

Shoggoth: A Formalised Logic for Strategic Rewriting

Xueying Qin¹

Rob van Glabbeek¹, Peter Höfner², Liam O'Connor¹, Michel Steuwer¹

¹ The University of Edinburgh

² Australian National University

July 17, 2023

Table of Contents

0. Introduction
1. Semantics of Our Strategic Rewriting Language
2. Location Based Weakest Precondition Calculus
3. Conclusion and Future Work

Introduction

Strategic Rewriting Languages

The core calculus of strategic rewriting languages like ELEVATE[1], Stratego[4] and Strafunski[2] has a simple construct: atomic strategies and strategy combinators.

Atomic strategy

An atomic strategy is a *rewrite rule*:

$add_{com} : a + b \rightsquigarrow b + a$

$mapFusion : map\ f\ (map\ g\ xs) \rightsquigarrow map\ (f \circ g)\ xs$

Composed strategy

$add_{com} ; add_{id} \quad add_{com} <+ mult_{com}$

$repeat(mapFusion)$

Strategy combinator

Strategy combinators compose strategies together and controls the application of atomic strategies:

$s ; t$ Sequential composition, apply s then t

$s <+ t$ Left choice, if fail to apply s then t

$repeat(s)$ Keep applying s until inapplicable

Importance of Strategic Rewriting Languages

- Strategic rewriting languages provide programmers with combinators and generic traversals that allow them to: 1) control the application of rewrite rules; 2) reuse rewrite rules
- Many application areas: for writing program optimisation (ELEVATE), writing interpreter/compiler for DSLs (Spoofox/Stratego) etc.

Strategies can go wrong

- **Result in error** - an atomic strategy is not defined for certain expressions or strategies are not well composed
- **Do not terminate** - for example : *repeat*(SKIP)
- **Do not rewrite an expression into desired form**

Therefore, we would like a formal understanding of these strategies and a framework that allows us to formally reason about these strategies.

- We design, formalise and mechanise using Isabelle **denotational semantics** of a strategic rewriting language, we prove computational soundness and adequacy between the denotational semantics and an existing big-step operational semantics.
- We design, formalise and mechanise using Isabelle the **weakest precondition calculus** for the strategic rewriting language. We prove its soundness w.r.t. the denotational semantics.
- We demonstrate how to use the weakest precondition calculus to prove properties of strategic rewriting.

Semantics of Our Strategic Rewriting Language

System S

System S [5] is the core calculus of the strategic rewriting languages we study that contains atomic strategies (rewrite rules) and strategy combinators which compose strategies and perform expression traversals.

Expression

The expressions being rewritten by strategies are in the form of:

$$\text{Expressions}(\mathbb{E}) \quad e := \text{Id} \mid \text{Fun } e \ e \mid \text{App } e \ e$$

Strategy

$Strategy(\mathcal{S}) \quad s ::=$ SKIP (Always success) | ABORT (Always fail)
| *atomic* (Atomic strategy)
| X (Variable)
| $s_1 ; s_2$ (Sequential composition)
| $s_1 <+ s_2$ (Left choice)
| $s_1 <+> t_2$ (Nondeterministic choice)
| *one*(s) (Apply s to one child)
| *some*(s) (Apply s to as many children as possible)
| *all*(s) (Apply s to all children)
| $\mu X.s$ (Fixed point operator)

Denotational Semantics - Domain (0)

We define a denotational semantics. Since our strategies include a fixed point operator, we need to ensure the denotational semantics of strategies is monotone. To do so, we provide a *Plotkin powerdomain* with *Egli-Milner ordering* where divergence (*div*) is the bottom element, and point-wise lift it to the domain (which is a CPO) for defining our denotational semantics.

The Plotkin powerdomain

$$\mathcal{D}_p = \mathcal{P}_{-\emptyset}(\mathbb{E} \cup \{err\} \cup \{div\})$$

Egli-Milner ordering

$$a \leq b \iff (\forall x \in a. \exists y \in b. x \leq y) \wedge (\forall y \in b. \exists x \in a. x \leq y)$$

The domain

$$\mathcal{D} = \mathbb{E} \rightarrow \mathcal{D}_p$$

Denotational Semantics by Examples - Skip, Abort and Atomic

Basics

Variable(\mathbb{V}) $X Y Z \dots$ $\llbracket S \rrbracket : \Gamma_S \rightarrow \mathcal{D}$
Semantic Environment(Γ_S) $\xi : \mathbb{V} \rightarrow \mathcal{D}$
 $\xi := \perp \mid \xi[X \mapsto d]$

Example

$add_{com} : a + b \rightsquigarrow b + a$ $1 + 3 \xrightarrow{add_{com}} 3 + 1$ $1 + 3 \xrightarrow{SKIP} 1 + 3$ $1 + 3 \xrightarrow{ABORT} err$

Semantics

$\llbracket atomic \rrbracket \xi = \lambda e. \{ atomic(e) \mid atomic(e) \text{ def} \} \cup \{ err \mid atomic(e) \text{ undef} \}$

$\llbracket SKIP \rrbracket \xi = \lambda e. \{ e \}$

$\llbracket ABORT \rrbracket \xi = \lambda e. \{ err \}$

Example

$$\text{add}_{id} : 0 + a \rightsquigarrow a$$

$$\text{add}_{com} : a + b \rightsquigarrow b + a$$

$$3 + 0 \xrightarrow{\text{add}_{com}; \text{add}_{id}} 3$$

Semantics

$$(\text{ ; }_s) : \mathcal{D} \rightarrow \mathcal{D} \rightarrow \mathcal{D}$$

$$(s \text{ ; }_s t)(e) = \bigcup \{t(x) \mid x \in s(e) \cap \mathbb{E}\} \cup \{x \mid x \in s(e) \cap \{\text{div}, \text{err}\}\}$$

$$\llbracket s \text{ ; } t \rrbracket \xi = \llbracket s \rrbracket \xi \text{ ; }_s \llbracket t \rrbracket \xi$$

Denotational Semantics by Examples - Left Choice

Example

$$add_{com} : a + b \rightsquigarrow b + a$$

$$3 + 6 \xrightarrow{add_{com} <+ mult_{com}} 6 + 3$$

$$mult_{com} : a * b \rightsquigarrow b * a$$

$$3 * 6 \xrightarrow{add_{com} <+ mult_{com}} 6 * 3$$

Example - Try

$$try(s) = s <+ SKIP$$

$$3 + 6 \xrightarrow{try(mult_{com})} 3 + 6$$

Semantics

$$(<+_s) : \mathcal{D} \rightarrow \mathcal{D} \rightarrow \mathcal{D}$$

$$(s <+_s t)(e) = (s(e) \setminus \{err\}) \cup \{y \mid y \in t(e) \wedge err \in s(e)\}$$

$$\llbracket s <+ t \rrbracket \xi = \llbracket s \rrbracket \xi <+_s \llbracket t \rrbracket \xi$$

Example

$$add_{id} : 0 + a \rightsquigarrow a$$

$$add_{com} : a + b \rightsquigarrow b + a$$

$$0 + 3 \xrightarrow{add_{id} \langle + \rangle add_{com}} 3$$

$$0 + 3 \xrightarrow{add_{id} \langle + \rangle add_{com}} 3 + 0$$

Semantics

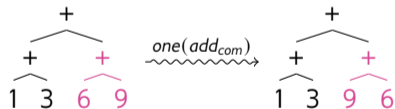
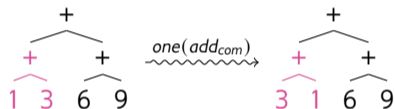
$$\langle + \rangle_s : \mathcal{D} \rightarrow \mathcal{D} \rightarrow \mathcal{D}$$

$$\begin{aligned} (s \langle + \rangle_s t)(e) = & \{x \mid x \in s(e) \cap \mathbb{E}\} \cup \{div \mid div \in s(e)\} \\ & \cup \{y \mid y \in t(e) \cap \mathbb{E}\} \cup \{div \mid div \in t(e)\} \\ & \cup \{err \mid err \in s(e) \cap t(e)\} \end{aligned}$$

$$\llbracket s \langle + \rangle t \rrbracket \xi = \llbracket s \rrbracket \xi \langle + \rangle_s \llbracket t \rrbracket \xi$$

Example

$add_{com} : a + b \rightsquigarrow b + a$



Example

$add_{com} : a + b \rightsquigarrow b + a$



Semantics

$(one_s) : \mathcal{D} \rightarrow \mathcal{D}$

$$one_s(s)(e) = \left\{ \begin{array}{l} n \\ x \quad e_2 \end{array} \mid e = \begin{array}{l} n \\ e_1 \quad e_2 \end{array} \wedge x \in s(e_1) \cap \mathbb{E} \right\}$$

$$\cup \left\{ \begin{array}{l} n \\ e_1 \quad x \end{array} \mid e = \begin{array}{l} n \\ e_1 \quad e_2 \end{array} \wedge x \in s(e_2) \cap \mathbb{E} \right\}$$

$$\cup \left\{ div \mid e = \begin{array}{l} n \\ e_1 \quad e_2 \end{array} \wedge div \in s(e_1) \cup s(e_2) \right\}$$

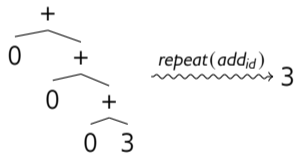
$$\cup \left\{ err \mid e = ld \vee (e = \begin{array}{l} n \\ e_1 \quad e_2 \end{array} \wedge err \in s(e_1) \cap s(e_2)) \right\}$$

$$\llbracket one(s) \rrbracket \xi = one_s(\llbracket s \rrbracket \xi)$$

Example - Repeat

$$\text{repeat}(s) = \mu X. \text{try}(s ; X)$$

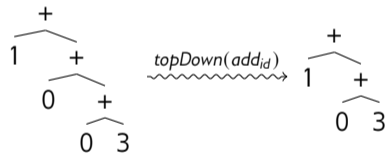
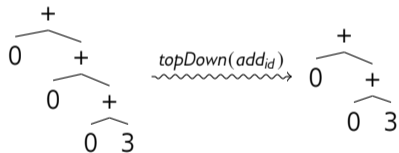
$$\text{add}_{id} : 0 + a \rightsquigarrow a$$



Example - Top down

$topDown(s) = \mu X.s <+ one(X)$

$add_{id} : 0 + a \rightsquigarrow a$



Semantics

$$\llbracket X \rrbracket \xi = \xi X$$

$$\llbracket \mu X.s \rrbracket \xi = \mu X. \llbracket s \rrbracket (\xi[X \mapsto X])$$

Big-Step operational semantics

The existing big-step operational semantics does not model divergence. In general, it takes the form of:

$$e \xrightarrow{s} e'$$

where e' can be either an expression or an error.

Correspondence to Existing Big-Step Operational Semantics(1)

Closed strategy

$$fv(s_\bullet) = \emptyset$$

Computational soundness

$$\frac{e \xrightarrow{s_\bullet} e'}{e' \in \llbracket s_\bullet \rrbracket \xi e}$$

Computational adequacy

$$\frac{e' \in \llbracket s_\bullet \rrbracket \xi e \wedge e' \neq div}{e \xrightarrow{s_\bullet} e'}$$

Equivalence?

We believe if we add divergence to the big-step operational semantics, we would be able to conclude that these two semantics are equivalent.

Location Based Weakest Precondition Calculus

Strategies Can Go Wrong

Error

$add_{id} : 0 + a \rightsquigarrow a$ $add_{com} : a + b \rightsquigarrow b + a$ $mult_{com} : a * b \rightsquigarrow b * a$

$6 + 3 \xrightarrow{add_{id}} err$

$\begin{array}{c} + \\ \underbrace{\quad} \\ 6 \quad 3 \end{array} \xrightarrow{one(add_{com})} err$

$0 + 3 \xrightarrow{add_{com}; mult_{com}} err$

$0 + 3 \xrightarrow{add_{id} <+ add_{com}; add_{com}} err$

Divergence

$e \xrightarrow{repeat(SKIP)} div$

$6 + 3 \xrightarrow{repeat(add_{com})} div$

Undesired result

We want $3 + 0$

$0 + 3 \xrightarrow{add_{id} <+ add_{com}} 3$

Weakest precondition

Given a program S and a postcondition Q , a weakest precondition is a predicate P_w such that for any precondition P :

$$\{P\}S\{Q\} \Leftrightarrow (P \Rightarrow P_w)$$

The challenge in design

We have strategies that can traverse the syntax tree and control at what location in the syntax tree to apply a strategy — we need a notion of “location” in our formulas.

Our solution

We introduce the location as a path in the syntax tree into our formulas.

Tag and environment

Tag(\mathbb{T}) $t := \cdot \mid \uparrow$

Logic Environment(Γ_L) $\zeta : (\mathbb{V} \times \mathbb{T}) \rightarrow (\mathbb{L} \rightarrow \mathcal{P}(\mathbb{E}) \rightarrow \mathcal{P}(\mathbb{E}))$

$\zeta := \perp \mid \zeta [(X, t) \mapsto f]$

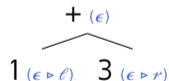
Position

Position $i := \ell \mid r$

Location

Location(\mathbb{L}) $l := \epsilon \mid l \triangleright i \mid i \triangleleft l$

Location in a Tree



Helper partial functions

$$\text{update} : \mathbb{L} \rightarrow \mathbb{E} \rightarrow \mathcal{P}_{-\emptyset}(\mathbb{E} \cup \{\text{err}\} \cup \{\text{div}\}) \rightarrow \mathcal{P}_{-\emptyset}(\mathbb{E} \cup \{\text{err}\} \cup \{\text{div}\})$$

$$\text{update}(\epsilon, e, xs) = xs$$

$$\text{update}(\ell \triangleleft l, \bigwedge_{e_1 e_2}^n, xs) = \{ \bigwedge_{x e_2}^n \mid x \in \text{update}(l, e_1, xs) \cap \mathbb{E} \} \cup (xs \cap \{\text{err}, \text{div}\})$$

$$\text{update}(\neg \triangleleft l, \bigwedge_{e_1 e_2}^n, xs) = \{ \bigwedge_{e_1 x}^n \mid x \in \text{update}(l, e_2, xs) \cap \mathbb{E} \} \cup (xs \cap \{\text{err}, \text{div}\})$$

$$\text{lookup} : \mathbb{L} \rightarrow \mathbb{E} \rightarrow \mathbb{E}$$

$$\text{lookup}(\epsilon, e) = e$$

$$\text{lookup}(\ell \triangleleft l, \bigwedge_{e_1 e_2}^n) = \text{lookup}(l, e_1)$$

$$\text{lookup}(\neg \triangleleft l, \bigwedge_{e_1 e_2}^n) = \text{lookup}(l, e_2)$$

Definition

$$wp_{s@l}\zeta(P) = Q$$

$$(wp_s : \mathbb{L} \rightarrow \Gamma_L \rightarrow \mathcal{P}(\mathbb{E}) \rightarrow \mathcal{P}(\mathbb{E}))$$

If for a set of expressions described by the postcondition P , a non-empty set of expressions described by the precondition Q can be found,

then each expression in the precondition set will be successfully transformed into an expression in the postcondition set, by applying strategy s at location l in the logic environment ζ .

Definition

$$wp_{s@l}^{\uparrow} \zeta(P) = Q$$

$$(wp_s^{\uparrow} : \mathbb{L} \rightarrow \Gamma_L \rightarrow \mathcal{P}(\mathbb{E}) \rightarrow \mathcal{P}(\mathbb{E}))$$

If for a set of expressions described by the postcondition P , a non-empty set of expressions described by the precondition Q can be found,

then each expression in the precondition set will be successfully transformed into an expression in the postcondition set **or result in error**, by applying strategy s at location l in the logic environment ζ .

It is used for defining the weakest precondition for total correctness.

Definition of Unsuccessful Execution

A strategy cannot execute successfully

$$wp_{s@l}\zeta(\mathbb{U}) = \emptyset$$

\Rightarrow

The strategy s cannot execute successfully on any input expression.

$$wp_{s@l}\zeta(P) = \emptyset$$

\Rightarrow

The strategy s cannot transform the any input expression into an output expression in P .

Invalid input expression

$$e \Rightarrow wp_{s@l}\zeta(\mathbb{U}) \neq \emptyset \wedge e \notin wp_{s@l}\zeta(\mathbb{U})$$

\Rightarrow

The strategy s cannot execute successfully on the input expression e .

$$e \Rightarrow wp_{s@l}\zeta(P) \neq \emptyset \wedge e \notin wp_{s@l}\zeta(P)$$

\Rightarrow

The strategy s cannot transform the input expression e into an output expression in P .

A Strategy Is Not Well-Composed

Example

$$add_{com} : a + b \rightsquigarrow b + a$$

$$mult_{com} : a * b \rightsquigarrow b * a \text{ ; } mult_{com} \text{ (Bad?)}$$

Wp for atomic strategy

$$wp_{atomic@l} \zeta(P) = \{e \mid update(l, e, \llbracket atomic \rrbracket \perp (lookup(l, e))) \subseteq P\}$$

$$wp_{atomic@l}^{\uparrow} \zeta(P) = \{e \mid update(l, e, \llbracket atomic \rrbracket \perp (lookup(l, e))) \subseteq P \cup \{err\}\}$$

Wp for add_{com} and $mult_{com}$

$$wp_{add_{com}@\epsilon} \zeta(\mathbb{U}) = \{e \mid e = App (+) m n\}$$

$$wp_{mult_{com}@\epsilon} \zeta(\mathbb{U}) = \{e \mid e = App (*) m n\}$$

Wp of sequential composition

$$wp_{s;t@l} \zeta(P) = wp_{s@l} \zeta(wp_{t@l} \zeta(P))$$

$$wp_{s;t@l}^{\uparrow} \zeta(P) = wp_{s@l}^{\uparrow} \zeta(wp_{t@l}^{\uparrow} \zeta(P))$$

Checking invalid composition

$$wp_{add_{com}; mult_{com}@\epsilon} \zeta(\mathbb{U}) = \emptyset \text{ (Bad!)}$$

A Strategy Is Not Well-Composed for Desired Output

Example

$$\text{mult}_{\text{zero}} : 0 * a \rightsquigarrow 0 \quad \text{mult}_{\text{com}} : a * b \rightsquigarrow b * a$$

$$e \xrightarrow{\text{mult}_{\text{com}} \langle + \rangle \text{mult}_{\text{zero}} ; \text{mult}_{\text{com}}} \{e \mid e = \text{App} (*) 0 m\} \quad (\text{Bad?})$$

Wp of non-deterministic choice

$$\text{wp}_{s \langle + \rangle t @ I} \zeta(P) = (\text{wp}_{t @ I}^{\uparrow} \zeta(P) \cap \text{wp}_{s @ I} \zeta(P)) \cup (\text{wp}_{s @ I}^{\uparrow} \zeta(P) \cap \text{wp}_{t @ I} \zeta(P))$$

$$\text{wp}_{s \langle + \rangle t @ I}^{\uparrow} \zeta(P) = \text{wp}_{s @ I}^{\uparrow} \zeta(P) \cap \text{wp}_{t @ I}^{\uparrow} \zeta(P)$$

Checking invalid composition for P

$$\text{wp}_{\text{mult}_{\text{com}} \langle + \rangle \text{mult}_{\text{zero}} ; \text{mult}_{\text{com}}} @ \epsilon \zeta(\{e \mid e = \text{App} (*) 0 m\}) = \emptyset \quad (\text{Bad!})$$

Example

$$\text{add}_{\text{com}} : a + b \rightsquigarrow b + a \quad \text{mult}_{\text{com}} : a * b \rightsquigarrow b * a$$

$$3 * 6 \xrightarrow{\text{mult}_{\text{com}} <+ \text{add}_{\text{com}} ; \text{mult}_{\text{com}}} 3 * 6 \quad 3 + 6 \xrightarrow{\text{mult}_{\text{com}} <+ \text{add}_{\text{com}} ; \text{mult}_{\text{com}}} \text{err}$$

Wp of left choice

$$\text{wp}_{s <+ t @ l} \zeta(P) = \text{wp}_{s @ l} \zeta(P) \cup (\text{wp}_{s @ l}^{\uparrow} \zeta(P) \cap \text{wp}_{t @ l} \zeta(P))$$

$$\text{wp}_{s <+ t @ l}^{\uparrow} \zeta(P) = \text{wp}_{s @ l} \zeta(P) \cup (\text{wp}_{s @ l}^{\uparrow} \zeta(P) \cap \text{wp}_{t @ l}^{\uparrow} \zeta(P))$$

Checking Invalid Input

$$\text{wp}_{\text{mult}_{\text{com}} <+ \text{add}_{\text{com}} ; \text{mult}_{\text{com}} @ \epsilon} \zeta(\mathbb{U}) = \{e \mid e = \text{App} (*) a b\}$$

$$3 * 6 \in \{e \mid e = \text{App} (*) a b\} \quad (\text{Good!})$$

$$3 + 6 \notin \{e \mid e = \text{App} (*) a b\} \quad (\text{Bad!})$$

Invalid Input for A Desired Output

Example

$$\text{mult}_{\text{zero}} : 0 * a \rightsquigarrow 0 \quad \text{mult}_{\text{com}} : a * b \rightsquigarrow b * a$$

$$3 * 4 \xrightarrow{\text{mult}_{\text{com}} <+ \text{mult}_{\text{zero}} ; \text{mult}_{\text{com}}} ? \{e \mid e = \text{App} (*) 0 m\}$$

Checking invalid input for P

$$\text{wp}_{\text{mult}_{\text{com}} <+ \text{mult}_{\text{zero}} ; \text{mult}_{\text{com}}} @ \epsilon \zeta (\{e \mid e = \text{App} (*) 0 m\}) = \{e \mid e = \text{App} (*) 0 m\}$$

$$3 * 4 \notin \{e \mid e = \text{App} (*) 0 m\} \quad (\text{Bad!})$$

Detect Divergence (0)

The given strategy will diverge, i.e., will not lead to any successful execution.

Example

repeat(SKIP) **Bad?**

Wp of fixed point operator

$$wp_{\mu X.s@l}\zeta(P) = [\text{LFP } \mathcal{X} : \Delta](l)(P) \quad wp_{\uparrow\mu X.s@l}\zeta(P) = [\text{LFP } \mathcal{Y} : \Delta](l)(P)$$

Where:

$$\Delta = \begin{cases} \mathcal{X}(l)(P) & = wp_{s@l}\zeta[(X, \cdot) \mapsto \mathcal{X}, (X, \uparrow) \mapsto \mathcal{Y}](P) \\ \mathcal{Y}(l)(P) & = wp_{s@l}^{\uparrow}\zeta[(X, \cdot) \mapsto \mathcal{X}, (X, \uparrow) \mapsto \mathcal{Y}](P) \end{cases}$$

$$wp_{X@l}\zeta(P) = \zeta(X, \cdot)(l)(P) \quad (\text{where } \zeta(X, \cdot) \text{ **def.**})$$

$$wp_{X@l}^{\uparrow}\zeta(P) = \zeta(X, \uparrow)(l)(P) \quad (\text{where } \zeta(X, \uparrow) \text{ **def.**})$$

Detect Divergence (1)

Example

$\text{repeat}(\text{SKIP})$ **Bad?**

Wp for repeat

$$\text{wp}_{\mu X.\text{try}(s; X) @ l} \zeta(P) = \text{wp}_{\mu X.\text{try}(s; X) @ l}^{\uparrow} \zeta(P) = [\text{LFP } \mathcal{X} : \Delta](l)(P)$$

Where:

$$\begin{aligned} \Delta = \mathcal{X}(l)(P) = & \text{wp}_{s @ l} \zeta[(X, \cdot) \mapsto \mathcal{X}, (X, \uparrow) \mapsto \mathcal{X}](\mathcal{X}(l)(P)) \\ & \cup (\text{wp}_{s @ l}^{\uparrow} \zeta[(X, \cdot) \mapsto \mathcal{X}, (X, \uparrow) \mapsto \mathcal{X}](\mathcal{X}(l)(P)) \cap P) \end{aligned}$$

Checking divergence

$$\text{wp}_{\text{repeat}(\text{SKIP}) @ \epsilon} \zeta(\mathbb{U}) = \emptyset \quad \text{Bad!}$$

Detect Divergence - Demonic Non-determinism

We take divergence seriously — a strategy which can either diverge or successfully execute leads to diverge.

Example

$\text{SKIP} \langle + \rangle \text{repeat}(\text{SKIP})$ **Bad?**

Checking divergence

$\text{wp}_{\text{SKIP} \langle + \rangle \text{repeat}(\text{SKIP}) @ \epsilon} \zeta(\mathbb{U}) = \emptyset$ **Bad!**

Example

$$\begin{array}{c}
 \text{add}_{com} : a + b \rightsquigarrow b + a \\
 \begin{array}{ccc}
 \begin{array}{c} + \\ \frown \\ 6 \quad 3 \end{array} & \xrightarrow{\text{one}(\text{add}_{com})} & \text{err} \\
 \begin{array}{c} + \\ \frown \quad \frown \\ + \quad + \\ \frown \quad \frown \\ 1 \quad 3 \quad 6 \quad 9 \end{array} & \xrightarrow{\text{one}(\text{add}_{com})} & \begin{array}{c} + \\ \frown \quad \frown \\ + \quad + \\ \frown \quad \frown \\ 3 \quad 1 \quad 6 \quad 9 \end{array}
 \end{array}
 \end{array}$$

Wp of one

$$\begin{aligned}
 \text{wp}_{\text{one}(s)@l} \zeta(P) &= (\text{wp}_{s@l \triangleright \ell}^{\uparrow} \zeta(P) \cap \text{wp}_{s@l \triangleright r} \zeta(P)) \cup (\text{wp}_{s@l \triangleright r}^{\uparrow} \zeta(P) \cap \text{wp}_{s@l \triangleright \ell} \zeta(P)) \\
 \text{wp}_{\text{one}(s)@l}^{\uparrow} \zeta(P) &= \{e \mid \text{lookup}(l, e) = \text{Id}\} \cup (\text{wp}_{s@l \triangleright 0}^{\uparrow} \zeta(P) \cap \text{wp}_{s@l \triangleright 1}^{\uparrow} \zeta(P))
 \end{aligned}$$

Checking invalid input

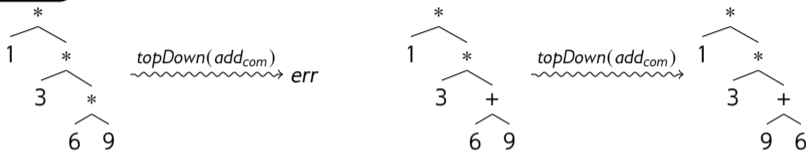
$$wp_{\text{One}(\text{add}_{\text{com}})} \circ \epsilon \zeta(\{e \mid e = \text{App} (*) 0 m\}) = \{e \mid e = \begin{array}{c} x \\ \wedge \\ + \quad e_r \\ \wedge \\ m \quad n \end{array}\} \cup \{e \mid e = \begin{array}{c} x \\ \wedge \\ e_l \quad + \\ \wedge \\ m \quad n \end{array}\}$$

$$\begin{array}{c} + \\ \wedge \\ 6 \quad 3 \end{array} \notin \{e \mid e = \begin{array}{c} x \\ \wedge \\ + \quad e_r \\ \wedge \\ m \quad n \end{array}\} \cup \{e \mid e = \begin{array}{c} x \\ \wedge \\ e_l \quad + \\ \wedge \\ m \quad n \end{array}\} \quad \text{Bad!}$$

$$\begin{array}{c} + \\ \wedge \\ + \quad + \\ \wedge \quad \wedge \\ 1 \quad 3 \quad 6 \quad 9 \end{array} \in \{e \mid e = \begin{array}{c} x \\ \wedge \\ + \quad e_r \\ \wedge \\ m \quad n \end{array}\} \cup \{e \mid e = \begin{array}{c} x \\ \wedge \\ e_l \quad + \\ \wedge \\ m \quad n \end{array}\} \quad \text{Good!}$$

Reasoning About Traversals - Top Down (0)

Example



Wp of top down

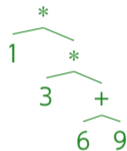
$$wp_{\mu X.s <+one(X)@l} \zeta(P) = [LFP \mathcal{X} : \Delta](l)(P) \quad wp_{\mu X.s <+one(X)@l}^{\uparrow} \zeta(P) = [LFP \mathcal{Y} : \Delta](l)(P)$$

Where:

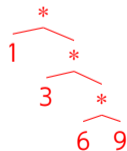
$$\Delta = \begin{cases} \mathcal{X}(l)(P) &= wp_{s@l} \zeta[(X, \cdot) \mapsto \mathcal{X}, (X, \uparrow) \mapsto \mathcal{Y}](P) \cup (wp_{s@l}^{\uparrow} \zeta[(X, \cdot) \mapsto \mathcal{X}, (X, \uparrow) \mapsto \mathcal{Y}](P) \\ &\cap ((\mathcal{Y}(l \triangleright \ell)(P) \cap \mathcal{X}(l \triangleright r)(P)) \cup (\mathcal{Y}(l \triangleright r)(P) \cap \mathcal{X}(l \triangleright \ell)(P)))) \\ \mathcal{Y}(l)(P) &= wp_{s@l} \zeta[(X, \cdot) \mapsto \mathcal{X}, (X, \uparrow) \mapsto \mathcal{Y}](P) \cup (wp_{s@l}^{\uparrow} \zeta[(X, \cdot) \mapsto \mathcal{X}, (X, \uparrow) \mapsto \mathcal{Y}](P) \\ &\cap (\mathcal{Y}(l \triangleright \ell)(P) \cap \mathcal{Y}(l \triangleright r)(P))) \end{cases}$$

Checking invalid input

$$WP_{topDown}(add_{com} @ \epsilon \zeta(\mathbb{U}) = \{e \mid \exists l. lookup \ l \ e = \begin{array}{c} + \\ \wedge \\ m \quad n \end{array}\}$$



$$\in \{e \mid \exists l. lookup \ l \ e = \begin{array}{c} + \\ \wedge \\ m \quad n \end{array}\} \text{ Good!}$$



$$\notin \{e \mid \exists l. lookup \ l \ e = \begin{array}{c} + \\ \wedge \\ m \quad n \end{array}\} \text{ Bad!}$$

Soundness Theorems

$$\frac{\begin{array}{l} \forall X \mid P. \zeta(X, \cdot)(l)(P) = \{e \mid \text{update}(l, e, (\xi(X)(\text{lookup}(l, e)))) \subseteq P\} \\ \wedge \zeta(X, \uparrow)(l)(P) = \{e \mid \text{update}(l, e, (\xi(X)(\text{lookup}(l, e)))) \subseteq P \cup \{\text{err}\}\} \end{array}}{wp_{s \circ l} \zeta(P) = \{e \mid \text{update}(l, e, \llbracket s \rrbracket \xi(\text{lookup}(l, e))) \subseteq P\}}$$

(Weakest Precondition for Total Correctness)

$$\frac{\begin{array}{l} \forall X \mid P. \zeta(X, \cdot)(l)(P) = \{e \mid \text{update}(l, e, (\xi(X)(\text{lookup}(l, e)))) \subseteq P\} \\ \wedge \zeta(X, \uparrow)(l)(P) = \{e \mid \text{update}(l, e, (\xi(X)(\text{lookup}(l, e)))) \subseteq P \cup \{\text{err}\}\} \end{array}}{wp_{s \circ l}^{\uparrow} \zeta(P) = \{e \mid \text{update}(l, e, \llbracket s \rrbracket \xi(\text{lookup}(l, e))) \subseteq P \cup \{\text{err}\}\}}$$

(Weakest May Error Precondition)

Conclusion and Furure Work

Conclusion

- We present the formalised denotational semantics of System S and demonstrate the correspondence (potentially equivalence) between the denotational semantics and big-step operational semantics.
- We present the formalised weakest precondition calculus for System S and demonstrated the usage of the weakest precondition calculus for reasoning about the execution of strategies.


Future Work

- Rewriting expressions represented in other forms such as graphs?
- Using weakest precondition calculus for automatic reasoning about the execution of strategies?

We were on the track ahead as the nightmare plastic column of foetid black iridescence oozed tightly onward through its fifteen-foot sinus; gathering unholy speed and driving before it a spiral, re-thickening cloud of the pallid abyss-vapour. It was a terrible, indescribable thing vaster than any subway train—a shapeless congeries of protoplasmic bubbles, faintly self-luminous, and with myriads of temporary eyes forming and unforming as pustules of greenish light all over the tunnel-filling front that bore down upon us, crushing the frantic penguins and slithering over the glistening floor that it and its kind had swept so evilly free of all litter. Still came that eldritch, mocking cry — “Tekeli-li! Tekeli-li!” And at last we remembered that the daemoniac shoggoths — given life, thought, and plastic organ patterns solely by the Old Ones, and having no language save that which the dot-groups expressed — had likewise no voice save the imitated accents of their bygone masters. —H. P. Lovecraft “From the Mountains of Madness”[3]

Thank you

Xueying Qin [xueying.qin@ed.ac.uk]

 Bastian Hagedorn, Johannes Lenfers, Thomas K  hler, Xueying Qin, Sergei Gorlatch, and Michel Steuwer.

Achieving high-performance the functional way: A functional pearl on expressing high-performance optimizations as rewrite strategies.

Proc. ACM Program. Lang., 4(ICFP), aug 2020.

 Markus Kaiser and Ralf L  mmel.

An isabelle/hol-based model of stratego-like traversal strategies.

In *Proceedings of the 11th ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*, PPDP '09, page 93–104, New York, NY, USA, 2009. Association for Computing Machinery.

 Howard P. Lovecraft.

At the mountains of madness.

1931.

 Eelco Visser.

Stratego: A language for program transformation based on rewriting strategies system description of stratego 0.5.

In Aart Middeldorp, editor, *Rewriting Techniques and Applications*, pages 357–361, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

 Eelco Visser and Zine el Abidine Benaissa.

A core language for rewriting.

Electronic Notes in Theoretical Computer Science, 15:422–441, 1998.

International Workshop on Rewriting Logic and its Applications.