Freely Extending Interpreters

Greg Brown

University of Edinburgh

November 6, 2024

Guile Scheme Interpreter

```
(define (interp t env)
 (match t
  (('fun x t) (lambda (v)
     (interp t `((,x . ,v) ,env))))
  (('app t u) ((interp t env) (interp u env)))
  (x (assq-ref env x))))
```

Why Partial Evaluation?

- Optimising compiler — fast in, fast out
- Dependent type checking — terms in types
- Running programs with holes — testing partial programs

Structure of the Talk

Approximate formal definitions for

- languages
- interpreters
- partial evaluators

This is work in progress

Structure of the Talk

Approximate formal definitions for

- languages — second order algebraic theories
- interpreters — setoid actions
- partial evaluators — setoid models

This is work in progress

Theory of STLC

Types

$$A \Rightarrow B$$

Operators

$$A \Rightarrow B, A \vdash \$: B$$
$$(A)B \vdash \lambda : A \Rightarrow B$$

Axioms

$$M : (A)B, N : A \rhd \vdash (\lambda x.M[x]) \; \$ \; N \cong M[N] \quad : B$$
$$M : A \Rightarrow B \rhd \vdash (\lambda x.M \; \$ \; x) \cong M \quad : A \Rightarrow B$$

Second Order Algebraic Theories

Definition

A *theory* Σ consists of:

T	types	A, B
O	binding operators	$((\Gamma_i)A_i)_{i<k} \vdash o : B$
E	axioms	$\Theta \triangleright \Gamma \vdash t \cong u : A$

$\Gamma \vdash A \ni t$ set of terms

$t[\sigma]$ capture-avoiding substitution

Set Model of STLC

Types

$$[\![A \Rightarrow B]\!] := [\![A]\!] \to [\![B]\!]$$

Expressions

$$M(\Gamma; A) := [\![\Gamma]\!] \to [\![A]\!]$$

Operations

$$[\![\$]\!]\ (f, g)\ \gamma := f\ \gamma\ (g\ \gamma)$$
$$[\![\lambda]\!]\ f\ \gamma := (x \mapsto f\ (\gamma, x))$$

Substitution

$$\eta\ i\ \gamma := \gamma(i)$$
$$\mu\ (f; \sigma)\ \gamma := f\ (\sigma\ \gamma)$$

Σ-Models in General

Definition

A Σ-*model* consists of:

$M(-;-)$ expressions

$[\![o]\!]$ semantics for each operator

η variable embedding

μ substitution operation

such that

- $[\![o]\!]$ commutes with μ
- (μ, η) is a substitution monoid

} substitution lemma

- all instantiations of the axioms hold

Partial Evaluators and Models

- Expressions have free variables
- Substitute variables for expressions
- Equivalent terms have equal expressions

Hypothesis

Σ-models formalise strict partial evaluators.

Interpreters are Not Models

- Interpreters have no free variables
- Interpreters have no substitution

Hypothesis

Σ-setoid actions formalise interpreters.

Define action structures and actions first

Σ-Action-Structures

Definition

A Σ-*action-structure* consists of:

$\mathsf{Val}(-)$	values	
$\mathsf{act}(-;-)$	action on terms	$(\Gamma \vdash A) \times \mathsf{Val}(\Gamma) \to \mathsf{Val}(A)$

Example

Closed Terms $\mathsf{Val}(A) := \bullet \vdash A$; $\mathsf{act}(t;\gamma) := t[\gamma]$

Closed Expressions

$$\mathsf{Val}(A) := M(\bullet; A)$$

$$\mathsf{act}(t;\gamma) := \mu(\llbracket t \rrbracket; \gamma)$$

Σ-Actions

Definition

A Σ-*action* is a Σ-action-structure $(\mathsf{Val}, \mathsf{act})$ such that

- $\Gamma \vdash t \approx u : A \implies \forall \gamma . \mathsf{act}(t; \gamma) = \mathsf{act}(u; \gamma)$
- $\mathsf{act}(x; \gamma) = \gamma(x)$
- $\mathsf{act}(t[\sigma]; \gamma) = \mathsf{act}(t; \mathsf{act}(\sigma; \gamma))$

Our interpreter is not a Σ-action.

Our Interpreter is Not A Σ-Action

`interp` does not respect many equivalences; e.g.

```
(let ((identity (lambda (x) x)))
  (equal?
   (interp 'f                `((f . ,identity)))
   (interp '(fun x (f x)) `((f . ,identity)))))
```

Also congruence under λ; beta at some functions

Choosing Our Equality

$$\texttt{v} \sim_A \texttt{w} \iff \texttt{v} = \texttt{w} \text{ for base types}$$
$$\texttt{f} \sim_{A \Rightarrow B} \texttt{g} \iff \forall \texttt{v} \sim_A \texttt{w}.\ (\texttt{f v}) \sim_B (\texttt{g w})$$

Σ-Setoid Actions

Definition

A Σ-*setoid action* is a Σ-action-structure with a type-indexed equivalence relation $\sim$ such that

- $\Gamma \vdash t \approx u : A \wedge \gamma \sim_\Gamma \delta \implies \mathsf{act}(t;\gamma) \sim_A \mathsf{act}(u;\delta)$
- $\mathsf{act}(x;\gamma) \sim_A \gamma(x)$
- $\mathsf{act}(t[\sigma];\gamma) \sim_A \mathsf{act}(t;\mathsf{act}(\sigma;\gamma))$

Our Interpreter Respects ∼

```
(let ((identity (lambda (x) x)))
  (obs-equal?
   (interp 'f                `((f . ,identity)))
   (interp '(fun x (f x)) `((f . ,identity)))))
```

Setoid Models

Definition

A Σ-setoid model is a Σ-model M with
a type-indexed equivalence relation $\sim$ on closed expressions
such that

$$\sigma_1 \sim_\Gamma \sigma_2 \implies \forall m \in M(\Gamma; A).\ \mu(m; \sigma_1) \sim_A \mu(m; \sigma_2)$$

I.e. $M(\bullet; -)$ is a setoid action.

Extending Interpreters

Definition

A setoid model M extends a setoid action Val when there are functions $\mathsf{val} : \mathsf{Val}(A) \to M(\bullet; A)$ such that

- $x \sim_A y \implies \mathsf{val}\ x \sim_A \mathsf{val}\ y$
- $\mathsf{val}\ (\mathsf{act}(t; \gamma)) \sim_A \mu([\![t]\!]; \mathsf{val}\ \gamma)$

Hypothesis

The free extension of a setoid action is its free partial evaluator

Future Work

- Construct free extension of our interpreter
- Apply to other languages
- Extend definitions to existing partial evaluators

Summary

Approximate formal definitions for

- languages — second order algebraic theories
- interpreters — setoid actions
- partial evaluators — setoid models

Email greg.brown01@ed.ac.uk