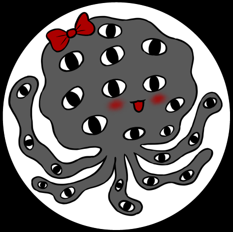


Shoggoth:

A Formal Foundation for Strategic Rewriting

Xueying Qin (秦雪莹)[1]

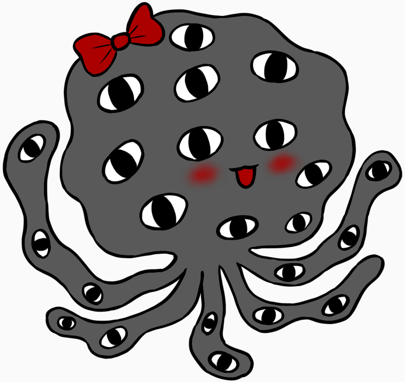
Liam O'Connor[1], Rob van Glabbeek[1 3], Peter Höfner[2], Ohad Kammar[1], Michel Steuwer[1 4]

[1] The University of Edinburgh [2] Australian National University

[3] UNSW [4] Technische Universität Berlin

January 17, 2024

Shoggoth and Strategic Rewriting



Shoggoth

A blob with a lot of eyes. It is a **shape-shifter**, making the sound 'Tekeli-li! Tekeli-li!' which can no longer be understood by anyone. [Lovecraft, 1931]

Strategic rewriting

A language performs **syntactic transformation**, which is lack of formal understanding.

Introduction

System S [Visser and el Abidine Benaissa, 1998], the core calculus of strategic rewriting languages like ELEVATE [Hagedorn et al., 2020], Stratego [Visser, 2001] and Strafunski [Kaiser and Lämmel, 2009] has atomic strategies and composed strategies.

Atomic strategy

An atomic strategy is a *rewrite rule*:

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

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

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

Composed strategy

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

$repeat(mapFusion)$

Strategy combinator

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

$s_1 \ ; \ s_2$ sequential composition, apply s_1 then s_2

$s_1 <+ s_2$ left choice, if fail to apply s_1 then s_2

$repeat(s)$ keep applying s until inapplicable

Importance of Strategic Rewriting Languages

- Strategic rewriting languages provide programmers with **combinators** and **generic traversals** that allow them to:
  - control the application of rewrite rules
  - reuse rewrite rules
- Many application areas: program optimisation (ELEVATE [Hagedorn et al., 2020]), writing interpreter/compiler for DSLs (Spoofax/Stratego [Visser, 2001]) etc.

Strategies can go wrong

- **Result in error** - an atomic strategy is not defined for certain expressions or strategies are not well composed, for example: $add_{com} \ ; \ mult_{com}$
- **Do not terminate** - for example: $repeat(\mathrm{SKIP})$
- **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 the execution of these strategies.

Existing Formal Works

- Big-step operational semantics of System S without modelling divergence [Visser and el Abidine Benaissa, 1998].
- Weakest preconditional calculus for System S using computational tree logic (CTL) [Kieburtz, 2001]. It has following issues:
  - not expressive enough to reason about nondeterminism in traversals
  - problematic fixed-point operator construction
  - soundness of the calculus is not proven

Our Contributions

- Providing the formal semantics of System S, including both **denotational** and **operational** models.
  - Featuring **nondeterminism**, **errors**, and **divergence**.
  - Proving these two semantics models are **equivalent**.
- Providing the **weakest precondition calculus** for the strategic rewriting language.
  - Proving its soundness w.r.t. the denotational semantics.
- Demonstrating how to use the weakest precondition calculus to **prove properties** of strategic rewriting.

Syntax of System S

Introduction to System S and Expressions to be Rewritten

System S

System S [Visser and el Abidine Benaissa, 1998] contains **atomic strategies** (rewrite rules), **strategy combinators** which compose strategies and **traversals** that traverse the expression AST.

Expression

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

$$\mathit{Expressions}(\mathbb{E}) \quad e := \quad \mathit{Leaf} \;\mid\; \begin{array}{c} n \\ \widehat{\;e \quad e\;} \end{array}$$

Syntax of Strategies

Strategy

$$
\begin{aligned}
\mathit{Strategy}(\mathbb{S}) \quad s \ := \quad & \mathrm{SKIP} \ \text{(Always succeeds)} \ \mid \ \mathrm{ABORT} \ \text{(Always results in error)} \\
& \mid \mathit{atomic} \ \text{(Atomic strategy)} \\
& \mid X \ \text{(Variable)} \\
& \mid s_1 \ ; \ s_2 \ \text{(Sequential composition)} \\
& \mid s_1 <+ s_2 \ \text{(Left choice)} \\
& \mid s_1 <+> t_2 \ \text{(Nondeterministic choice)} \\
& \mid \mathit{one}(s) \ \text{(Apply } s \text{ to one child, nondeterministic)} \\
& \mid \mathit{some}(s) \ \text{(Apply } s \text{ to as many children as possible, nondeterministic)} \\
& \mid \mathit{all}(s) \ \text{(Apply } s \text{ to all children, nondeterministic)} \\
& \mid \mu X.s \ \text{(Fixed-point operator)}
\end{aligned}
$$

Semantics of System S

Semantics by Examples - Skip, Abort and Atomic

Examples

$add_{com} : a + b \rightsquigarrow b + a \quad 1+3 \overset{add_{com}}{\rightsquigarrow} 3+1 \qquad 1+3 \overset{\text{SKIP}}{\rightsquigarrow} 1+3 \quad 1+3 \overset{\text{ABORT}}{\rightsquigarrow} err$

Operational semantics

$$\frac{}{e \xrightarrow{atomic} atomic(e)}\ (\text{Atomic})$$

$$\frac{}{e \xrightarrow{\text{SKIP}} e}\ (\text{Skip}) \qquad \frac{}{e \xrightarrow{\text{ABORT}} err}\ (\text{Abort})$$

Denotational semantics

$$[\![atomic]\!]\xi = \lambda e.\{atomic(e) \mid atomic(e)\ \mathbf{def}\} \cup \{err \mid atomic(e)\ \mathbf{undef}\}$$

$$[\![\text{SKIP}]\!]\xi = \lambda e.\{e\}$$

$$[\![\text{ABORT}]\!]\xi = \lambda e.\{err\}$$

Basic definitions for denotational semantics

$[\![\ \mathbb{S}\ ]\!] : \quad \Gamma_S \to \mathfrak{D}$ | Variable$(\mathbb{V})$ $\quad X\ Y\ Z \ldots$

Domain $\quad \mathfrak{D} = \mathbb{E} \to \mathfrak{D}_p$ | where: $\mathfrak{D}_p = \mathcal{P}_{\neg\emptyset}(\mathbb{E} \cup \{err\} \cup \{div\})$

Semantic Environment(Γ_S) $\quad \xi : \mathbb{V} \to \mathfrak{D}$ | $\xi := \quad \emptyset \mid \xi[X \mapsto d]$

Divergence in Sequential Composition

Example

$$add_{id} : 0 + a \rightsquigarrow a \qquad add_{com} : a + b \rightsquigarrow b + a \qquad 3 + 0 \overset{add_{com}\,;\,add_{id}}{\rightsquigarrow} 3$$

$repeat(\mathrm{SKIP})\ ;\ add_{com}$ diverges

$add_{com}\ ;\ repeat(\mathrm{SKIP})$ diverges

- We need to consider divergence as a possible outcome when providing the semantics of the sequential composition.

Big-Step Operational Semantics - Handling of Divergence

Prior operational semantics does not handle divergence

It takes the form of:

$$e \xrightarrow{s} r$$

where r can be either an expression or an error.

Our extended operational semantics handles divergence

We extend the big-step operational semantics to include divergence as a possible outcome, encoded using coinduction, taking the form of:

$$e \xrightarrow[\infty]{s}$$

Semantics by Examples - Sequential Composition

Example

$$add_{id} : 0 + a \rightsquigarrow a \qquad add_{com} : a + b \rightsquigarrow b + a \qquad 3 + 0 \overset{add_{com}\,;\,add_{id}}{\rightsquigarrow} 3$$

Operational semantics

$$\frac{e \xrightarrow{s_1} e_1 \qquad e_1 \xrightarrow{s_2} e_2}{e \xrightarrow{s_1\,;\,s_2} e_2} \text{(SC)}$$

$$\frac{e \xrightarrow{s_1} err}{e \xrightarrow{s_1\,;\,s_2} err} \text{(SCErr(1))} \qquad \frac{e \xrightarrow{s_1} e_1 \qquad e_1 \xrightarrow{s_2} err}{e \xrightarrow{s_1\,;\,s_2} err} \text{(SCErr(2))}$$

$$\frac{e \xrightarrow[\infty]{s_1}}{e \xrightarrow[\infty]{s_1\,;\,s_2}} \text{(SCDiv(1))} \qquad \frac{e \xrightarrow{s_1} e_1 \qquad e_1 \xrightarrow[\infty]{s_2}}{e \xrightarrow[\infty]{s_1\,;\,s_2}} \text{(SCDiv(2))}$$

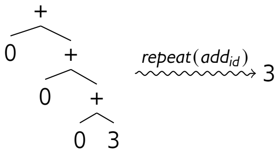
Denotational semantics

$$\begin{aligned} &[\![s_1 \;;\; s_2]\!]\xi \\ &\quad = \lambda e. \bigcup\{[\![s_2]\!]\xi(x) \mid x \in [\![s_1]\!]\xi(e) \cap \mathbb{E}\} \\ &\quad \cup \{x \mid x \in [\![s_1]\!]\xi(e) \cap \{div, err\}\} \end{aligned}$$

The Need of A Fixed-Point Operator

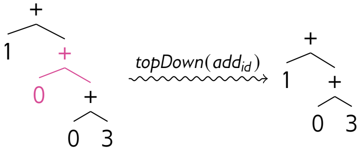
Example - Repeat

$repeat(s) = \mu X.try(s \,;\, X) \quad add_{id} : 0 + a \rightsquigarrow a$



Example - Top Down

$topDown(s) = \mu X.s <+ one(X) \quad add_{id} : 0 + a \rightsquigarrow a$



- We need make sure the fixed point is the least fixed point and thus the denotational semantics are monotonic and continuous functions.

Power Domain, Domain and Ordering

The Plotkin powerdomain

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

The domain

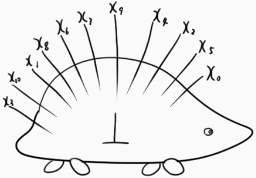
$$\mathfrak{D} = \mathbb{E} \longrightarrow \mathfrak{D}_p$$

Egli-Milner ordering

$$A \leq B \iff (\forall x \in A.\ \exists y \in B.\ x \leq y) \wedge (\forall y \in B.\ \exists x \in A.\ x \leq y)$$

Porcupine ordering

$$A \leq B \iff A = B \vee ((\perp \in A) \wedge A \backslash \{\perp\} \subseteq B)$$



- Defining denotational semantics in such a domain can ensure the semantics to be monotone and continuous.

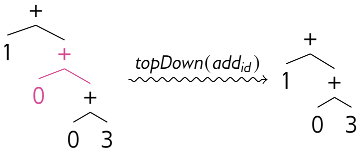
A 2500BC Porcupine

Photo by Michel Steuwer

Semantics by Examples - Fixed Point Operator

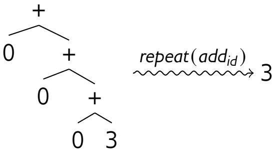
Example - Top Down

$topDown(s) = \mu X.s <+ one(X)$ $\quad add_{id} : 0 + a \rightsquigarrow a$



Example - Repeat

$repeat(s) = \mu X.try(s \ ; \ X)$ $\quad add_{id} : 0 + a \rightsquigarrow a$



Operational semantics

$$\frac{e \xrightarrow{s[X:=\mu X.s]} e_1}{e \xrightarrow{\mu X.s} e_1} \text{(FP)}$$

$$\frac{e \xrightarrow{s[X:=\mu X.s]} err}{e \xrightarrow{\mu X.s} err} \text{(FPErr)} \qquad \frac{e \xrightarrow[\infty]{s[X:=\mu X.s]}}{e \xrightarrow[\infty]{\mu X.s}} \text{(FPDiv)}$$

Denotational semantics

$$[\![X]\!]\xi = \xi X$$

$$[\![\mu X.s]\!]\xi = \mu \mathcal{X}.[\![s]\!](\xi[X \mapsto \mathcal{X}])$$

We Show the Denotational and Operational Semantics are Equivalent

Closed strategy

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

Computational soundness

$$\frac{e \xrightarrow{s_\clubsuit} e'}{e' \in [\![s_\clubsuit]\!]\xi e} \qquad \frac{e \xrightarrow[\infty]{s_\clubsuit}}{div \in [\![s_\clubsuit]\!]\xi e}$$

Computational adequacy

$$\frac{e' \in [\![s_\clubsuit]\!]\xi e \wedge e' \neq div}{e \xrightarrow{s_\clubsuit} e'} \qquad \frac{div \in [\![s_\clubsuit]\!]\xi e}{e \xrightarrow[\infty]{s_\clubsuit}}$$

Semantics equivalence

$$[\![s_\clubsuit]\!]\xi e = \{r \mid e \xrightarrow{s_\clubsuit} r\} \cup \{div \mid e \xrightarrow[\infty]{s_\clubsuit}\}$$

Mechanised proofs are available at: https://github.com/XYUnknown/Shoggoth

Location Based Weakest Precondition Calculus

Strategies Can Go Wrong

Errors

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

$6 + 3 \overset{add_{id}}{\rightsquigarrow} err$

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

$0 + 3 \overset{add_{com} \,;\, mult_{com}}{\rightsquigarrow} err$

$0 + 3 \overset{(add_{id} <\!+ add_{com}) \,;\, add_{com}}{\rightsquigarrow} err$

Divergence

$e \overset{repeat(\text{SKIP})}{\rightsquigarrow} div$

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

Undesired result

We want $3 + 0$

$0 + 3 \overset{add_{id} <\!+ add_{com}}{\rightsquigarrow} 3$

Observations

- **Bad strategies can never lead to any successful execution.**
- **Good strategies may be unsuccessfully executed on some inputs.**

Introduction of Weakest Precondition Calculus

Motivations

- To characterise **good** and **bad strategies**.
- To characterise **successful** and **unsuccessful executions**.
- To **detect** bad strategies and unsuccessful executions, by:
  - specifying a property to be satisfied after the execution of a strategy and calculating the set expressions that can lead to a result satisfying such a property.

Background: 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 of traversals

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 formulae.

Our solution

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

Definition

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

A *weakest must succeed precondition* is the set of those expressions that, by applying strategy s at location l under the logic environment ζ, will be successfully transformed into expressions satisfying P.

Weakest May Error Precondition

Definition

$$wp^{\uparrow}_{\zeta \Vdash s@l}(P)$$

A *weakest may error precondition* is the set of those expressions that, by applying strategy s at location l under the logic environment ζ, will be successfully transformed into expressions satisfying P, or result in error.

Is A Strategy Well-Composed?

Example

$add_{com} : a + b \rightsquigarrow b + a \quad mult_{com} : a * b \rightsquigarrow b * a \qquad add_{com} \,;\, mult_{com}$ (Bad?)

Wp for atomic strategy

$$wp_{\zeta \Vdash atomic @ l}(P) = \{e \mid update(l, e, [\![atomic]\!]\varnothing(lookup(l, e))) \subseteq P\}$$

$$wp^{\uparrow}_{\zeta \Vdash atomic @ l}(P) = \{e \mid update(l, e, [\![atomic]\!]\varnothing(lookup(l, e))) \subseteq P \cup \{err\}\}$$

Wp of sequential composition

$$wp_{\zeta \Vdash s \,;\, t @ l}(P) = wp_{\zeta \Vdash s @ l}(wp_{\zeta \Vdash t @ l}(P)) \qquad wp^{\uparrow}_{\zeta \Vdash s \,;\, t @ l}(P) = wp^{\uparrow}_{\zeta \Vdash s @ l}(wp^{\uparrow}_{\zeta \Vdash t @ l}(P))$$

Checking invalid composition

$$wp_{\zeta \Vdash add_{com} @ \epsilon}(\mathbb{E}) = \{e \mid e = \begin{array}{c} + \\ m \quad n \end{array}\} \qquad wp_{\zeta \Vdash mult_{com} @ \epsilon}(\mathbb{E}) = \{e \mid e = \begin{array}{c} * \\ m \quad n \end{array}\}$$

$wp_{\zeta \Vdash add_{com} \,;\, mult_{com} @ \epsilon}(\mathbb{E}) = \emptyset$ (Bad!)

Does A Strategy Diverge? (0)

Does the given strategy diverge, i.e., does not lead to any successful execution?

Example

$repeat(\mathrm{SKIP})$ Bad?

Wp of fixed point operator

$$wp_{\zeta \Vdash \mu X.s@l}(P) = [\mathrm{LFP}\, \mathscr{X} : \Delta] \,/\, P \qquad wp^{\uparrow}_{\zeta \Vdash \mu X.s@l}(P) = [\mathrm{LFP}\, \mathscr{Y} : \Delta] \,/\, P$$

Where:

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

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

$wp^{\uparrow}_{\zeta \Vdash X@l}(P) = \zeta(X,\uparrow) \,/\, P$ (where $\zeta(X,\uparrow)$ **def.**)

Does A Strategy Diverge? (1)

Example

$$repeat(\mathrm{SKIP}) \quad \textcolor{red}{\text{Bad?}}$$

Wp for repeat

$$wp_{\zeta \Vdash repeat(s) @ l}(P) = wp^{\uparrow}_{\zeta \Vdash repeat(s) @ l}(P) = [\mathrm{LFP}\, \mathscr{X} : \Delta] \ l \ P$$

where Δ is the fixed-point equation

$$\begin{aligned} \mathscr{X}(l)(P) = {} & wp_{\zeta[(X,\cdot)\mapsto \mathscr{X}\ ,\ (X,\uparrow)\mapsto \mathscr{X}] \Vdash s @ l}(\mathscr{X} \ l \ P) \\ & \cup \left(P \cap wp^{\uparrow}_{\zeta[(X,\cdot)\mapsto \mathscr{X}\ ,\ (X,\uparrow)\mapsto \mathscr{X}] \Vdash s @ l}(\mathscr{X} \ l \ P)\right) \end{aligned}$$

Checking divergence

$$wp_{repeat(\mathrm{SKIP}) @ \epsilon}\, \zeta(\mathbb{E}) = \emptyset \quad \textcolor{red}{\text{Bad!}}$$

Good and Bad Strategies, Successful and Unsuccessful Executions

Good strategies

A strategy s is good iff for a given postcondition P:

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

Bad strategies

A strategy s is bad iff for a given postcondition P:

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

Successful executions

An execution of a good strategy s, on an input expression e is successful iff for a given postcondition P:

$$e \in wp_{\zeta \Vdash s @ l}(P) \quad \text{(where: } wp_{\zeta \Vdash s @ l}(P) \neq \emptyset)$$

Unsuccessful executions

An execution of a good strategy s on an input expression e is unsuccessful iff for a given postcondition P:

$$e \notin wp_{\zeta \Vdash s @ l}(P) \quad \text{(where: } wp_{\zeta \Vdash s @ l}(P) \neq \emptyset)$$

Soundness theorems

$$\frac{\begin{array}{c}\forall X\, l\, P.\ \zeta(X, \cdot)\ l\ P = \{e \mid \xi(X)(\pitchfork_l\ e) \ \square\!\!\Rightarrow_l\ e \subseteq P\} \\ \wedge \zeta(X, \uparrow)\ l\ P = \{e \mid \xi(X)(\pitchfork_l\ e) \ \square\!\!\Rightarrow_l\ e \subseteq P \cup \{err\}\}\end{array}}{wp_{\zeta \Vdash s @ l}(P) = \{e \mid ([\![s]\!]\xi(\pitchfork_l\ e)) \ \square\!\!\Rightarrow_l\ e \subseteq P\}}$$

(Weakest Must Succeed Precondition)

$$\frac{\begin{array}{c}\forall X\, l\, P.\ \zeta(X, \cdot)\ l\ P = \{e \mid \xi(X)(\pitchfork_l\ e) \ \square\!\!\Rightarrow_l\ e \subseteq P\} \\ \wedge \zeta(X, \uparrow)\ l\ P = \{e \mid \xi(X)(\pitchfork_l\ e) \ \square\!\!\Rightarrow_l\ e \subseteq P \cup \{err\}\}\end{array}}{wp^{\uparrow}_{\zeta \Vdash s @ l}(P) = \{e \mid ([\![s]\!]\xi(\pitchfork_l\ e)) \ \square\!\!\Rightarrow_l\ e \subseteq P \cup \{err\}\}}$$

(Weakest May Error Precondition)

Mechanised proofs are available at: https://github.com/XYUnknown/Shoggoth

Conclusion and Future Work

Our paper features

- Formal semantics of System S and equivalence proofs of the denotational semantics and big-step operational semantics.
- The formalised weakest precondition calculus for System S, soundness proofs and more case studies demonstrating the usage of the weakest precondition calculus for reasoning about the execution of strategies.
- All formalised semantics and calculus as well as proofs are mechanised in Isabelle/HOL. (Artifact: https://doi.org/10.5281/zenodo.10125602)

Future works

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

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 . . . 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"

Thank you (^w^)

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

[https://xyunknown.github.io]

Hagedorn, B., Lenfers, J., Kœhler, T., Qin, X., Gorlatch, S., and Steuwer, M. (2020). **Achieving high-performance the functional way: A functional pearl on expressing high-performance optimizations as rewrite strategies.** *Proc. ACM Program. Lang.*, 4(ICFP).

Kaiser, M. and Lämmel, R. (2009). **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. Association for Computing Machinery.

Kieburtz, R. B. (2001).
A logic for rewriting strategies.
Electronic Notes in Theoretical Computer Science, 58(2):138–154.
STRATEGIES 2001, 4th International Workshop on Strategies in Automated Deduction - Selected Papers (in connection with IJCAR 2001).

Lovecraft, H. P. (1931).
At the mountains of madness.

Visser, E. (2001).
Stratego: A language for program transformation based on rewriting strategies system description of stratego 0.5.
In Middeldorp, A., editor, *Rewriting Techniques and Applications*, pages 357–361, Berlin, Heidelberg. Springer Berlin Heidelberg.

Visser, E. and el Abidine Benaissa, Z. (1998).
A core language for rewriting.
Electronic Notes in Theoretical Computer Science, 15:422–441.
International Workshop on Rewriting Logic and its Applications.