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

Goal: apply this to all of these

Software product line
Solution: analyze variational program

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

configure

program analysis

✓

variational program

configure

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

implementing these things = research

variational result
Outline

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

A core metalanguage for describing variation

\[ e ::= a\{e, \ldots, e\} \mid D\langle e, e \rangle \]

object language AST

choice between two alternatives
synchronized by a dimension

\[ A\langle 2, 3 \rangle + B\langle 4, A\langle 5, 6 \rangle \rangle \]

\[
\begin{align*}
\{A\} & \quad \{\neg A\} \\
2 + B\langle 4, 5 \rangle & \quad 3 + B\langle 4, 6 \rangle \\
\{A,B\} & \quad \{A,\neg B\} & \quad \{\neg A,B\} & \quad \{\neg A,\neg B\} \\
2 + 4 & \quad 2 + 5 & \quad 3 + 4 & \quad 3 + 6
\end{align*}
\]
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

\[ e ::= c \]
\[ \mid x \]
\[ \mid \lambda x. e \]
\[ \mid e \; e \]
\[ \mid D\langle e, e \rangle \]

\textit{variational lambda calculus}

\[ T ::= \kappa \]
\[ \mid a \]
\[ \mid T \rightarrow T \]
\[ \mid D\langle T, T \rangle \]

\textit{variational types}

can instantiate the equivalence laws too:

\[ D\langle T_1 \rightarrow T_2, T_3 \rightarrow T_4 \rangle \equiv D\langle T_1, T_3 \rangle \rightarrow D\langle T_2, T_4 \rangle \]
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 $T_L \sigma \equiv T_R \sigma$

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

$\text{Int} \equiv? a$
$a \equiv? \text{Bool}$

stuck!

$\sigma = \{a \mapsto A\langle\text{Int, Bool}\rangle\}$
Variational unification

\[ A\langle \text{Int}, a \rangle \equiv^? A\langle a, \text{Bool} \rangle \]

\[ A\langle \text{Int}, a_{\bar{A}} \rangle \equiv^? A\langle a_A, \text{Bool} \rangle \]

1. Qualification

\[ \text{Int} \equiv^? a_A \quad a_{\bar{A}} \equiv^? \text{Bool} \]

\{a_A \rightarrow \text{Int}, a_{\bar{A}} \rightarrow \text{Bool}\}

2. Solve qualified problem

\{a \rightarrow A\langle \text{Int}, \text{Bool} \rangle\}

3. Completion
Variational unification

\[ A \langle B \langle \text{Int}, \text{Bool} \rangle, a \rangle \equiv^? B \langle a, \text{Bool} \rangle \]

1. Qualification

\[ A \langle B \langle \text{Int}, \text{Bool} \rangle, a \ A \rangle \equiv^? B \langle a_B, \text{Bool} \rangle \]

swap

\[ B \langle A \langle \text{Int}, a \ \text{\~A}B \rangle, A \langle \text{Bool}, a \ \text{\~A}B \rangle \rangle \equiv^? B \langle a_B, \text{Bool} \rangle \]

2. Solve qualified problem

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

1. Qualification

2. Solve qualified problem

3. Completion

\[ A \langle B \langle \text{Int}, \text{Bool} \rangle, a \rangle \equiv? B \langle a, \text{Bool} \rangle \]

\[ A \langle B \langle \text{Int}, \text{Bool} \rangle, a \tilde{A} \rangle \equiv? B \langle a_B, \text{Bool} \rangle \]

\[ B \langle A \langle \text{Int}, a \tilde{A}B \rangle, A \langle \text{Bool}, a \tilde{A}B \rangle \rangle \equiv? B \langle a_B, \text{Bool} \rangle \]

\[ A \langle \text{Int}, a \tilde{A}B \rangle \equiv? a_B \]

\[ A \langle \text{Bool}, a \tilde{A}B \rangle \equiv? \text{Bool} \]

\[ \text{Bool} \equiv? \text{Bool} \]

\[ a \tilde{A}B \equiv? \text{Bool} \]

\[ \{ a_B \mapsto A \langle \text{Int}, a \tilde{A}B \rangle, a \tilde{A}B \mapsto \text{Bool} \} \]

\[ \{ a \mapsto B \langle A \langle \text{Int}, c \rangle, A \langle d, \text{Bool} \rangle \rangle \} \]
Important results

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

Variational type inference is sound, complete, and principal
Surprising applications

- 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
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