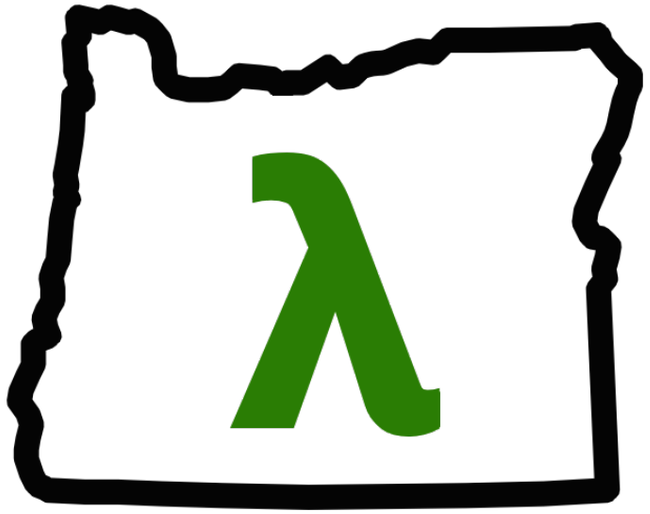
Choice Calculus & Variational Programming

PNW PL/SE Meeting 2016

Eric Walkingshaw
Oregon State University

PL Research @ OSU



Lots of research related to **variability**:

- variational program analyses
- variational program execution
- variational data structures
- view-based editing of variational programs

Other active areas:

- language design & domain-specific languages
- functional programming

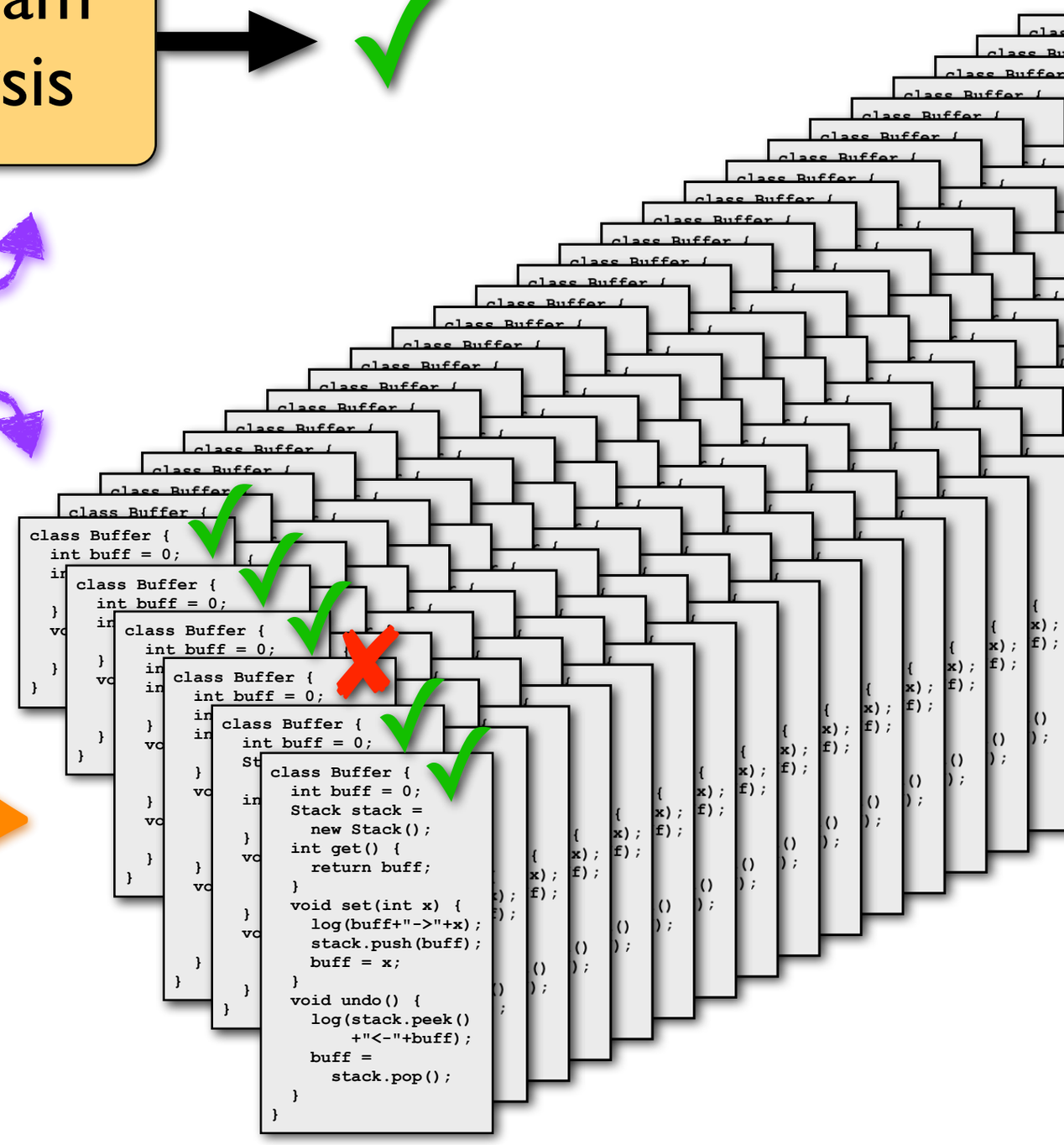
Motivation: a problem from SPLs

```
class Buffer {
  int buff = 0;
  int get() {
    return buff;
  }
  void set(int x) {
    buff = x;
  }
}
```

Goal: apply this to all of these

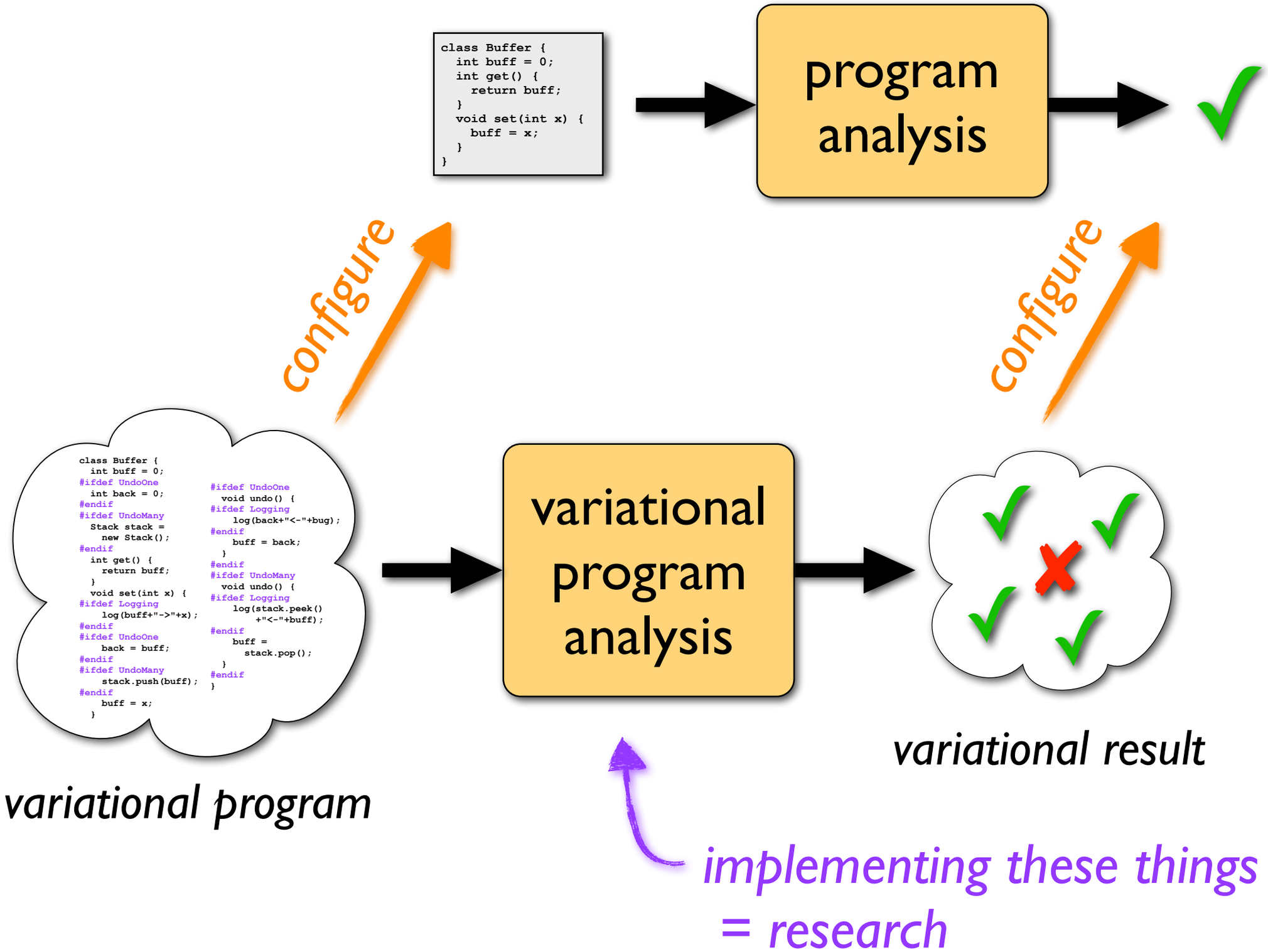
```
class Buffer {
  int buff = 0;
#ifdef UndoOne
  int back = 0;
#endif
#ifdef UndoMany
  Stack stack =
    new Stack();
#endif
  int get() {
    return buff;
  }
  void set(int x) {
#ifdef Logging
    log(buff+"->"+x);
#endif
#ifdef UndoOne
    back = buff;
#endif
#ifdef UndoMany
    stack.push(buff);
#endif
    buff = x;
  }

#ifdef UndoOne
  void undo() {
#ifdef Logging
    log(back+"<-"+bug);
#endif
    buff = back;
  }
#endif
#ifdef UndoMany
  void undo() {
#ifdef Logging
    log(stack.peek()
        +"<-"+buff);
#endif
    buff =
      stack.pop();
  }
#endif
}
```

configure



software product line

Solution: analyze variational program



Outline

- Intro and motivation
- Choice calculus
- Variational type inference + applications
- Brief note on variational data structures

Choice calculus

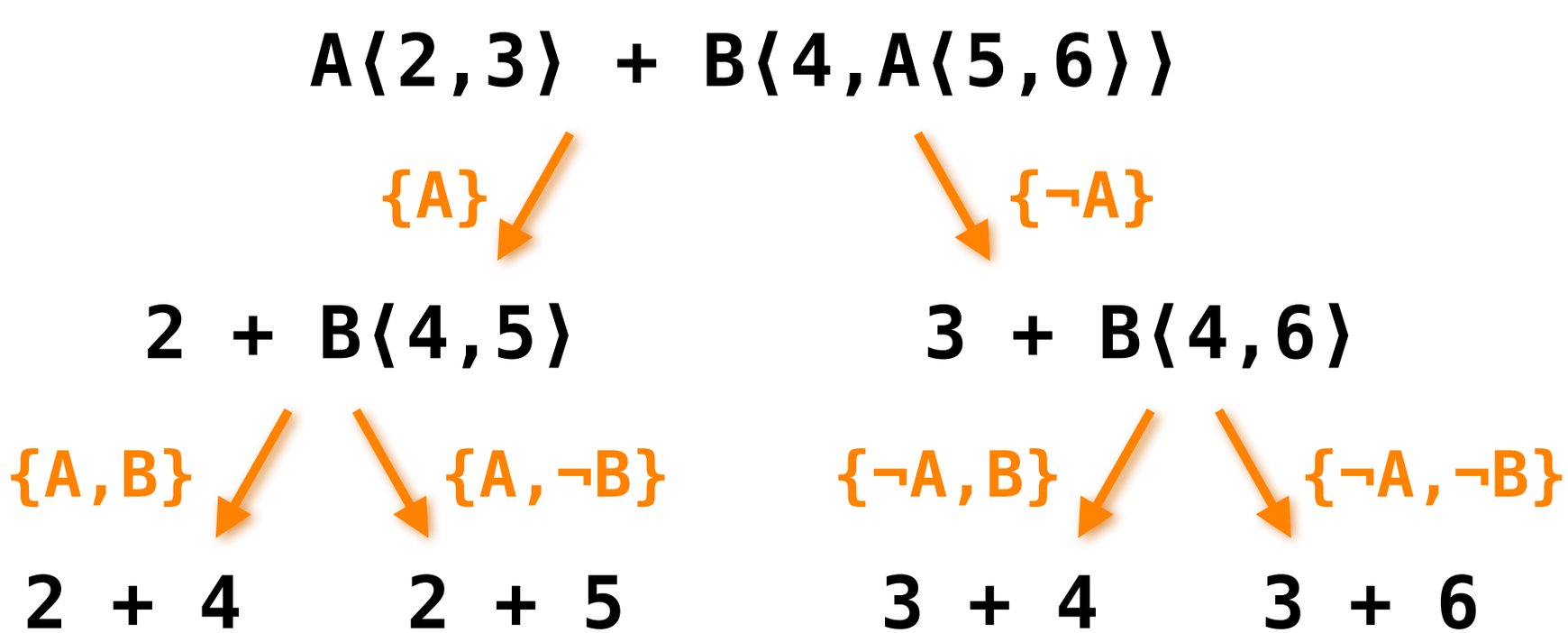
Martin Erwig

A core metalanguage for describing variation

```
e ::= a{e,...,e}
    |  D⟨e,e⟩
```

object language AST

choice between two alternatives synchronized by a dimension



Equivalence laws

$$D\langle e,e\rangle \equiv e$$ idempotency

$$D\langle e_1,D\langle e_2,e_3\rangle\rangle \equiv D\langle e_1,e_3\rangle$$ domination

$$D_1\langle D_2\langle e_1,e_2\rangle,D_2\langle e_3,e_4\rangle\rangle \equiv D_2\langle D_1\langle e_1,e_3\rangle,D_1\langle e_2,e_4\rangle\rangle$$ distribution

$$D\langle a\{e_1,e_2\},a\{e_3,e_4\}\rangle \equiv a\{D\langle e_1,e_3\rangle,D\langle e_2,e_4\rangle\}$$ factoring

Instantiating the choice calculus

```
e ::= c
    |  x
    |  λx.e
    |  e e
    |  D⟨e,e⟩
```

variational lambda calculus

```
T ::= κ
    |  a
    |  T -> T
    |  D⟨T,T⟩
```

variational types

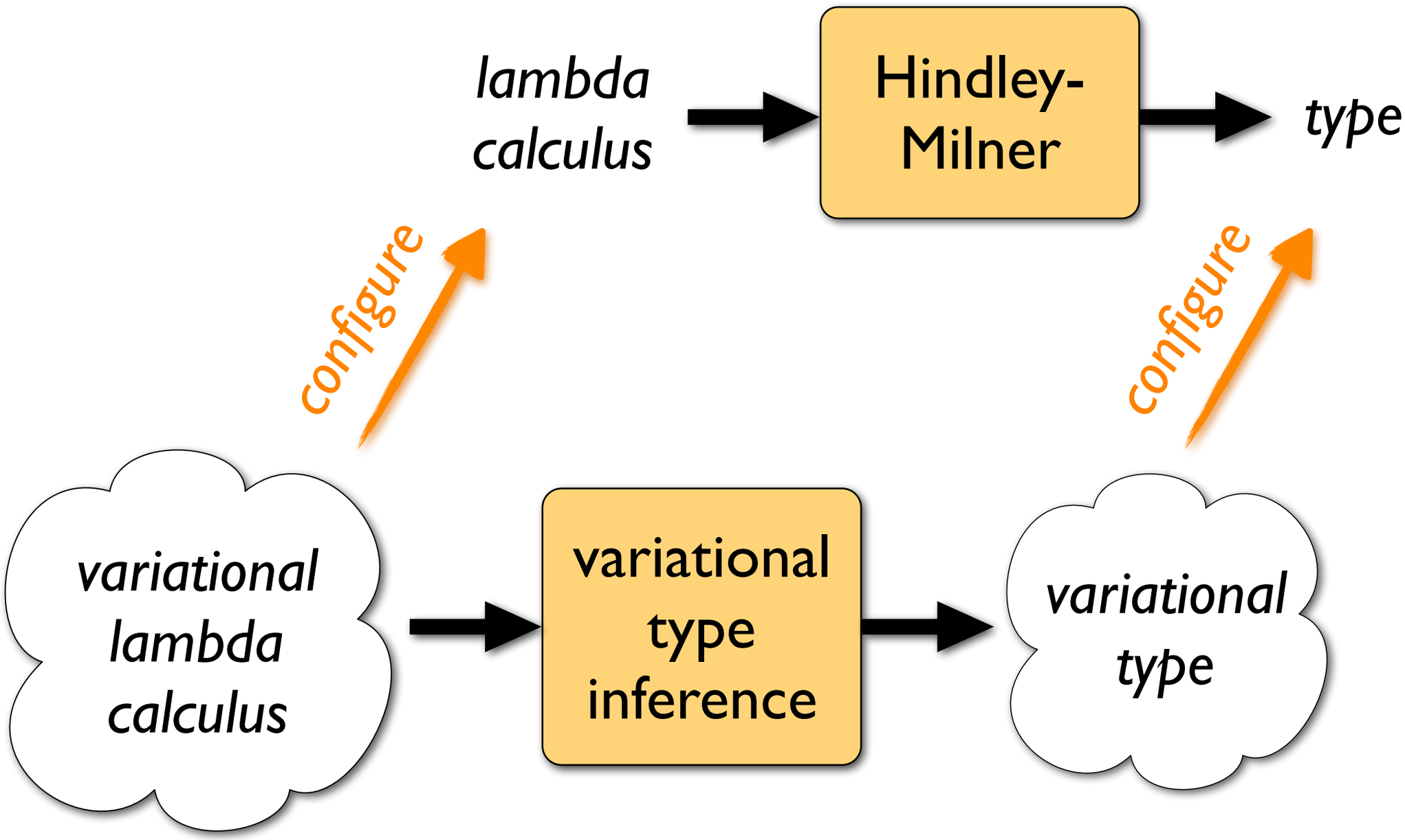
can instantiate the equivalence laws too:

$$D\langle T_1 \to T_2,\ T_3 \to T_4\rangle \equiv D\langle T_1, T_3\rangle \to D\langle T_2, T_4\rangle$$

Outline

- Intro and motivation
- Choice calculus
- Variational type inference + applications
- Brief note on variational data structures

Variational type inference



Variational unification

Goal: find a mapping $\sigma = \{a \mapsto T\}$ so that $T_L\sigma \equiv T_R\sigma$

$$A\langle \texttt{Int}, a\rangle \equiv^? A\langle a, \texttt{Bool}\rangle$$

decompose by alternatives

$$\texttt{Int} \equiv^? a \qquad a \equiv^? \texttt{Bool}$$

stuck!

$$\sigma = \{a \mapsto A\langle \texttt{Int}, \texttt{Bool}\rangle\}$$

Variational unification

$$A\langle \texttt{Int}, a\rangle \equiv^{?} A\langle a, \texttt{Bool}\rangle$$

$$A\langle \texttt{Int}, a_{\tilde{A}}\rangle \equiv^{?} A\langle a_A, \texttt{Bool}\rangle$$

1. Qualification

$$\texttt{Int} \equiv^{?} a_A \qquad a_{\tilde{A}} \equiv^{?} \texttt{Bool}$$

2. Solve qualified problem

$$\{a_A \mapsto \texttt{Int}, a_{\tilde{A}} \mapsto \texttt{Bool}\}$$

$$\{a \mapsto A\langle \texttt{Int}, \texttt{Bool}\rangle\}$$

3. Completion

Variational unification

$$A\langle B\langle \texttt{Int},\texttt{Bool}\rangle, a\rangle \equiv^{?} B\langle a, \texttt{Bool}\rangle$$

1. Qualification

$$A\langle B\langle \texttt{Int},\texttt{Bool}\rangle, a_{\tilde{A}}\rangle \equiv^{?} B\langle a_B, \texttt{Bool}\rangle$$

swap

2. Solve qualified problem

$$B\langle A\langle \texttt{Int}, a_{\tilde{A}B}\rangle, A\langle \texttt{Bool}, a_{\tilde{A}\tilde{B}}\rangle\rangle \equiv^{?} B\langle a_B, \texttt{Bool}\rangle$$

Goal: transform so we can decompose by alternatives

C-C-SWAP1

$$D'\langle D\langle T_1, T_2\rangle, T_3\rangle \equiv D\langle D'\langle T_1, T_3\rangle, D'\langle T_2, T_3\rangle\rangle$$

Variational unification

$$A\langle B\langle \texttt{Int},\texttt{Bool}\rangle, a\rangle \equiv^? B\langle a,\texttt{Bool}\rangle$$

1. Qualification

$$A\langle B\langle \texttt{Int},\texttt{Bool}\rangle, a_{\tilde{A}}\rangle \equiv^? B\langle a_B,\texttt{Bool}\rangle$$

swap

2. Solve qualified problem

$$B\langle A\langle \texttt{Int}, a_{\tilde{A}B}\rangle, A\langle \texttt{Bool}, a_{\tilde{A}\tilde{B}}\rangle\rangle \equiv^? B\langle a_B,\texttt{Bool}\rangle$$

$$A\langle \texttt{Int}, a_{\tilde{A}B}\rangle \equiv^? a_B \qquad A\langle \texttt{Bool}, a_{\tilde{A}\tilde{B}}\rangle \equiv^? \texttt{Bool}$$

$$\texttt{Bool} \equiv^? \texttt{Bool} \qquad a_{\tilde{A}\tilde{B}} \equiv^? \texttt{Bool}$$

C-IDEMP

$$\frac{T_1 \equiv T \qquad T_2 \equiv T}{D\langle T_1, T_2\rangle \equiv T}$$

$$\{a_B \mapsto A\langle \texttt{Int}, a_{\tilde{A}B}\rangle,\ a_{\tilde{A}\tilde{B}} \mapsto \texttt{Bool}\}$$

3. Completion

$$\{a \mapsto B\langle A\langle \texttt{Int}, c\rangle, A\langle d, \texttt{Bool}\rangle\rangle\}$$

Important results

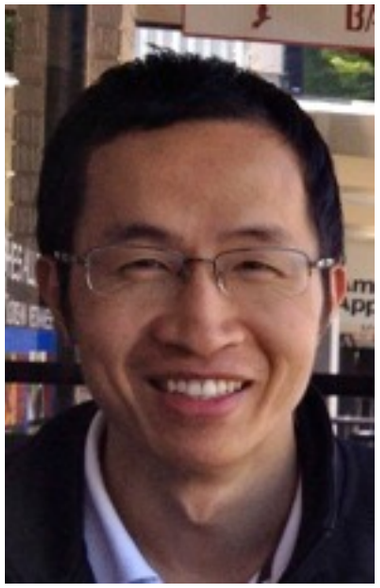
Variational unification is:

- sound, complete, and most general
- polynomially bounded: $O(l^2 r + l r^2)$

Variational type inference is sound, complete, and principal

Extending Type Inference to Variational Programs
TOPLAS, 2014

Surprising applications



Sheng Chen
@ Lafayette

- Improving error location in *plain* Haskell

  Counter-Factual Typing for Debugging Type Errors
  POPL, 2014

- Improving precision of type inference for GADTs

  Principal Type Inference for GADTs
  POPL, 2016

Outline

- Intro and motivation
- Choice calculus
- Variational type inference + applications
- Brief note on variational data structures

Variational analysis strike force

Christian Kästner
Carnegie Mellon

Martin Erwig
Oregon State

Sven Apel
Passau

Eric Bodden
Darmstadt

Variational …

- parsing in TypeChef
- type checking in TypeChef & FFJ_{PL}
- type inference in VLC
- data-flow analysis in SPL^{LIFT}
- execution in a Haskell DSL
- test execution in VarEx(J)

When implementing variational analyses … the variation gets everywhere!

Must manage variation in …

- lists – token streams, TLDs
- trees – ASTs, types
- maps – symbol tables
- graphs – data-flow, control-flow
- and all kinds of intermediate values

so far we've dealt with this in an ad hoc way …

Variational execution and data structures

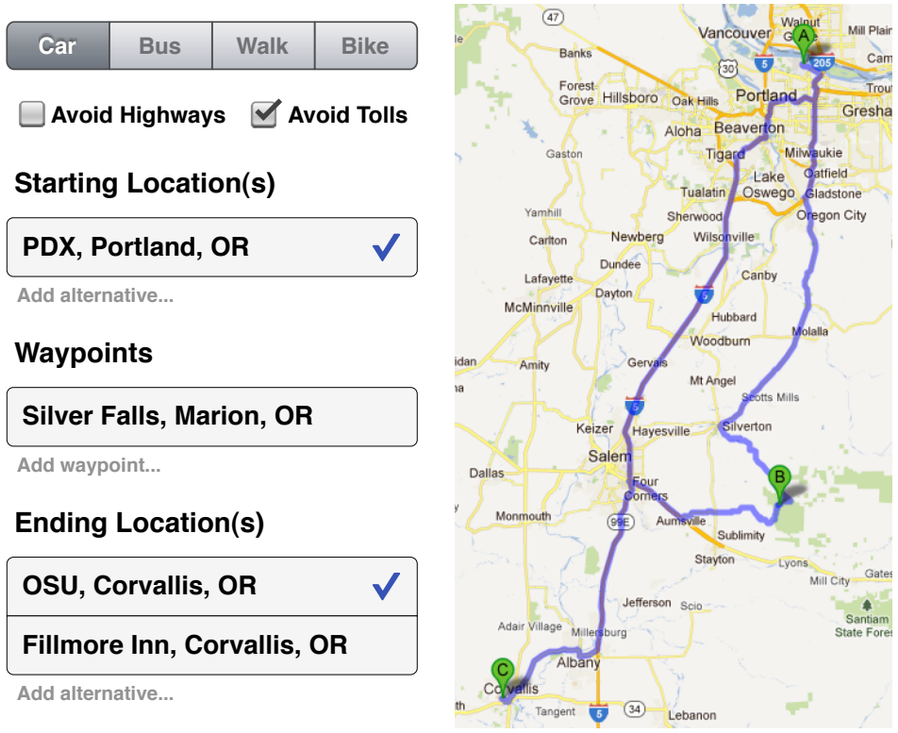
Needed: foundational work

- deal with variation systematically
- solve problems by *considering tradeoffs* among *well-understood, general-purpose* variational data structures

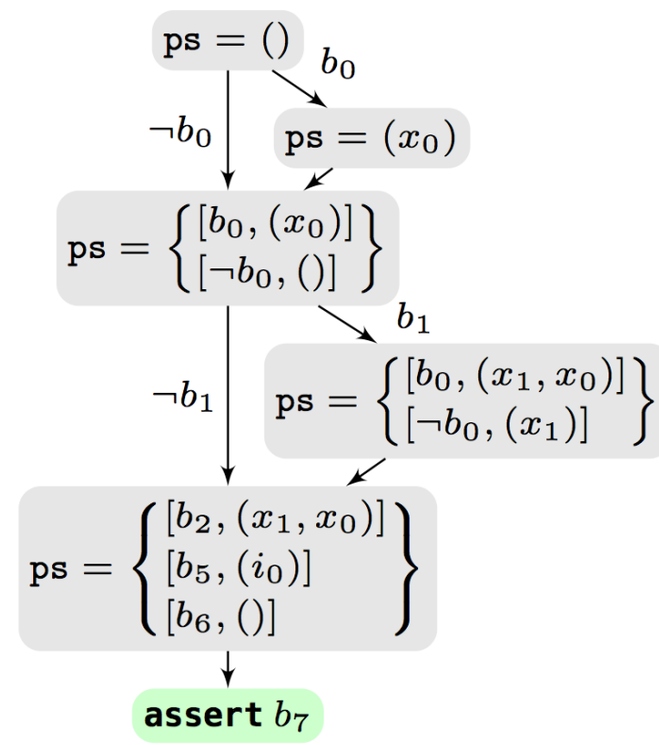
support *reuse* (libraries) and *intentional design decisions*

Variational Data Structures: Exploring Tradeoffs in Computing with Variability
Onward!, 2014

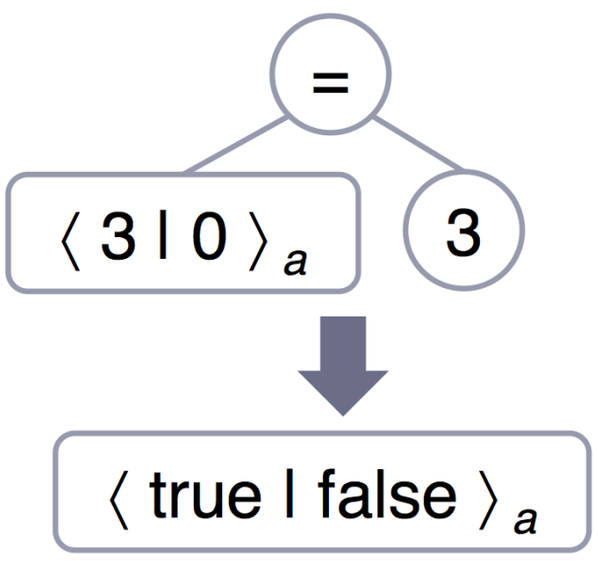
Beyond variational program analyses



route planning software



encoding lists in symbolic execution
Torlak & Bodik (PLDI'14)



evaluation w/ multiple
privacy policies
Austin et al. (PLAS'13)

representations similar to ones we've used