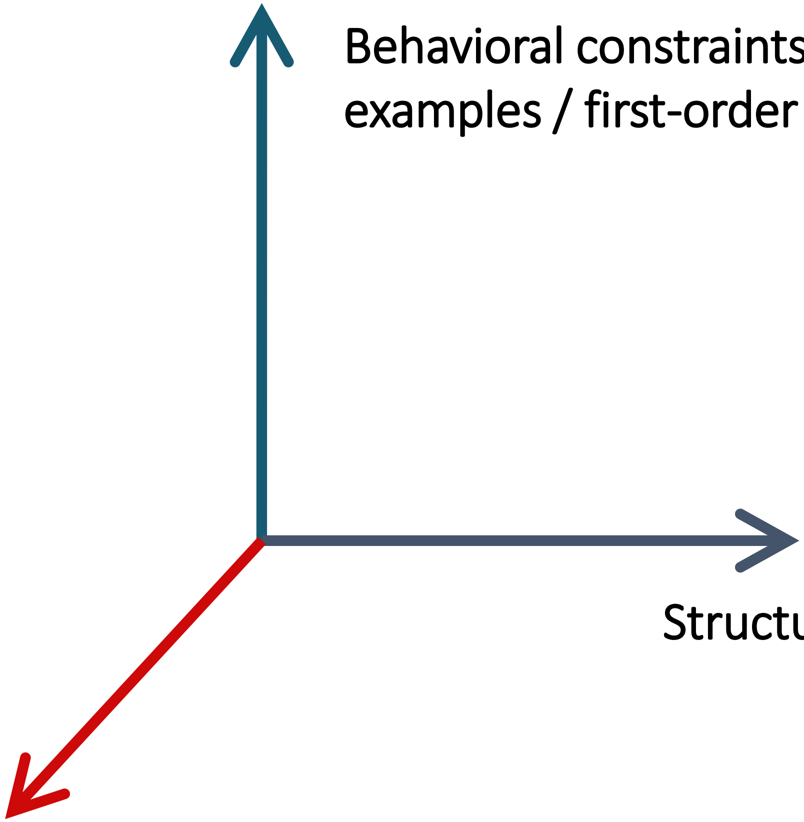
Lecture 8
Constraint-based search

Nadia Polikarpova

The problem statement

Behavioral constraints =
examples / first-order formula



Search strategy?

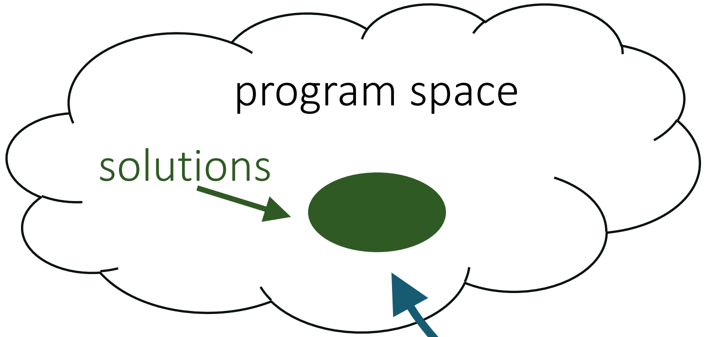
Enumerative
Representation-based
Stochastic
Constraint-based

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

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

Define a formula $\mathrm{wf}(c_1, \ldots, c_n)$

- 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 $c_1, \ldots, c_n$
(**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 in 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→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

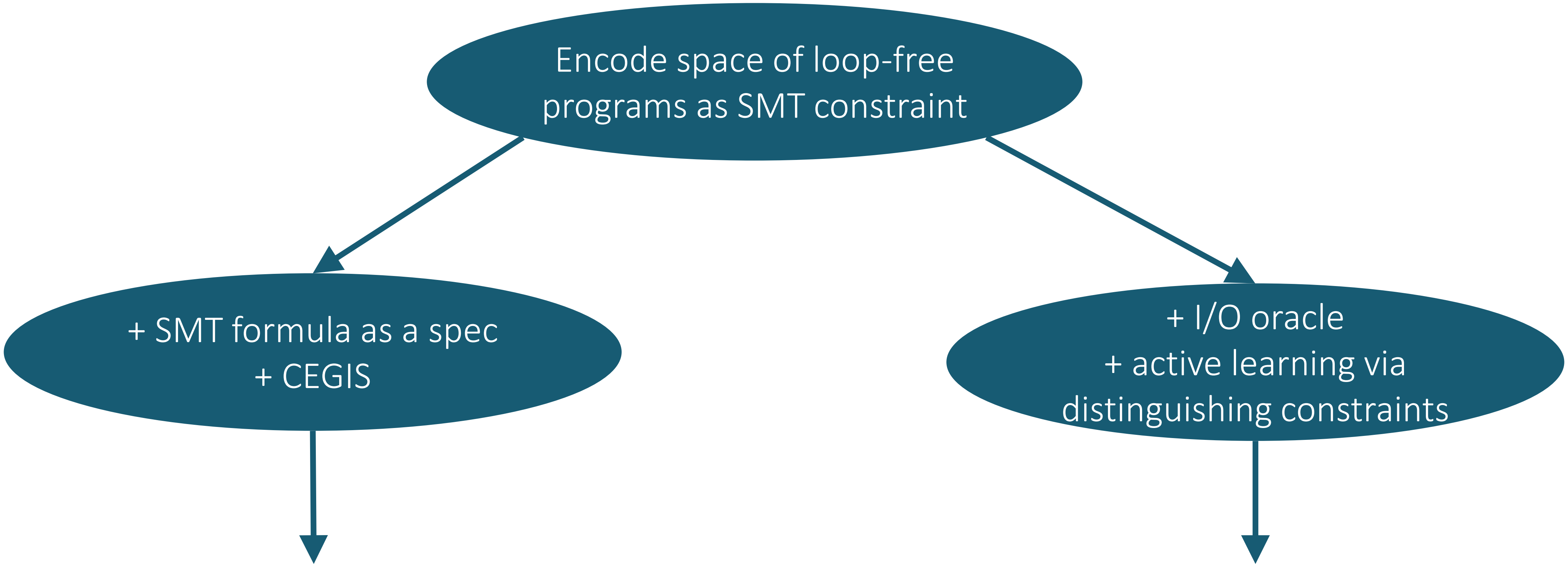
- Counterexample:
```
L ::= sort(L)  |  L[N..N]
    |  L + L  |  [N]  |  x
N ::= find(L,N)  |  0
```

- 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



[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: take 1

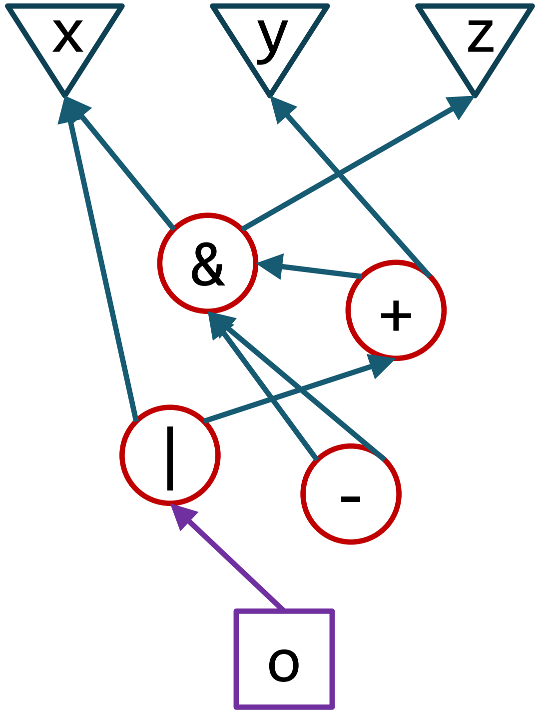
program = DAG

I

inputs

F

components

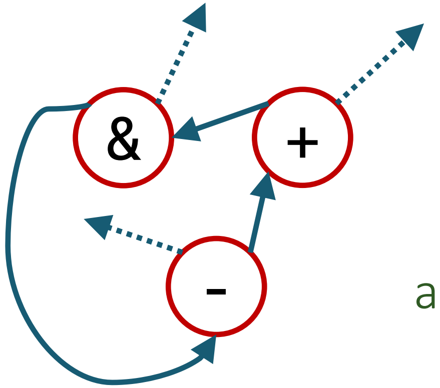


$V = I \cup F$ nodes

parameter space

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

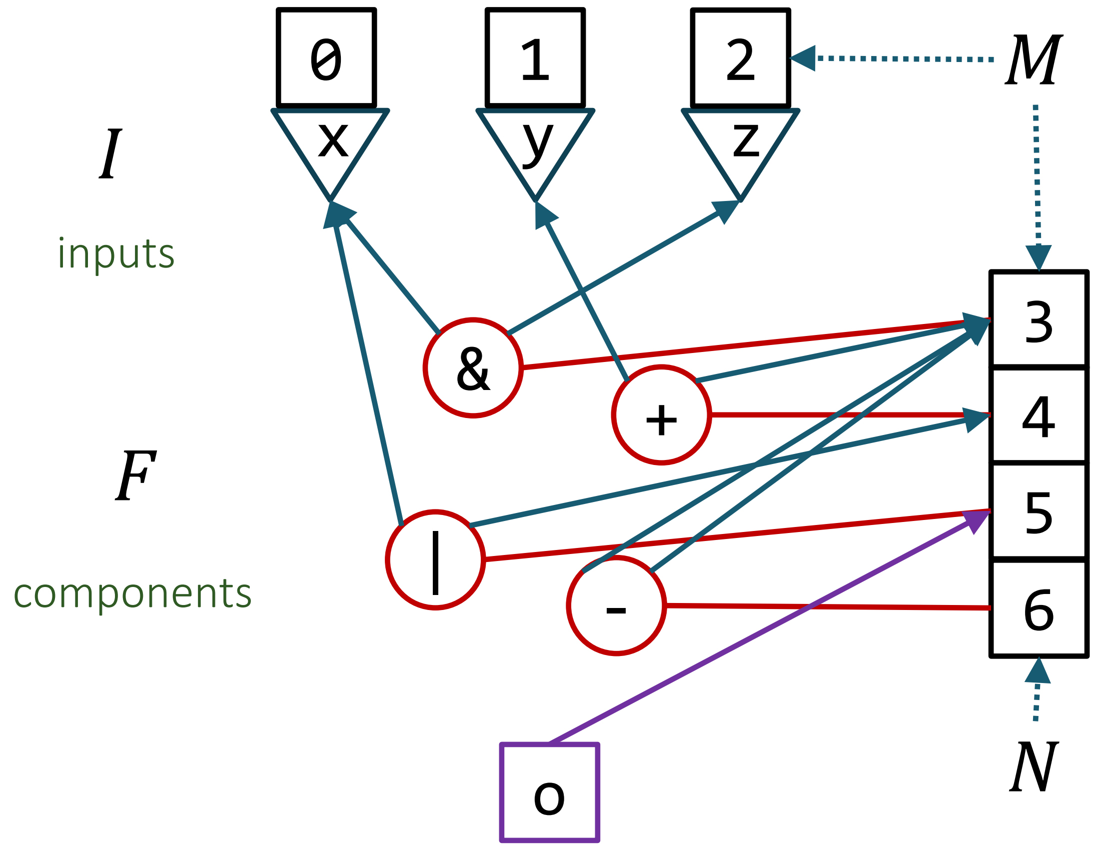
$\mathrm{wf}(C) \equiv ?$



hard to encode acyclicity!

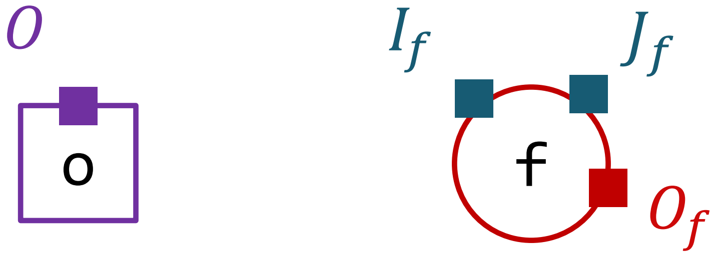
Brahma encoding: take 2

program = DAG



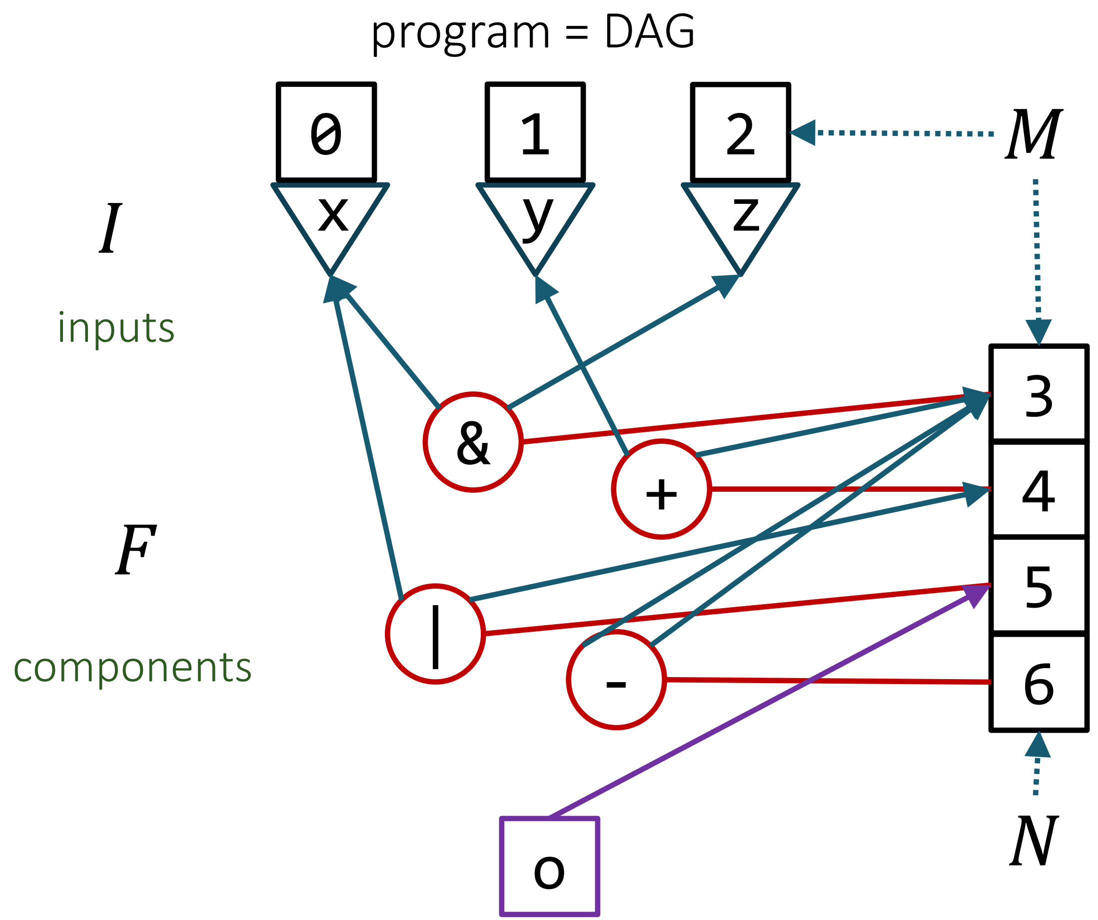
parameter space

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



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

Brahma encoding: take 2



parameter space

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

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

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

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

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
    - Can we add these things?

Brahma: questions

Behavioral Constraints? Structural Constraints? Search Strategy?

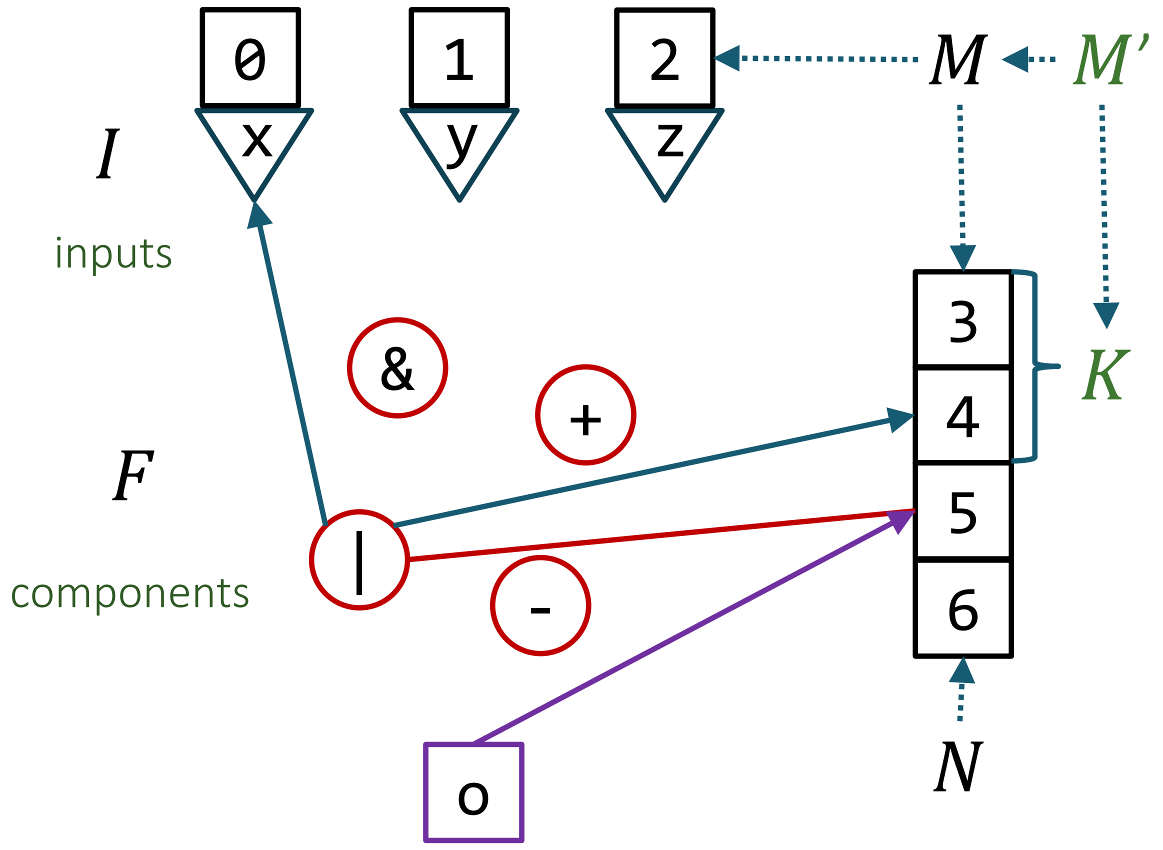
- First-order formula
- A multiset of components + straight-line program
- Constraint based + CEGIS

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

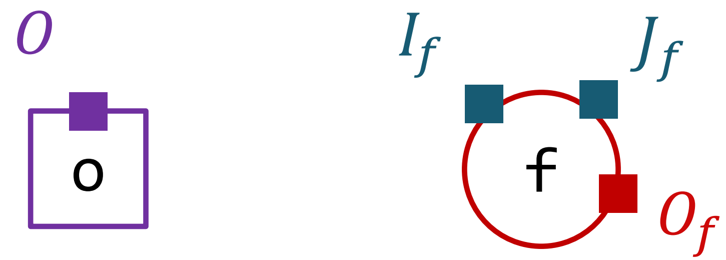
Limit #components to K?

program = DAG



parameter space

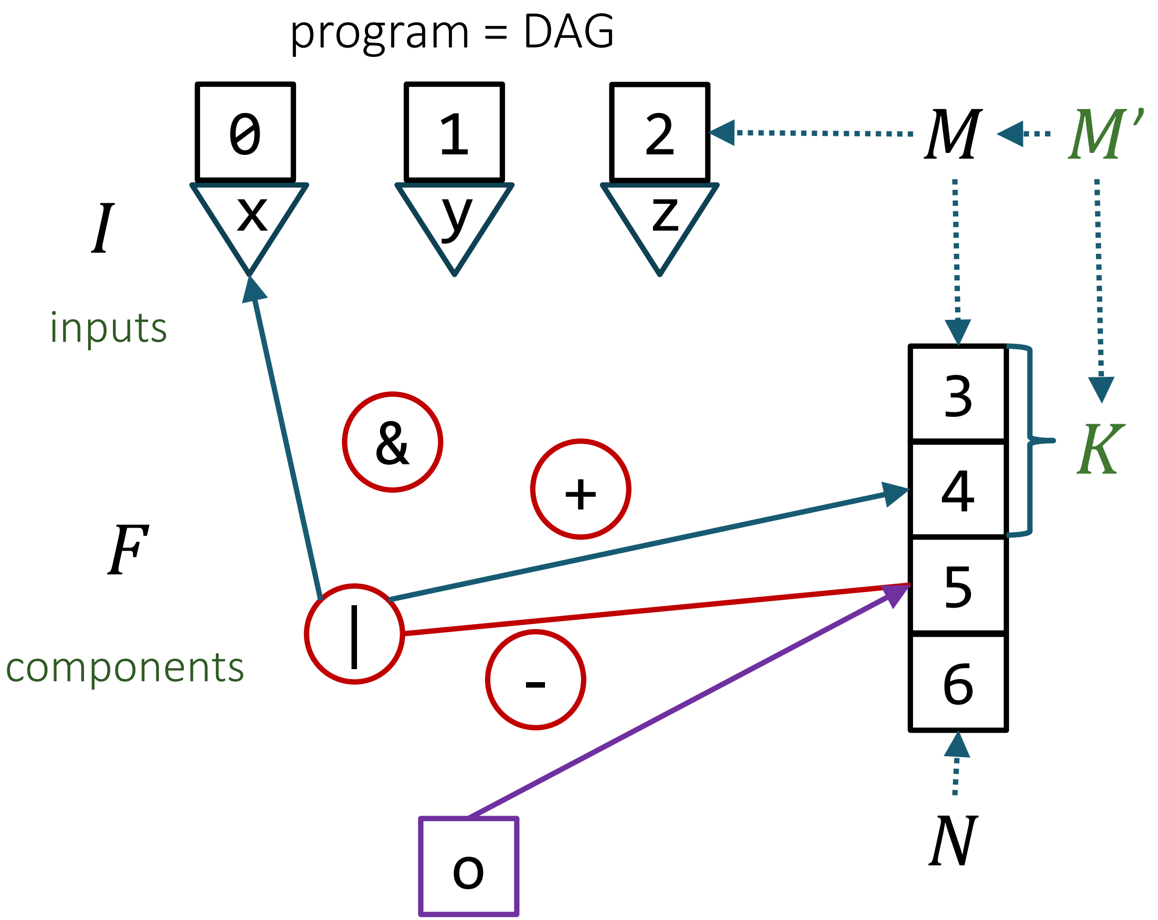
$$C = \{c_O : \mathrm{Int}\} \cup \bigcup_{f \in F} \{c_{O_f}, c_{I_f}, c_{J_f} : \mathrm{Int}\}$$



$$\mathrm{wf}(C) \equiv c_O \in \overset{M'}{\cancel{M}} \wedge \bigwedge_{f \in F} c_{O_f} \in \overset{K}{\cancel{N}} \wedge c_{I/J_f} \in \overset{M'}{\cancel{M}}$$

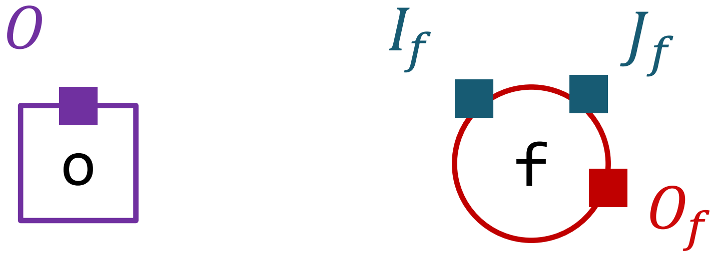
$$\wedge \bigwedge_{f,g \in F, f \not\equiv g} c_{O_f} \neq c_{O_g} \wedge \bigwedge_{f \in F} c_{I/J_f} < c_{O_f}$$

Limit #components to K?



parameter space

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



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

$$\wedge \bigwedge_{f,g \in F, f \not\equiv g} c_{O_f} \neq c_{O_g} \wedge \bigwedge_{f \in F} c_{I/J_f} < c_{O_f}$$

Comparison of search strategies