Lecture 7
Constraint-based search

Nadia Polikarpova

Logistics

Project proposals

- Due this Friday (Oct 26)
- Upload to the Proposals directory inside the shared Google folder
- Can be a Google Doc or a PDF
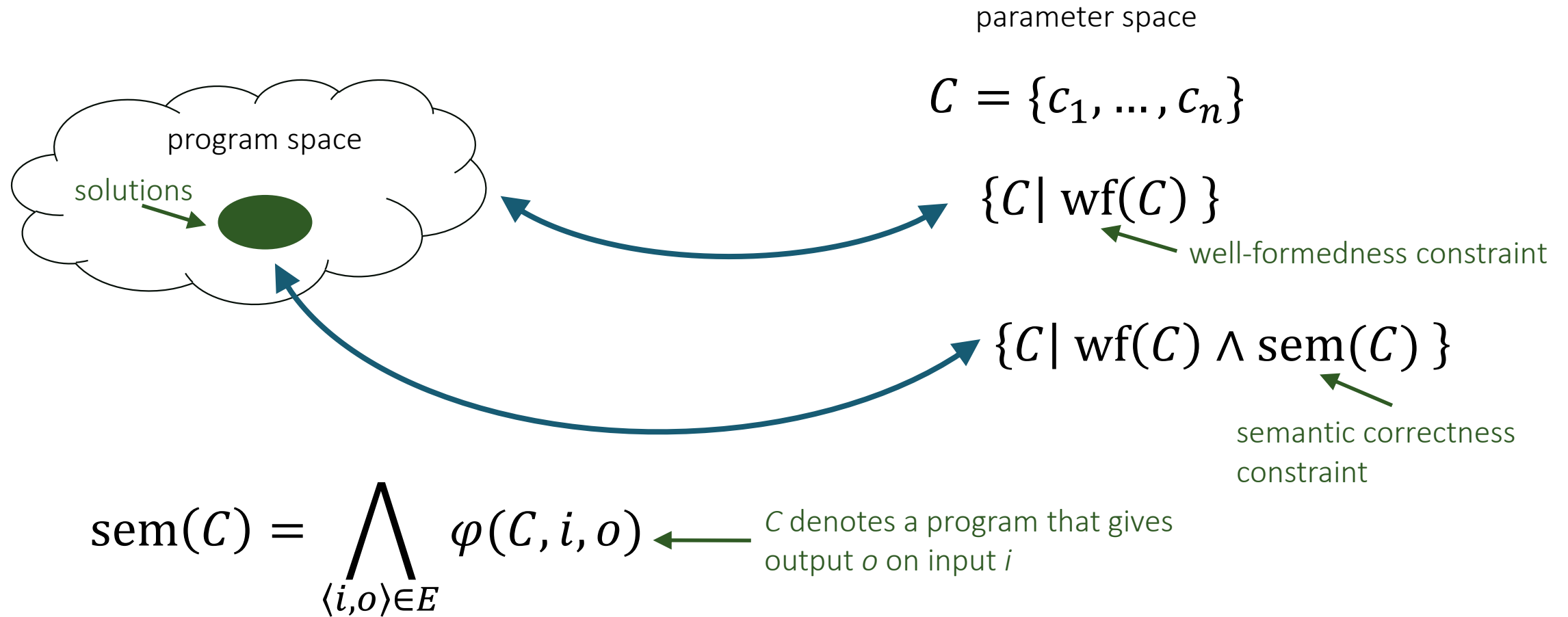- File name must be "Team-N", where N is your team ID

The problem statement

Behavioral constraints = examples

```
[1,4,7,2,0,6,9,2,5]  →  [1,2,4,7,0]
[0] → [0]
[5,1] → [1,5,0]
```

Search strategy?

Enumerative
Stochastic
Representation-based
Constraint-based

Structural constraints = we'll see...
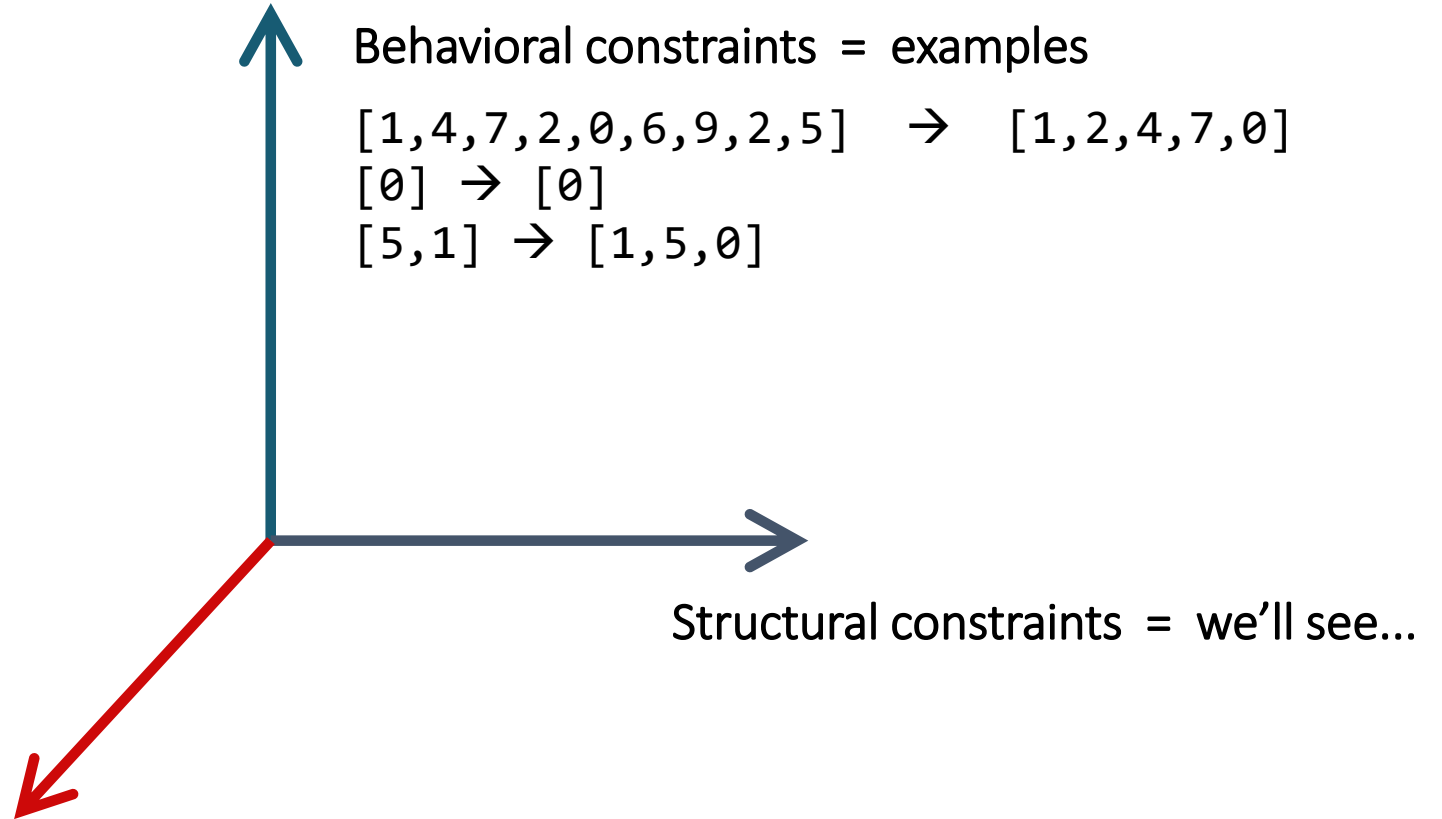
Constraint-based search

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

What is an encoding?

parameter space

program space

solutions

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

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

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

- `encode : Prog` → `C`
- `decode : C` → `Prog` (might not be defined for all C)

Define a formula $\text{wf}(c_1, \ldots, c_n)$
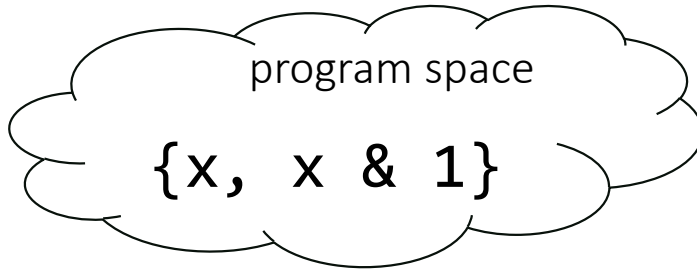
- that holds iff `decode[C]` is a "well-formed" program

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, ..., c_n
(i and o are fixed)

SAT encoding: example

program space

{x, x & 1}

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

$E = [11 \rightarrow 01]$

parameter space

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

decode[0] $\rightarrow$ x
decode[1] $\rightarrow$ x & 1

$\text{wf}(c) \equiv T$

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

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

$\text{SAT}((\neg c \Rightarrow 0 = 1 \wedge 1 = 1) \wedge (c \Rightarrow 0 = 0 \wedge 1 = 1))$

SAT solver

Model {c$\rightarrow$1}

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

SMT encoding: example

program space

x + N | x * N

N is an in integer literal
x is an integer

E = [2 → 9]

parameter space

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

decode[0,N] → x + N
decode[1,N] → x * N
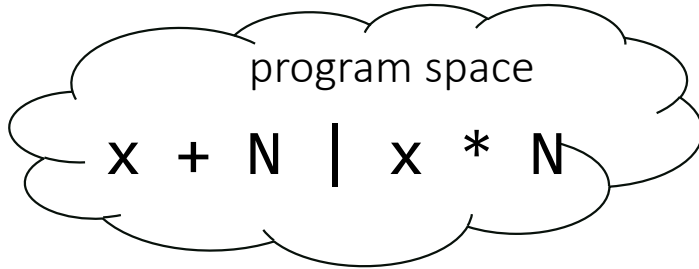
$$\mathrm{wf}(c_{op}, c_N) \equiv \top$$

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

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

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

SMT solver

Model {c_{op}→0, c_N→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

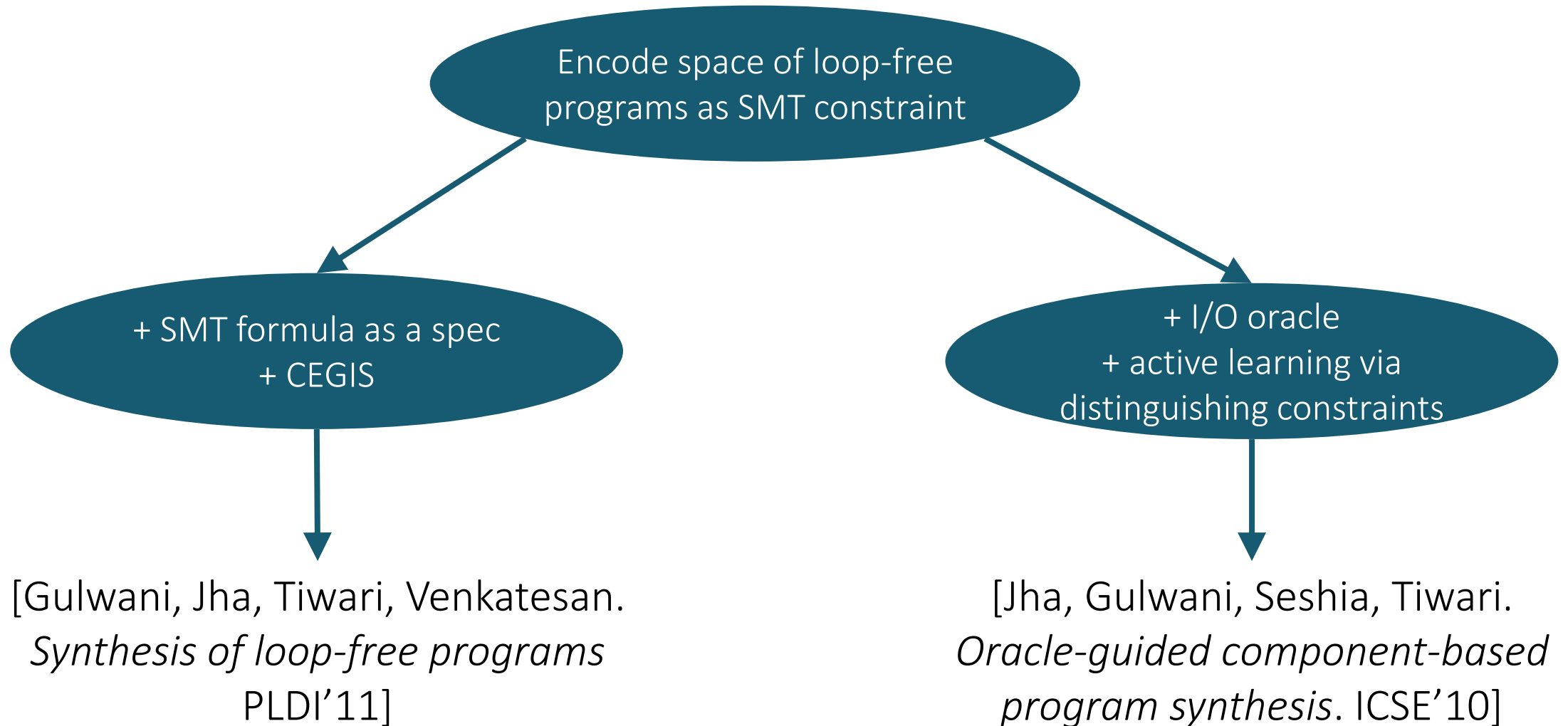Program space can be parameterized with a finite set of parameters

- 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

Brahma

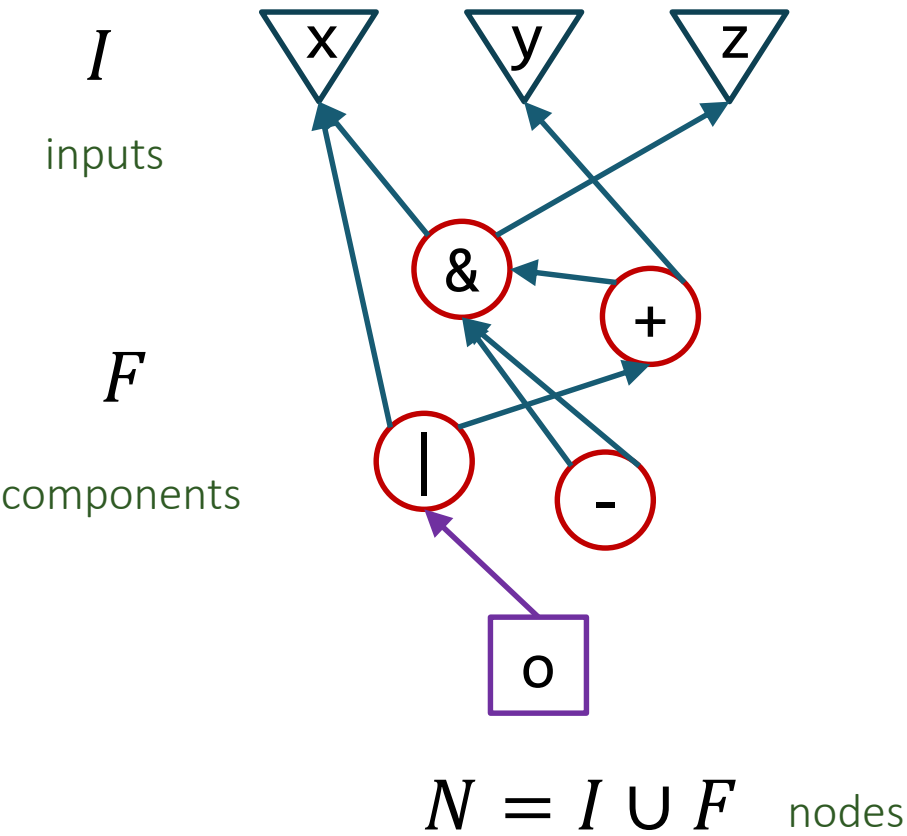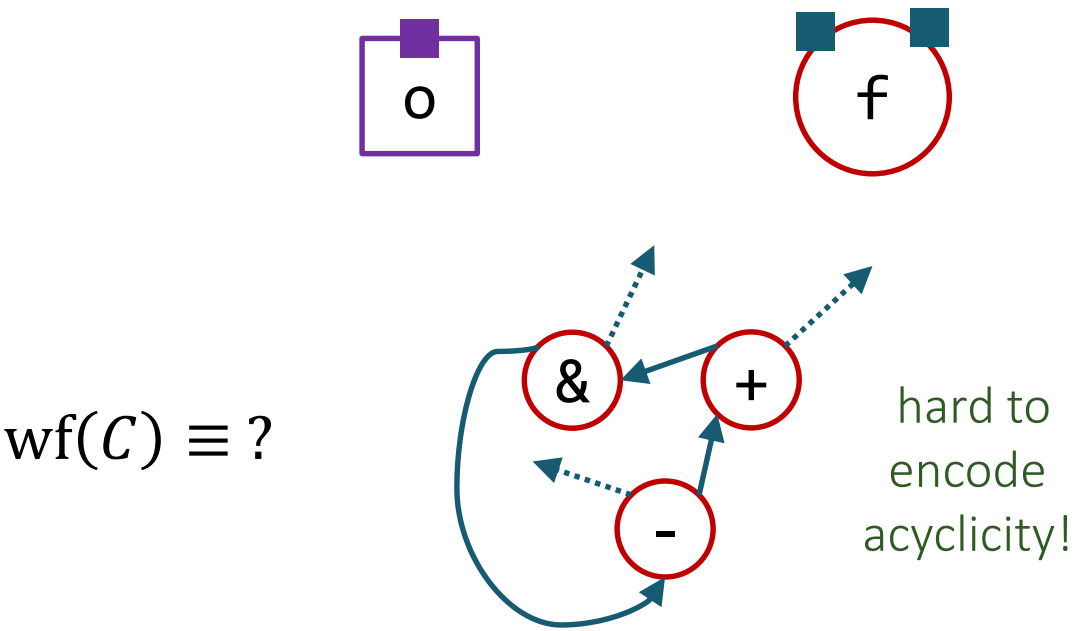Encode space of loop-free programs as SMT constraint

+ SMT formula as a spec
+ CEGIS

+ I/O oracle
+ active learning via distinguishing constraints

[Gulwani, Jha, Tiwari, Venkatesan.
Synthesis of loop-free programs
PLDI'11]

[Jha, Gulwani, Seshia, Tiwari.
*Oracle-guided component-based
program synthesis*. ICSE'10]

Brahma encoding

program space = DAG

parameter space

I

inputs

F

components

$$C = \{c_o : N\} \cup \bigcup_{f \in F} \{c_1^f, c_2^f : N\}$$

$\text{wf}(C) \equiv ?$

hard to encode acyclicity!
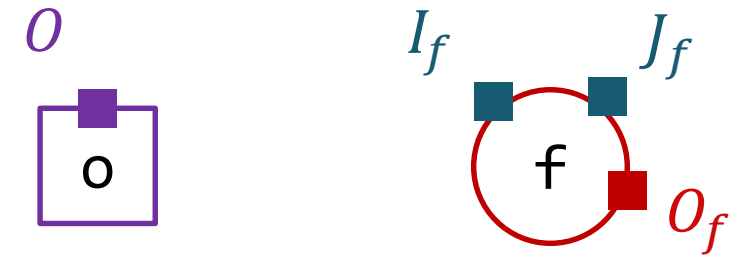
$N = I \cup F$ nodes

Brahma encoding: take 2

[Jha et al. '10]

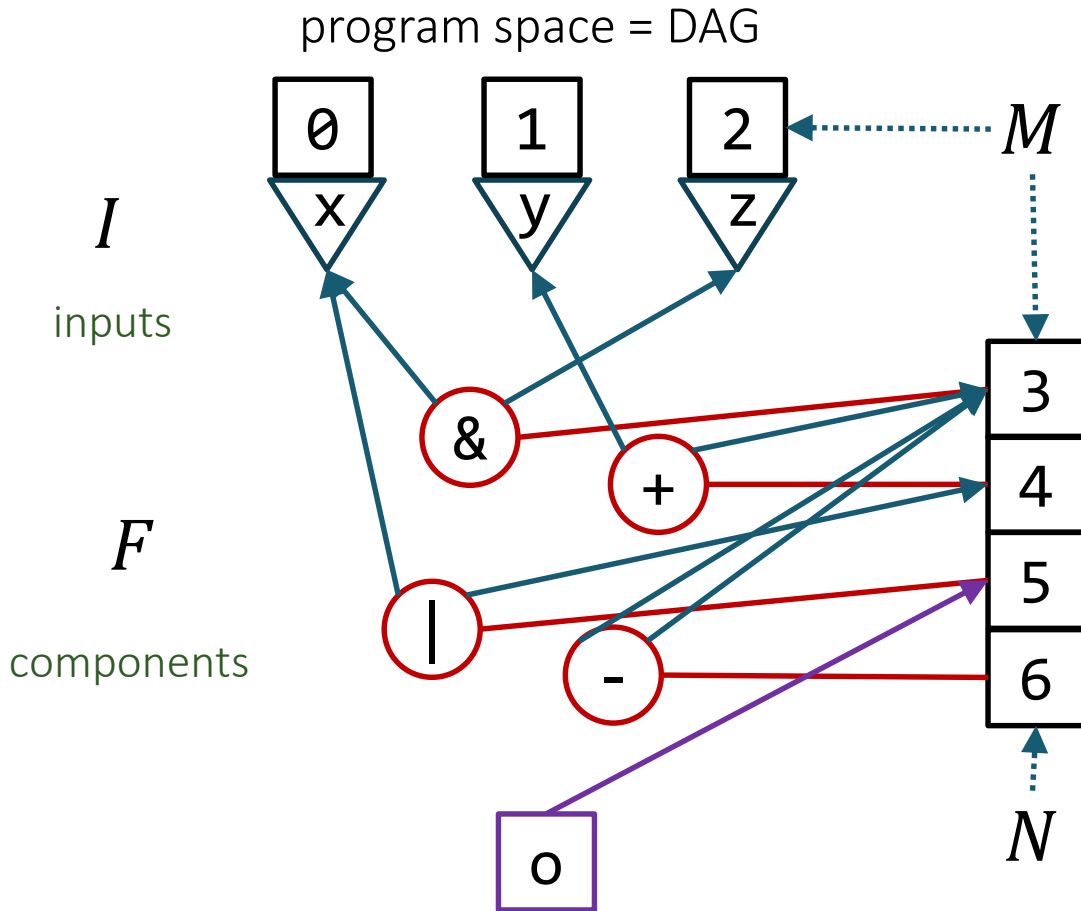program space = DAG
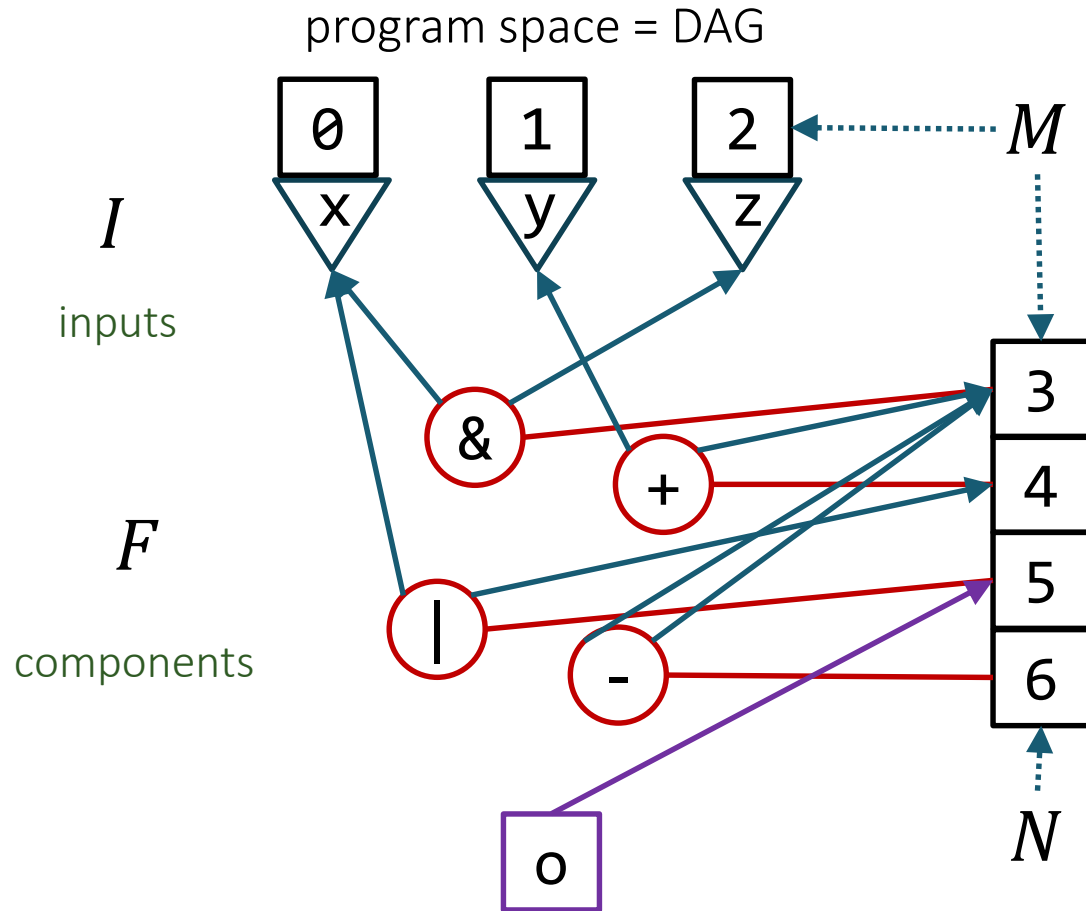
parameter space

$$C = \{c_o : \text{Int}\} \cup \bigcup_{f \in F} \{c_{O_f}, c_{I_f}, c_{J_f} : \text{Int}\}$$

$$\text{wf}(C) \equiv c_O \in M \wedge \bigwedge_{f \in F} c_{O_f} \in N \wedge c_{I/J_f} \in M$$

Brahma encoding: take 2

[Jha et al. '10]

program space = DAG

parameter space

$$C = \{c_o : \mathrm{Int}\} \cup \bigcup_{f \in F} \{c_{O_f}, c_{I_f}, c_{J_f} : \mathrm{Int}\}$$

$$P = \bigcup_{f \in F} \{I_f, J_f\} \qquad R = \bigcup_{f \in F} \{O_f\}$$

$$\varphi(C, I, o) \equiv \exists P, R. \bigwedge_{f \in F} O_f = F(I_f, J_f)$$

$$\wedge \bigwedge_{x \in P \cup R \cup I \cup \{O\}} c_x = c_y \Rightarrow x = y$$

Comparison of search strategies