#17: BlinkFill and Constraint-based Search

Sankha Narayan Guria

EECS 700: Introduction to Program Synthesis

BlinkFill

- Strengths? Weaknesses?
  - differences between FlashFill and BlinkFill language? which one is more expressive?

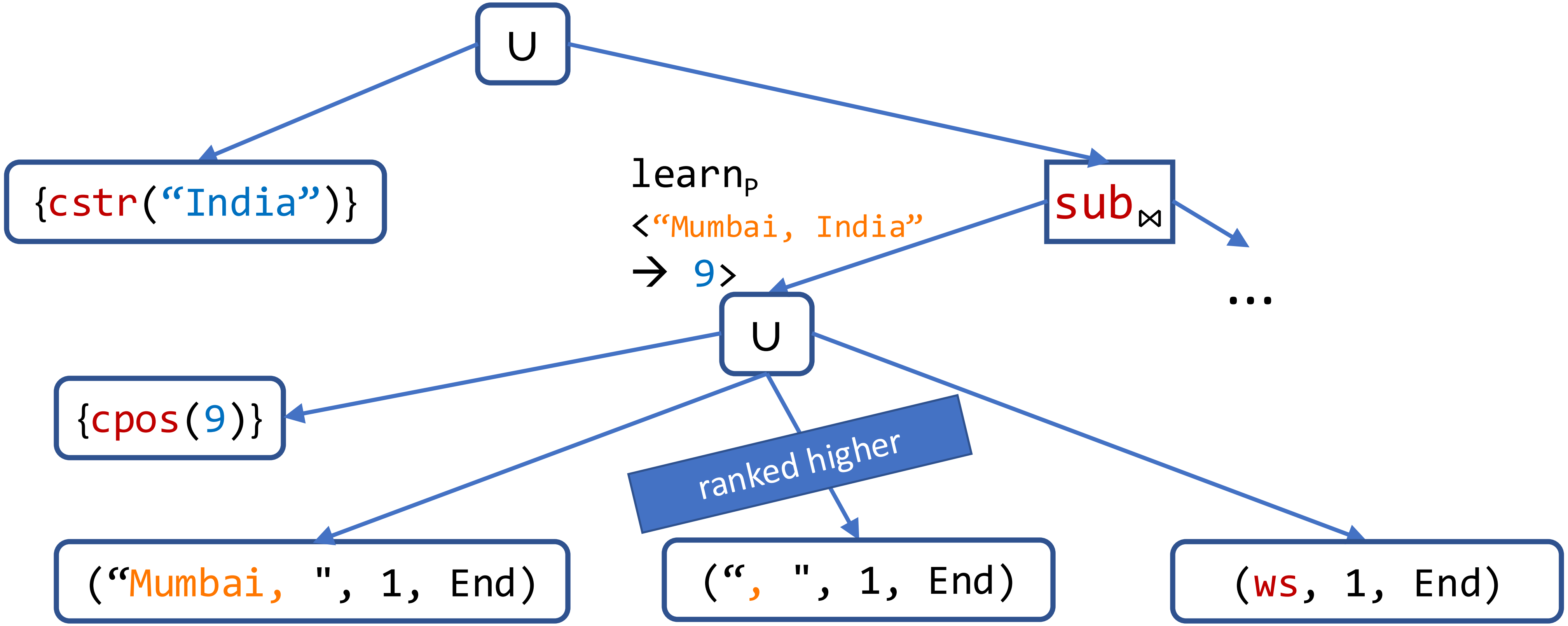
BlinkFill

- What does BlinkFill use as behavioral constraints? Structural constraints? Search strategy?
  - input-output examples (+ input examples); custom string DSL; VSA
- What is the main technical insight of BlinkFill wrt FlashFill?
  - BlinkFill uses the available inputs (with no outputs) to infer structure (segmentation) common to all inputs
  - it uses this structure to shrink the DAG and to rank substring expressions

Example

“Los Angeles, United States"

- learn_F <“Mumbai, India” → “India”>



BlinkFill

- Write a BlinkFill program that satisfies:
  - "Programming Language Design and Implementation (PLDI), 2019, Phoenix AZ" -> "PLDI 2019"
  - "Principles of Programming Languages (POPL), 2020, New Orleans LA" -> "POPL 2020"

  - Between first parentheses and between first and last comma:

```
Concat(SubStr(v1, ("(", 1, End), (")",1, Start)),
       SubStr(v1, (",", 1, End), (",", -1, Start)))
```

BlinkFill

- Could we extend the algorithm to support sequences of tokens?
  - More expressive:
    - "Programming Language Design and Implementation: PLDI 2019" -> "PLDI 2019"
    - “POPL 2020 started on January 22" -> "POPL 2020“
    - `SubStr(v1, (C ws d, 1, Start), (C ws d, 1, End))`
  - Each edge of the single-string IDG would have more labels
  - Extra edges from 0 and to the last node
  - More edges left after intersection (might be a problem, but unclear)
  - Need fewer primitive tokens (no need for ProperCase)

On to today’s topic

The problem statement

Behavioral constraints = examples / first-order formula

Search strategy?

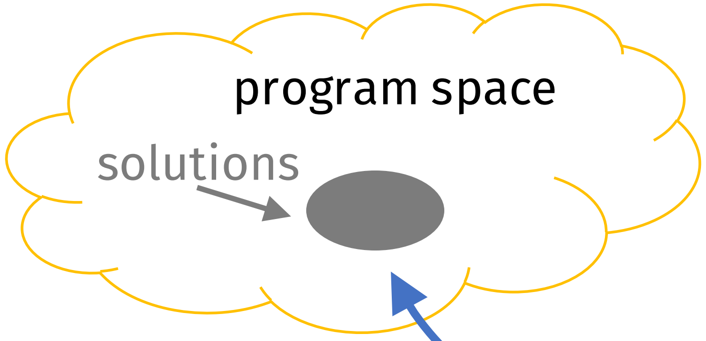
Enumerative
Representation-based
Stochastic
Constraint-based

Structural constraints = we'll see...

Constraint-based search

- **Idea:** encode the synthesis problem as a SAT/SMT problem and let a solver deal with it

What is an encoding?



parameter space

$$C = \{c_1, \ldots, c_n\}$$

$$\{C \mid \mathrm{wf}(C)\}$$

well-formedness constraint

$$\{C \mid \mathrm{wf}(C) \wedge \mathrm{sem}(C)\}$$

semantic correctness constraint

$$\mathrm{sem}(C) = \bigwedge_{\langle i,o\rangle \in E} \varphi(C, i, o)$$

C denotes a program that gives output *o* on input *i*

How to define an encoding

- Define the parameter space $C = \{c_1, \ldots, c_n\}$
  - `decode` : C → Prog (might not be defined for all C)
- Define a formula wf$(c_1, \ldots, c_n)$
  - that holds iff `decode`[C] is defined
- Define a formula $\varphi(c_1, \ldots, c_n, i, o)$
  - that holds iff (`decode`[C])(i) = o

Constraint-based search

```
constraint-based (wf, φ, E = [i → o]) {
  match SAT(wf(C) ∧ ⋀⟨i,o⟩∈E φ(C, i, o)) with
    Unsat -> return “No solution”
    Model C* -> return decode[C*]
}
```

Find a satisfying assignment for $c_1, \ldots, c_n$ (i and o are fixed)

SAT encoding: example

program space

`{x, x & 1}`

x is a two-bit word
($x = x_h x_l$)

`E = [11 → 01]`

parameter space

$$C = \{c : \mathrm{Bool}\}$$

`decode[0] → x`
`decode[1] → x & 1`

$$\mathrm{wf}(c) \equiv \top$$

$$\varphi(c, i_h, i_l, o_h, o_l) \equiv (\neg c \;\Rightarrow\; o_h = i_h \wedge o_l = i_l)$$
$$\wedge\; (c \;\Rightarrow\; o_h = 0 \wedge o_l = i_l)$$

$$\mathrm{SAT}(\varphi(c, 1, 1, 0, 1))$$

$$\mathrm{SAT}((\neg c \;\Rightarrow\; 0 = 1 \wedge 1 = 1) \wedge (c \;\Rightarrow 0 = 0 \wedge 1 = 1)) \xrightarrow{\text{SAT solver}} \texttt{Model \{c→1\}}$$

return `decode[1]` i.e. `x & 1`

SMT encoding: example

program space

`x + N | x * N`

N is an integer literal
x is an integer input

`E = [2 → 9]`

parameter space

$C = \{c_{op}: \mathrm{Bool}, c_N: \mathrm{Int}\}$

`decode[0,N] → x + N`
`decode[1,N] → x * N`

$\mathrm{wf}(c_{op}, c_N) \equiv \top$

$$\varphi(c_{op}, c_N, i, o) \equiv (\neg c_{op} \Rightarrow o = i + c_N) \wedge (c_{op} \Rightarrow o = i * c_N)$$

$$\mathrm{SAT}(\varphi(c_{op}, c_N, 2, 9))$$

$$\mathrm{SAT}((\neg c_{op} \Rightarrow 9 = 2 + c_N) \wedge (c_{op} \Rightarrow 9 = 2 * c_N))$$

SMT solver →

Model $\{c_{op} \to 0, c_N \to 7\}$

return `decode[0,7]` i.e. `x + 7`

What is a good encoding?

- Sound
  - if $\mathrm{wf}(C) \wedge \mathrm{sem}(C)$ then `decode`[C] is a solution
- Complete
  - if `decode`[C] is a solution then $\mathrm{wf}(C) \wedge \mathrm{sem}(C)$
- Small parameter space
  - avoid symmetries
- Solver-friendly
  - decidable logic, compact constraint

DSL limitations

- Program space can be parameterized with a finite set of parameters
  - Counterexample:

```
L ::= sort(L)  |  L[N..N]
   |  L + L  |  [N]  |  x
N ::= find(L,N)  |  0
```

  - Workaround

```
L0 ::= x   L1 ::= sort(L0)  |  L0[N0..N0]
N0 ::= 0        |  L0 + L0  |  [N0]  |  L0
           N1 ::= find(L0,N0)  |  N0
```

- Program semantics $\varphi(C, i, o)$ is expressible as a (decidable) SAT/SMT formula
  - Counterexample

Comparison of search strategies