# in search of deductive possibility spaces

Chris Martens

March 22, 2024

UPenn PL Club

logic programming

is

neat.

# programming with relations

```
append-nil : append [] L L.
append-cons : append (X::Xs) Ys (X::L)
              :- append Xs Ys L.
```

?- append X Y [1, 2, 3, 4]

- X = [] Y = [1, 2, 3, 4]
- X = [1] Y = [2, 3, 4]
- X = [1, 2] Y = [3, 4]
- X = [1, 2, 3] Y = [4]
- X = [1, 2, 3, 4] Y = []

one question, many answers

one program, many questions

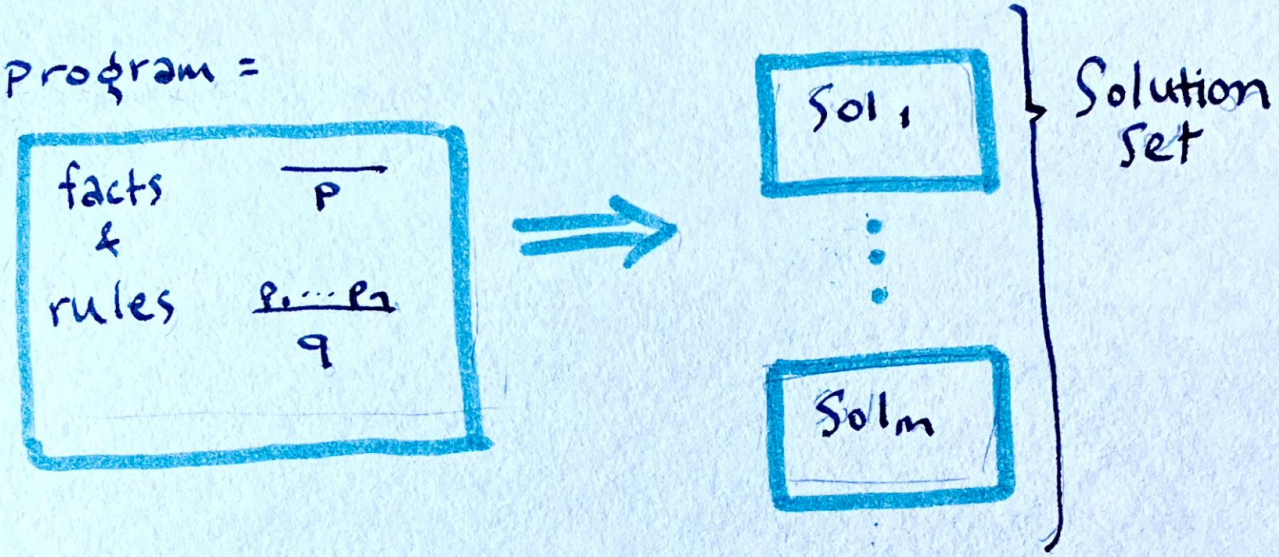
Logic programming is useful

- mechanizing metatheory (judgments as relations)
- typeclasses & trait systems
- set comprehensions
- constraint solving
- expressive pattern matching

I want to tell you about


a new logic programming language

# DUSA



# Dusa examples

- Tarot
- Spanning trees
- Maze generation
- Rock, paper, scissors

DUSA

functional dependencies

a IS V  a IS V'

$\Rightarrow$

V = V'

# Dusa: Closed & Open Rules

p is $\{V_1, V_2\}$ — p must be one of $V_i$

p is $\{V_1, V_2 ?\}$ — p may be one of $V_i$, and must have some assignment in sol.

# Dusa: Operational Semantics

For $F \in \mathcal{D}$, $C \leftarrow F$ rule, if

$C = p\ \text{is}\ \{v_1 \ldots v_n\}$ (closed)

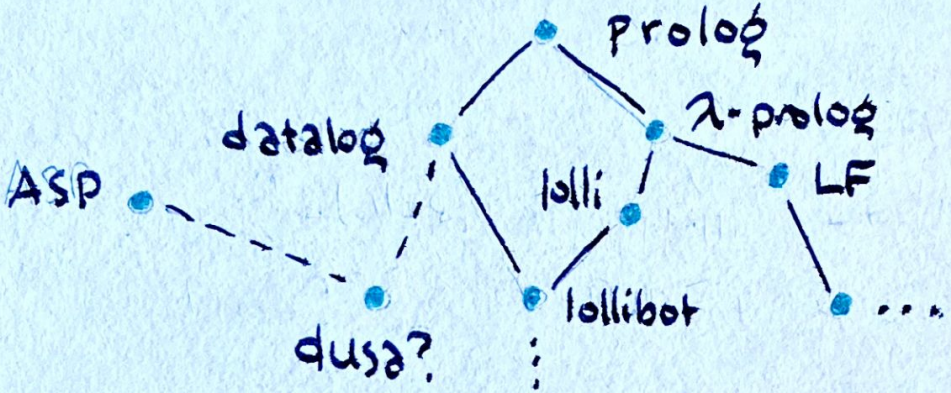
$\mathcal{D} \Rightarrow \{\mathcal{D} + \{p \mapsto v_1\}, \ldots, \mathcal{D} + \{p \mapsto v_n\}\}$

if $C = p\ \text{is}\ \{v_1 \ldots v_n ?\}$ (open)

$\mathcal{D} \Rightarrow \{\mathcal{D} + \{p \mapsto v_1\}, \ldots, \mathcal{D} + \{p \mapsto v_n\}, \mathcal{D}\}$

Solutions are saturated & consistent

but what justifies a logic programming language's design?



# 3 theories of meaning

## for logic programs

1. proofs — $\dfrac{\quad}{\Gamma \vdash A}$
2. models — $I \models \varphi$
3. least fixed points — $\lim_i f^i(\bot)$

# PART 1:

# PROOF THEORY

# Horn Clauses

$$q :- p_1, \ldots, p_n$$

$$\approx$$

$$\forall \vec{x}.\ p_1 \wedge \ldots \wedge p_n \supset q$$

# Proof Theory

program $\Gamma$
query $A$
Solution Proof of $\Gamma \Rightarrow A$

"Execution as proof search"

Miller, Nadathur, Pfenning, Scedrov. "Uniform Proofs as a foundation for logic programming", 1991.

Solutions = Proofs

?- ∃x,y. append x y [1,2,3]

$$\frac{\Gamma \Rightarrow \text{append } [] \ [1,2,3] \ [1,2,3]}{\Gamma \Rightarrow \exists x,y.\ \text{append } x\ y\ [1,2,3]}\ \exists R$$

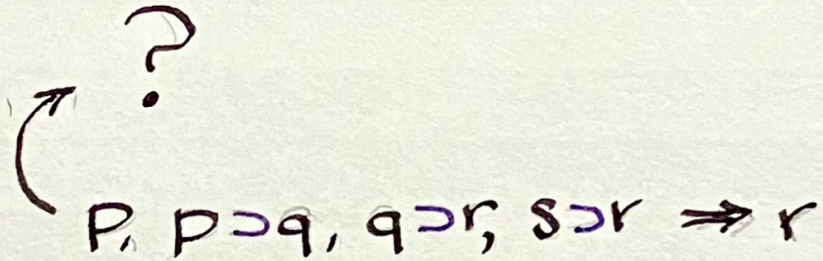
# Sequent Calculus

$$\frac{\Gamma, A, B \Rightarrow C}{\Gamma, A \wedge B \Rightarrow C}\ \wedge L \qquad \frac{\Gamma \Rightarrow A \quad \Gamma \Rightarrow B}{\Gamma \Rightarrow A \wedge B}\ \wedge R$$

$$\frac{\Gamma, A \supset B \Rightarrow A \quad \Gamma, A \supset B, B \Rightarrow C}{\Gamma, A \supset B \Rightarrow C}\ \supset L \qquad \frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \supset B}\ \supset R$$

$$\frac{}{\Gamma, P \Rightarrow P}\ id$$

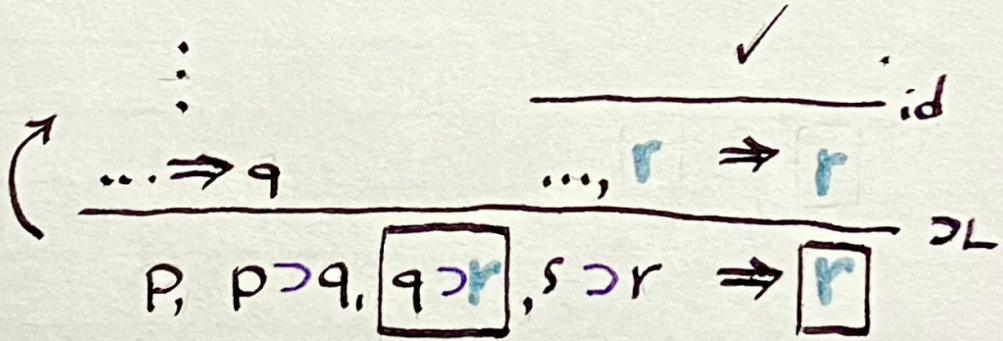
# Chaining



# Chaining

**backward chaining:**

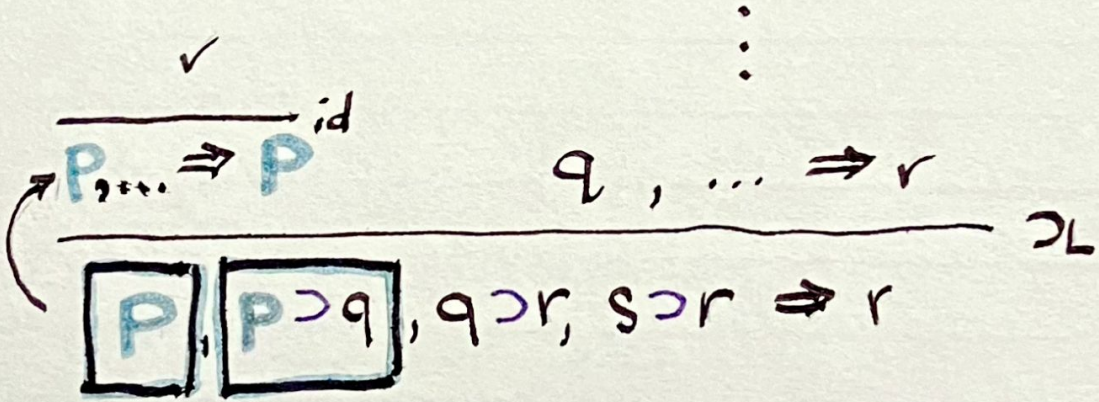
dress for the job you want



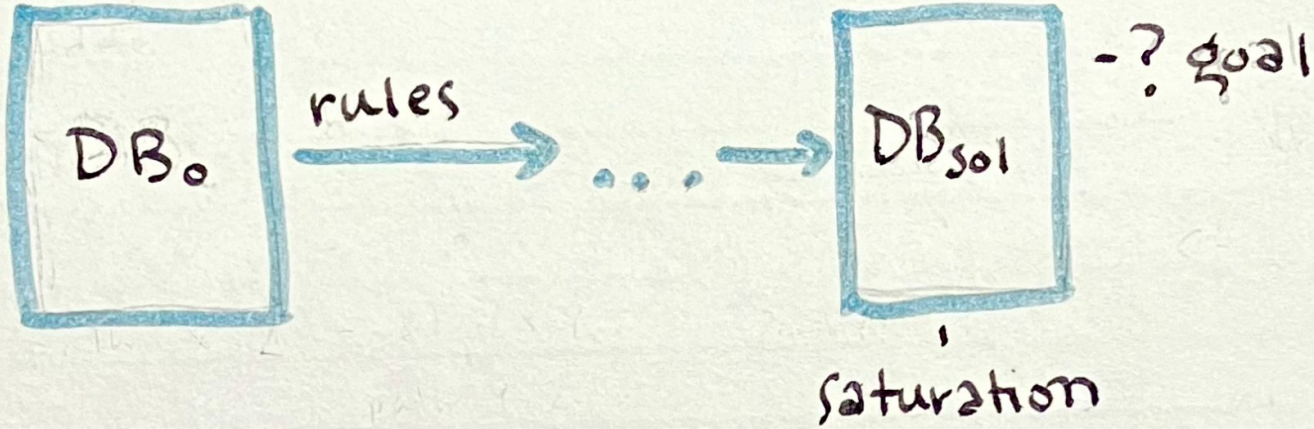
# Chaining

forward chaining:

work with what you've got

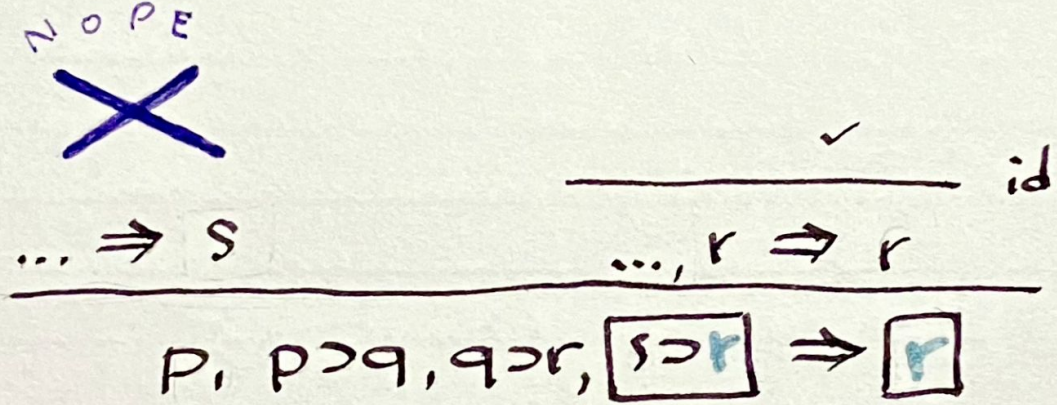


# Datalog



# Invertibility

The wrong choice might cause the proof to fail!



# Invertibility

Invertible rules can be eagerly applied—
they don't introduce backtracking points.

| Invertible | Non-invertible |
| --- | --- |
| $\supset R$ | $\supset L$ |
| $\wedge L$ | |
| $\wedge R$ | |
| id | |

1 Andreoli. Logic programming with focusing proofs in linear logic (1992)

# Focusing

Label connectives with invertible right rules negative.

Label connectives with invertible left rules positive.

$$A^+, B^+ ::= p^+ \mid A \wedge B$$

$$A^-, B^- ::= p^- \mid A \supset B$$

# Focusing

Two phases of proof search:

**Inversion** — Apply invertible rules eagerly

**Focusing** — Apply rules to one thing as much as possible

mediated by focus and blur rules

Cool thing #1

about proof theory:

focusing explains

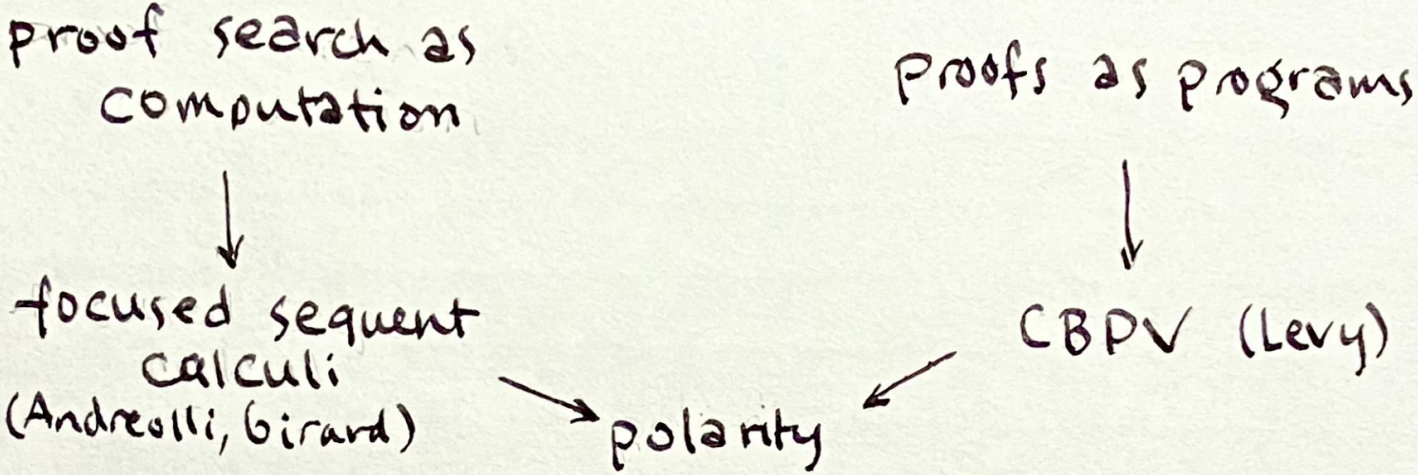
forward & backward chaining

$p^+, p^+ \supset q^+, q^+ \supset r^+ \Rightarrow r^+$ forward

$p^-, p^- \supset q^-, p^- \supset r^- \Rightarrow r^-$ backward

(ask me later for details!)

# double discovery




"Their convergence is a sign of their significance."

– Brock-Nannestad, Guenot, Gustafsson.
"Computation in Focused Intuitionistic Logic."

forward & backward chaining:

equivalent provability

for intuitionistic Logic

(one canonical "solution")

Cool thing #2 about proof theory:

**extensibility** to other logics

# Linear Forward Chaining

spanning tree generation:

pickRoot: start $\otimes$ unvisited $N$
$\multimap$ visited $N$,

addEdge: visited $N_1$ $\otimes$ edge $N_1$, $N_2$
$\otimes$ unvisited $N_2$
$\multimap$ visited $N_1$ $\otimes$ stEdge $N_1$, $N_2$
$\otimes$ visited $N_2$.

# Linear logic programming (resources)

- Rob Simmons and Frank Pfenning: Linear Logical Algorithms
- My thesis language, Ceptre
- My Strange Loop 2013 talk

Predecessors/related work:

- Lolli (Hodas & Miller);
- Lollimon; CLF/Celf (Lopez, Polakow Pfenning, Watkins)

# PART 2:

# STABLE MODELS

# Motivation: representing possibility spaces

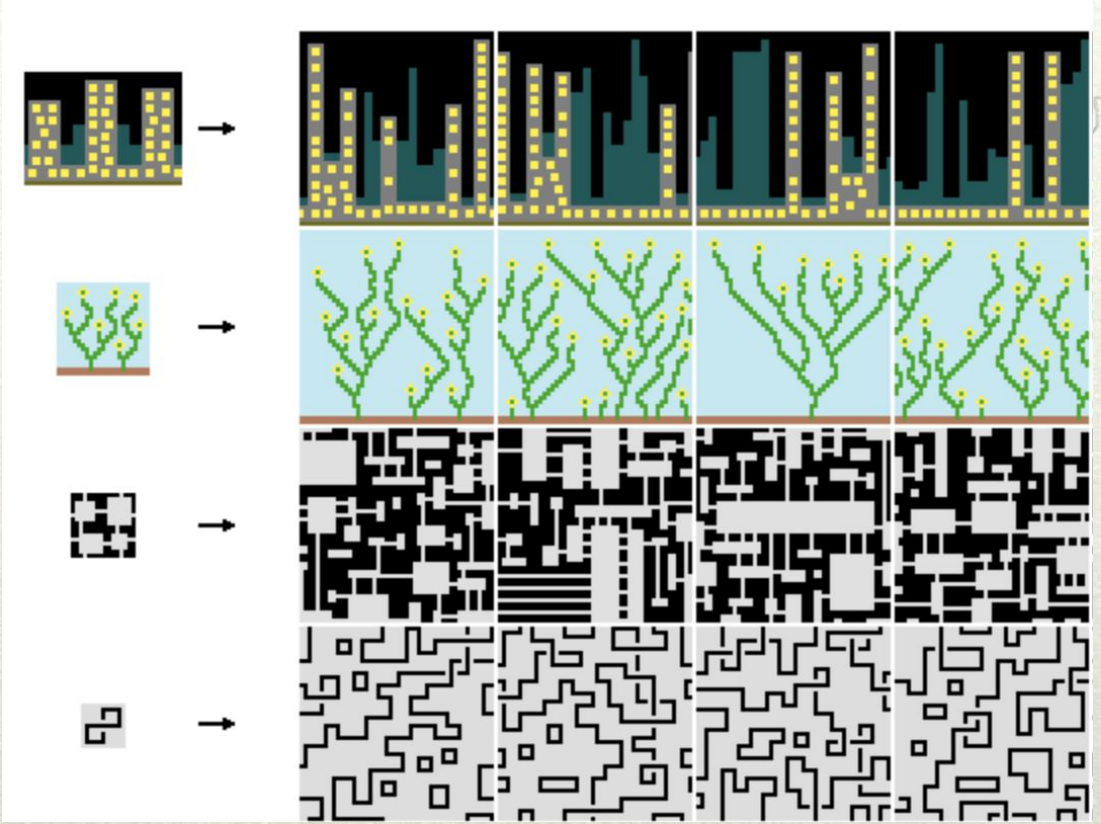
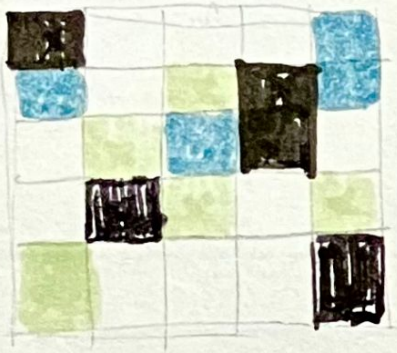


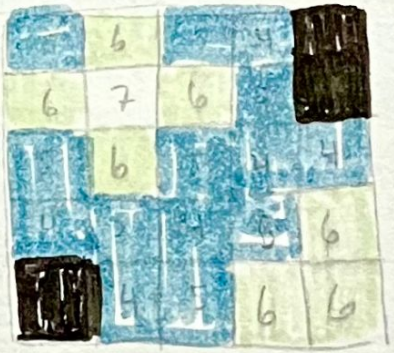
Image:
Wave Function Collapse
by Maxim Gumin

https://github.com/mxgmn/WaveFunctionCollapse

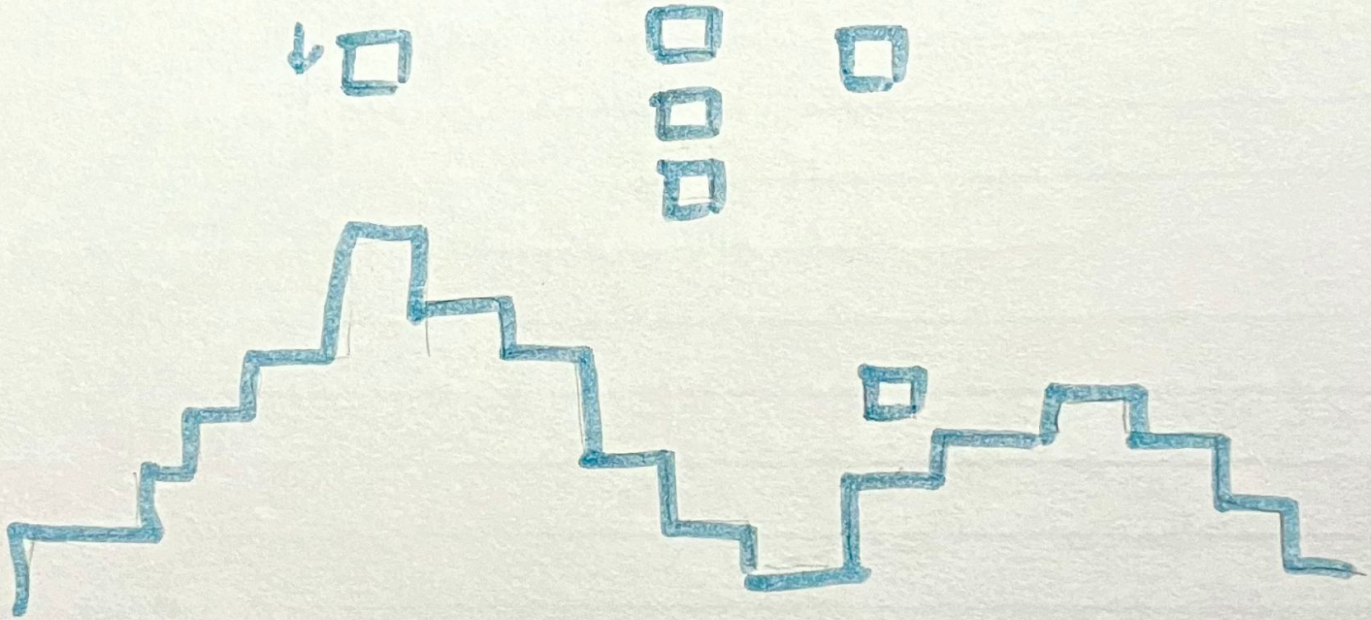
# terrain smoothness

NO.

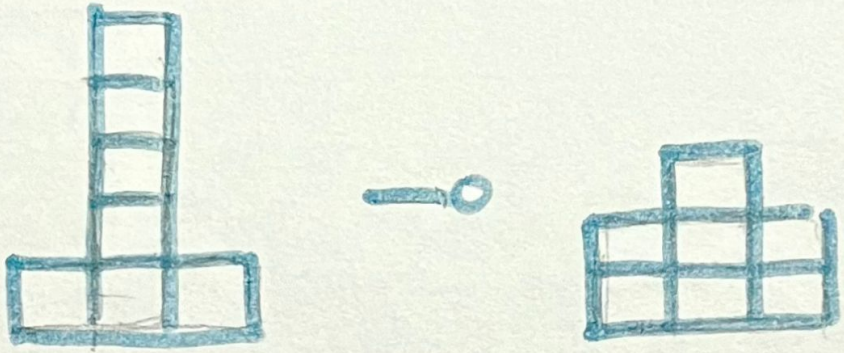




# simulating erosion:
## sandpile cellular automata



# Cellular automata as rewrite rules



local rewrites vs. global properties

$$\forall c, n.\ \text{neighbor}(c, n) \supset |\text{height}(c) - \text{height}(n)| \leq 1$$

chiseling possibility spaces

:- not propertyIwant.

# Stable Models

(Gelfond & Lifschitz '88)

$P \text{ :- } p_1, \ldots, p_n, \neg q_1, \ldots, \neg q_m$

Let $X$ be an instance. Reduct $P^X$:

1. Delete rules in $P$ w/ premises $\neg q$ where $q \in X$.
2. Delete negations in remaining rules.

$X$ is <u>stable</u> if $\text{wfmodel}(P^X) = X$.

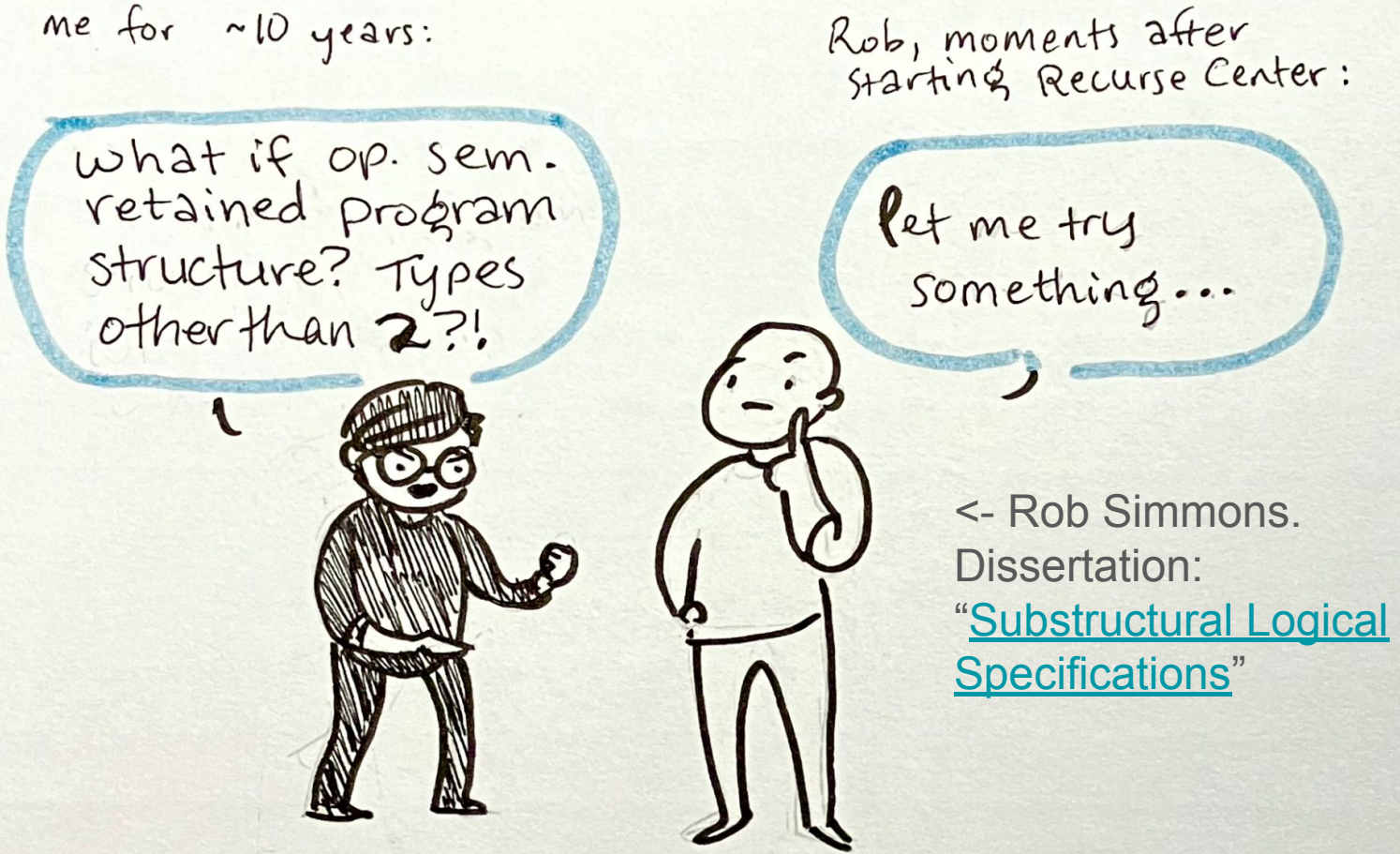
# Answer Set Programming (ASP)

+ successful! (for PCG: Smith 2011 "PCG via ASP")

+ mature, fast tooling!

-: ground → solve pipeline
∴ all domains finite

- black box
- logical justification?
- program structure




<- Rob Simmons. Dissertation: "Substructural Logical Specifications"

enter Dusa!

but, again: how to guide and justify the design?

# PART 3:

# LEAST FIXED POINTS

Dusa: Denotational Semantics?

"run to saturation" has Datalog nature...


Michael Arntzenius, dissertation: "Deconstructing Datalog"

http://www.rntz.net/files/thesis.pdf

# Datalog Semantics

Database $\mathcal{D}$ : DB = set of ground predicates

Programs $P$ contain rules $C \leftarrow F$

that can <u>fire</u> in $\mathcal{D}$ w/ subst $\sigma$

when $\sigma F \in \mathcal{D}$.

The <u>immediate consequences</u> $T_P(\mathcal{D}) =$

$$\{ \sigma C \mid C \leftarrow F \in P,\ \sigma F \in \mathcal{D} \}$$

# Datalog Semantics

Thm: for any Datalog prog. $P$,

$T_P$ has a least fixed point,

equal to $T_P^i(\emptyset)$ for some $i$.

(Instance of Kleene fixed point theorem)

# Immediate Consequence for Dusa?

LFP theorem requires $T_P$ monotone:

if $D_1 \subseteq D_2$ then $T_P(D_1) \subseteq T_P(D_2)$

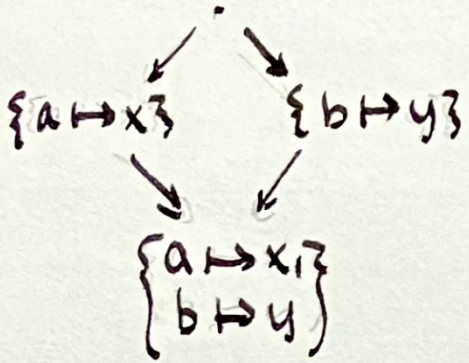
Rules in Dusa map $DB \to 2^{DB}$

Q1: How to combine DB sets?

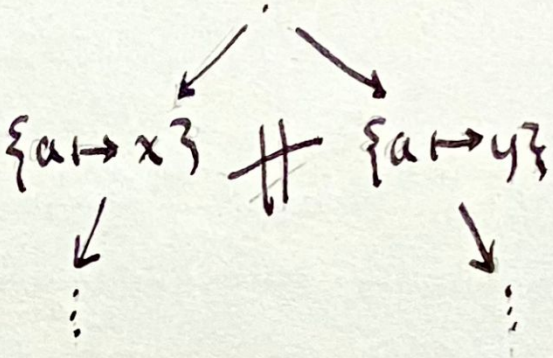
Q2: How to order them?

# Compatibility

Arbitrary rule firing order
(a la Datalog)




Incompatible choices




LUBs (∨) exist when compatible,
but not otherwise.

# Dusa Semantics: Choice Sets

**Choice set**: Set of pairwise incompatible DBs (candidate/partial solutions)

Can define $\leq$ (~Smith powerdomain) & get all LUBs

$\bot = \{\bot_{DB}\}$, $\top = \emptyset$

$\therefore$ complete lattice!

## Dusa Semantics:

## Immediate Consequence

$$T_P(\mathcal{D}) = \bigvee \{ C \mid \mathcal{D} \Rightarrow C \}$$

$$T_P^*(\mathcal{C}) = \bigcup \{ \mathcal{C}' \mid \mathcal{D} \in \mathcal{C}, T_P(\mathcal{D}) = \mathcal{C}' \}$$

## Knaster-Tarski Theorem

Any monotone function on a complete lattice has a least fixed point.

NTS $T_P^*$ : Choice set → Choice set

monotone (it is). □

Also want: correspondence w/ op. sem.!

(// TODO)

# OUTRO

# Open Questions

- Are stable models "found in nature"?

- & is Dusa a canonical presentation?

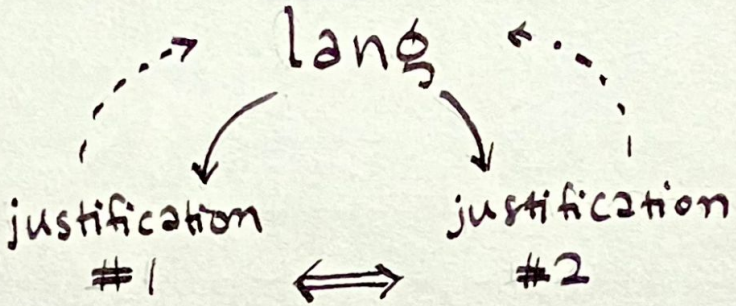
- Can it be proof-theoretically justified?

# Open Questions

- Possible relationships to:
  - Probabilistic languages?
  - Coalgebras?
- Potential use cases:
  - Property-based testing?
    - Generating (co)inductive data

Blanco, Miller, Momigliano:
"Property-Based Testing via Proof Reconstruction"

# Guiding principles in language design



# Thanks!

- Email c.martens@northeastern.edu
- Website https://www.khoury.northeastern.edu/~cmartens/
- Mastodon https://hci.social/@chrisamaphone