% (0)
DataXFormer-UB	38% (90)	7% (3)	36% (16)	35% (17)	62% (31)	**46% (23)**
System-A	13% (30)	52% (24)	2% (1)	10% (5)	0% (0)	0% (0)
OpenRefine-Menu[8]	4% (9)	13% (6)	2% (1)	4% (2)	0% (0)	0% (0)

- TDE and FlashFill focused on row-to-row transformations
- Foofah considers a wider range of transformations (table reformatting)

[Y. He et al., 2018]

TDE Benchmarks

System	Total cases (239)	FF-GR-Trifacta (46)	Head cases (44)	StackOverflow (49)	BingQL-Unit (50)	BingQL-Other (50)
TDE	**72% (173)**	**91% (42)**	**82% (36)**	**63% (31)**	**96% (48)**	**32% (16)**
TDE-NF	53% (128)	87% (40)	41% (18)	35% (17)	96% (48)	10% (5)
FlashFill	23% (56)	57% (26)	34% (15)	31% (15)	0% (0)	0% (0)
Foofah	3% (7)	9% (4)	2% (1)	4% (2)	0% (0)	0% (0)
DataXFormer-UB	38% (90)	7% (3)	36% (16)	35% (17)	62% (31)	**46% (23)**
System-A	13% (30)	52% (24)	2% (1)	10% (5)	0% (0)	0% (0)
OpenRefine-Menu[8]	4% (9)	13% (6)	2% (1)	4% (2)	0% (0)	0% (0)

- TDE and FlashFill focused on row-to-row transformations
- Foofah considers a wider range of transformations (table reformatting)

[Y. He et al., 2018]

Trifacta's Transform by Example