

Lecture 8

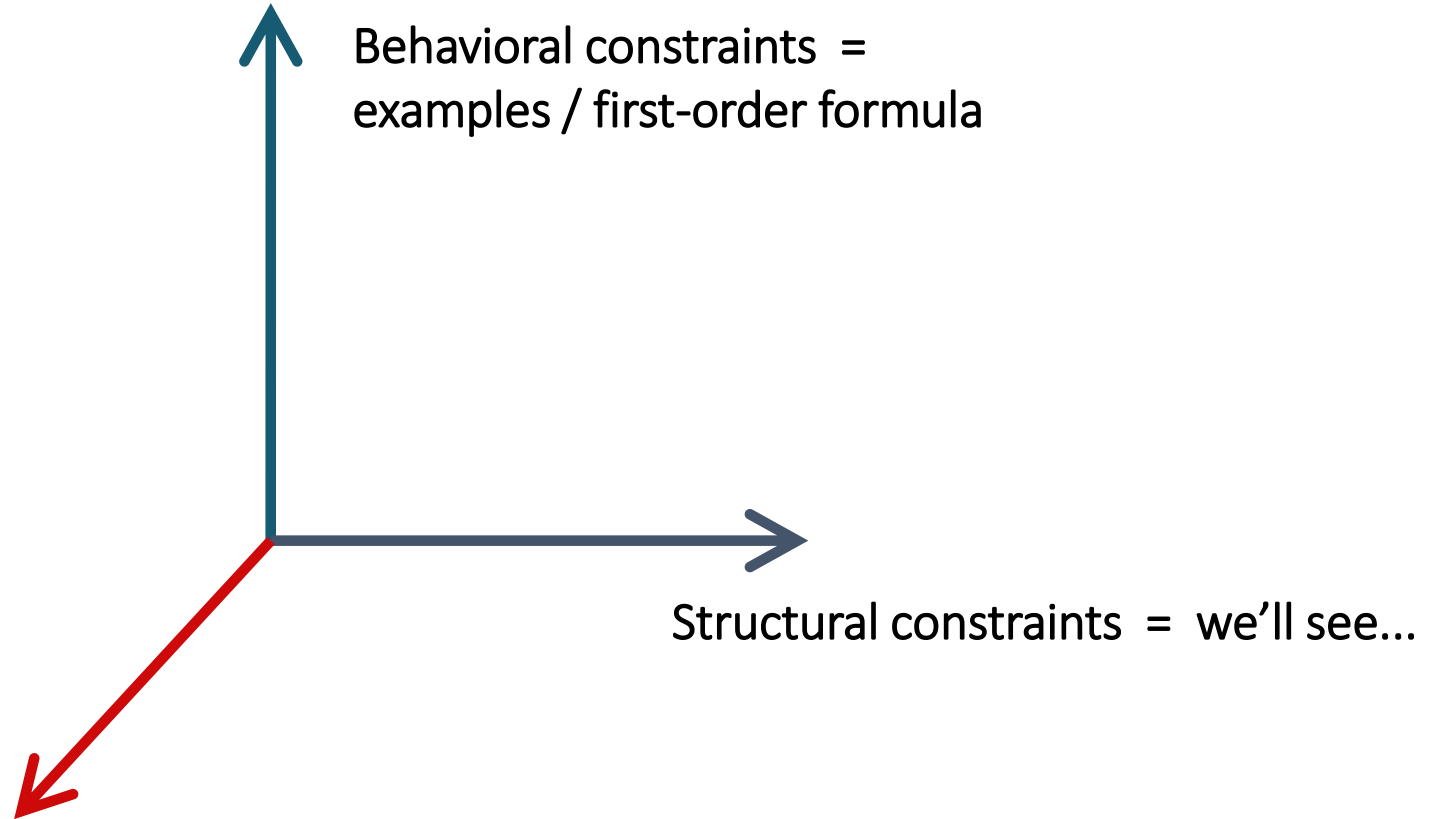
Constraint-based search

Nadia Polikarpova

The problem statement

Search strategy?

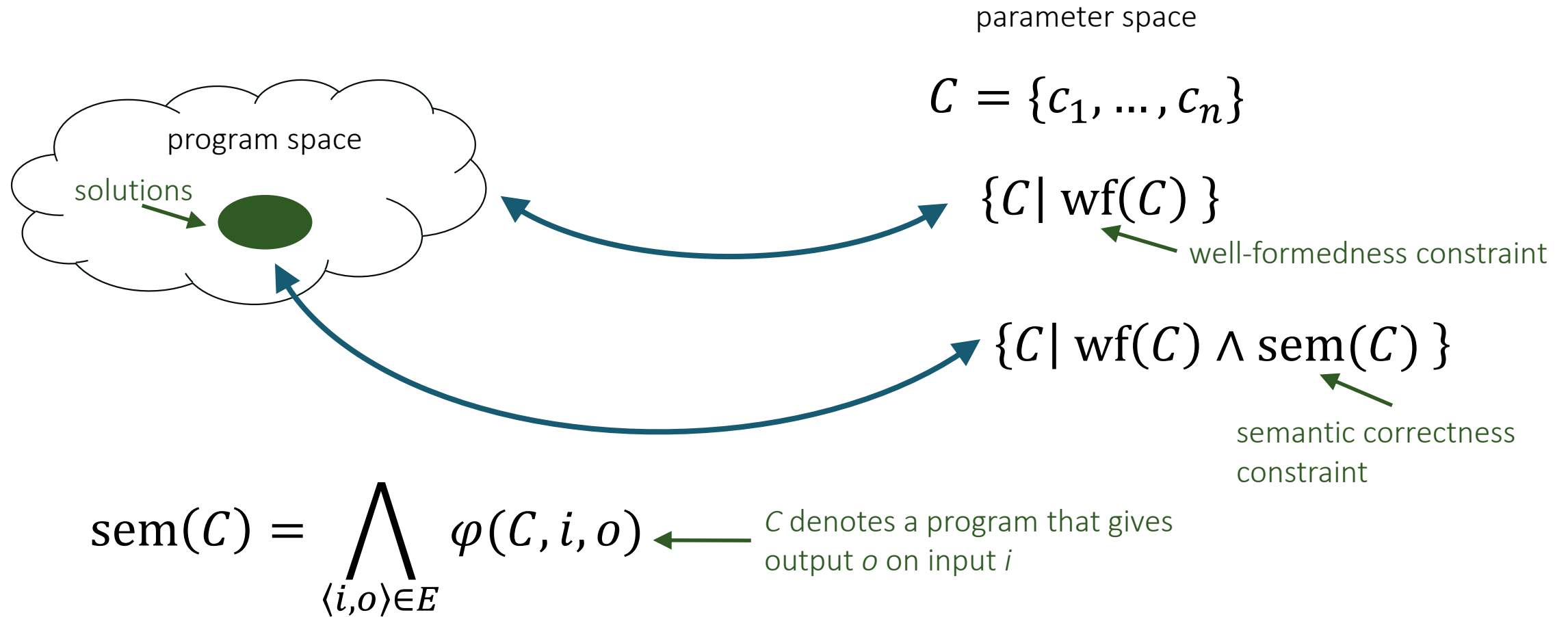
Enumerative
Representation-based
Stochastic
Constraint-based



Constraint-based search

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

What is an encoding?



How to define an encoding

Define the parameter space $\mathcal{C} = \{c_1, \dots, c_n\}$

- **encode** : $\text{Prog} \rightarrow \mathcal{C}$
- **decode** : $\mathcal{C} \rightarrow \text{Prog}$ (might not be defined for all \mathcal{C})


Define a formula $\text{wf}(c_1, \dots, c_n)$

- that holds iff **decode**[\mathcal{C}] is a “well-formed” program

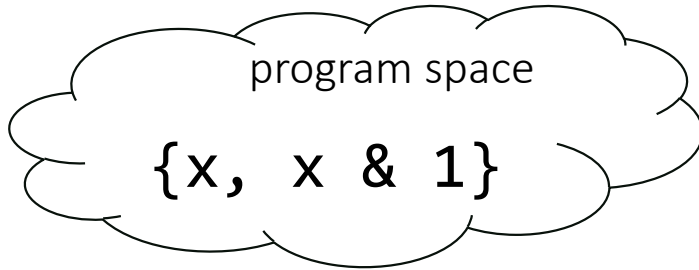
Define a formula $\varphi(c_1, \dots, c_n, i, o)$

- that holds iff $(\text{decode}[\mathcal{C}])(i) = o$

Constraint-based search

```
constraint-based (wf,  $\varphi$ ,  $E = [i \rightarrow o]$ ) {  
  match SAT(wf( $C$ )  $\wedge \bigwedge_{\langle i, o \rangle \in E} \varphi(C, i, o)$ ) with  Find a satisfying assignment  
    Unsat -> return "No solution" for  $c_1, \dots, c_n$   
    Model  $C^*$  -> return decode[ $C^*$ ] ( $i$  and  $o$  are fixed)  
}
```

SAT encoding: example



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

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

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

$$\text{SAT}((\neg c \Rightarrow 0 = 1 \wedge 1 = 1) \wedge (c \Rightarrow 0 = 0 \wedge 1 = 1)) \xrightarrow{\text{SAT solver}} \text{Model } \{c \rightarrow 1\}$$

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

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

$$E = [11 \rightarrow 01]$$

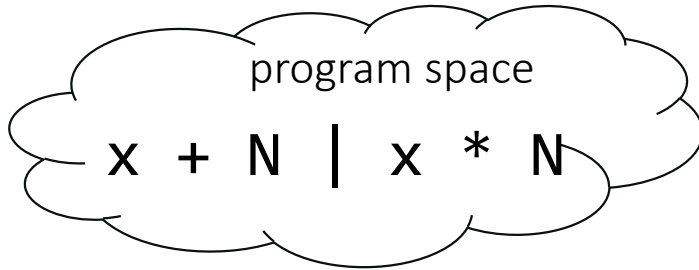
parameter space

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

$$\text{decode}[0] \rightarrow x$$

$$\text{decode}[1] \rightarrow x \& 1$$

SMT encoding: example



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

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

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

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

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

N is an integer literal
 x is an integer input

$$E = [2 \rightarrow 9]$$

parameter space

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

$$\text{decode}[0, N] \rightarrow x + N$$

$$\text{decode}[1, N] \rightarrow x * N$$

SMT solver



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

What is a good encoding?

Sound

- if $\text{wf}(C) \wedge \text{sem}(C)$ then $\text{decode}[C]$ is a solution

Complete

- if $\text{decode}[C]$ is a solution then $\text{wf}(C) \wedge \text{sem}(C)$

Small parameter space

- avoid symmetries

Solver-friendly

- decidable logic, compact constraint

DSL limitations

Program space can be parameterized with a finite set of parameters

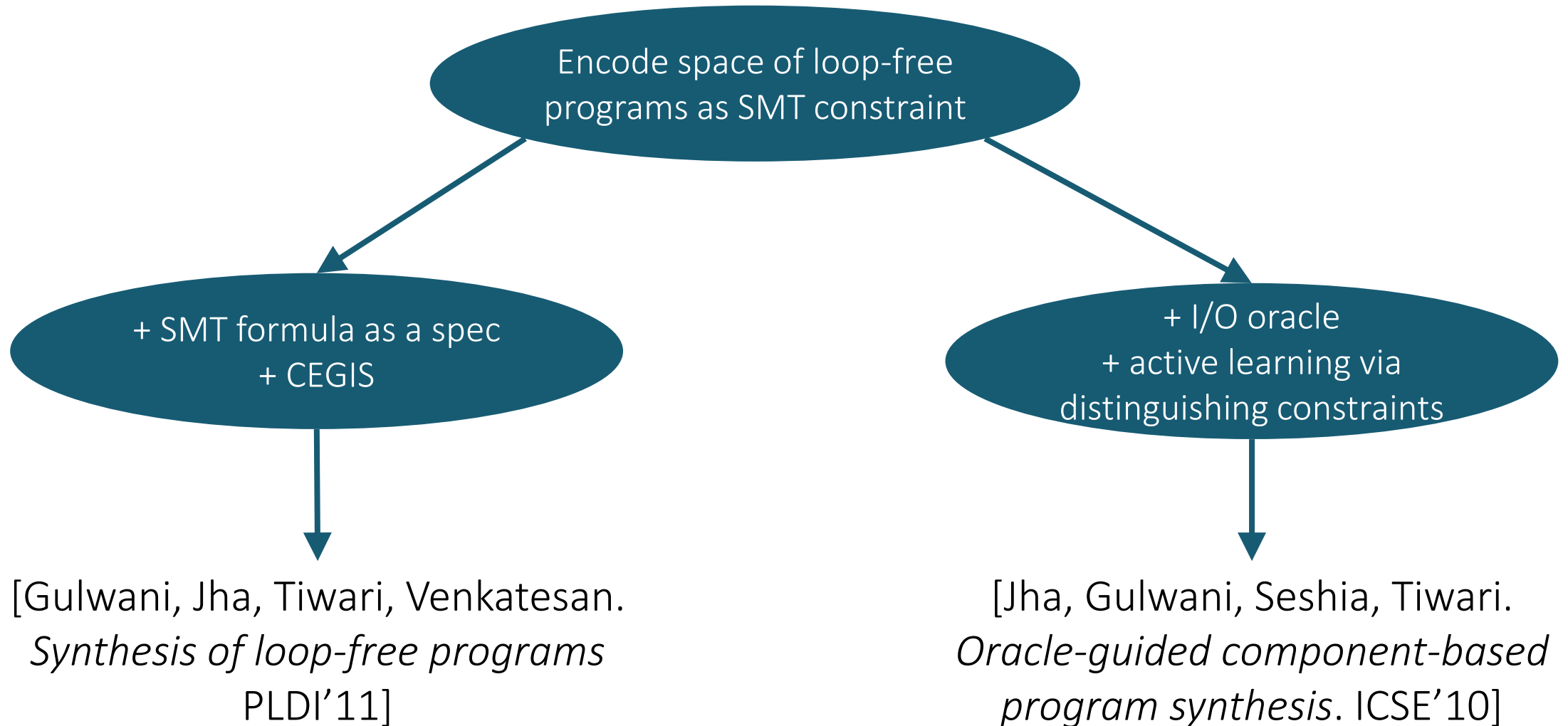
- Counterexample:
$$\begin{array}{lcl} L & ::= & \text{sort}(L) \quad | \quad L[N..N] \\ & & | \quad L + L \quad | \quad [N] \quad | \quad x \\ N & ::= & \text{find}(L, N) \quad | \quad \emptyset \end{array}$$

- Workaround
$$\begin{array}{lcl} L0 & ::= & x \quad L1 ::= \text{sort}(L0) \quad | \quad L0[N0..N0] \\ N0 & ::= & \emptyset \quad | \quad L0 + L0 \quad | \quad [N0] \quad | \quad L0 \\ & & N1 ::= \text{find}(L0, N0) \quad | \quad N0 \end{array}$$

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

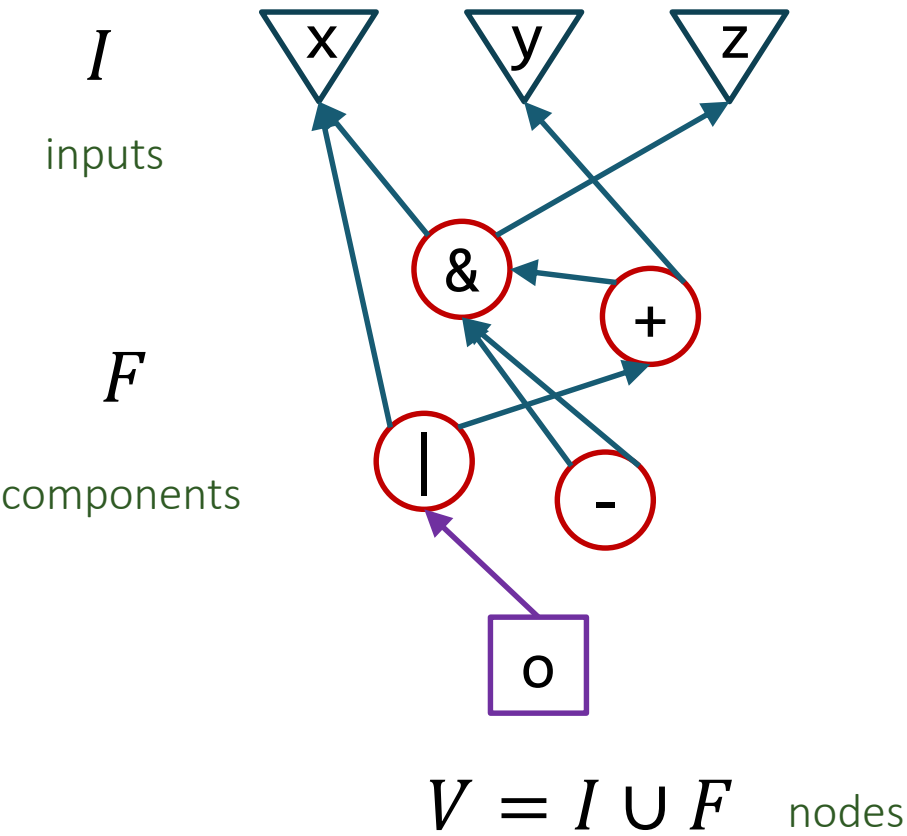
- Counterexample

Brahma



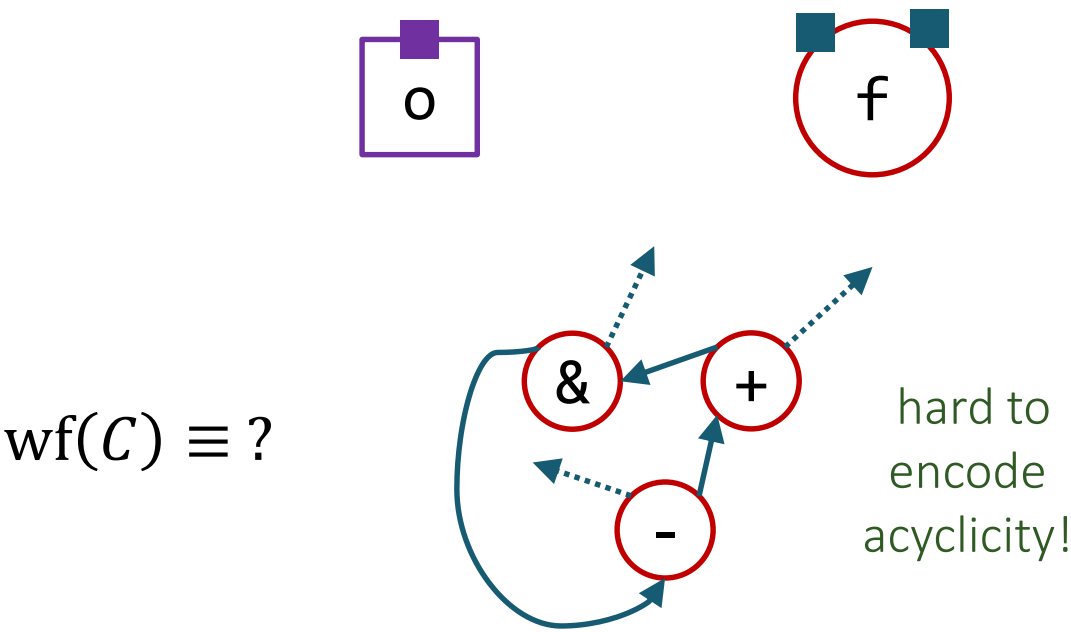
Brahma encoding

program space = DAG

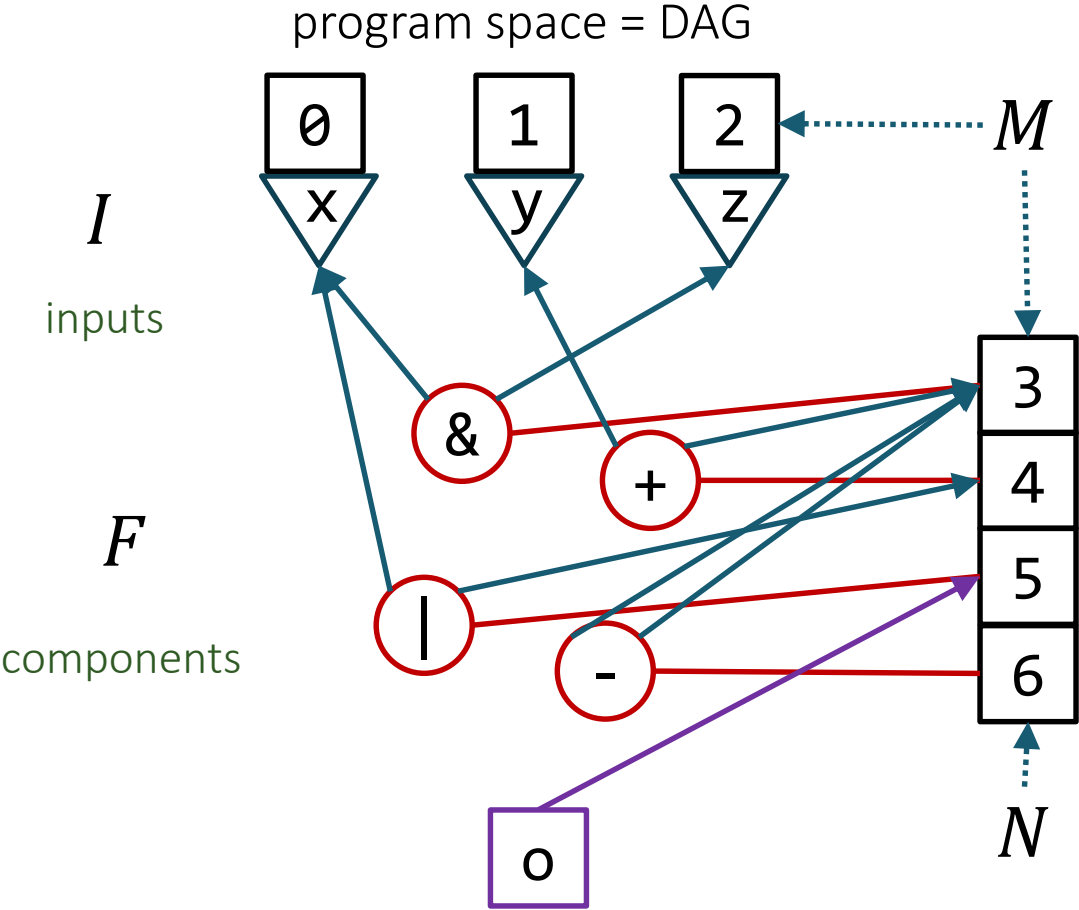


parameter space

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

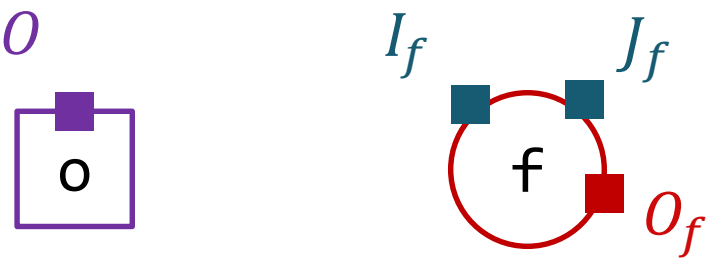


Brahma encoding: take 2



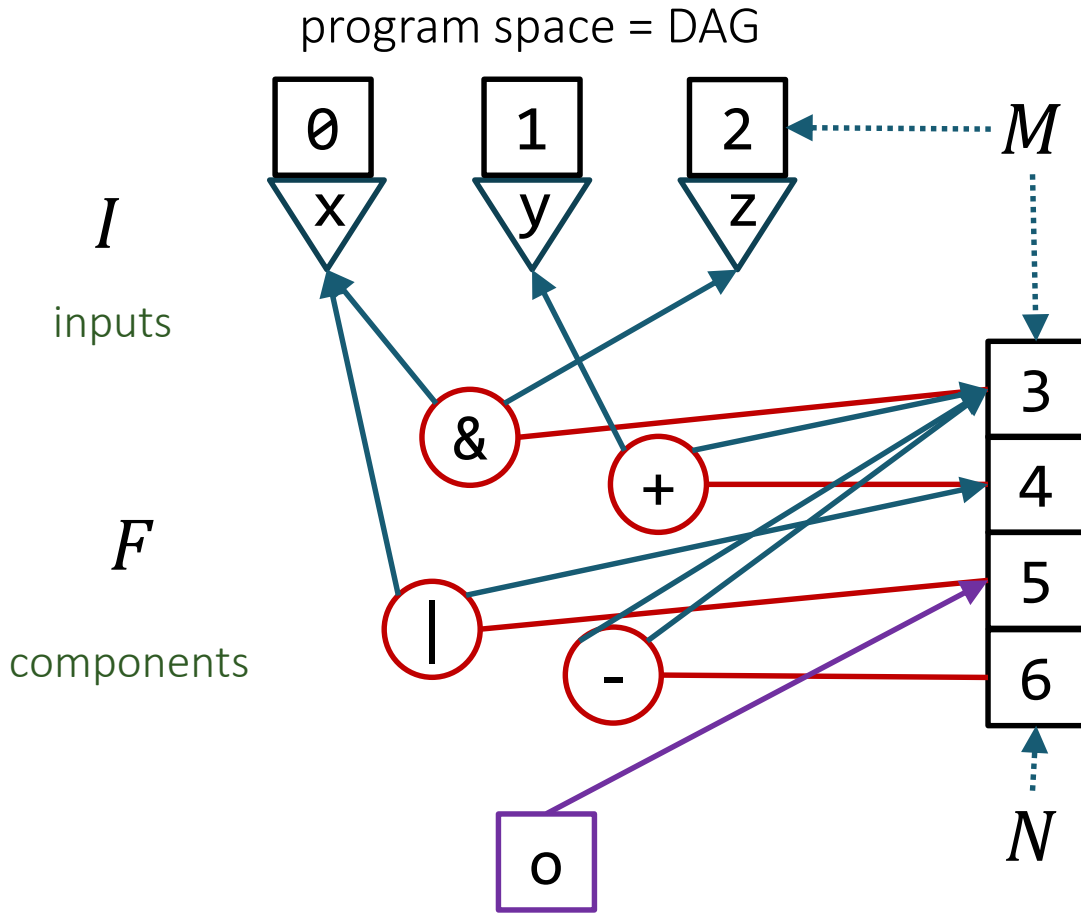
parameter space

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



$$\text{wf}(C) \equiv c_o \in M \wedge \bigwedge_{f \in F} c_{o_f} \in N \wedge c_{I_f/J_f} \in M$$

Brahma encoding: take 2



parameter space

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

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

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

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

Brahma: contributions

SMT encoding of program space

- sound?
- complete?
- solver-friendly?
- why does line 5 in ExAllSolver use conjunction instead of implication?

SMT solver can guess constants

- e.g. 0x55555555 in P23

Brahma: limitations

Requires component multiplicities

- What happens if user provides too many? too few?
- What's the alternative to including dead code?
- How would you extend this approach to work without multiplicities?

Requires *precise* SMT specs for components

- What happens if we give an over-approximate spec?

Brahma: limitations

No ranking

Cannot handle:

- loops
- types
- noise
 - Can we add these things?

Brahma: questions

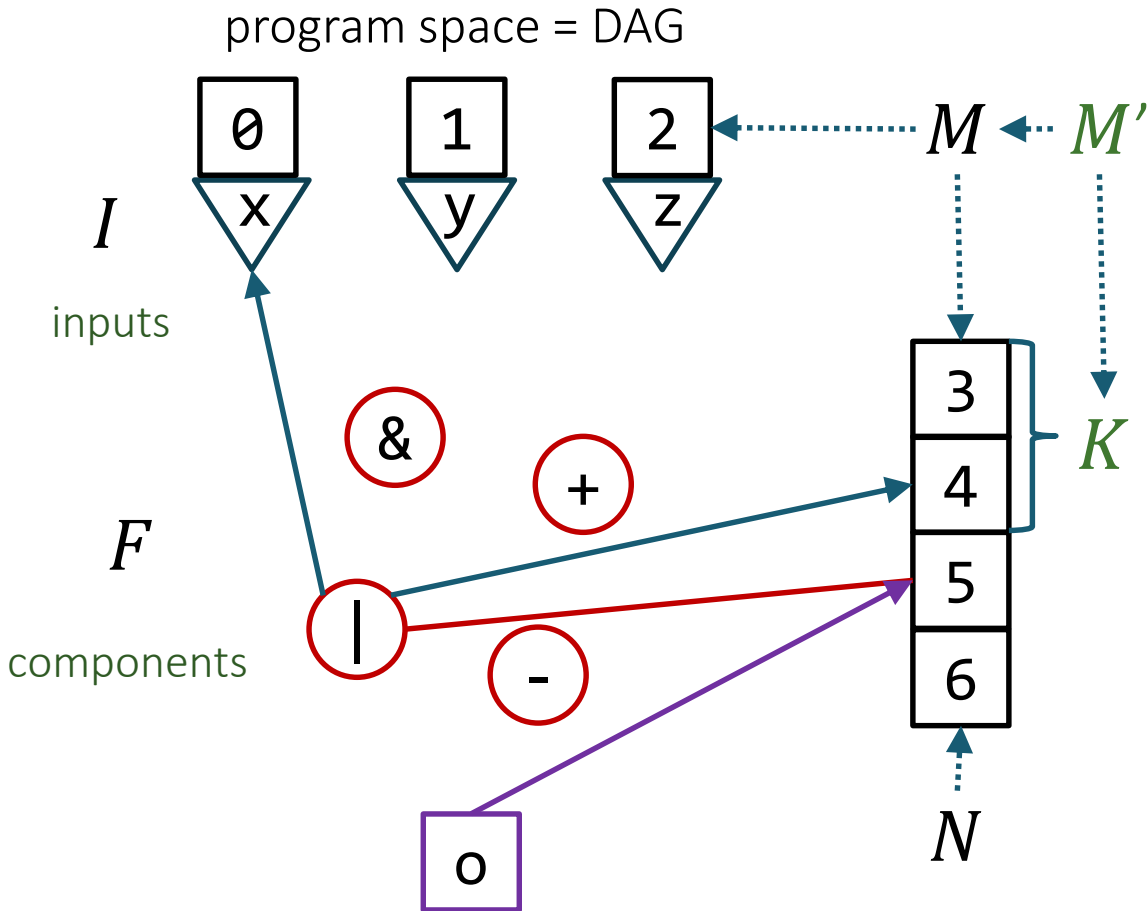
Behavioral Constraints? Structural Constraints? Search Strategy?

- First-order formula
- A multiset of components + straight-line program
- Constraint based + CEGIS

Can we represent these structural constraints as a grammar?

- Yes and no
- No because grammars cannot encode multiplicities
- Yes because the set is finite, so we can simply enumerate all possible programs
 - but this is not useful for synthesis

Limit #components to K?



parameter space

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

parameter space

memory locations: o , I_f , J_f , o_f

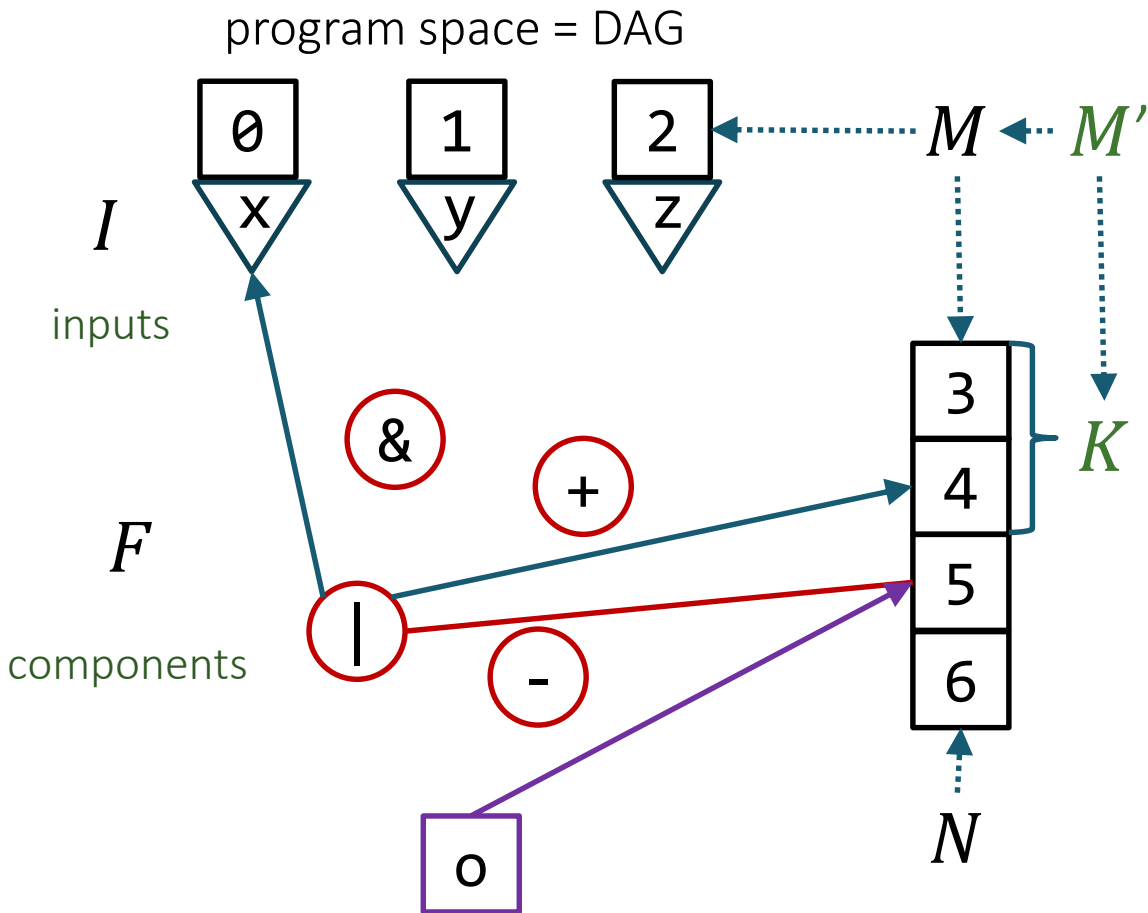
components: o , f

values: M' , K , M'

$$\text{wf}(C) \equiv c_o \in \cancel{M} \wedge \bigwedge_{f \in F} c_{o_f} \in \cancel{N} \wedge c_{I_f/J_f} \in \cancel{M}$$

$$\wedge \bigwedge_{f, g \in F, f \neq g} c_{o_f} \neq c_{o_g} \wedge \bigwedge_{f \in F} c_{I_f/J_f} < c_{o_f}$$

Limit #components to K?



parameter space

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

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

$$\wedge \bigwedge_{f, g \in F, f \neq g} c_{o_f} \neq c_{o_g} \wedge \bigwedge_{f \in F} c_{I/J_f} < c_{o_f}$$

Comparison of search strategies

