ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

ROSAEC MEMO
2009-05
September 4, 2009

A Control Flow Analysis for 2-staged Programming Languages

Taeksu Kim[1], Chunwoo Lee[1], Kiljoo Lee[1], Soohyun Baik[1], and Kwangkeun Yi[2]
{dolicoli,oniguni,laazycat,shbaik82}@selab.snu.ac.kr[1], kwang@ropas.snu.ac.kr[2]

September 4, 2009

Abstract

As a program written in multi-staged language can generate and execute code fragments in excution time, it is hard to predict what code fragments will be generated and run in execution time. Therefore we need a new method to predict what code fragments would be generated in execution time to analyze a control flow of software.

In this article, we present static analysis which detects code fragments generated in execution time using abstract interpretation. Moreover we prove the correctness of analyzer.

1 Introduction

Multi-staged language is a language which can generate and execute new program codes in execution time. The key aspect of multi-staged languages is to have code templates (program fragments) as first-class objects [1]. Recently multi-staged languages are used widely because they can make software development faster and prevent side effects occurred with single-staged language. Many Web programming languages(e.g. Ruby, Python) or script languages(e.g. PHP, Lisp, JavaScript) adopt multi-staged features in their specification so that they support rapid development.

Even if multi-staged languages are useful for rapid software development, it is hard to estimate what code fragments will be generated and executed. Therefore, the estimation of a control flow of software written in multi-staged language is hard, too. To analyze the control flow of the software, it is needed to analyze how many and what code fragments would be generated in execution time.

In this article, we design a static analysis for detecting code fragments generated in execution time using abstract interpretation[2] and prove the correctness of analyzer. We expect that this analyzer could help software developers to analyze a control flow of multi-staged programming language.

2 Language

Figure 1 shows a syntax of a simple 2-staged language. In the language, a program consists of expressions. Functions and code fragments are the first-class objects in the language. This language use quasi-quotations[3] to generate and execute code fragments in the execution time. Boxing(‘) operator makes a subexpression as a code fragments at one-stage and unboxing(,) operation substitutes a subexpression by a value evaluated at zero-stage. Run operation evaluates a code fragment. Boxing and unboxing operators are labeled by unique label and alphabet respectively. Detailed semantics are shown in the next chapter.

To decrease the complexity of the analysis we make some assumptions on the language as follows.

$$
\begin{array}{rcl}
e & \rightarrow & c \\
 & | & x \\
 & | & \textbf{let } x\ e\ e \\
 & | & \textbf{if } e\ e\ e \\
 & | & f\lambda x.e \\
 & | & e\ \ e \\
 & | & {}^{\backprime}{}_{l}e \\
 & | & {}_{,\alpha}e \\
 & | & \textbf{run } e \\
l & \in & \mathit{Label} \\
\alpha & \in & \mathit{Alphabet}
\end{array}
$$

Figure 1: Language specification of simple two-staged language

- The language only support 2-stage processing.
- Expression **run** e cannot generate code fragments.
- All *labels* and *alphabets* are unique values in program code.

From now on, we will only concern this two-staged language. Three or more staged languages can be treated similarly.

3 Operational Semantics

In this chapter we define the semantics of the target language. A semantic function

$$\mathcal{E} = \mathit{Stage} \rightarrow \mathit{Exp} \rightarrow \mathit{Env} \rightarrow \mathit{Val} \times \mathit{Env}$$

is the least fixpoint of a function

$$\mathcal{F} \in (\mathit{Stage} \rightarrow \mathit{Exp} \rightarrow \mathit{Env} \rightarrow \mathit{Val} \times \mathit{Env}) \rightarrow (\mathit{Stage} \rightarrow \mathit{Exp} \rightarrow \mathit{Env} \rightarrow \mathit{Val} \times \mathit{Env})$$

.

$$
\begin{array}{rcl}
e & \in & \mathit{Exp} \\
x & \in & \mathit{Var} \\
l & \in & \mathit{Label} \\
\alpha & \in & \mathit{Alphabet} \\
v & \in & \mathit{Val} = \mathit{Code} + \mathit{Closure} + \mathit{Constant} \\
\sigma & \in & \mathit{Env} = \mathit{Var} \xrightarrow{fin} \mathit{Val} \\
\mathit{Stage} & = & \{0, 1\}
\end{array}
$$

1. $$\mathcal{F}\ \mathcal{E}\ 0\ [\![c]\!]\ \sigma \quad := \quad \langle c, \sigma \rangle$$

2. $$\mathcal{F}\ \mathcal{E}\ 0\ [\![x]\!]\ \sigma \quad := \quad \langle \sigma(x), \sigma \rangle$$

3.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 0\ [\![\mathbf{let}\ x\ e_1\ e_2]\!]\ \sigma\ :=\ & \\ & \mathbf{let}\ \langle v, \sigma'\rangle = \mathcal{E}\ 0\ [\![e_1]\!]\ \sigma\ \mathbf{in} \\ & \mathcal{E}\ 0\ [\![e_2]\!]\ \{\sigma'[x \mapsto v]\}\end{aligned}$$

4.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 0\ [\![\mathbf{if}\ e_1\ e_2\ e_3]\!]\ \sigma\ :=\ & \\ & \mathbf{let}\ \langle v, \sigma'\rangle = \mathcal{E}\ 0\ [\![e_1]\!]\ \sigma\ \mathbf{in} \\ & \mathbf{if}\ v \neq 0\ \mathbf{then}\ \mathcal{E}\ 0\ [\![e_2]\!]\ \sigma' \\ & \mathbf{else}\ \mathcal{E}\ 0\ [\![e_3]\!]\ \sigma'\end{aligned}$$

5.

$$\mathcal{F}\ \mathcal{E}\ 0\ [\![f\lambda x.e]\!]\ \sigma\ :=\ \langle\langle f\lambda x.e, \sigma\rangle, \sigma\rangle$$

6.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 0\ [\![e_1\ \ e_2]\!]\ \sigma\ :=\ & \\ & \mathbf{let}\ \langle\langle f\lambda x.e, \sigma_0\rangle, \sigma_1\rangle = \mathcal{E}\ 0\ [\![e_1]\!]\ \sigma\ \mathbf{in} \\ & \mathbf{let}\ \langle v, \sigma_2\rangle = \mathcal{E}\ 0\ [\![e_2]\!]\ \sigma_1\ \mathbf{in} \\ & \mathcal{E}\ 0\ [\![e]\!]\ \sigma_2[x \mapsto v][f \mapsto \langle f\lambda x.b, \sigma_0\rangle]\end{aligned}$$

7.

$$\mathcal{F}\ \mathcal{E}\ 0\ [\![^{\backprime\iota}e]\!]\ \sigma\ :=\ \mathcal{E}\ 1\ [\![e]\!]\ \sigma$$

8.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 0\ [\![\mathbf{run}\ e]\!]\ \sigma\ :=\ & \\ & \mathbf{let}\ \langle [e'], \sigma'\rangle = \mathcal{E}\ 0\ [\![e]\!]\ \sigma\ \mathbf{in} \\ & \mathcal{E}\ 0\ [\![e']\!]\ \sigma'\end{aligned}$$

9.

$$\mathcal{F}\ \mathcal{E}\ 1\ [\![c]\!]\ \sigma\ :=\ \langle [c], \sigma\rangle$$

10.

$$\mathcal{F}\ \mathcal{E}\ 1\ [\![x]\!]\ \sigma\ :=\ \langle [x], \sigma\rangle$$

11.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 1\ [\![\mathbf{let}\ x\ e_1\ e_2]\!]\ \sigma\ :=\ & \\ & \mathbf{let}\ \langle [e_1'], \sigma_1\rangle = \mathcal{E}\ 1\ [\![e_1]\!]\ \sigma\ \mathbf{in} \\ & \mathbf{let}\ \langle [e_2'], \sigma_2\rangle = \mathcal{E}\ 1\ [\![e_2]\!]\ \sigma_1\ \mathbf{in} \\ & \langle [\mathbf{let}\ x\ e_1'\ e_2'], \sigma_2\rangle\end{aligned}$$

12.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 1\ [\![\textbf{if}\ e_1\ e_2\ e_3]\!]\ \sigma\ \ :=\ & \\ & \textbf{let}\ \langle [e_1'], \sigma_1\rangle = \mathcal{E}\ 1\ [\![e_1]\!]\ \sigma\ \textbf{in} \\ & \textbf{let}\ \langle [e_2'], \sigma_2\rangle = \mathcal{E}\ 1\ [\![e_2]\!]\ \sigma_1\ \textbf{in} \\ & \textbf{let}\ \langle [e_3'], \sigma_3\rangle = \mathcal{E}\ 1\ [\![e_3]\!]\ \sigma_1\ \textbf{in} \\ & \langle [\textbf{if}\ e_1'\ e_2'\ e_3'], \sigma_3\rangle\end{aligned}$$

13.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 1\ [\![f\lambda x.e]\!]\ \sigma\ \ :=\ & \\ & \textbf{let}\ \langle [e_1'], \sigma_1\rangle = \mathcal{E}\ 1\ [\![e_1]\!]\ \sigma\ \textbf{in} \\ & \langle [f\lambda x.e_1'], \sigma_1\rangle\end{aligned}$$

14.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 1\ [\![e_1\ \ e_2]\!]\ \sigma\ \ :=\ & \\ & \textbf{let}\ \langle [e_1'], \sigma_1\rangle = \mathcal{E}\ 1\ [\![e_1]\!]\ \sigma\ \textbf{in} \\ & \textbf{let}\ \langle [e_2'], \sigma_2\rangle = \mathcal{E}\ 1\ [\![e_2]\!]\ \sigma_1\ \textbf{in} \\ & \langle [e_1'\ \ e_2'], \sigma_2\rangle\end{aligned}$$

15.

$$\mathcal{F}\ \mathcal{E}\ 1\ [\![,_\alpha e]\!]\ \sigma\ \ :=\ \ \mathcal{E}\ 0\ [\![e]\!]\ \sigma\ \textbf{in}$$

16.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 1\ [\![\textbf{run}\ e]\!]\ \sigma\ \ :=\ & \\ & \textbf{let}\ \langle [e_1'], \sigma_1\rangle = \mathcal{E}\ 1\ [\![e_1]\!]\ \sigma\ \textbf{in} \\ & \langle [\textbf{run}\ e_1'], \sigma_1\rangle\end{aligned}$$

4 Collecting Sementics

In this section, we define the collecting semantics of the target language. A semantic function

$$\mathcal{E} = Stage \to Exp \to 2^{Env} \to 2^{Val \times Env}$$

is the least fixpoint of a function

$$\mathcal{F} \in (Stage \to Exp \to 2^{Env} \to 2^{Val \times Env}) \to (Stage \to Exp \to 2^{Env} \to 2^{Val \times Env})$$

.

$$\begin{aligned} e &\in Exp \\ x &\in Var \\ l &\in Label \\ \alpha &\in Alphabet \\ v &\in Val = Code + Closure + Constant \\ \sigma &\in Env = Var \xrightarrow{fin} Val \\ Stage &= \{0, 1\} \\ \Sigma &\in 2^{Env} \\ T &\in 2^{Val \times Env} \end{aligned}$$

1.

$$\mathcal{F}\ \mathcal{E}\ 0\ [\![c]\!]\ \Sigma \quad := \quad \{\langle c, \sigma\rangle : \sigma \in \Sigma\}$$

2.

$$\mathcal{F}\ \mathcal{E}\ 0\ [\![x]\!]\ \Sigma \quad := \quad \{\langle \sigma(x), \sigma\rangle : \sigma \in \Sigma\}$$

3.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 0\ [\![\textbf{let}\ x\ e_1\ e_2]\!]\ \Sigma \quad := \quad & \\ & \textbf{let}\ T = \mathcal{E}\ 0\ [\![e_1]\!]\ \Sigma\ \textbf{in} \\ & \bigcup_{\langle v,\sigma\rangle \in T} \mathcal{E}\ 0\ [\![e_2]\!]\ \{\sigma[x \mapsto v]\}\end{aligned}$$

4.

$$\mathcal{F}\ \mathcal{E}\ 0\ [\![\textbf{if}\ e_1\ e_2\ e_3]\!]\ \Sigma \quad := \quad \mathcal{E}\ 0\ [\![e_2]\!]\ (\neg\mathcal{B}\ [\![e_1]\!]\ \Sigma) \cup \mathcal{E}\ 0\ [\![e_3]\!]\ (\mathcal{B}\ [\![e_1]\!]\ \Sigma)$$

where for $\Sigma \in 2^{Env}$ and $e \in Exp$,

$$\begin{aligned}\mathcal{B}\ [\![e]\!]\ \Sigma \quad &:= \quad \{\sigma \in \Sigma : e \text{ becomes 0 under } \sigma\} \\ \neg\mathcal{B}\ [\![e]\!]\ \Sigma \quad &:= \quad \Sigma - \mathcal{B}\ [\![e]\!]\ \Sigma\end{aligned}$$

5.

$$\mathcal{F}\ \mathcal{E}\ 0\ [\![f\lambda x.e]\!]\ \Sigma \quad := \quad \{\langle\langle f\lambda x.e, \sigma\rangle, \sigma\rangle : \sigma \in \Sigma\}$$

6.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 0\ [\![e_1\ \ e_2]\!]\ \Sigma \quad := \quad & \\ & \textbf{let}\ T_1 = \mathcal{E}\ 0\ [\![e_1]\!]\ \Sigma\ \textbf{in} \\ & \textbf{let}\ T_2 = \bigcup_{\langle\langle f\lambda x.b,\sigma\rangle,\sigma_1\rangle \in T_1} \mathcal{E}\ 0\ [\![e_2]\!]\ \{\sigma_1\}\ \textbf{in} \\ & \bigcup_{\langle\langle f\lambda x.b,\sigma\rangle,\sigma_1\rangle \in T_1, \langle v,\sigma_2\rangle \in T_2} \mathcal{E}\ 0\ [\![b]\!]\ \{\sigma_2[x \mapsto v][f \mapsto \langle f\lambda x.b, \sigma\rangle]\}\end{aligned}$$

7.

$$\mathcal{F}\ \mathcal{E}\ 0\ [\![^{\iota}e]\!]\ \Sigma \quad := \quad \mathcal{E}\ 1\ [\![e]\!]\ \Sigma$$

8.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 0\ [\![\textbf{run}\ e]\!]\ \Sigma \quad := \quad & \\ & \textbf{let}\ T = \mathcal{E}\ 0\ [\![e]\!]\ \Sigma\ \textbf{in} \\ & \bigcup_{\langle [e'],\sigma\rangle \in T} \mathcal{E}\ 0\ [\![e']\!]\ \{\sigma\}\end{aligned}$$

9.

$$\mathcal{F}\ \mathcal{E}\ 1\ [\![c]\!]\ \Sigma \quad := \quad \{\langle [c], \sigma\rangle : \sigma \in \Sigma\}$$

10.

$$\mathcal{F}\ \mathcal{E}\ 1\ [\![x]\!]\ \Sigma \quad := \quad \{\langle [x], \sigma\rangle : \sigma \in \Sigma\}$$

11.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 1\ [\![\textbf{let}\ x\ e_1\ e_2]\!]\ \Sigma \quad := \quad & \\ & \textbf{let}\ T_1 = \mathcal{E}\ 1\ [\![e_1]\!]\ \Sigma\ \textbf{in} \\ & \textbf{let}\ T_2 = \bigcup_{\langle [e_1'], \sigma_1\rangle \in T_1} \mathcal{E}\ 1\ [\![e_2]\!]\ \{\sigma_1\}\ \textbf{in} \\ & \{\langle [\textbf{let}\ x\ e_1'\ e_2'], \sigma_2\rangle : \langle [e_1'], \sigma_1\rangle \in T_1, \langle [e_2'], \sigma_2\rangle \in T_2\}\end{aligned}$$

12.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 1\ [\![\textbf{if}\ e_1\ e_2\ e_3]\!]\ \Sigma \quad := \quad & \\ & \textbf{let}\ T_1 = \mathcal{E}\ 1\ [\![e_1]\!]\ \Sigma\ \textbf{in} \\ & \textbf{let}\ T_2 = \bigcup_{\langle [e_1'], \sigma_1\rangle \in T_1} \mathcal{E}\ 1\ [\![e_2]\!]\ \{\sigma_1\}\ \textbf{in} \\ & \textbf{let}\ T_3 = \bigcup_{\langle [e_2'], \sigma_2\rangle \in T_2} \mathcal{E}\ 1\ [\![e_3]\!]\ \{\sigma_2\}\ \textbf{in} \\ & \{\langle [\textbf{if}\ e_1'\ e_2'\ e_3'], \sigma_3\rangle : \langle [e_1'], \sigma_1\rangle \in T_1, \langle [e_2'], \sigma_2\rangle \in T_2, \langle [e_3'], \sigma_3\rangle \in T_3\}\end{aligned}$$

13.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 1\ [\![f\lambda x.e]\!]\ \Sigma \quad := \quad & \\ & \textbf{let}\ T = \mathcal{E}\ 1\ [\![e]\!]\ \Sigma\ \textbf{in} \\ & \{\langle [f\lambda x.e'], \sigma\rangle : \langle [e'], \sigma\rangle \in T\}\end{aligned}$$

14.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 1\ [\![e_1\ \ e_2]\!]\ \Sigma \quad := \quad & \\ & \textbf{let}\ T_1 = \mathcal{E}\ 1\ [\![e_1]\!]\ \Sigma\ \textbf{in} \\ & \textbf{let}\ T_2 = \bigcup_{\langle [e_1'], \sigma_1\rangle \in T_1} \mathcal{E}\ 1\ [\![e_2]\!]\ \{\sigma_1\}\ \textbf{in} \\ & \{\langle [e_1'\ \ e_2'], \sigma_2\rangle : \langle [e_1'], \sigma_1\rangle \in T_1, \langle [e_2'], \sigma_2\rangle \in T_2\}\end{aligned}$$

15.

$$\mathcal{F}\ \mathcal{E}\ 1\ [\![_{,\alpha} e]\!]\ \Sigma \quad := \quad \mathcal{E}\ 0\ [\![e]\!]\ \Sigma$$

16.

$$\begin{aligned}\mathcal{F}\ \mathcal{E}\ 1\ [\![\textbf{run}\ e]\!]\ \Sigma \quad := \quad & \\ & \textbf{let}\ T = \mathcal{E}\ 1\ [\![e]\!]\ \Sigma\ \textbf{in} \\ & \{\langle [\textbf{run}\ e'], \sigma\rangle : \langle [e'], \sigma\rangle \in T\}\end{aligned}$$

Lemma 1. *2^{Val}, 2^{Env} and $2^{Val\times Env}$ are CPOs(Complete Partial Orders).*

Proof. Let $\subseteq$ be an order of 2^{Val}.Then

- Trivially, 2^{Val} is a partially ordered set for $\subseteq$.
- 2^{Val} has a least element $\emptyset$.
- 2^{Val} has a least upper bound (universe) for all chains.

Therefore, 2^{Val} is CPO.
Similarly, 2^{Env} and $2^{Val \times Env}$ are CPOs, too.

□

5 Abstractions

For an abstract interpretation, we define abstract domains, operations on each domain and abstractions in this chapter.

5.1 Grammar

A value returned by an expression can be a code, a closure or a constant. Codes can contain subexpressions with unboxing operations and the value of the codes is decided by substitution of unboxing operations with some codes. That means a prediction is required that what unboxing operation the code has and what code the unboxing expression evaluates. To represent code fragments, let us define a new data type, *Grammar* as a production rule of formal grammar[4] to represent a code.

Rules are as follows.

- (Terminal)
$$Z \rightarrow \triangle \mid label \mid Z[Terminal^*]$$
  - Label l: a code fragment whose label is l.
  - $\triangle$: a code fragment whose label is unknown.
  - $Z[Nonterminal^*]$: a code fragment which has unboxing operations represented with non-terminal alphabets. Its label is detemined by Z.
- (Nonterminal)
$$N \rightarrow \mathfrak{s} \mid alphabet$$
  - $\mathfrak{s}$: a start symbol
  - Alphabet a: an unboxing operation whose alphabet is a.
- (Start Symbol)
$$\Sigma \rightarrow \mathfrak{s}$$

Terminal and non-terminal represent label of a boxing expression and alphabet of an unboxing expression respectively, with the consequence that one code fragment can be represented as a formal grammar and one unboxing subexpression is decided by production rule of formal grammar. Let us simply call this production rule as a *grammar*.

$$Grammar = Box + Slot$$

A slot is a grammar whose start terminal is $\triangle$ and a box is a grammar whose start terminal is not $\triangle$.

5.1.1 Functions

$$\begin{aligned} \psi &\in Grammar \rightarrow 2^{Exp} \\ \xi &\in Code \rightarrow Grammar \end{aligned}$$

- $\psi(g)$: Set of code fragments which can be generated from grammar g.
- $\xi(c)$: A grammar that represent code c.

5.1.2 Operations

- (Slot Append) For $g \in Box$ and $\alpha \in Alphabet$,

$$\begin{aligned} g' &= \alpha \oplus g \in Slot \\ \psi(g') &= \{c \in Code : c \text{ has exactly one subexpress labeled by } \alpha \\ &\qquad \text{and the subexpression can be replaced with } \psi(g)\} \end{aligned}$$

- (Box Append) For $g \in Slot$ and $l \in Label$,

$$\begin{aligned} g' &= l \otimes g \in Box \\ \psi(g') &= \{c \in Code : c \in \psi(g) \text{ and labeled by } l\} \end{aligned}$$

- (Slot Join) For $g_1, g_2 \in Slot$,

$$\begin{aligned} g' &= g_1 \sqcup_s g_2 \in Slot \\ \psi(g') &= \{c \in Code : c \text{ has two subexpress.} \\ &\qquad \text{The subexpressions can be replaced with } \psi(g_1) \text{ and } \psi(g_2) \text{ respectively}\} \end{aligned}$$

- (Box Join) For $g_1, g_2 \in Box$,

$$\begin{aligned} g' &= g_1 \sqcup_b g_2 \in Box \\ \psi(g') &= \psi(g_1) \cup \psi(g_2) \end{aligned}$$

- (Grammar Join) For $g_1, g_2 \in Grammar$,

$$g_1 \sqcup g_2 = \begin{cases} g_1 \sqcup_b g_2 & \text{if } g_1, g_2 \in \text{Box} \\ g_1 \sqcup_s g_2 & \text{if } g_1, g_2 \in \text{Slot} \end{cases}$$

5.2 Abstraction: Value Set

Let us define an abstract value as follows.

$$\hat{Val} = Grammar + 2^{Exp} + \{\cdot\} + \hat{\bot}_v + \hat{\top}_v$$

5.2.1 Operations

- (Abstract Value Join) $\forall \hat{v_1}, \hat{v_2} \in \hat{Val}$,

$$\hat{v_1} \sqcup \hat{v_2} = \begin{cases} \hat{v_1} & \text{if } \hat{v_2} = \hat{\bot}_v \\ \hat{v_2} & \text{if } \hat{v_1} = \hat{\bot}_v \\ \{\cdot\} & \text{if } \hat{v_1} = \{\cdot\}, \hat{v_2} = \{\cdot\} \\ \hat{v_1} \cup \hat{v_2} & \text{if } \hat{v_1} \in 2^{Exp}, \hat{v_2} \in 2^{Exp} \\ \hat{v_1} \sqcup \hat{v_2} & \text{if } \hat{v_1} \in Grammar, \hat{v_2} \in Grammar \\ \hat{\top}_v & \text{otherwise} \end{cases}$$

Lemma 2. *$\hat{Val}$ is a CPO.*

Proof. Let us define an order $\sqsubseteq_v$ of $\hat{Val}$ as follows.

$$\begin{array}{rcll} \hat{\bot}_v & \sqsubseteq_v & \hat{v} & \text{for all } \hat{v} \in \hat{Val} \\ \hat{v} & \sqsubseteq_v & \hat{\top}_v & \text{for all } \hat{v} \in \hat{Val} \end{array}$$

$$\hat{v_1} \sqsubseteq_v \hat{v_2} \iff \begin{cases} \hat{v_1} \subseteq \hat{v_2} & \text{if } \hat{v_1} \in 2^{Exp}, \hat{v_2} \in 2^{Exp} \\ \hat{v_1} \sqsubseteq_g \hat{v_2} & \text{if } \hat{v_1} \in Grammar, \hat{v_2} \in Grammar \\ \hat{v_1} = \hat{v_2} & \text{if } \hat{v_1} = \{\cdot\}, \hat{v_2} = \{\cdot\} \end{cases}$$

- Trivially, $\hat{Val}$ is a partially ordered set for $\sqsubseteq_v$.
- $\hat{Val}$ has a least element $\hat{\bot}_v$.
- $\hat{Val}$ has a least upper bound $\hat{\top}_v$ for all chains.

Therefore, $\hat{Val}$ is CPO. □

5.2.2 Abstraction

Then we can define the abstraction map $\alpha_v : 2^{Val} \to \hat{Val}$ as follows.

$$\alpha_v V = \begin{cases} \hat{\bot}_v & \text{if } V = \emptyset \\ \{\cdot\} & \text{if } V \in 2^{Constant} \\ \{f \mid \langle f, \sigma \rangle \in V\} & \text{if } V \in 2^{Closure} \\ \bigsqcup_{e \in V} \xi(e) & \text{if } V \in 2^{Code} \\ \hat{\top}_v & \text{otherwise} \end{cases}$$

Naturally, we can define the inverse map $\gamma_v : \hat{Val} \to 2^{Val}$ as follows.

$$\gamma_v \hat{v} = \begin{cases} \emptyset & \text{if } \hat{v} = \hat{\bot}_v \\ \text{Set of all Constants} & \text{if } \hat{v} = \{\cdot\} \\ \{\langle f, \sigma \rangle : f \in \hat{v}, \sigma \in \text{Set of all environments}\} & \text{if } \hat{v} \in 2^{Exp} \\ \psi(\hat{v}) & \text{if } \hat{v} \in Grammar \\ \text{Set of all Values} & \text{if } \hat{v} = \hat{\top}_v \end{cases}$$

Lemma 3. *2^{Val} and $\hat{Val}$ are Galois connected with α_v and γ_v.*

$$2^{Val} \underset{\gamma_v}{\overset{\alpha_v}{\rightleftarrows}} \hat{Val}$$

Proof. 2^{Val} and $\hat{Val}$ are CPOs from lemma 1, 2.

1. Assume that $V \in 2^{Val}, \hat{v} \in \hat{Val}$ such that $\alpha_v V \sqsubseteq_v \hat{v}$.
   (a) If $V = \emptyset$, trivially, $V = \emptyset \subseteq \gamma_v \hat{v}$.
   (b) If $V \in 2^{Constant}$, $\hat{v} \sqsupseteq_v \alpha_v V = \{\cdot\}$. Therefore $\hat{v} = \hat{\top}_v$ or $\hat{v} = \{\cdot\}$.
   When $\hat{v} = \hat{\top}_v$, $\gamma_v \hat{v} =$ Set of all values. Therefore $V \subseteq \gamma_v \hat{v}$.
   When $\hat{v} = \{\cdot\}$, $\gamma_v \hat{v} =$ Set of all constants. Therefore $V \subseteq \gamma_v \hat{v}$.

(c) If $V \in 2^{Closure}$, $\hat{v} \sqsupseteq_v \alpha_v V = \{f \mid \langle f, \sigma\rangle \in V\}$. Therefore $\hat{v} = \hat{\top}_v$ or $\hat{v} \in 2^{Exp}$ and $\hat{v} \supseteq \alpha_v V$.
When $\hat{v} = \hat{\top}_v$, $\gamma_v \hat{v}$ = Set of all values. Therefore $V \subseteq \gamma_v \hat{v}$.
When $\hat{v} \in 2^{Exp}$ and $\hat{v} \supseteq \alpha_v V$, $\forall \langle f, \sigma\rangle \in V, f \in \hat{v}$ because $\hat{v} \supseteq \alpha_v V$. Thus $\gamma_v \hat{v} = \bigcup_{f \in \hat{v}, \sigma' \in \Sigma^*} \{\langle f, \sigma'\rangle\} \supseteq \bigcup_{\langle f,\sigma\rangle \in V, \sigma' \in \Sigma^*} \{\langle f, \sigma'\rangle\} \supseteq \bigcup_{\langle f,\sigma\rangle \in V} \{\langle f, \sigma\rangle\} = V$. Therefore, $V \subseteq \gamma_v \hat{v}$.

(d) If $V \in 2^{Code}$, $\hat{v} \sqsupseteq_v \alpha_v V = \xi(e_1) \sqcup \xi(e_2) \sqcup \cdots$ where $V = \{e_1, e_2, \cdots\}$. Therefore $\hat{v} = \hat{\top}_v$ or $\hat{v} = g$ such that $g \sqsupseteq_v \xi(e_1) \sqcup \xi(e_2) \sqcup \cdots$.
When $\hat{v} = \hat{\top}_v$, $\gamma_v \hat{v}$ = Set of all values. Therefore $V \subseteq \gamma_v \hat{v}$.
When $\hat{v} = g$, $\psi(g) \supseteq \{e_1, e_2, \cdots\}$ by definition. Therefore $\gamma_v \hat{v} = \psi(g) \supseteq \{e_1, e_2, \cdots\} = V$. Therefore, $V \subseteq \gamma_v \hat{v}$.

(e) Otherewise, $\hat{v} \sqsupseteq_v \alpha_v V = \hat{\top}_v$. Thus, $\hat{v} = \hat{\top}_v$ and $\gamma_v \hat{v}$ = Set of all values. Therefore $V \subseteq \gamma_v \hat{v}$.

Thus,

$$\alpha_v V \sqsubseteq_v \hat{v} \implies V \subseteq \gamma_v \hat{v}$$

2. Assume that $V \in 2^{Val}, \hat{v} \in \hat{Val}$ such that $V \subseteq \gamma_v \hat{v}$.

(a) If $\hat{v} = \hat{\bot}_v$, $V \subseteq \gamma_v \hat{v} = \gamma_v \hat{\bot}_v = \emptyset$ and $V = \emptyset$. Therefore, $\alpha_v V = \hat{\bot}_v \sqsubseteq_v \hat{v}$.

(b) If $\hat{v} = \{\cdot\}$, $V \subseteq \gamma_v \hat{v}$ = Set of all constants. Therefore $V = \emptyset$ or $V \in 2^{Constant}$.
When $V = \emptyset$, $\alpha_v V = \hat{\bot}_v \sqsubseteq_v \hat{v}$.
When $V \in 2^{Constant}$, $\alpha_v V = \{\cdot\} = \hat{v}$. Therefore $\alpha_v V \sqsubseteq_v \hat{v}$.

(c) If $\hat{v} \in 2^{Exp}$, $V \subseteq \gamma_v \hat{v} = \bigcup_{f \in \hat{v}, \sigma \in \Sigma^*} \{\langle f, \sigma\rangle\}$. Therefore $V = \emptyset$ or $V \in 2^{Closure}$ and $\forall \langle f, \sigma\rangle \in V, \exists \langle f', \sigma'\rangle \in \gamma_v \hat{v}$ such that $f = f'$ and $\sigma = \sigma'$.
When $V = \emptyset$, $\alpha_v V = \hat{\bot}_v \sqsubseteq_v \hat{v}$.
When $V \in 2^{Closure}$, $\alpha_v V = \bigcup_{\langle f,\sigma\rangle \in V} \{f\} \subseteq \bigcup_{\langle f,\sigma\rangle \in \gamma_v \hat{v}} \{f\} = \bigcup_{f \in \hat{v}} \{f\} = \hat{v}$. Therefore $\alpha_v V \sqsubseteq_v \hat{v}$.

(d) If $\hat{v} \in Grammar$, $V \subseteq \gamma_v \hat{v} = \psi(\hat{v})$. Therefore $V = \emptyset$ or $V \in 2^{Code}$ and $\forall e \in V, \exists e' \in \psi(\hat{v})$ such that $e = e'$.
When $V = \emptyset$, $\alpha_v V = \hat{\bot}_v \sqsubseteq_v \hat{v}$.
When $V \in 2^{Code}$, $\alpha_v V = \xi(e_1) \sqcup \xi(e_2) \sqcup \cdots$ where $V = \{e_1, e_2, \cdots\}$. Thus, $\alpha_v V \sqsubseteq_v \hat{v}$ by definition of ψ.

(e) If $\hat{v} = \hat{\top}_v$, trivially $\alpha_v V \sqsubseteq_v \hat{\top}_v = \hat{v}$.

Thus,

$$V \subseteq \gamma_v \hat{v} \implies \alpha_v V \sqsubseteq_v \hat{v}$$

By 1 and 2,

$$\alpha_v V \sqsubseteq_v \hat{v} \iff V \subseteq \gamma_v \hat{v}$$

□

5.3 Abstraction: Environment Set

Let us define an abstract environment as follows.

$$\hat{Env} = Var \xrightarrow{fin} \hat{Val}$$

5.3.1 Operations

- (Abstract Environment Join) $\forall \hat{\sigma_1}, \hat{\sigma}_2 \in \hat{Env}$,

$$(\hat{\sigma_1} \sqcup \hat{\sigma_2})x = \begin{cases} \hat{\sigma_1}x & \text{if } x \in \text{dom } (\hat{\sigma_1}) \text{ and } x \notin \text{dom } (\hat{\sigma_2}) \\ \hat{\sigma_2}x & \text{if } x \notin \text{dom } (\hat{\sigma_1}) \text{ and } x \in \text{dom } (\hat{\sigma_2}) \\ \hat{\sigma_1}x \ \sqcup \ \hat{\sigma_2}x & \text{otherwise} \end{cases}$$

Lemma 4. *$\hat{Env}$ is a CPO.*

Proof. Let us define an order $\sqsubseteq_e$ of $\hat{Env}$ as follows.

$$\hat{\sigma_1} \sqsubseteq_e \hat{\sigma_2} \iff \text{dom}(\hat{\sigma_1}) \subseteq \text{dom}(\hat{\sigma_1}) \wedge \forall x \in \text{dom}(\hat{\sigma_1}).\hat{\sigma_1}(x) \sqsubseteq_v \hat{\sigma_2}(x)$$

- Trivially, $\hat{Env}$ is a partially ordered set for $\sqsubseteq_v$.
- $\hat{Env}$ has a least element $[\ ]$.
- $\hat{Env}$ has a least upper bound for all chains. (Let $X = \{x_1, x_2, \cdots\}$ be a set of all variables. Then, $[x_1 \mapsto \hat{\top}_v, x_2 \mapsto \hat{\top}_v, \cdots]$ is the least upper bound.)

Therefore, $\hat{Env}$ is a CPO. □

5.3.2 Abstraction

Then we can define the abstraction map $\alpha_e : 2^{Env} \to \hat{Env}$ and inverse map γ_e as follows.

$$\begin{aligned} \alpha_e\ \Sigma &= \lambda x.\alpha_v \bigcup_{\sigma \in \Sigma} \{\sigma x\} \\ \gamma_e\ \hat{\sigma} &= \bigcup_{i=1,2,\cdots,n, v_i \in \gamma_v(\hat{\sigma} x_i)} \{[x_1 \mapsto v_1, x_2 \mapsto v_2, \cdots, x_n \mapsto v_n]\} \quad \{x_1, x_2, \cdots, x_n\} = \text{dom } (\hat{\sigma}) \end{aligned}$$

Lemma 5. *2^{Env} and $\hat{Env}$ are Galois connected with α_e and γ_e.*

Proof. 2^{Env} and $\hat{Env}$ are CPOs from lemma 1, 4.

1. Assume that $\Sigma \in 2^{Env}, \hat{\sigma} \in \hat{Env}$ such that $\alpha_e \Sigma \sqsubseteq_e \hat{\sigma}$.
   $\forall x \in Var$,

$$\begin{aligned} \hat{\sigma}x &\sqsupseteq_v (\alpha_e \Sigma)x \quad \text{(by assumption)} \\ &= (\lambda x.\alpha_v \bigcup_{\sigma \in \Sigma} \{\sigma x\})x \quad \text{(by definition)} \\ &= \alpha_v \bigcup_{\sigma \in \Sigma} \{\sigma x\} \\ &= \bigsqcup_{\sigma \in \Sigma} \alpha_v \{\sigma x\} \quad \text{(by lemma 3)} \end{aligned}$$

   Therefore, $\forall \sigma \in \Sigma$, $\alpha_v\{\sigma x\} \sqsubseteq_v \hat{\sigma} x$.

$$\begin{aligned} \alpha_v\{\sigma x\} &\sqsubseteq_v \hat{\sigma}x \\ \gamma_v \circ \alpha_v\{\sigma x\} &\subseteq \gamma_v \hat{\sigma} x \quad \text{(by lemma 3)} \\ \therefore \{\sigma x\} &\subseteq \gamma_v \hat{\sigma} x \quad \text{(by lemma 3)} \\ &= (\gamma_e \hat{\sigma})x \quad \text{(by definition)} \end{aligned}$$

   Thus,

$$\alpha_e \Sigma \sqsubseteq_e \hat{\sigma} \implies \Sigma \subseteq \gamma_e \hat{\sigma}$$

2. Assume that $\Sigma \in 2^{Env}, \hat{\sigma} \in \hat{Env}$ such that $\Sigma \subseteq \gamma_e \hat{\sigma}$.

$$\begin{aligned}(\alpha_e \Sigma)x &= \alpha_v \bigcup_{\sigma \in \Sigma} \{\sigma x\} \quad \text{(by definition)} \\ &\sqsubseteq_v \alpha_v \bigcup_{\sigma' \in \gamma_e \hat{\sigma}} \{\sigma' x\} \quad \text{(by assumption)}\end{aligned}$$

By definition of γ_e, $\sigma' x = v$ such that $v \in \gamma_v(\hat{\sigma} x)$.

$$\begin{aligned}\therefore (\alpha_e \Sigma)x &\sqsubseteq_v \alpha_v \bigcup_{\sigma' \in \gamma_e \hat{\sigma}} \{\sigma' x\} \\ &\sqsubseteq_v \alpha_v \bigcup_{v \in \gamma_v(\hat{\sigma} x)} \{v\} \\ &\sqsubseteq_v (\alpha_v \circ \gamma_v)(\hat{\sigma} x) \\ &\sqsubseteq_v \hat{\sigma} x \quad \text{(by lemma 3)}\end{aligned}$$

Thus,

$$\Sigma \subseteq \gamma_e \hat{\sigma} \implies \alpha_e \Sigma \sqsubseteq_e \hat{\sigma}$$

By 1 and 2,

$$\Sigma \subseteq \gamma_e \hat{\sigma} \iff \alpha_e \Sigma \sqsubseteq_e \hat{\sigma}$$

□

Lemma 6. *For an arbitrary set of environment* Σ*,*

$$\bigsqcup_{\sigma \in \Sigma} \alpha_e \{\sigma\} = \alpha_e \Sigma$$

.

Proof.

$$\begin{aligned}\bigsqcup_{\sigma \in \Sigma} \alpha_e \{\sigma\} &= \bigsqcup_{\sigma \in \Sigma} \lambda x.\alpha_v \{\sigma x\} \quad \text{(by definition)} \\ &= \lambda x. \bigsqcup_{\sigma \in \Sigma} \alpha_v \{\sigma x\} \\ &= \lambda x.\alpha_v \bigcup_{\sigma \in \Sigma} \{\sigma x\} \quad \text{(by lemma 3)} \\ &= \alpha_e \Sigma\end{aligned}$$

□

5.4 Abstraction: Set of Value and Environment Tuple

From the results of previous chatpers, we can define the abstraction map $\alpha_{v \times e} : 2^{Val \times Env} \to \hat{Val} \times \hat{Env}$ and inverse map $\gamma_{v \times e}$ as follows.

$$\alpha_{v \times e}\ T = \begin{cases} \langle \hat{\bot}_v, \hat{\bot}_e \rangle & \text{if } T = \emptyset \\ \langle \bigsqcup_{\langle v, \sigma \rangle \in T} \alpha_v \{v\}, \bigsqcup_{\langle v, \sigma \rangle \in T} \alpha_e \{\sigma\} \rangle & \text{otherwise} \end{cases}$$

$$\gamma_{v\times e}\ \langle \hat{v},\hat{\sigma}\rangle = \begin{cases} \emptyset & \text{if } \gamma_v\hat{v}=\emptyset, \gamma_e\hat{\sigma}=\emptyset \\ \{\langle \emptyset,\sigma\rangle : \sigma\in\gamma_e\hat{\sigma}\} & \text{if } \gamma_v\hat{v}=\emptyset, \gamma_e\hat{\sigma}\neq\emptyset \\ \{\langle v,\emptyset\rangle : v\in\gamma_v\hat{v}\} & \text{if } \gamma_v\hat{v}\neq\emptyset, \gamma_e\hat{\sigma}=\emptyset \\ \{\langle v,\sigma\rangle : v\in\gamma_v\hat{v}, \sigma\in\gamma_e\hat{\sigma}\} & \text{otherwise} \end{cases}$$

Lemma 7. *$\hat{Val}\times\hat{Env}$ is a CPO.*

Proof. As $\hat{Val}\times\hat{Env}$ is a production set of CPOs from lemma 2 and 4, it is a CPO, too. □

Lemma 8. *$2^{Val\times Env}$ and $\hat{Val}\times\hat{Env}$ are Galois connected with $\alpha_{v\times e}$ and $\gamma_{v\times e}$.*

Proof. $2^{Val\times Env}$ and $\hat{Val}\times\hat{Env}$ are CPOs from lemma 1, 7.

1. Assume that $T\in 2^{Val\times Env}, \langle\hat{v},\hat{\sigma}\rangle\in\hat{Val}\times\hat{Env}$ such that $\alpha_{v\times e}T\sqsubseteq\langle\hat{v},\hat{\sigma}\rangle$.
   Let $T=\bigcup_{i\in N}\{\langle v_i,\sigma_i\rangle\}$. Then, by assumption

$$\begin{aligned}\alpha_{v\times e}T &= \langle\bigsqcup_i\alpha_v\{v_i\},\bigsqcup_i\alpha_e\{\sigma_i\}\rangle \\ &\sqsubseteq \langle\hat{v},\hat{\sigma}\rangle\end{aligned}$$

   Therefore $\bigsqcup_i\alpha_v\{v_i\}\sqsubseteq_v\hat{v}$ and $\alpha_v\{v_i\}\sqsubseteq_v\hat{v}$. Thus, $\{v_i\}\subseteq\gamma_v\hat{v}$ by lemma 3 and $v_i\in\gamma_v\hat{v}$.
   Similarly, $\sigma_i\in\gamma_e\hat{\sigma}$.
   Let

$$\begin{aligned}A &= \gamma_{v\times e}\langle\hat{v},\hat{\sigma}\rangle \\ &= \bigcup_{v\in\gamma_v\hat{v},\sigma\in\gamma_e\hat{\sigma}}\{\langle v,\sigma\rangle\}\end{aligned}$$

   Trivially, $\langle v_i,\sigma_i\rangle\in A$. Therefore

$$\begin{aligned}T &= \bigcup_{i\in N}\{\langle v_i,\sigma_i\rangle\} \\ &\subseteq A \\ &= \gamma_{v\times e}\langle\hat{v},\hat{\sigma}\rangle\end{aligned}$$

   Thus,

$$\alpha_{v\times e}T\sqsubseteq\langle\hat{v},\hat{\sigma}\rangle \implies T\subseteq\gamma_{v\times e}\langle\hat{v},\hat{\sigma}\rangle$$

2. Assume that $T\in 2^{Val\times Env}, \langle\hat{v},\hat{\sigma}\rangle\in\hat{Val}\times\hat{Env}$ such that $T\subseteq\gamma_{v\times e}\langle\hat{v},\hat{\sigma}\rangle$.
   Let $T=\bigcup_{i\in N}\{\langle v_i,\sigma_i\rangle\}$. Then, by assumption

$$\begin{aligned}\bigcup_{i\in N}\{\langle v_i,\sigma_i\rangle\} &= T \\ &\subseteq \gamma_{v\times e}\langle\hat{v},\hat{\sigma}\rangle \\ &= \bigcup_{v\in\gamma_v\hat{v},\sigma\in\gamma_e\hat{\sigma}}\{\langle v,\sigma\rangle\}\end{aligned}$$

So, $\forall\langle v_i, \sigma_i\rangle \in T, \exists\langle v', \sigma'\rangle \in \gamma_v\langle\hat{v}, \hat{\sigma}\rangle$ such that $v_i = v'$and $\sigma_i = \sigma'$. Therefore

$$\begin{array}{rcl} v' \in \gamma_v\hat{v} & \Longleftrightarrow & \{v'\} \subseteq \gamma_v\hat{v} \\ & \Longleftrightarrow & \{v_i\} \subseteq \gamma_v\hat{v} \\ & \Longleftrightarrow & \alpha_v\{v_i\} \sqsubseteq_v \hat{v} \end{array}$$

Similarly, $\alpha_e\{\sigma_i\} \sqsubseteq_e \hat{\sigma}$.

$$\begin{array}{rcl} \therefore \alpha_{v\times e}T & = & \alpha_{v\times e}(\bigcup_{i\in N}\{\langle v_i, \sigma_i\rangle\}) \\ & = & \langle \bigsqcup_{\cdots}\alpha_v\{v_i\}, \bigsqcup_{\cdots}\alpha_e\{\sigma_i\}\rangle \\ & \sqsubseteq & \langle\hat{v}, \hat{\sigma}\rangle \end{array}$$

Thus,

$$T \subseteq \gamma_{v\times e}\langle\hat{v}, \hat{\sigma}\rangle \implies \alpha_{v\times e}T \sqsubseteq \langle\hat{v}, \hat{\sigma}\rangle$$

By 1 and 2,

$$\alpha_{v\times e}T \sqsubseteq \langle\hat{v}, \hat{\sigma}\rangle \Longleftrightarrow T \subseteq \gamma_{v\times e}\langle\hat{v}, \hat{\sigma}\rangle$$

□

6 Abstract Sementics

In this section, we define the abstract semantics of the target language. A semantic function

$$\hat{\mathcal{E}} = Stage \to Exp \to \hat{Env} \to \hat{Val} \times \hat{Env}$$

is the least fixpoint of a function

$$\hat{\mathcal{F}} \quad \in \quad (Stage \to Exp \to \hat{Env} \to \hat{Val} \times \hat{Env}) \to (Stage \to Exp \to \hat{Env} \to \hat{Val} \times \hat{Env})$$

$$\begin{array}{rcl} e & \in & Exp \\ x & \in & Var \\ l & \in & Label \\ \alpha & \in & Alphabet \\ \hat{v} & \in & \hat{Val} = Grammar + 2^{Exp} + \{\cdot\} + \hat{\bot}_v + \hat{\top}_v \\ \hat{\sigma} & \in & \hat{Env} = Var \xrightarrow{fin} \hat{Val} \\ Stage & = & \{0, 1\} \end{array}$$

1.

$$\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![c]\!]\ \hat{\sigma} \quad := \quad \langle\{\cdot\}, \hat{\sigma}\rangle$$

2.

$$\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![x]\!]\ \hat{\sigma} \quad := \quad \langle\hat{\sigma}x, \hat{\sigma}\rangle$$

3.

$$\begin{aligned}\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![\mathbf{let}\ x\ e_1\ e_2]\!]\ \hat{\sigma}\quad :=\quad & \\ & \mathbf{let}\ \langle \hat{v}', \hat{\sigma}' \rangle = \hat{\mathcal{E}}\ 0\ [\![e_1]\!]\ \hat{\sigma}\ \mathbf{in} \\ & \hat{\mathcal{E}}\ 0\ [\![e_2]\!]\ \hat{\sigma}'[x \mapsto \hat{v}']\end{aligned}$$

4.

$$\begin{aligned}\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![\mathbf{if}\ e_1\ e_2\ e_3]\!]\ \hat{\sigma}\quad :=\quad & \\ & \mathbf{let}\ \langle \hat{v}_2, \hat{\sigma}_2 \rangle = \hat{\mathcal{E}}\ 0\ [\![e_2]\!]\ \hat{\sigma}\ \mathbf{in} \\ & \mathbf{let}\ \langle \hat{v}_3, \hat{\sigma}_3 \rangle = \hat{\mathcal{E}}\ 0\ [\![e_3]\!]\ \hat{\sigma}\ \mathbf{in} \\ & \langle \hat{v}_2 \sqcup \hat{v}_3, \hat{\sigma}_2 \sqcup \hat{\sigma}_3 \rangle\end{aligned}$$

5.

$$\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![f\lambda x.e]\!]\ \hat{\sigma}\quad :=\quad \langle \{f\lambda x.e\}, \hat{\sigma} \rangle$$

6.

$$\begin{aligned}\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![e_1\ \ e_2]\!]\ \hat{\sigma}\quad :=\quad & \\ & \mathbf{let}\ \langle \hat{v}, \hat{\sigma}' \rangle = \hat{\mathcal{E}}\ 0\ [\![e_2]\!]\ \hat{\sigma}\ \mathbf{in} \\ & \mathbf{let}\ \langle \hat{f}, \hat{\sigma}_0 \rangle = \hat{\mathcal{E}}\ 0\ [\![e_1]\!]\ \hat{\sigma}'\ \mathbf{in} \\ & \mathbf{let}\ \langle \hat{v_1}, \hat{\sigma}_1 \rangle = \hat{\mathcal{E}}\ 0\ [\![b_1]\!]\ \hat{\sigma}_0[x_1 \mapsto \hat{v}][f_1 \mapsto \{f_1\lambda x_1.b_1\}]\ \mathbf{in} \\ & \mathbf{let}\ \langle \hat{v_2}, \hat{\sigma}_2 \rangle = \hat{\mathcal{E}}\ 0\ [\![b_2]\!]\ \hat{\sigma}_0[x_2 \mapsto \hat{v}][f_2 \mapsto \{f_2\lambda x_2.b_2\}]\ \mathbf{in} \\ & \cdots \\ & \mathbf{let}\ \langle \hat{v_n}, \hat{\sigma}_n \rangle = \hat{\mathcal{E}}\ 0\ [\![b_n]\!]\ \hat{\sigma}_0[x_n \mapsto \hat{v}][f_n \mapsto \{f_n\lambda x_n.b_n\}]\ \mathbf{in} \\ & \langle \hat{v_1} \sqcup \hat{v_2} \sqcup \cdots \sqcup \hat{v_n}, \hat{\sigma}_1 \sqcup \hat{\sigma}_2 \sqcup \cdots \sqcup \hat{\sigma}_n \rangle \\ & \text{where } \hat{f} = \{f_1\lambda x_1.b_1, f_2\lambda x_2.b_2, \cdots, f_n\lambda x_n.b_n\}\end{aligned}$$

7.

$$\begin{aligned}\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![^{\iota}e]\!]\ \hat{\sigma}\quad :=\quad & \\ & \mathbf{let}\ \langle g, \hat{\sigma}' \rangle = \hat{\mathcal{E}}\ 1\ [\![e]\!]\ \hat{\sigma}\ \mathbf{in} \\ & \langle l \otimes g, \hat{\sigma}' \rangle\end{aligned}$$

8.

$$\begin{aligned}\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![\mathbf{run}\ e]\!]\ \hat{\sigma}\quad :=\quad & \\ & \mathbf{let}\ \langle g, \hat{\sigma}' \rangle = \hat{\mathcal{E}}\ 0\ [\![e]\!]\ \hat{\sigma}\ \mathbf{in} \\ & \mathbf{let}\ \langle \hat{v}_1, \hat{\sigma}_1 \rangle = \hat{\mathcal{E}}\ 0\ [\![e_1]\!]\ \hat{\sigma}'\ \mathbf{in} \\ & \mathbf{let}\ \langle \hat{v}_2, \hat{\sigma}_2 \rangle = \hat{\mathcal{E}}\ 0\ [\![e_2]\!]\ \hat{\sigma}'\ \mathbf{in} \\ & \cdots \\ & \mathbf{let}\ \langle \hat{v}_n, \hat{\sigma}_n \rangle = \hat{\mathcal{E}}\ 0\ [\![e_n]\!]\ \hat{\sigma}'\ \mathbf{in} \\ & \langle \hat{v_1} \sqcup \hat{v_2} \sqcup \cdots \sqcup \hat{v_n}, \hat{\sigma}_1 \sqcup \hat{\sigma}_2 \sqcup \cdots \sqcup \hat{\sigma}_n \rangle \\ & \text{where } \psi(g) = \{^{\iota_1}e_1, ^{\iota_2}e_2, \cdots, ^{\iota_n}e_n\}\end{aligned}$$

$$\begin{aligned}\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![\mathbf{run}\ e]\!]\ \hat{\sigma}\quad :=\quad & \\ & \mathbf{let}\ \langle g, \hat{\sigma}' \rangle = \hat{\mathcal{E}}\ 0\ [\![e]\!]\ \hat{\sigma}\ \mathbf{in} \\ & \langle \top_g, \hat{\sigma}' \rangle \\ & \text{if } \psi(g) \text{ is infinite}\end{aligned}$$

9.

$$\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 1\ [\![c]\!]\ \hat{\sigma} \quad := \quad \langle \emptyset, \hat{\sigma} \rangle$$

10.

$$\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 1\ [\![x]\!]\ \hat{\sigma} \quad := \quad \langle \emptyset, \hat{\sigma} \rangle$$

11.

$$\begin{aligned}\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 1\ [\![\textbf{let}\ x\ e_1\ e_2]\!]\ \hat{\sigma} \quad := \\ &\textbf{let}\ (\hat{v_1}, \hat{\sigma_1}) = \hat{\mathcal{E}}\ 1\ [\![e_1]\!]\ \hat{\sigma}\ \textbf{in} \\ &\textbf{let}\ (\hat{v_2}, \hat{\sigma_2}) = \hat{\mathcal{E}}\ 1\ [\![e_2]\!]\ \hat{\sigma_1}\ \textbf{in} \\ &\langle \hat{v_1} \sqcup \hat{v_2}, \hat{\sigma_2} \rangle\end{aligned}$$

12.

$$\begin{aligned}\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 1\ [\![\textbf{if}\ e_1\ e_2\ e_3]\!]\ \hat{\sigma} \quad := \\ &\textbf{let}\ \langle \hat{v}_1, \hat{\sigma}_1 \rangle = \hat{\mathcal{E}}\ 1\ [\![e_1]\!]\ \hat{\sigma}\ \textbf{in} \\ &\textbf{let}\ \langle \hat{v}_2, \hat{\sigma}_2 \rangle = \hat{\mathcal{E}}\ 1\ [\![e_2]\!]\ \hat{\sigma}_1\ \textbf{in} \\ &\textbf{let}\ \langle \hat{v}_3, \hat{\sigma}_3 \rangle = \hat{\mathcal{E}}\ 1\ [\![e_3]\!]\ \hat{\sigma}_2\ \textbf{in} \\ &\langle \hat{v_1} \sqcup \hat{v_2} \sqcup \hat{v_3}, \hat{\sigma}_3 \rangle\end{aligned}$$

13.

$$\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 1\ [\![f\lambda x.e]\!]\ \hat{\sigma} \quad := \quad \hat{\mathcal{E}}\ 1\ [\![e]\!]\ \hat{\sigma}$$

14.

$$\begin{aligned}\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 1\ [\![e_1\ \ e_2]\!]\ \hat{\sigma} \quad := \\ &\textbf{let}\ (\hat{v_1}, \hat{\sigma_1}) = \hat{\mathcal{E}}\ 1\ [\![e_1]\!]\ \hat{\sigma}\ \textbf{in} \\ &\textbf{let}\ (\hat{v_2}, \hat{\sigma_2}) = \hat{\mathcal{E}}\ 1\ [\![e_2]\!]\ \hat{\sigma_1}\ \textbf{in} \\ &\langle \hat{v_1} \sqcup \hat{v_2}, \hat{\sigma_2} \rangle\end{aligned}$$

15.

$$\begin{aligned}\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 1\ [\![_{,a} e]\!]\ \hat{\sigma} \quad := \\ &\textbf{let}\ \langle g, \hat{\sigma}' \rangle = \hat{\mathcal{E}}\ 0\ [\![e]\!]\ \hat{\sigma}\ \textbf{in} \\ &\langle a \oplus g, \hat{\sigma}' \rangle\end{aligned}$$

16.

$$\hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 1\ [\![\textbf{run}\ e]\!]\ \hat{\sigma} \quad := \quad \hat{\mathcal{E}}\ 1\ [\![e]\!]\ \hat{\sigma}$$

7 Correctness of Abstract Semantics

We prove the correctness of the abstract semantics shown in the previous chapter. To prove the correctness, we need a lemma.

Lemma 9. *For arbitrary $x \in Var, \sigma \in Env, v \in Val$.*

$$\alpha_e\{\sigma[x \mapsto v]\} \sqsubseteq_e (\alpha_e\{\sigma\})[x \mapsto \alpha_v\{v\}]$$

Proof. Recall that

$$\begin{aligned}\alpha_e\{\sigma[x \mapsto v]\} &= \lambda z.\alpha_v \bigcup_{\sigma' \in \{\sigma[x \mapsto v]\}} \{\sigma' z\} \\ &= \lambda z.\alpha_v\{\sigma[x \mapsto v]z\}\end{aligned}$$

For arbitrary $y \in Var$,

1. If $y \neq x$,

$$\begin{aligned}\alpha_e\{\sigma[x \mapsto v]\}y &= (\lambda z.\alpha_v\{\sigma[x \mapsto v]z\})y \\ &= \alpha_v\{\sigma y\} \\ (\alpha_e\{\sigma\})[x \mapsto \alpha_v\{v\}]y &= (\alpha_e\{\sigma\})y \\ &= (\lambda z.\alpha_v \bigcup_{\sigma' \in \{\sigma\}} \{\sigma' z\})y \\ &= (\lambda z.\alpha_v\{\sigma z\})y \\ &= \alpha_v\{\sigma y\}) \\ \therefore \alpha_e\{\sigma[x \mapsto v]\}y &= (\alpha_e\{\sigma\})[x \mapsto \alpha_v\{v\}]y\end{aligned}$$

2. If $y = x$,

$$\begin{aligned}\alpha_e\{\sigma[x \mapsto v]\}y &= (\lambda z.\alpha_v\{\sigma[x \mapsto v]z\})y \\ &= (\lambda z.\alpha_v\{\sigma[x \mapsto v]z\})x \\ &= \alpha_v\{v\} \\ (\alpha_e\{\sigma\})[x \mapsto \alpha_v\{v\}]y &= (\alpha_e\{\sigma\})[x \mapsto \alpha_v\{v\}]x \\ &= \alpha_v\{v\} \\ \therefore \alpha_e\{\sigma[x \mapsto v]\}y &= (\alpha_e\{\sigma\})[x \mapsto \alpha_v\{v\}]y\end{aligned}$$

From 1 and 2,

$$\alpha_e\{\sigma[x \mapsto v]\}y = (\alpha_e\{\sigma\})[x \mapsto \alpha_v\{v\}]y$$

□

Theorem 1. *For arbitrary expression e,*

$$\alpha_{v \times e} \circ fix\mathcal{F}[\![e]\!] \sqsubseteq fix\hat{\mathcal{F}}[\![e]\!] \circ \alpha_e$$

Proof. We will prove the theorem by the fixpoint induction[5].

Let $P(f, g)$ be an assertion

$$P(f, g) = \forall s \in Stage, \forall e \in Exp : \alpha_{v \times e} \circ f\ s\ [\![e]\!] \sqsubseteq g\ s\ [\![e]\!] \circ \alpha_e$$

.

Base case: From lemma 5 and lemma 8, α_e and $\alpha_{v \times e}$ are strict. Therefore, $P(\perp, \perp)$ holds trivially.

Inductive case: Assume that for continuous functions $\mathcal{E}$ and $\hat{\mathcal{E}}$, $P(\mathcal{E}, \hat{\mathcal{E}})$ holds. Then we should show that

$$P(\mathcal{F}(\mathcal{E}), \hat{\mathcal{F}}(\hat{\mathcal{E}}))$$

1. (c: 0-stage)

$$\begin{aligned}
\alpha_{v\times e}(\mathcal{F}\ \mathcal{E}\ 0\ [\![c]\!]\ \Sigma) &= \alpha_{v\times e}(\bigcup_{\sigma\in\Sigma}\{\langle c,\sigma\rangle\}) \\
&= \langle \bigsqcup_{\sigma\in\Sigma}\alpha_v\{c\}, \bigsqcup_{\sigma\in\Sigma}\alpha_e\{\sigma\}\rangle \\
&= \langle \bigsqcup_{\sigma\in\Sigma}\{\cdot\}, \alpha_e\Sigma\rangle \quad \text{(by lemma 6)} \\
&= \langle\{\cdot\}, \alpha_e\Sigma\rangle \\
&= \hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![c]\!]\ \alpha_e\Sigma
\end{aligned}$$

2. (x: 0-stage)

$$\begin{aligned}
\alpha_{v\times e}(\mathcal{F}\ \mathcal{E}\ 0\ [\![x]\!]\ \Sigma) &= \alpha_{v\times e}(\bigcup_{\sigma\in\Sigma}\{\langle \sigma x,\sigma\rangle\}) \\
&= \langle \bigsqcup_{\sigma\in\Sigma}\alpha_v\{\sigma x\}, \bigsqcup_{\sigma\in\Sigma}\alpha_e\{\sigma\}\rangle \\
&= \langle \bigsqcup_{\sigma\in\Sigma}\alpha_v\{\sigma x\}, \alpha_e\Sigma\rangle \quad \text{(by lemma 6)} \\
&= \langle \alpha_e\Sigma x, \alpha_e\Sigma\rangle \quad \text{(by definition)} \\
&= \hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![x]\!]\ \alpha_e\Sigma
\end{aligned}$$

3. (**let** $x\ e_1\ e_2$: 0-stage) Let

$$\begin{aligned}
T &= \mathcal{E}\ 0\ [\![e_1]\!]\ \Sigma \\
\langle \hat{v}', \hat{\sigma}'\rangle &= \hat{\mathcal{E}}\ 0\ [\![e_1]\!]\ \alpha_e\Sigma
\end{aligned}$$

As $\alpha_{v\times e}T \sqsubseteq \langle \hat{v}', \hat{\sigma}'\rangle$ by induction hypothesis, $\forall\langle v,\sigma\rangle \in T$, $\alpha_v\{v\} \sqsubseteq_v \hat{v}'$ and $\alpha_e\{\sigma\} \sqsubseteq_e \hat{\sigma}'$ by definition. Therefore

$$\begin{aligned}
\alpha_e\{\sigma[x\mapsto v]\} &\sqsubseteq_e \alpha_e\{\sigma\}[x\mapsto \alpha_v\{v\}] \quad \text{(by lemma 9)} \\
&\sqsubseteq_e \hat{\sigma}'[x\mapsto \hat{v}'] \quad \text{(by definition)}
\end{aligned}$$

$$\begin{aligned}
\alpha_{v\times e}(\mathcal{F}\ \mathcal{E}\ 0\ [\![\textbf{let}\ x\ e_1\ e_2]\!]\ \Sigma) &= \alpha_{v\times e}(\bigcup_{\langle v,\sigma\rangle\in VS}\mathcal{E}\ 0\ [\![e_2]\!]\ \{\sigma[x\mapsto v]\}) \\
&\sqsubseteq \bigsqcup_{\langle v,\sigma\rangle\in VS}\alpha_{v\times e}(\mathcal{E}\ 0\ [\![e_2]\!]\ \{\sigma[x\mapsto v]\}) \quad \text{(by lemma 8)} \\
&\sqsubseteq \bigsqcup_{\langle v,\sigma\rangle\in VS}\hat{\mathcal{E}}\ 0\ [\![e_2]\!]\ \alpha_e\{\sigma[x\mapsto v]\} \quad \text{(by induction hypothesis)} \\
&\sqsubseteq \bigsqcup_{\langle v,\sigma\rangle\in VS}\hat{\mathcal{E}}\ 0\ [\![e_2]\!]\ \hat{\sigma}'[x\mapsto \hat{v}'] \quad \text{(by continuity of } \hat{\mathcal{E}}) \\
&= \hat{\mathcal{E}}\ 0\ [\![e_2]\!]\ \hat{\sigma}'[x\mapsto \hat{v}'] \\
&= \hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![\textbf{let}\ x\ e_1\ e_2]\!]\ \alpha_e\Sigma
\end{aligned}$$

4. (**if** e_1 e_2 e_3: 0-stage)

$$\begin{array}{rcl}
\alpha_{v\times e}(\mathcal{F}\ \mathcal{E}\ 0\ [\![\textbf{if}\ e_1\ e_2\ e_3]\!]\ \Sigma) & = & \alpha_{v\times e}(\mathcal{E}\ 0\ [\![e_2]\!]\ (\neg\mathcal{B}\ [\![e_1]\!]\ \Sigma) \cup \mathcal{E}\ 0\ [\![e_3]\!]\ (\mathcal{B}\ [\![e_1]\!]\ \Sigma)) \\
& \sqsubseteq & \alpha_{v\times e}(\mathcal{E}\ 0\ [\![e_2]\!]\ (\neg\mathcal{B}\ [\![e_1]\!]\ \Sigma)) \sqcup \alpha_{v\times e}(\mathcal{E}\ 0\ [\![e_3]\!]\ (\mathcal{B}\ [\![e_1]\!]\ \Sigma)) \\
& & \text{(by lemma 8)} \\
& \sqsubseteq & \hat{\mathcal{E}}\ 0\ [\![e_2]\!]\ \alpha_e(\neg\mathcal{B}\ [\![e_1]\!]\ \Sigma) \sqcup \hat{\mathcal{E}}\ 0\ [\![e_3]\!]\ \alpha_e(\mathcal{B}\ [\![e_1]\!]\ \Sigma) \\
& & \text{(by induction hypothesis)} \\
& \sqsubseteq & (\hat{\mathcal{E}}\ 0\ [\![e_2]\!]\ \alpha_e\Sigma) \sqcup (\hat{\mathcal{E}}\ 0\ [\![e_3]\!]\ \alpha_e\Sigma) \quad \text{(by continuity of } \hat{\mathcal{E}} \text{ and lemma 5)} \\
& = & \hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![\textbf{if}\ e_1\ e_2\ e_3]\!]\ \alpha_e\Sigma
\end{array}$$

5. ($f\lambda x.e$: 0-stage)

$$\begin{array}{rcl}
\alpha_{v\times e}(\mathcal{F}\ \mathcal{E}\ 0\ [\![f\lambda x.e]\!]\ \Sigma) & = & \alpha_{v\times e}(\bigcup_{\sigma\in\Sigma}\{\langle\langle f\lambda x.e,\sigma\rangle,\sigma\rangle\}) \\
& \sqsubseteq & \bigsqcup_{\sigma\in\Sigma}\alpha_{v\times e}\{\langle\langle f\lambda x.e,\sigma\rangle,\sigma\rangle\} \quad \text{(by lemma 8)} \\
& = & \bigsqcup_{\sigma\in\Sigma}\langle\alpha_v\{\langle f\lambda x.e,\sigma\rangle\},\alpha_e\{\sigma\}\rangle \\
& = & \langle\bigsqcup_{\sigma\in\Sigma}\alpha_v\{\langle f\lambda x.e,\sigma\rangle\},\bigsqcup_{\sigma\in\Sigma}\alpha_e\{\sigma\}\rangle \\
& = & \langle\bigsqcup_{\sigma\in\Sigma}\{f\lambda x.e\},\alpha_e\Sigma\rangle \quad \text{(by lemma 6)} \\
& = & \langle\{f\lambda x.e\},\alpha_e\Sigma\rangle \\
& = & \hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![f\lambda x.e]\!]\ \alpha_e\Sigma
\end{array}$$

6. (e_1 e_2: 0-stage) Let

$$\begin{array}{rcl}
T & = & \mathcal{E}\ 0\ [\![e_2]\!]\ \Sigma \\
\langle\hat{v},\hat{\sigma}_2'\rangle & = & \hat{\mathcal{E}}\ 0\ [\![e_2]\!]\ \alpha_e\Sigma
\end{array}$$

As $\alpha_{v\times e}T \sqsubseteq \langle\hat{v},\hat{\sigma}_2'\rangle$ by induction hypothesis, $\forall\langle v,\sigma_2\rangle \in T$, $\alpha_v\{v\} \sqsubseteq_v \hat{v}$ and $\alpha_e\{\sigma_2\} \sqsubseteq_e \hat{\sigma}_2'$ by definition.

Let

$$\begin{array}{rcl}
T' & = & \mathcal{E}\ 0\ [\![e_1]\!]\ \{\sigma_2\} \\
\langle\hat{f},\hat{\sigma}_1'\rangle & = & \hat{\mathcal{E}}\ 0\ [\![e_1]\!]\ \hat{\sigma}_2'
\end{array}$$

Then,

$$\begin{array}{rcl}
\alpha_{v\times e}T' & \sqsubseteq & \hat{\mathcal{E}}\ 0\ [\![e_1]\!]\ \alpha_e\{\sigma_2\} \quad \text{(by induction hypothesis)} \\
& \sqsubseteq & \hat{\mathcal{E}}\ 0\ [\![e_1]\!]\ \hat{\sigma}_2' \quad \text{(by continuity of } \hat{\mathcal{E}}\text{)} \\
& = & \langle\hat{f},\hat{\sigma}_1'\rangle
\end{array}$$

Therefore, $\forall\langle f,\sigma_1\rangle \in T'$, $\alpha_v\{f\} \sqsubseteq_v \hat{f}$ and $\alpha_e\{\sigma_1\} \sqsubseteq_e \hat{\sigma}_1'$ by definition.

$$\begin{aligned}
\alpha_{v\times e}(\mathcal{F}\ \mathcal{E}\ 0\ [\![e_1\ \ e_2]\!]\ \Sigma) &= \alpha_{v\times e}(\bigcup_{\langle\langle f\lambda x.b,\sigma\rangle,\sigma_1\rangle\in FS,\langle v,\sigma_2\rangle\in VS} \mathcal{E}\ 0\ [\![b]\!]\ \{\sigma_2[x\mapsto v][f\mapsto\langle f\lambda x.b,\sigma\rangle]\}) \\
&\sqsubseteq \bigsqcup_{\cdots} \alpha_{v\times e}(\mathcal{E}\ 0\ [\![b]\!]\ \{\sigma_2[x\mapsto v][f\mapsto\langle f\lambda x.b,\sigma\rangle]\}) \quad \text{(by lemma 8)} \\
&\sqsubseteq \bigsqcup_{\cdots} \hat{\mathcal{E}}\ 0\ [\![b]\!]\ \alpha_e\{\sigma_2[x\mapsto v][f\mapsto\langle f\lambda x.b,\sigma\rangle]\} \quad \text{(by induction hypothesis)} \\
&\sqsubseteq \bigsqcup_{\cdots} \hat{\mathcal{E}}\ 0\ [\![b]\!]\ \alpha_e\{\sigma_2\}[x\mapsto\alpha_v\{v\}][f\mapsto\alpha_v\{\langle f\lambda x.b,\sigma\rangle\}] \quad \text{(by lemma 9)} \\
&\sqsubseteq \bigsqcup_{\cdots} \hat{\mathcal{E}}\ 0\ [\![b]\!]\ \hat{\sigma}_2'[x\mapsto\hat{v}][f\mapsto\{f\lambda x.b\}] \quad \text{(by continuity of } \hat{\mathcal{E}}) \\
&= \hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![e_1\ \ e_2]\!]\ \alpha_e\Sigma
\end{aligned}$$

7. ($^{\iota}e$: 0-stage) Let

$$\begin{aligned}
T &= \mathcal{E}\ 1\ [\![e]\!]\ \Sigma \\
\langle g,\hat{\sigma}'\rangle &= \hat{\mathcal{E}}\ 1\ [\![e]\!]\ \alpha_e\Sigma
\end{aligned}$$

As $\alpha_{v\times e}T \sqsubseteq \langle g,\hat{\sigma}'\rangle$ by induction hypothesis, $\forall\langle e',\sigma\rangle\in T$, $\alpha_v\{e\}\sqsubseteq_v g$ and $\alpha_e\{\sigma\}\sqsubseteq_e\hat{\sigma}'$ by definition. Therefore, $\{e'\}\subseteq\psi(g)$ and $e'\in\psi(g)$.

$\gamma_v(l\sqcup g)=\psi(l\sqcup g)$ and it represents the set of all expression whose label is l and whose subexpressions can be replaced with $\psi(g)$. Therefore, $^{\iota}e'\in\psi(l\sqcup g)$ and $\{^{\iota}e'\}\in\gamma_v(l\sqcup g)$. Thus, $\alpha_v\{^{\iota}e'\}\sqsubseteq_v l\sqcup g$

$$\begin{aligned}
\alpha_{v\times e}(\mathcal{F}\ \mathcal{E}\ 0\ [\![^{\iota}e]\!]\ \Sigma) &= \alpha_{v\times e}\bigcup_{\langle e',\sigma\rangle\in CS}\{\langle^{\iota}e',\sigma\rangle\} \\
&= \langle\bigsqcup_{\cdots}\alpha_v\{^{\iota}e'\},\bigsqcup_{\cdots}\alpha_e\{\sigma\}\rangle \\
&\sqsubseteq \langle l\sqcup g,\hat{\sigma}'\rangle \\
&= \hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![^{\iota}e]\!]\ \alpha_e\Sigma
\end{aligned}$$

8. (**run** e: 0-stage) Let

$$\begin{aligned}
T &= \mathcal{E}\ 0\ [\![e]\!]\ \Sigma \\
\langle g,\hat{\sigma}'\rangle &= \hat{\mathcal{E}}\ 0\ [\![e]\!]\ \alpha_e\Sigma
\end{aligned}$$

As $\alpha_{v\times e}T\sqsubseteq\langle g,\hat{\sigma}'\rangle$ by induction hypothesis, $\forall\langle e',\sigma\rangle\in T$, $\alpha_v\{e'\}\sqsubseteq_v g$ and $\alpha_e\{\sigma\}\sqsubseteq_e\hat{\sigma}'$ by definition. Thus, $\{e'\}\subseteq\gamma_v g$ and $e'\in\psi(g)$.

$$\begin{aligned}
\alpha_{v\times e}(\mathcal{F}\ \mathcal{E}\ 0\ [\![\mathbf{run}\ e]\!]\ \Sigma) &= \alpha_{v\times e}(\bigcup_{\langle e',\sigma\rangle\in VS} \mathcal{E}\ 0\ [\![e']\!]\ \{\sigma\}) \\
&\sqsubseteq \bigsqcup_{\langle e',\sigma\rangle\in VS} \alpha_{v\times e}(\mathcal{E}\ 0\ [\![e']\!]\ \{\sigma\}) \quad \text{(by lemma 8)} \\
&\sqsubseteq \bigsqcup_{\langle e',\sigma\rangle\in VS} \hat{\mathcal{E}}\ 0\ [\![e']\!]\ \alpha_e\{\sigma\} \quad \text{(by induction hypothesis)} \\
&\sqsubseteq \bigsqcup_{\langle e',\sigma\rangle\in VS} \hat{\mathcal{E}}\ 0\ [\![e']\!]\ \hat{\sigma} \quad \text{(by continuity of } \hat{\mathcal{E}}) \\
&\sqsubseteq \bigsqcup_{e'\in\psi(g)} \hat{\mathcal{E}}\ 0\ [\![e']\!]\ \hat{\sigma} \\
&= \hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 0\ [\![\mathbf{run}\ e]\!]\ \alpha_e\Sigma
\end{aligned}$$

9. (c: 1-stage)

$$\begin{aligned}
\alpha_{v\times e}(\mathcal{F}\ \mathcal{E}\ 1\ [\![c]\!]\ \Sigma) &= \alpha_{v\times e}(\bigcup_{\sigma\in\Sigma}\{\langle c,\sigma\rangle\}) \\
&= \langle \bigsqcup_{\sigma\in\Sigma} \alpha_v\{c\}, \bigsqcup_{\sigma\in\Sigma} \alpha_e\{\sigma\}\rangle \\
&= \langle \bigsqcup_{\sigma\in\Sigma} \xi(c), \alpha_e\Sigma\rangle \quad \text{(by lemma 6)} \\
&= \langle \bigsqcup_{\sigma\in\Sigma} \bot_g, \alpha_e\Sigma\rangle \\
&= \langle \bot_g, \alpha_e\Sigma\rangle \\
&= \hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 1\ [\![c]\!]\ \hat{\sigma}
\end{aligned}$$

10. (x: 1-stage)

$$\begin{aligned}
\alpha_{v\times e}(\mathcal{F}\ \mathcal{E}\ 1\ [\![x]\!]\ \Sigma) &= \alpha_{v\times e}(\bigcup_{\sigma\in\Sigma}\{\langle x,\sigma\rangle\}) \\
&= \langle \bigsqcup_{\sigma\in\Sigma} \alpha_v\{x\}, \bigsqcup_{\sigma\in\Sigma} \alpha_e\{\sigma\}\rangle \\
&= \langle \bigsqcup_{\sigma\in\Sigma} \xi(x), \alpha_e\Sigma\rangle \quad \text{(by lemma 6)} \\
&= \langle \bigsqcup_{\sigma\in\Sigma} \bot_g, \alpha_e\Sigma\rangle \\
&= \langle \bot_g, \alpha_e\Sigma\rangle \\
&= \hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 1\ [\![x]\!]\ \hat{\sigma}
\end{aligned}$$

11. (**let** $x\ e_1\ e_2$: 1-stage) Let

$$\begin{aligned}
T_1 &= \mathcal{E}\ 1\ [\![e_1]\!]\ \Sigma \\
\langle g_1, \hat{\sigma}_1\rangle &= \hat{\mathcal{E}}\ 1\ [\![e_1]\!]\ \alpha_e\Sigma
\end{aligned}$$

As $\alpha_{v\times e}T_1 \sqsubseteq \langle g_1, \hat{\sigma}_1\rangle$ by induction hypothesis, $\forall\langle e_1', \sigma_1\rangle \in T_1$, $\alpha_v\{e_1'\} \sqsubseteq_v \hat{g}_1$ and $\alpha_e\{\sigma_1\} \sqsubseteq_e \hat{\sigma}_1$ by definition. Thus, $e_1' \in \gamma_v(g_1) = \psi(g_1)$.

Let

$$\begin{aligned} T_2 &= \bigcup_{\langle e', \sigma_1 \rangle \in T_1} \mathcal{E}\ 1\ [\![e_2]\!]\ \{\sigma_1\} \\ \langle g_2, \hat{\sigma}_2 \rangle &= \hat{\mathcal{E}}\ 1\ [\![e_2]\!]\ \hat{\sigma}_1 \end{aligned}$$

$$\begin{aligned} \alpha_{v \times e} T_2 &= \alpha_{v \times e}(\bigcup_{\langle e', \sigma_1 \rangle \in T_1} \mathcal{E}\ 1\ [\![e_2]\!]\ \{\sigma_1\}) \\ &\sqsubseteq \bigsqcup_{\cdots} \alpha_{v \times e}(\mathcal{E}\ 1\ [\![e_2]\!]\ \{\sigma_1\}) \\ &\sqsubseteq \bigsqcup_{\cdots} \hat{\mathcal{E}}\ 1\ [\![e_2]\!]\ \alpha_e\{\sigma_1\} \quad \text{(by induction hypothesis)} \\ &\sqsubseteq \bigsqcup_{\cdots} \hat{\mathcal{E}}\ 1\ [\![e_2]\!]\ \hat{\sigma}_1 \quad \text{(by continuity of } \hat{\mathcal{E}}) \\ &= \hat{\mathcal{E}}\ 1\ [\![e_2]\!]\ \hat{\sigma}_1 \\ &= \langle g_2, \hat{\sigma}_2 \rangle \end{aligned}$$

Therefore, $\forall \langle e_2', \sigma_2 \rangle \in T_2$, $\alpha_v\{e_2'\} \sqsubseteq_v g_2$ and $\alpha_e\{\sigma_2\} \sqsubseteq_e \hat{\sigma}_2$ by definition. Thus, $e_2' \in \gamma_v(g_2) = \psi(g_2)$.

$$\begin{aligned} \alpha_{v \times e}(\mathcal{F}\ \mathcal{E}\ 1\ [\![\textbf{let}\ x\ e_1\ e_2]\!]\ \Sigma) &= \alpha_{v \times e}(\bigcup_{\langle e_1', \sigma_1 \rangle \in CS_1, \langle e_2', \sigma_2 \rangle \in CS_2} \{\langle \textbf{let}\ x\ e_1'\ e_2', \sigma_2 \rangle\}) \\ &= \langle \bigsqcup_{\cdots} \alpha_v\{\textbf{let}\ x\ e_1'\ e_2'\}, \bigsqcup_{\cdots} \alpha_e\{\sigma_2\} \rangle \end{aligned}$$

$e_1' \in \gamma_v(g_1) = \psi(g_1)$ and a code fragment **let** $x\ e_1'\ e_2'$ represents *let* expression whose subexpressions can be replaced with e_1' and e_2', it is naturally true that $\{\textbf{let}\ x\ e_1'\ e_2'\} \subseteq \psi(g_1) \sqcup \psi(g_2)$ from the definition of ψ. Therefore $\alpha_v\{\textbf{let}\ x\ e_1'\ e_2'\} \sqsubseteq_v g_1 \sqcup g_2$.

$$\begin{aligned} \therefore \alpha_{v \times e}(\mathcal{F}\ \mathcal{E}\ 1\ [\![\textbf{let}\ x\ e_1\ e_2]\!]\ \Sigma) &\sqsubseteq \langle \bigsqcup_{\cdots} \alpha_v\{\textbf{let}\ x\ e_1'\ e_2'\}, \bigsqcup_{\cdots} \alpha_e\{\sigma_2\} \rangle \\ &\sqsubseteq \langle \bigsqcup_{\cdots} g_1 \sqcup g_2, \bigsqcup_{\cdots} \hat{\sigma}_2 \rangle \\ &= \langle g_1 \sqcup g_2, \hat{\sigma}_2 \rangle \\ &= \hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 1\ [\![\textbf{let}\ x\ e_1\ e_2]\!]\ \alpha_e \Sigma \end{aligned}$$

12. (**if** $e_1\ e_2\ e_3$: 1-stage) It is easy to show that the theorem holds using similar steps in (**let** $x\ e_1\ e_2$: 1-stage).

13. ($f\lambda x.e$: 1-stage) It is easy to show that the theorem holds using similar steps in (**let** $x\ e_1\ e_2$: 1-stage).

14. ($e_1\ \ e_2$: 1-stage) It is easy to show that the theorem holds using similar steps in (**let** $x\ e_1\ e_2$: 1-stage).

15. ($_{,\alpha}e$: 1-stage)) Let

$$\langle g, \hat{\sigma}' \rangle \quad = \quad \hat{\mathcal{E}}\ 0\ [\![e]\!]\ \alpha_e \Sigma$$

$\psi(a \oplus g)$ represents the set of all expressions that have unbox subexpression whose label is a and the subexpression can be replaced with expressions in $\psi(g)$. Naturally, $\psi(g) \subseteq \psi(a \oplus g)$ and $g \sqsubseteq_v a \oplus g$.

$$\begin{array}{rcl}
\alpha_{v \times e}(\mathcal{F}\ \mathcal{E}\ 1\ [\![_{,\alpha}e]\!]\ \Sigma) & = & \alpha_{v \times e}(\mathcal{E}\ 0\ [\![e]\!]\ \Sigma) \\
& \sqsubseteq & \hat{\mathcal{E}}\ 0\ [\![e]\!]\ \alpha_e \Sigma \quad \text{(by induction hypothesis)} \\
& = & \langle g, \hat{\sigma}' \rangle \\
& \sqsubseteq & \langle a \oplus g, \hat{\sigma}' \rangle \\
& = & \hat{\mathcal{F}}\ \hat{\mathcal{E}}\ 1\ [\![_{,\alpha}e]\!]\ \alpha_e \Sigma
\end{array}$$

16. (**run** e: 1-stage) It is easy to show that the theorem holds using similar steps in (**let** $x\ e_1\ e_2$: 1-stage).

□

8 Conclusion

We designed a control flow analysis for a simple two-staged language and prove the correctness of the analyzer in this article. As Two-staged language can generate and execute code fragments in execution time, to analyze features in the language it is need to predict what code fragments would be generated in execution time. To achieve the goal, we introduce new data type, grammar, and design a analyzer with it. Moreover we use a fixpoint induction to prove the correctness of the analyzer.

The analyzer have some explicit limitations as follows.

- The target language only support two-stage processing.

  We assumed that the target language only support 2-stage processing to simplify the semantics of the language. Without this assumption, the semantics would be more complex and the analyzer should be more complex, too.

- The target language does not contain data types and operators in main stream languages(e.g. Ruby, Python, etc.)

  Many languages have various data types (integer, floating numbers, arrays, strings, etc.) and operations for the data types. The target language only have one type (constant) and does not provide any operators on the data type. Therefore to apply the analyzer to main stream language, we should extend syntax and semantics of the target language.

- Accuracy of the analyzer sould be improved.

References

[1] I.-S. Kim, K. Yi, and C. Calcagno, "A polymorphic modal type system for lisp-like multi-staged languages," in *Proceedings of The ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2006, pp. 257–269.

[2] P. Cousot and R. Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *POPL '77: Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. New York, NY, USA: ACM, 1977, pp. 238–252.

[3] W. V. Quine, *Mathematical Logic, Revised Edition*. Harvard University Press, 2003.

[4] N. Chomsky, "Three models for the description of language," *Information Theory, IEEE Transactions on*, vol. 2, no. 3, pp. 113–124, 1956. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1056813

[5] J. D. U. Hopcroft, John E.; Rajeev Motwani, *Introduction to Automata Theory, Languages, and Computation (2nd ed.)*. Addison-Wesley, 2001.