Syntax and Semantics of V^-

Roger Burtonpatel

October 22, 2023

1 Syntax

We present a grammar of V^-:

Programs	P	$::=$	$\{d\}$	definition
Definitions	d	$::=$	`val` x `=` *ealph*	bind name to expression
Expressions	e_α	$::=$	x, y, z	names
		$\mid$	`if [` g_α $\{[]g_\alpha\}$ `] fi`	if-fi
		$\mid$	$K\{e_\alpha\}$	value constructor application
		$\mid$	$e_{\alpha_1}\ e_{\alpha_2}$	function application
Guarded Expressions	g_α	$::=$	$\rightarrow \alpha$	terminating α
		$\mid$	$e_\alpha;\ g_\alpha$	intermediate expression
		$\mid$	$\exists\{x\}.\, g_\alpha$	existential
		$\mid$	$e_{\alpha_1} = e_{\alpha_2};\ g_\alpha$	equation
Values	v	$::=$	$K(v)$	value constructor application
Value Constructors	K	$::=$	`::`	cons
		$\mid$	`[]`	empty list
		$\mid$	`#`x	name beginning with `#`
		$\mid$	`A-Z`x	name beggining with capital letter
		$\mid$	[-|+](0 − 9)+	signed integer literal

A *name* is any token that is not an integer literal, does not contain whitespace, a bracket, or parenthesis, and is not a value constructor name or a reserved word.

2 Refinement ordering on environments

$$\begin{aligned} \rho \subseteq \rho' \text{ when } & \operatorname{dom}\rho \subseteq \operatorname{dom}\rho' \\ \text{and } & \forall x \in \operatorname{dom}\rho : \rho(x) \subseteq \rho'(x) \end{aligned}$$

3 Forms of Judgement for V^-:

Metavariables	
ϑ	a value produced from evaluating α.
eq	equation
reject	equation rejection
r	$\vartheta \mid$ **reject**: a result of ϑ or rejection
ρ	environment: $name \rightarrow \mathcal{V}_\perp$
$\rho\{x \mapsto y\}$	environment extended with name x mapping to y
$\mathcal{T}$	Context of all temporarily stuck equations (a sequence)
e_α	An expression
g_α	A guarded expression
$\not\in$	Inability to compile to a decision tree; a compile time error

Sequences	
ε	the empty sequence
$S_1 \cdot S_2$	Concatenate sequence S_1 and sequence S_2
$x \cdot S_2$	Cons x onto sequence S_2

Expressions

An expression in core Verse evaluates to produce possibly-empty sequence of values. In V^{-}, values depend on the form of abstract expression α. If α is a Verse-like expression, ϑwill be a value sequence. If it is an ML-like expression, it will be a single value.
A guarded expression evaluates to produce a **result**. A result is either a possibly-empty sequence of values or reject.

$$r ::= \vartheta \mid \textbf{reject}$$

$$\rho; \mathcal{T} \vdash \alpha \Downarrow r \quad (\textsc{Eval-Expr})$$
$$\rho; \mathcal{T} \vdash g_\alpha \Downarrow r \quad (\textsc{Eval-Guarded-Expr})$$

If a guarded expression cannot be evaluated without producing logical variables at runtime, it cannot be expressed as a decision tree. This notation indicates this failure (think of $\rotatebox[origin=c]{90}{ψ}$ as a fallen tree), which results in a compile-time error.

$$\rho; \mathcal{T} \vdash g_\alpha \Downarrow r \leadsto \rotatebox[origin=c]{90}{ψ} \quad (\textsc{NoTree})$$

4 Sequences

The trivial sequence is ε. Sequences can be concatenated with infix $\cdot$. In an appropriate context, a value like x stands for the singleton sequence containing x.

$$\varepsilon \cdot ys \equiv ys$$
$$ys \cdot \varepsilon \equiv ys$$
$$(xs \cdot ys) \cdot zs \equiv xs \cdot (ys \cdot zs)$$

5 Rules (Big-step Operational Semantics) for V^{-}:

Evaluating Guarded Expressions

Evaluating simple parts of guarded expressions

$$(\textsc{Eval-ArrowExpr}) \quad \frac{\rho; \varepsilon \vdash \alpha \Downarrow \vartheta}{\rho; \varepsilon \vdash\!\rightarrow \alpha \Downarrow \vartheta}$$

$$(\textsc{Eval-Exists}) \quad \frac{\rho\{x \mapsto \bot\}; \mathcal{T} \vdash g_\alpha \Downarrow r}{\rho; \mathcal{T} \vdash \exists x.\ g_\alpha \Downarrow r}$$

$$(\textsc{Eval-Expseq}) \quad \frac{\rho;\ \mathcal{T} \vdash ealph \Downarrow \vartheta \qquad \rho;\ \mathcal{T} \vdash g_\alpha \Downarrow r}{\rho; \mathcal{T} \vdash ealph;\ g_\alpha \Downarrow r}$$

Shifting an equation to the context

$$(\textsc{G-Move-To-Ctx}) \quad \frac{\rho; eq \cdot \mathcal{T} \vdash g_\alpha \Downarrow r}{\rho; \mathcal{T} \vdash eq;\ g_\alpha \Downarrow r}$$

Evaluating with different types of equations

$$(\textsc{G-EqExps}) \quad \frac{\begin{array}{c} x,\ y \text{ are distinct and fresh} \\ \rho\{x \mapsto \bot,\ y \mapsto \bot\}; x = e_{\alpha_1} \cdot y = e_{\alpha_2} \cdot x = y \cdot \mathcal{T} \cdot \mathcal{T}' \vdash g_\alpha \Downarrow r \end{array}}{\rho; \mathcal{T} \cdot e_{\alpha_1} = e_{\alpha_2} \cdot \mathcal{T}' \vdash g_\alpha \Downarrow r}$$

$$(\textsc{G-EqNameExp}) \quad \frac{\rho;\ \mathcal{T} \vdash ealph \Downarrow \vartheta \qquad \rho\{x \mapsto \vartheta\}; \mathcal{T} \cdot \mathcal{T}' \vdash g_\alpha \Downarrow r'}{\rho; \mathcal{T} \cdot x = ealph \cdot \mathcal{T}' \vdash g_\alpha \Downarrow r'}$$

$$\text{(G-EqNames-Vals-Succ)}\quad \frac{\begin{array}{c} x,\ y \in \operatorname{dom}\rho \\ \rho(x) = \vartheta,\ \rho(y) = \vartheta \\ \rho; \mathcal{T} \cdot \mathcal{T}' \vdash g_\alpha \Downarrow r \end{array}}{\rho; \mathcal{T} \cdot x = y \cdot \mathcal{T}' \vdash g_\alpha \Downarrow r}$$

$$\text{(G-EqNames-Vals-Fail)}\quad \frac{\begin{array}{c} x,\ y \in \operatorname{dom}\rho \\ \rho(x) = \vartheta,\ \rho(y) = \vartheta' \\ \vartheta \neq \vartheta' \end{array}}{\rho; \mathcal{T} \cdot x = y \cdot \mathcal{T}' \vdash g_\alpha \Downarrow \textbf{reject}}$$

$$\text{(G-EqNames-Bots-Fail)}\quad \frac{\begin{array}{c} x,\ y \in \operatorname{dom}\rho \\ \rho(x) = \bot,\ \rho(y) = \bot \\ x,\ y \text{ do not appear in } \mathcal{T},\ \mathcal{T}' \end{array}}{\rho; \mathcal{T} \cdot x = y \cdot \mathcal{T}' \vdash g_\alpha \Downarrow r \rightsquigarrow \not\in}$$

$$\text{(G-EqNames-BotVal-Succ)}\quad \frac{\begin{array}{c} x,\ y \in \operatorname{dom}\rho \\ \rho(x) = \bot,\ \rho(y) = \vartheta \\ \rho\{x \mapsto \vartheta\}; \mathcal{T} \cdot \mathcal{T}' \vdash g_\alpha \Downarrow r' \end{array}}{\rho; \mathcal{T} \cdot x = y \cdot \mathcal{T}' \vdash g_\alpha \Downarrow r'}$$

$$\text{(G-Vcon-Single-Fail)}\quad \frac{K \neq K'}{\rho; \mathcal{T} \cdot K = K' \cdot \mathcal{T}' \vdash g_\alpha \Downarrow \textbf{reject}}$$

$$\text{(G-Vcon-Single-Succ)}\quad \frac{\rho; \mathcal{T} \cdot \mathcal{T}' \vdash g_\alpha \Downarrow r}{\rho; \mathcal{T} \cdot K = K \cdot \mathcal{T}' \vdash g_\alpha \Downarrow r}$$

$$\text{(G-Vcon-Multi-Fail)}\quad \frac{K \neq K'}{\rho; \mathcal{T} \cdot K(e_{\alpha_1}, \ldots e_{\alpha_n}) = K'(e'_{\alpha_1}, \ldots e'_{\alpha_n}) \cdot \mathcal{T}' \vdash g_\alpha \Downarrow \textbf{reject}}$$

$$\text{(G-Vcon-Multi-Arity-Fail)}\quad \frac{n \neq m}{\rho; \mathcal{T} \cdot K(e_{\alpha_1}, \ldots e_{\alpha_n}) = K(e'_{\alpha_1}, \ldots e'_{\alpha_m}) \cdot \mathcal{T}' \vdash g_\alpha \Downarrow \textbf{reject}}$$

$$\text{(G-Vcon-Multi-Succ)}\quad \frac{\rho; [e_{\alpha_i} = e'_{\alpha_i} \mid 1 \leq i \leq n] \cdot \mathcal{T} \cdot \mathcal{T}' \vdash g_\alpha \Downarrow r}{\rho; \mathcal{T} \cdot K(e_{\alpha_1}, \ldots e_{\alpha_n}) = K(e'_{\alpha_1}, \ldots e'_{\alpha_n}) \cdot \mathcal{T}' \vdash g_\alpha \Downarrow r}$$

Evaluating General Expressions

$$\text{(If-Fi-Success)}\quad \frac{\rho; \mathcal{T} \vdash g_\alpha \Downarrow r \Downarrow \vartheta}{\rho;\ \mathcal{T} \vdash \text{IF } [\ g_\alpha \ \Box\ \ldots\ ]\ \text{FI} \Downarrow \vartheta}$$

$$\text{(If-Fi-Reject)}\quad \frac{\rho; \mathcal{T} \vdash g_\alpha \Downarrow \textbf{reject} \qquad \rho;\ \mathcal{T} \vdash \text{IF } [\ \ldots\ ]\ \text{FI} \Downarrow \vartheta}{\rho;\ \mathcal{T} \vdash \text{IF } [\ g_\alpha \ \Box\ \ldots\ ]\ \text{FI} \Downarrow \vartheta}$$

$$\text{(Vcon-Empty)}\quad \frac{}{\rho;\ \mathcal{T} \vdash K \Downarrow K}$$

$$\text{(Vcon-Multi)}\quad \frac{\rho;\ \mathcal{T} \vdash e_{\alpha_i} \Downarrow \vartheta_i \qquad 1 \leq i \leq n}{\rho;\ \mathcal{T} \vdash K(e_{\alpha_1}, \ldots e_{\alpha_n}) \Downarrow K(\vartheta_1, \ldots \vartheta_i)}$$