~~FlashMeta~~ Microsoft PROSE SDK: A Framework for Inductive Program Synthesis

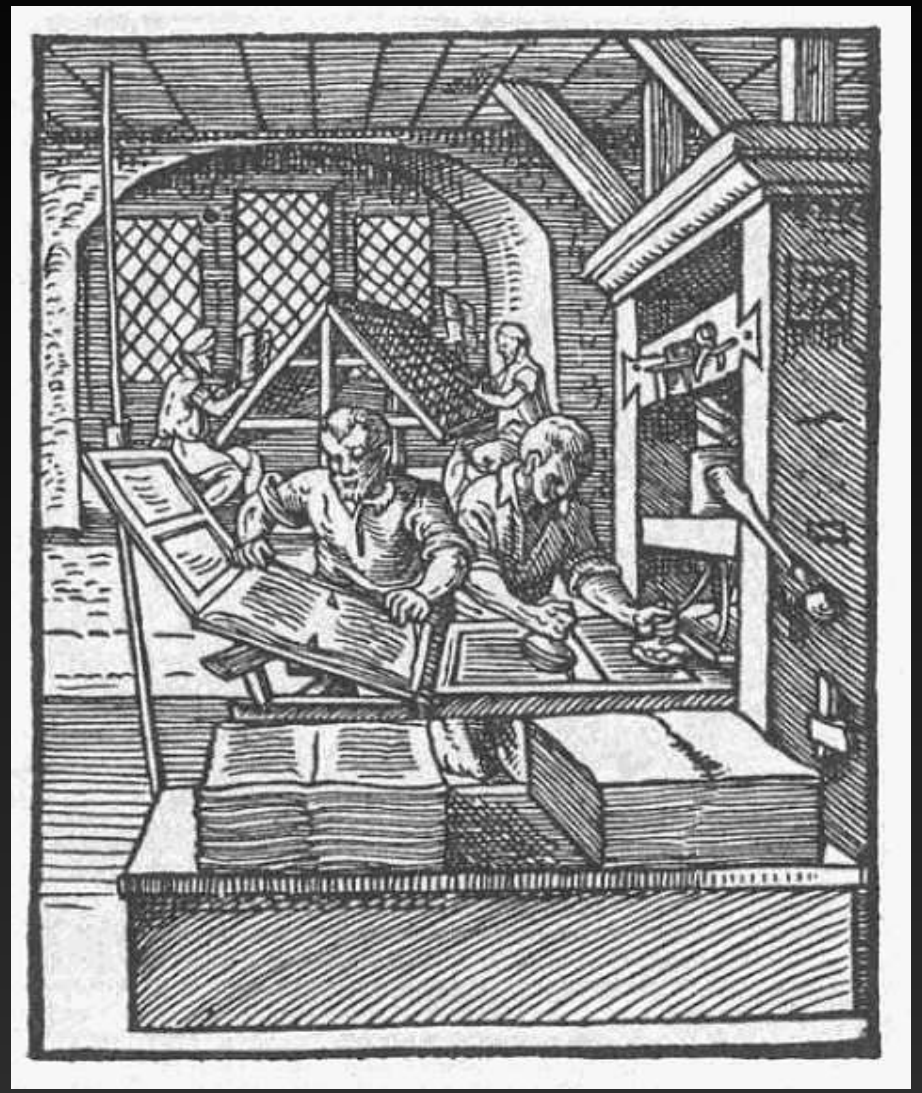
Oleksandr Polozov
University of Washington

Sumit Gulwani
Microsoft Research

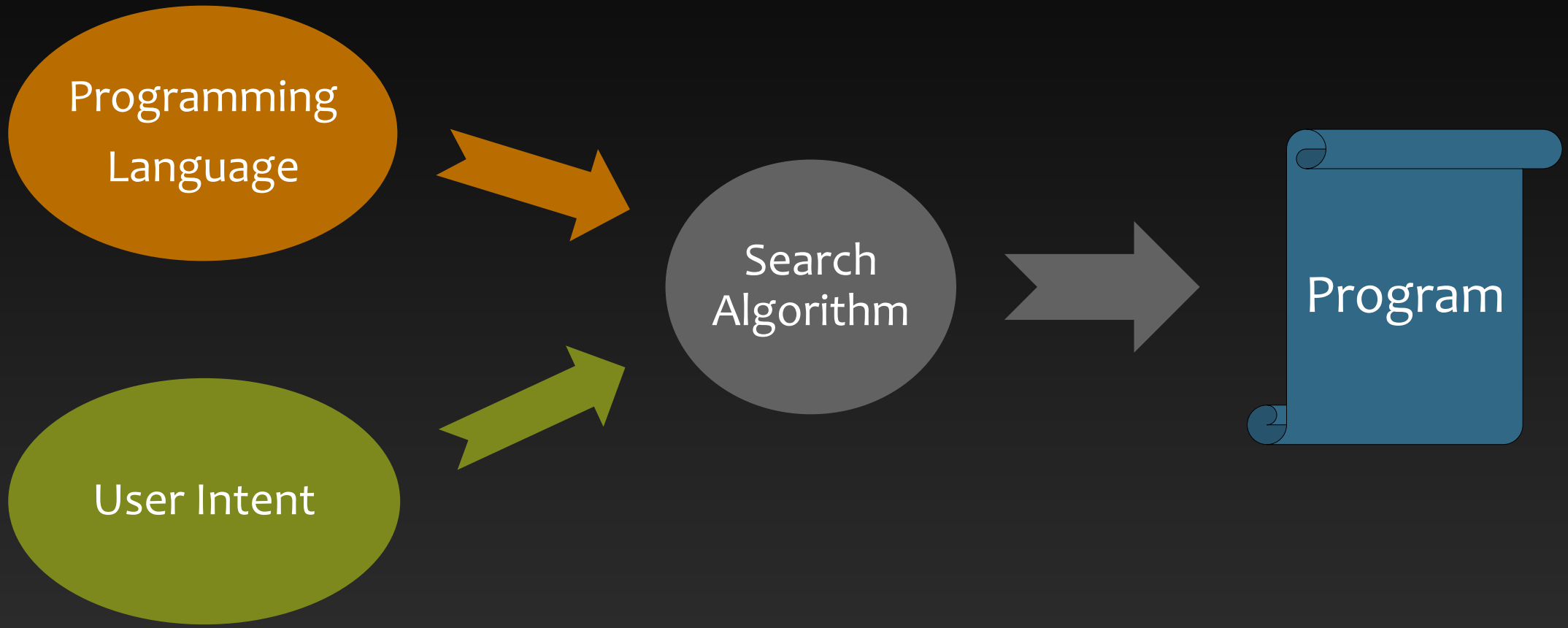
Microsoft Research

Why do people create frameworks?

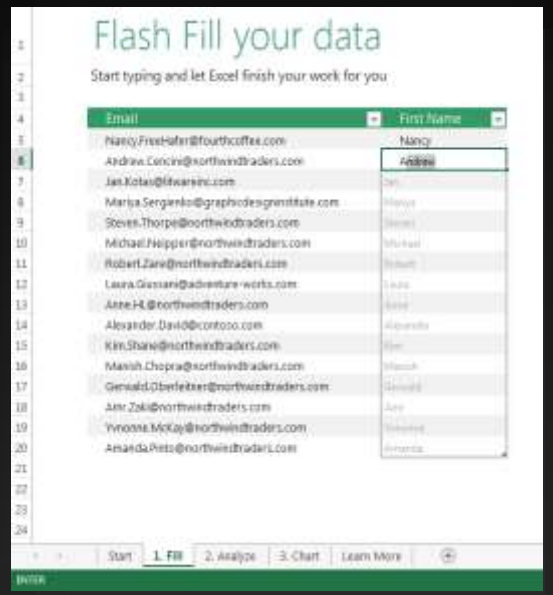
Industrialization (a.k.a. “Tech Transfer”)

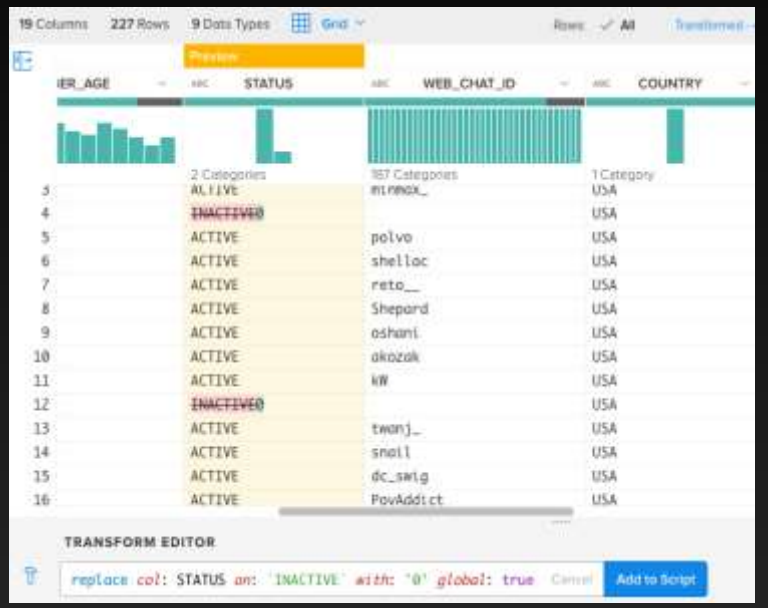


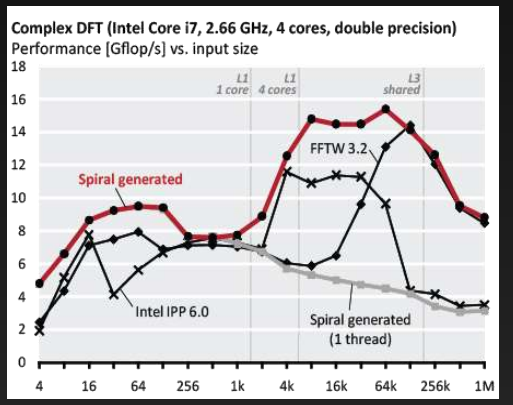
Program Synthesis: “The Ultimate Dream” of CS



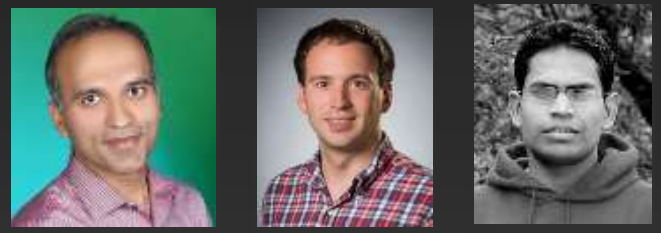
Industrialization Time?



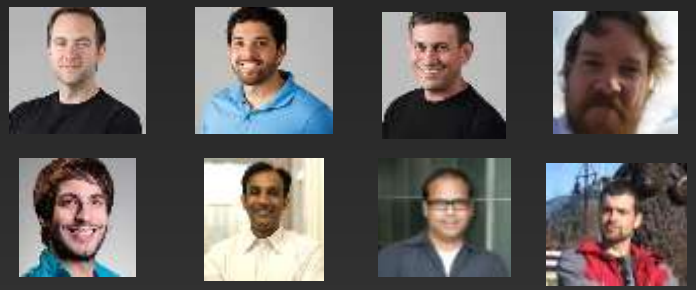




Flash Fill (2010-2012)



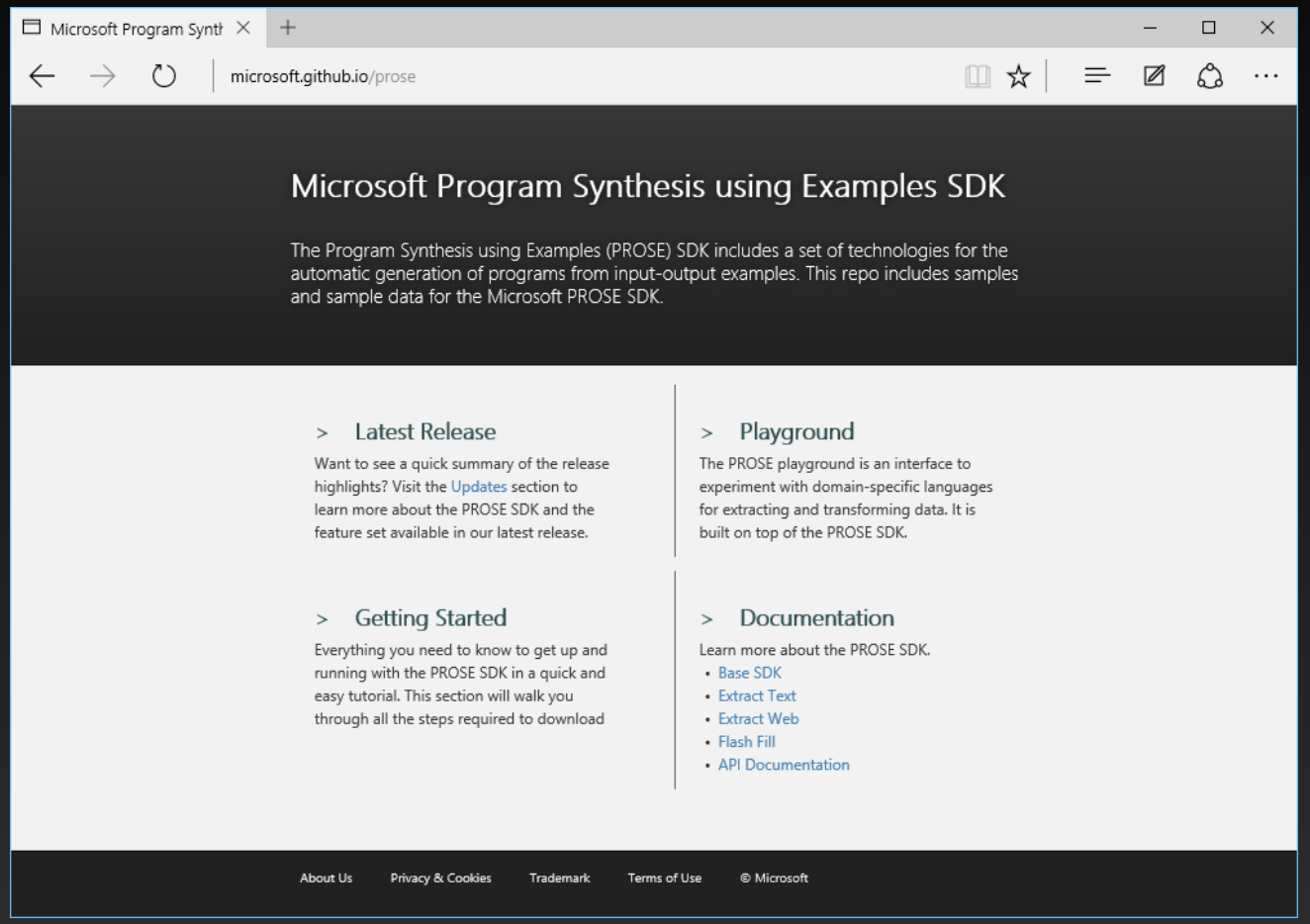
Trifacta (2012-2015)



SPIRAL (2000-2015)

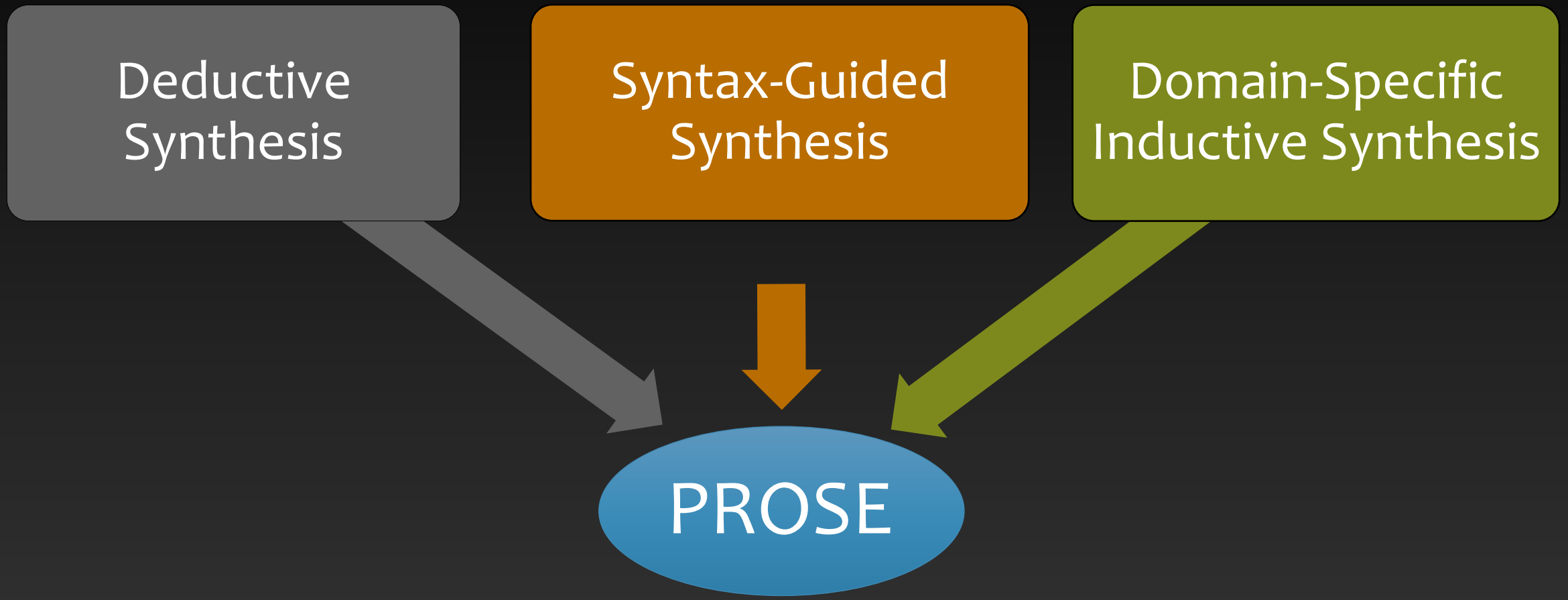
+114 more

Microsoft Program Synthesis using Examples SDK



https://microsoft.github.io/prose

Shoulders of Giants



Shoulders of Giants

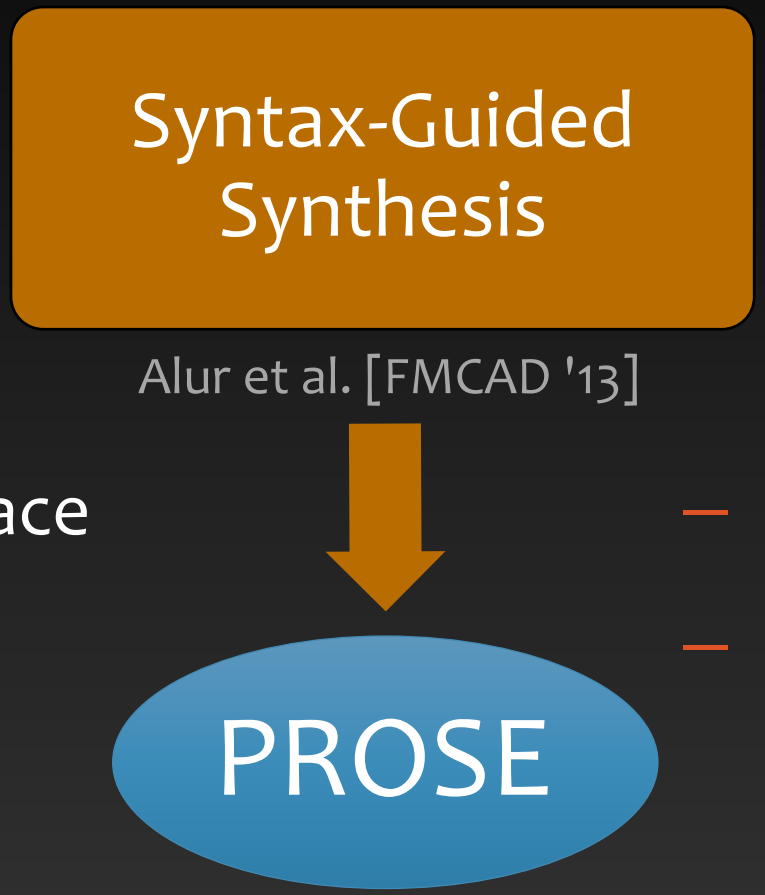
Deductive Synthesis

Püschel et al. [IEEE '05]
Panchekha et al. [PLDI '15]
Manna, Waldinger [TOPLAS '80]

+ No invalid candidates ⟹ fast
– [Usually] complete specs
– Domain axiomatization

PROSE

Shoulders of Giants

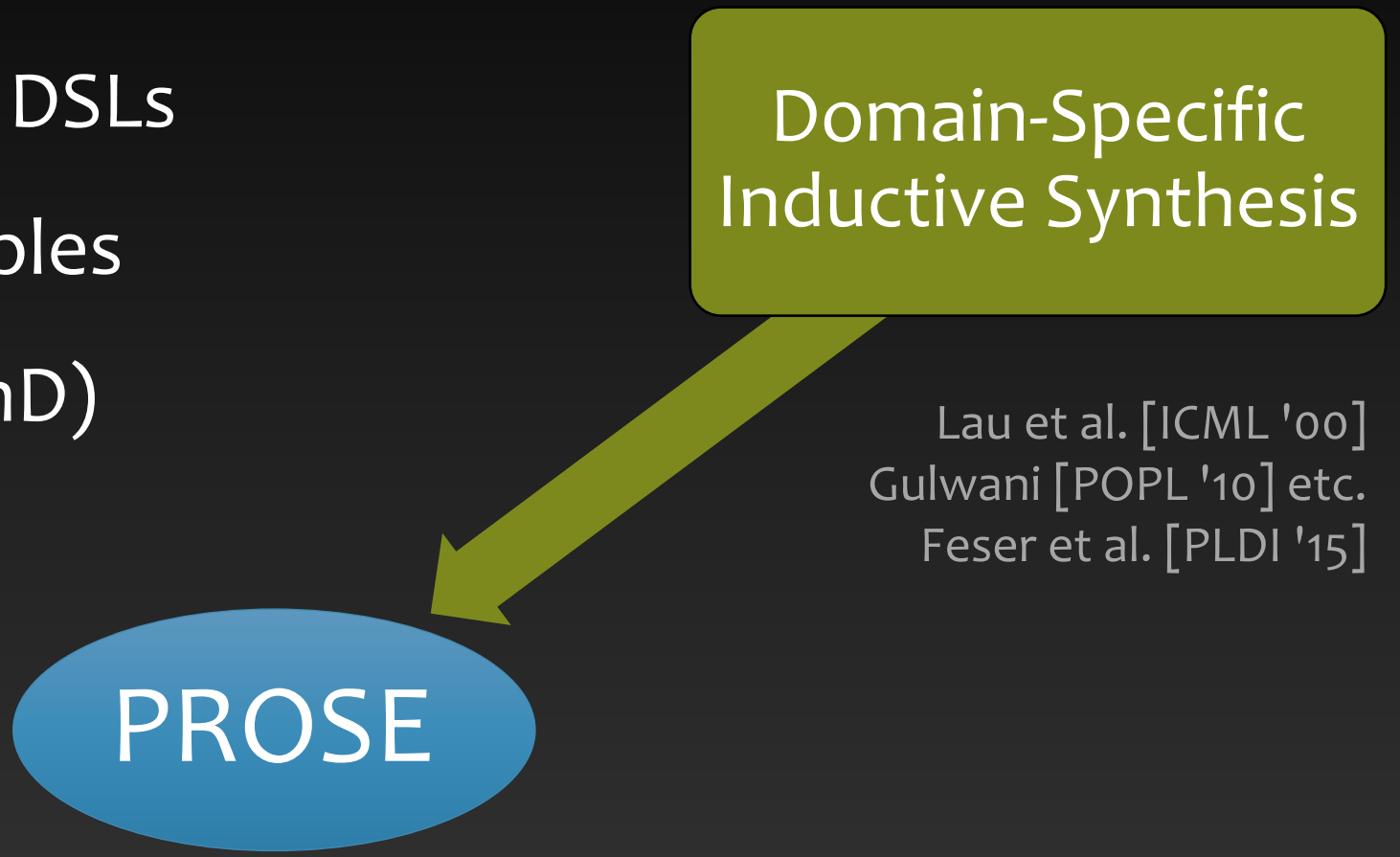


+ Shrinks the search space
+ Generic algorithms

– No domain-specific insights
– Limited to SMT-LIB

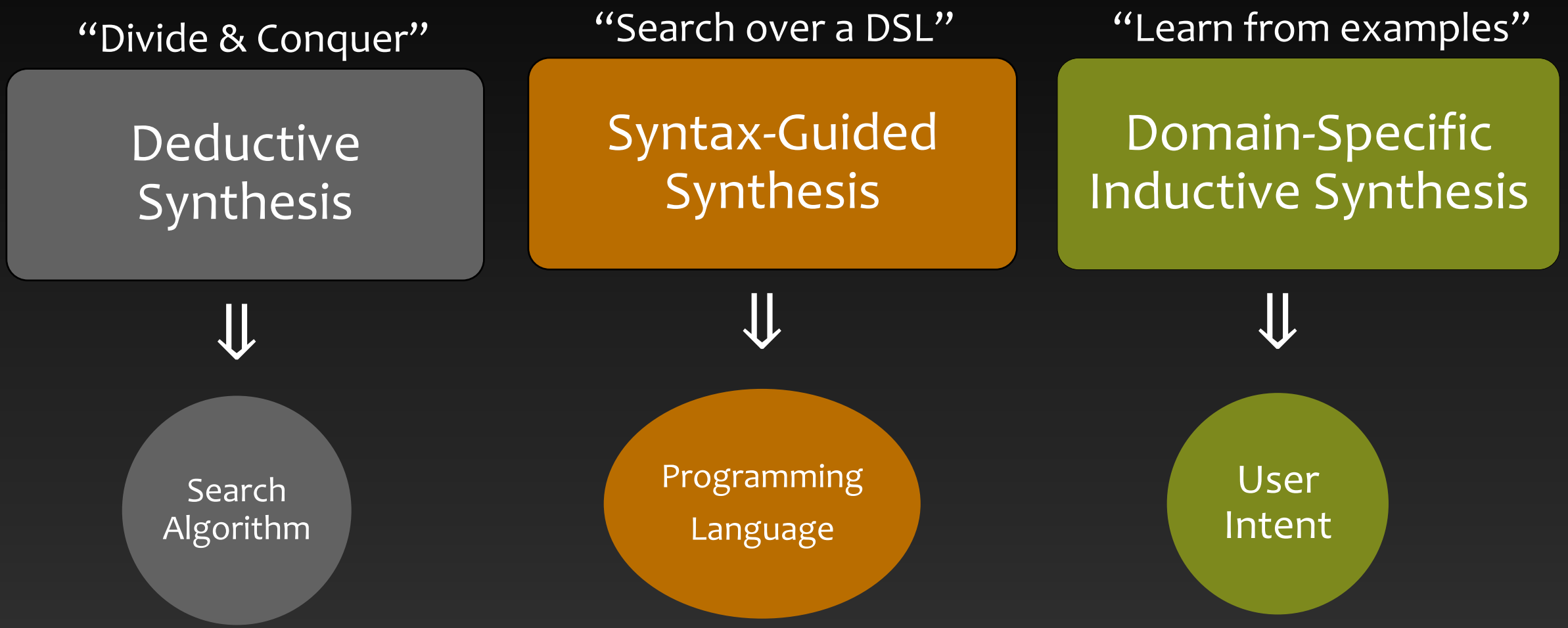
Shoulders of Giants

+ Arbitrarily complex DSLs
+ Input/output examples
– 1-2 person-years (PhD)
– One-off



Lau et al. [ICML '00]
Gulwani [POPL '10] etc.
Feser et al. [PLDI '15]

Shoulders of Giants



PROSE

Meta-synthesizer framework

Synthesis Strategies

PROSE

DSL Definition

App

Synthesizer

I/O Specification

Input

Programs

Output

Domain-Specific Language

FlashFill (portion) as a PROSE DSL

```
string output(string[] inputs) :=
    | ConstantString(s)
    | let string x = std.list.Kth(inputs, k) in
          Substring(x, positionPair(x));

Tuple<int, int> positionPair(string s) :=
    std.Pair(positionIn(s), positionIn(s));

int positionIn(string s) := AbsolutePosition(s, k)
                          | RegexPosition(s, std.Pair(r, r), k);

const int k;      const RegularExpression r;    const string s;
```

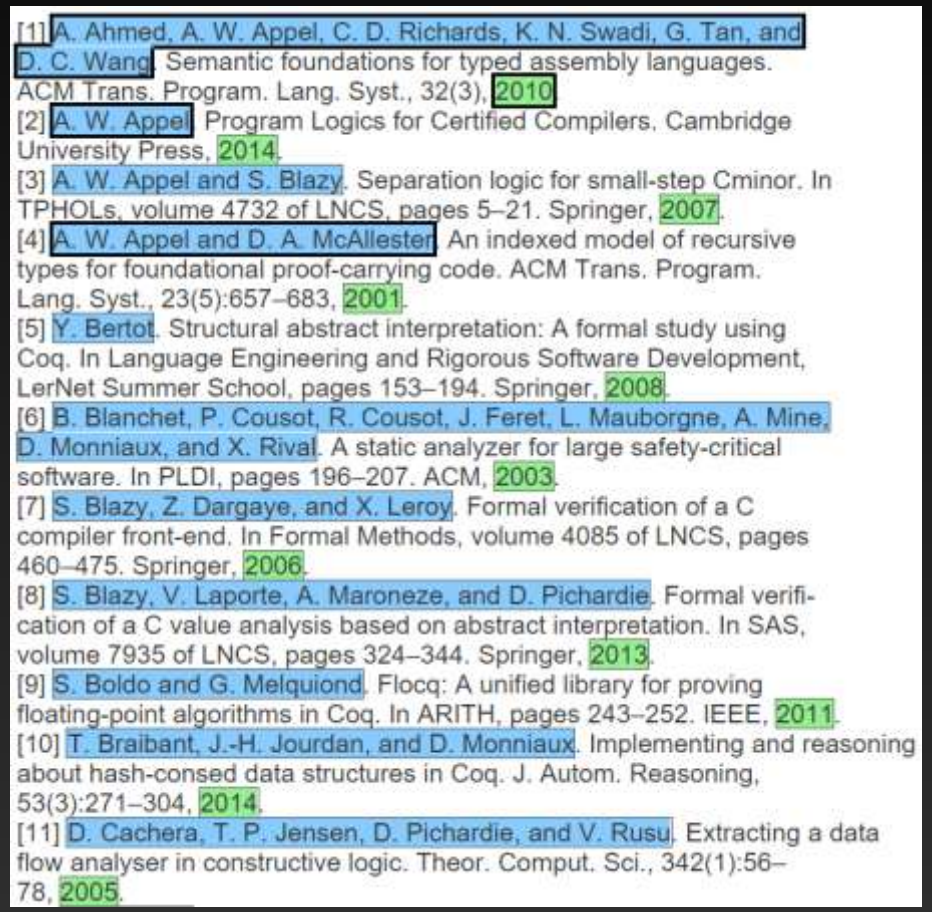
DSL design = Art + *Lots* of iterations

Inductive Specification

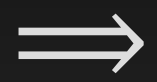
Input-Output Examples

input state σ	$\Longrightarrow$	*output value* out
`"206-279-6261"`	$\Longrightarrow$	`"(206) 279-6261"`
`"415.413.0703"`	$\Longrightarrow$	`"(415) 413-0703"`
`"(646) 408 6649"`	$\Longrightarrow$	`"(646) 408-6649"`

When one example is too many



[1] A. Ahmed, A. W. Appel, C. D. Richards, K. N. Swadi, G. Tan, and D. C. Wang. Semantic foundations for typed assembly languages. ACM Trans. Program. Lang. Syst., 32(3), 2010.
[2] A. W. Appel. Program Logics for Certified Compilers. Cambridge University Press, 2014.
[3] A. W. Appel and S. Blazy. Separation logic for small-step Cminor. In TPHOLs, volume 4732 of LNCS, pages 5–21. Springer, 2007.
[4] A. W. Appel and D. A. McAllester. An indexed model of recursive types for foundational proof-carrying code. ACM Trans. Program. Lang. Syst., 23(5):657–683, 2001.
[5] Y. Bertot. Structural abstract interpretation: A formal study using Coq. In Language Engineering and Rigorous Software Development, LerNet Summer School, pages 153–194. Springer, 2008.
[6] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Mine, D. Monniaux, and X. Rival. A static analyzer for large safety-critical software. In PLDI, pages 196–207. ACM, 2003.
[7] S. Blazy, Z. Dargaye, and X. Leroy. Formal verification of a C compiler front-end. In Formal Methods, volume 4085 of LNCS, pages 460–475. Springer, 2006.
[8] S. Blazy, V. Laporte, A. Maroneze, and D. Pichardie. Formal verification of a C value analysis based on abstract interpretation. In SAS, volume 7935 of LNCS, pages 324–344. Springer, 2013.
[9] S. Boldo and G. Melquiond. Flocq: A unified library for proving floating-point algorithms in Coq. In ARITH, pages 243–252. IEEE, 2011.
[10] T. Braibant, J.-H. Jourdan, and D. Monniaux. Implementing and reasoning about hash-consed data structures in Coq. J. Autom. Reasoning, 53(3):271–304, 2014.
[11] D. Cachera, T. P. Jensen, D. Pichardie, and V. Rusu. Extracting a data flow analyser in constructive logic. Theor. Comput. Sci., 342(1):56–78, 2005.



List of Authors	Publication Year
A. Ahmed, A. W. Appel, C. D. Richards, K. N. Swadi, G. Tan, and D. C. Wang	2010
A. W. Appel	2014
A. W. Appel and S. Blazy	2007
A. W. Appel and D. A. McAllester	2001
Y. Bertot	2008
B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Mine, D. Monniaux, and X. Rival	2003
S. Blazy, Z. Dargaye, and X. Leroy	2006
S. Blazy, V. Laporte, A. Maroneze, and D. Pichardie	2013
S. Boldo and G. Melquiond	2011
T. Braibant, J.-H. Jourdan, and D. Monniaux	2014
D. Cachera, T. P. Jensen, D. Pichardie, and V. Rusu	2005
A. Chlipala	2008
S. Cho, J. Kang, J. Choi, C.-K. Hur, and K. Yi	2013
P. Cousot and R. Cousot	1977
P. Cousot and R. Cousot	1979
P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Mine, and X. Rival	2009
P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Mine, D. Monniaux, and X. Rival	2006
A. Fouilhe, D. Monniaux, and M. Perin	2013
A. Fouilhe and S. Boulme	2014
D. Greenaway, J. Andronick, and G. Klein	2012

Inductive Specification

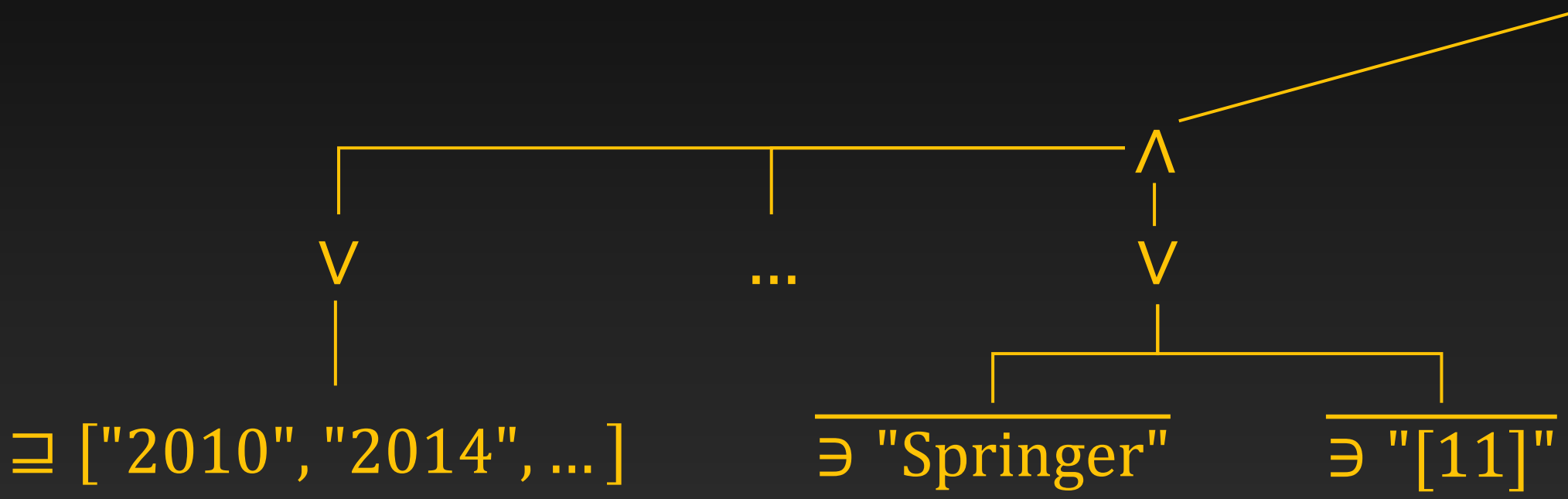
input state σ $\Longrightarrow$ *output constraint* $\varphi(out)$

[1] A. Ahmed, A. W. Appel, C. D. Richards, K. N. Swadi, G. Tan, and D. C. Wang. Semantic foundations for typed assembly languages. ACM Trans. Program. Lang. Syst., 32(3), 2010.
[2] A. W. Appel. Program Logics for Certified Compilers. Cambridge University Press, 2014.
[3] A. W. Appel and S. Blazy. Separation logic for small-step Cminor. In TPHOLs, volume 4732 of LNCS, pages 5–21. Springer, 2007.
[4] A. W. Appel and D. A. McAllester. An indexed model of recursive types for foundational proof-carrying code. ACM Trans. Program. Lang. Syst., 23(5):657–683, 2001.
[5] Y. Bertot. Structural abstract interpretation: A formal study using Coq. In Language Engineering and Rigorous Software Development, LerNet Summer School, pages 153–194. Springer, 2008.
[6] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Mine, D. Monniaux, and X. Rival. A static analyzer for large safety-critical software. In PLDI, pages 196–207. ACM, 2003.
[7] S. Blazy, Z. Dargaye, and X. Leroy. Formal verification of a C compiler front-end. In Formal Methods, volume 4085 of LNCS, pages 460–475. Springer, 2006.
[8] S. Blazy, V. Laporte, A. Maroneze, and D. Pichardie. Formal verification of a C value analysis based on abstract interpretation. In SAS, volume 7935 of LNCS, pages 324–344. Springer, 2013.
[9] S. Boldo and G. Melquiond. Flocq: A unified library for proving floating-point algorithms in Coq. In ARITH, pages 243–252. IEEE, 2011.

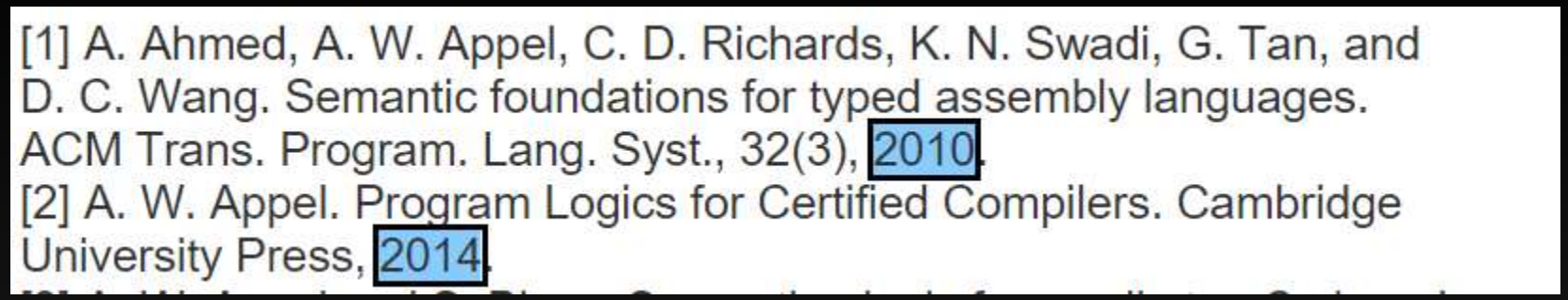
$\Longrightarrow$ $out \sqsupseteq$ ["2010", "2014", ...]

Inductive Specification

input state σ $\Longrightarrow$ *output constraint* $\varphi(out)$



Examples are ambiguous!



From:

all lines ending with “Number ∘ Dot”

“Space ∘ Number ∘ Dot”

starting with “Word ∘ Space ∘ CamelCase”

… and up to 10^{20} more candidates

Extract:

the first “Number” before a “Dot”

the last “Number” before a “Dot”

the last “Number” before a “Dot ∘ LineBreak”

the last “Number”

text between the last “Space” and the last “Dot”

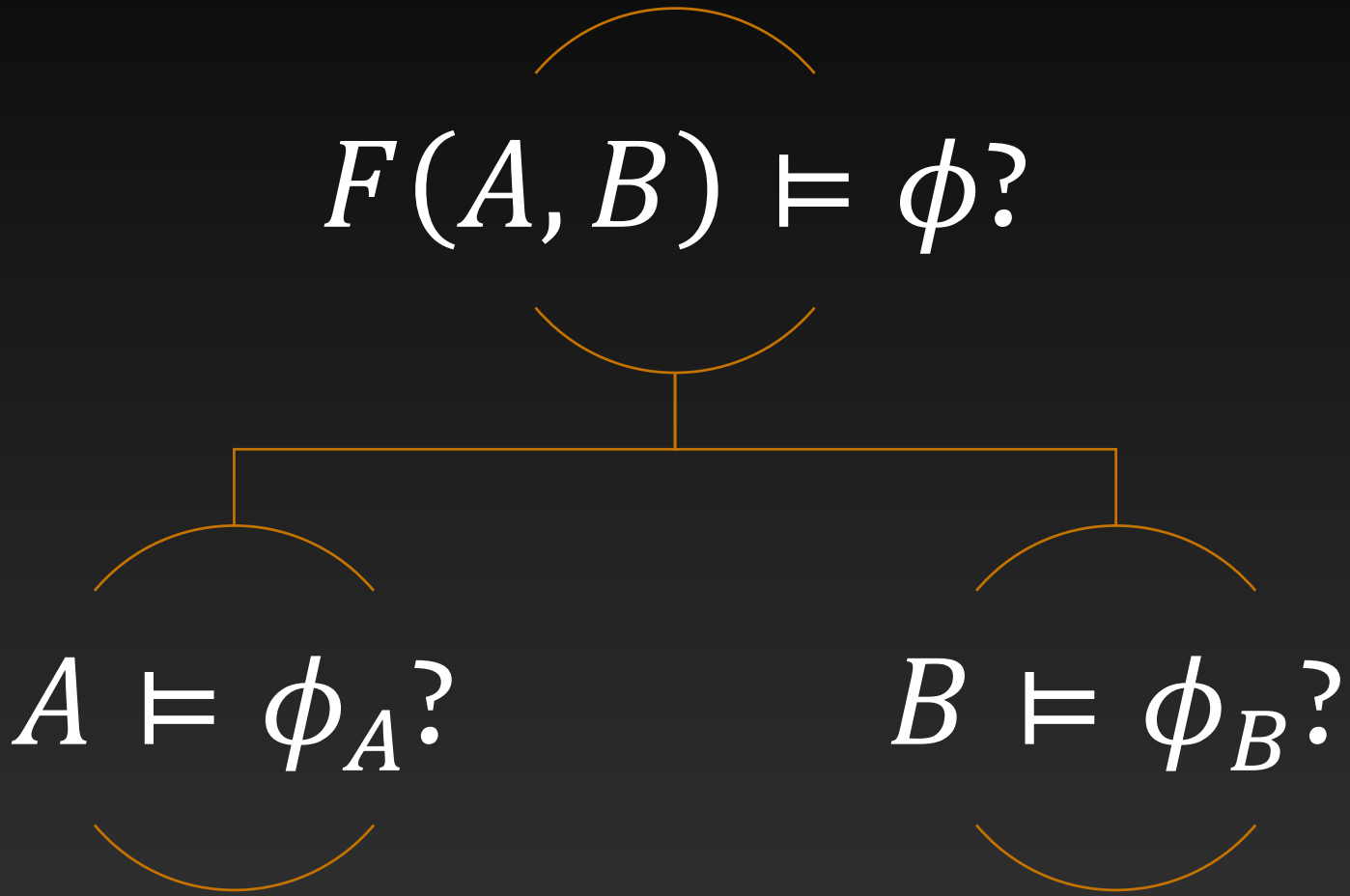
the first “Comma ∘ Space” and the last “Dot ∘ LineBreak”

One program is insufficient.

Program Set $\Longrightarrow$ Ranking
(Version Space Algebra) User interaction
Runtime correction
…

Synthesis Strategy

Observation 1: Inverse Semantics



Concat(F, E)

φ: { "Kathleen S. Fisher" ⟹ "Dr. Fisher"
"Bill Gates, Sr." ⟹ "Dr. Gates"

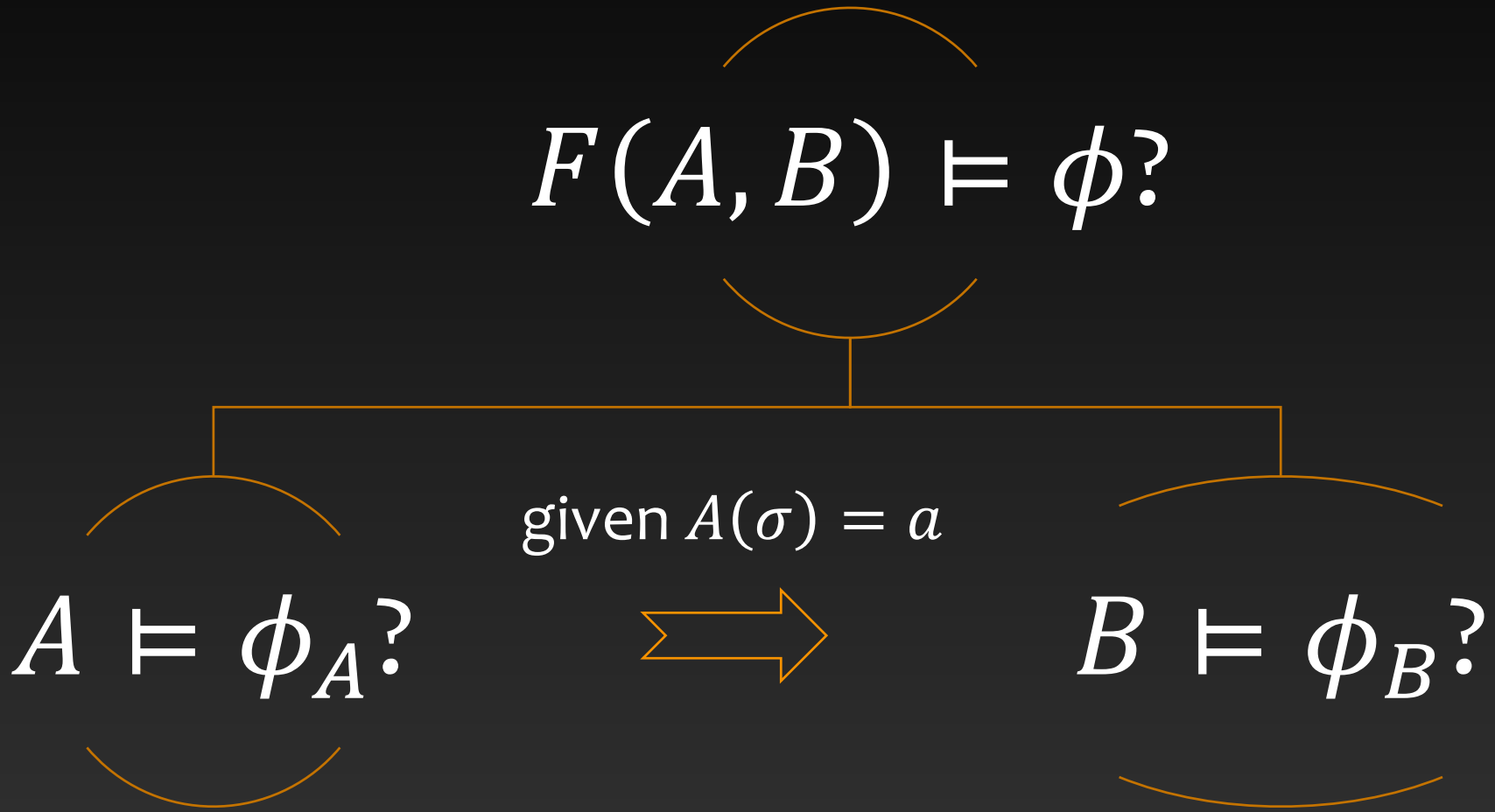
$\exists E$: Concat(F, E) satisfies φ if and only if F satisfies _____ ?

φ_f: { "Kathleen S. Fisher" ⟹ "D" ∨ "Dr" ∨ "Dr." ∨ "Dr. " ∨ "Dr. F" ∨ …
"Bill Gates, Sr." ⟹ "D" ∨ "Dr" ∨ "Dr." ∨ "Dr. " ∨ "Dr. G" ∨ …

$\exists$F: Concat(F, E) satisfies φ if and only if E satisfies _____ ?

F and E are not independent!

Observation 2: Skolemization



Concat(F, E)

φ: {"Kathleen S. Fisher" ⇒ "Dr. Fisher"; "Bill Gates, Sr." ⇒ "Dr. Gates"}

∃E: Concat(F, E) satisfies φ if and only if F satisfies _____ ?

φ_f: {"Kathleen S. Fisher" ⇒ "D" ∨ "Dr" ∨ "Dr." ∨ "Dr. " ∨ "Dr. F" ∨ …; "Bill Gates, Sr." ⇒ "D" ∨ "Dr" ∨ "Dr." ∨ "Dr. " ∨ "Dr. G" ∨ …}

Given an output of F, Concat(F, E) satisfies φ if and only if E satisfies _____ ?

F = {"Kathleen S. Fisher" ⇒ "Dr. "; "Bill Gates, Sr." ⇒ "Dr. "} ⟹ φ_E: {"Kathleen S. Fisher" ⇒ "Fisher"; "Bill Gates, Sr." ⇒ "Gates"}

Inverse Semantics + Skolemization = Witness Function

Witness function: $\varphi \mapsto \varphi_F$

$\exists E\colon \mathrm{Concat}(F, E)$ satisfies φ if and only if F satisfies _____ ?

Conditional witness function: $(\varphi \mid F(\sigma) = f) \mapsto \varphi_E$

Given an output of F, $\mathrm{Concat}(F, E)$ satisfies φ if and only if E satisfies _____ ?

Domain-Specific

Modular

No synthesis reasoning

Enable efficient deduction

Results

Unifies 10+ prior POPL/PLDI/… papers

- Lau, T., Domingos, P., & Weld, D. S. (2000). Version Space Algebra and its Application to Programming by Demonstration. In *ICML* (pp. 527–534).
- Kitzelmann, E. (2011). A combined analytical and search-based approach for the inductive synthesis of functional programs. *KI-Künstliche Intelligenz*, *25*(2), 179–182.
- Gulwani, S. (2011). Automating string processing in spreadsheets using input-output examples. In *POPL* (Vol. 46, p. 317).
- Singh, R., & Gulwani, S. (2012). Learning semantic string transformations from examples. *VLDB*, *5*(8), 740–751.
- Andersen, E., Gulwani, S., & Popovic, Z. (2013). A Trace-based Framework for Analyzing and Synthesizing Educational Progressions. In *CHI* (pp. 773–782).
- Yessenov, K., Tulsiani, S., Menon, A., Miller, R. C., Gulwani, S., Lampson, B., & Kalai, A. (2013). A colorful approach to text processing by example. In *UIST* (pp. 495–504).
- Le, V., & Gulwani, S. (2014). FlashExtract : A Framework for Data Extraction by Examples. In *PLDI* (p. 55).
- Barowy, D. W., Gulwani, S., Hart, T., & Zorn, B. (2015). FlashRelate: Extracting Relational Data from Semi-Structured Spreadsheets Using Examples. In *PLDI*.
- Kini, D., & Gulwani, S. (2015). FlashNormalize : Programming by Examples for Text Normalization. *IJCAI*.
- Osera, P.-M., & Zdancewic, S. (2015). Type-and-Example-Directed Program Synthesis. In *PLDI*.
- Feser, J., Chaudhuri, S., & Dillig, I. (2015). Synthesizing Data Structure Transformations from Input-Output Examples. In *PLDI*.
- …

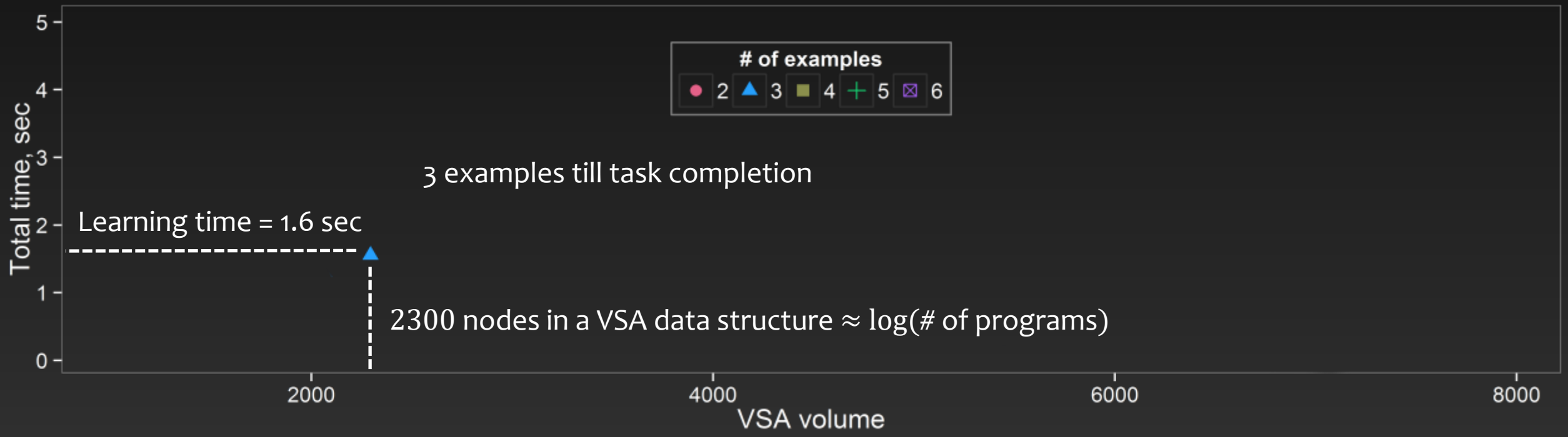
Program Synthesis meets Software Engineering

Project	Reference	Lines of Code		Development Time	
		Original	PROSE	Original	PROSE
Flash Fill	POPL 2010	12K	3K	9 months	1 month
Text Extraction	PLDI 2014	7K	4K	8 months	1 month
Text Normalization	IJCAI 2015	17K	2K	7 months	2 months
Spreadsheet Layout	PLDI 2015	5K	2K	8 months	1 month
Web Extraction	—	—	2.5K	—	1.5 months

Performance: 0.5 – 3X Original

More general ⇒ Slower Algorithmic advances ⇒ Faster

Example: FlashExtract

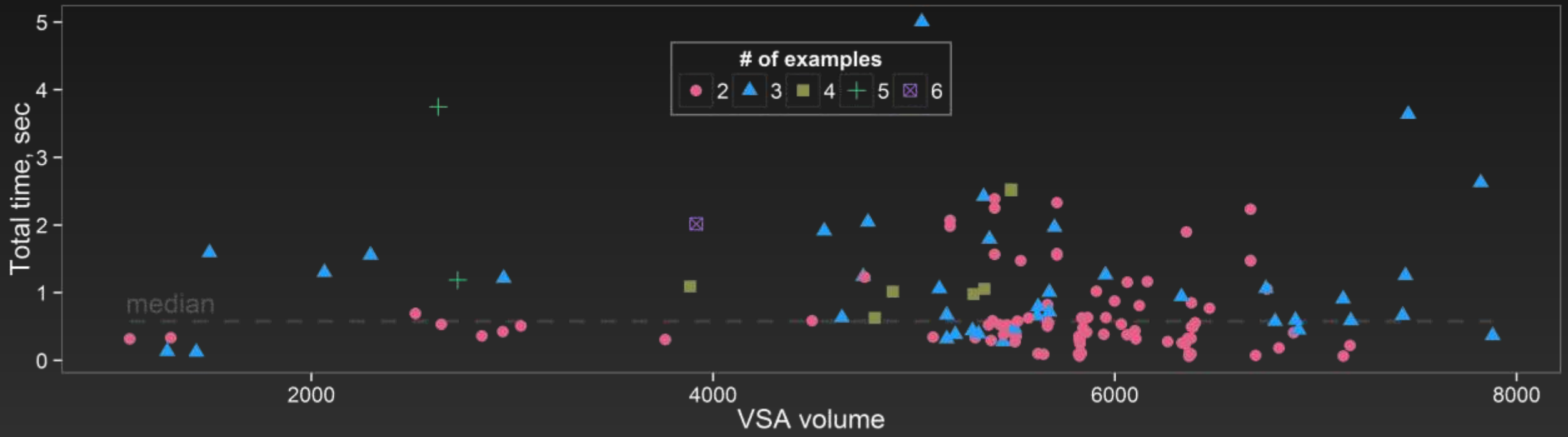


Performance: 0.5 – 3X Original

More general ⇒ Slower

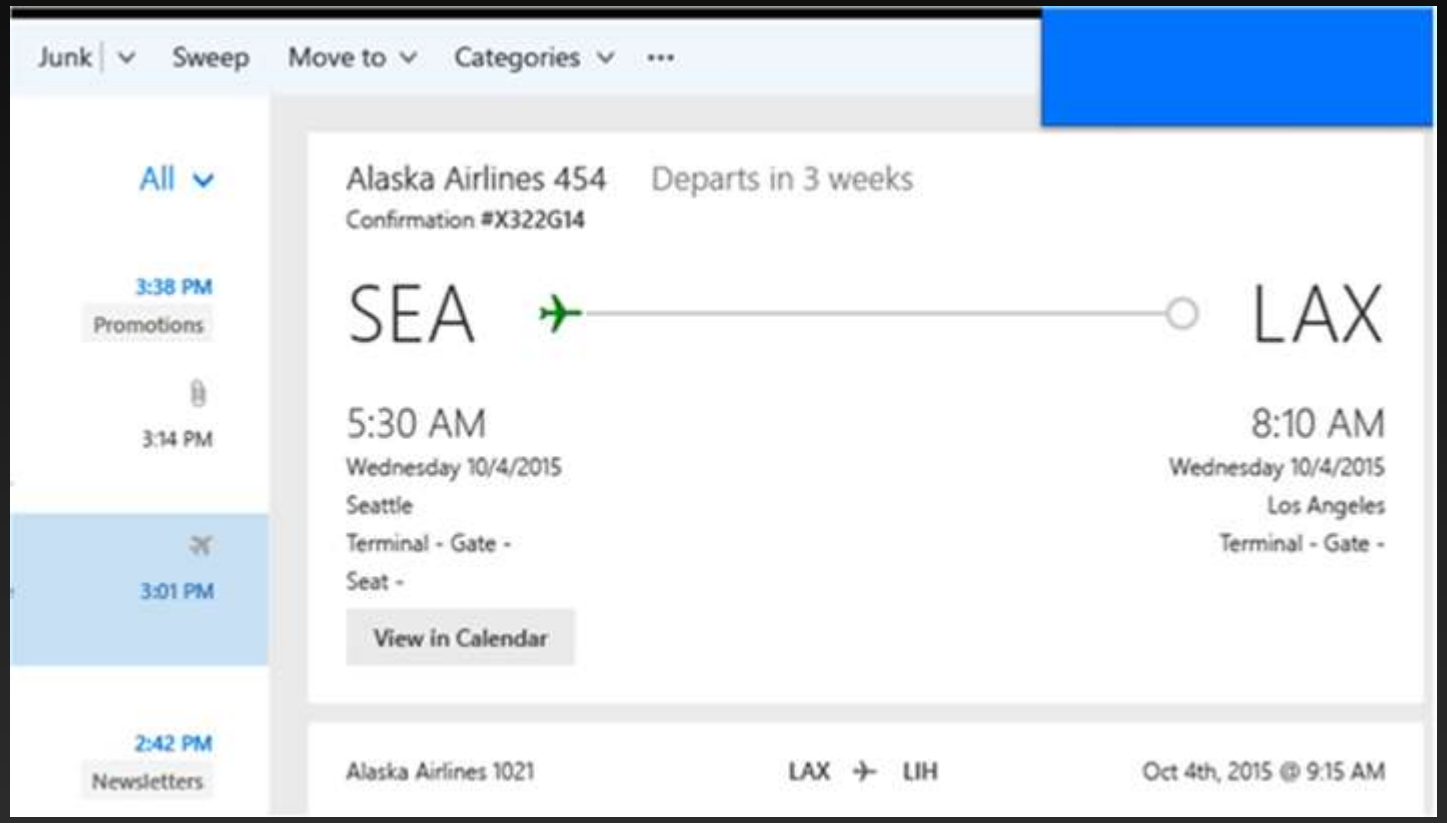
Algorithmic advances ⇒ Faster

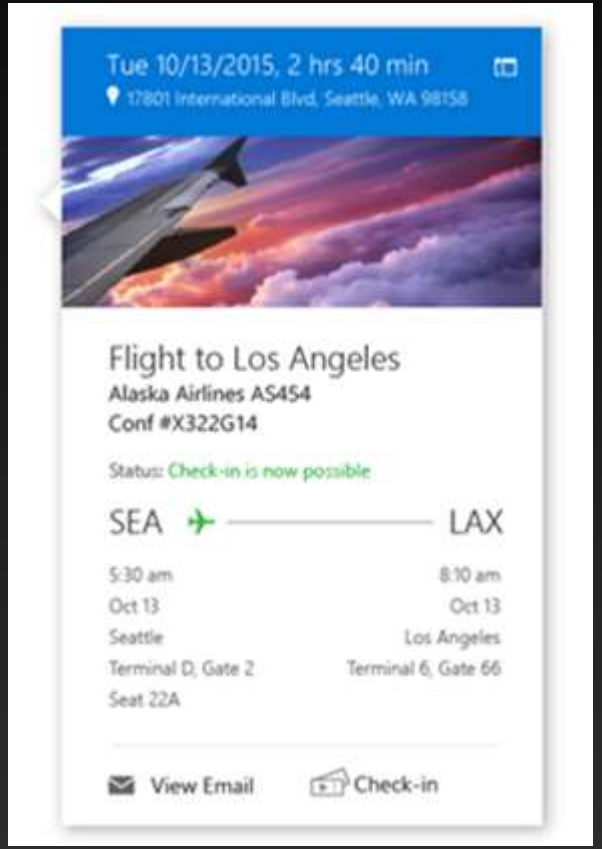
Example: FlashExtract



Applications

Email Parsing in Cortana





ConvertFrom-String in PowerShell

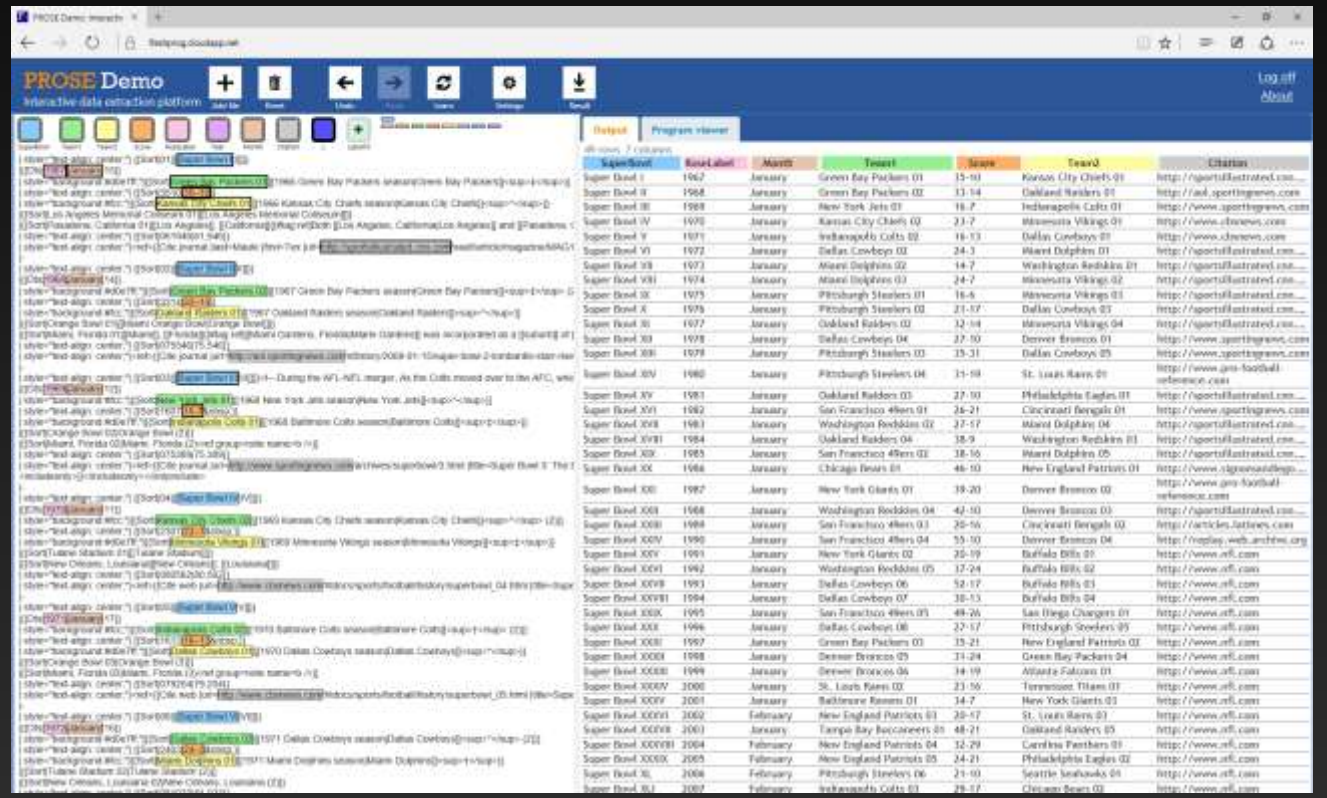
```
PS C:\> $template = @'

  1     {Time*:1} ms    <1 ms    <1 ms  cusred024ca901-tengige0-007-13.network.microsoft.com [{IP:10.31.196.2}]
 10     {Time*:2} ms     1 ms     1 ms  {IP:104.44.81.80}
 12     2 ms     2 ms     1 ms  a-0001.a-msedge.net [204.79.197.200]

'@

PS C:\> tracert bing.com | ConvertFrom-String -TemplateContent $template

Time IP
---- --
1    10.31.196.2
1    10.37.1.174
1    10.37.66.201
1    10.37.44.94
1    10.37.67.230
1    10.37.45.69
1    131.107.202.162
2    131.107.200.18
1    207.46.36.105
2    104.44.81.80
1    10.201.196.145
1    204.79.197.200
```

Research: https://microsoft.github.io/prose

Play: https://microsoft.github.io/prose/demo

Contact: prose-contact@microsoft.com

See our demo @ MSR table:



Thank you!

Questions?