Syntax and Semantics of V^-

Roger Burtonpatel

October 22, 2023

1 Syntax

NR: I recommend macros for optional and sequence, not literal brackets.
We present a grammar of V^-:

⟨*program*⟩ ::= {⟨*def*⟩}

⟨*def*⟩ ::= `val` ⟨*name*⟩ ⟨*exp*⟩
| ⟨*exp*⟩

⟨*name*⟩ ::= any token that is not an *int-lit*, does not contain whitespace, and is not a ⟨*value-constructor-name*⟩ or a reserved word.

⟨*value-constructor-name*⟩ ::= `cons` | any token that begins with a capital letter or a colon

⟨*exp*⟩ ::= ⟨*integer-literal*⟩
| ⟨*name*⟩
| ⟨*guarded-if*⟩
| ⟨*lambda*⟩

⟨*integer-literal*⟩ ::= token composed only of digits, possibly prefixed with a + or −.

⟨*lambda*⟩ ::= λ{⟨*name*⟩}`.` ⟨*exp*⟩

⟨*guarded-if*⟩ ::= `if` [⟨*guarded-exp*⟩ { [] ⟨*guarded-exp*⟩ }] `fi`

⟨*guarded-exp*⟩ ::= →⟨*exp*⟩
| `E` {⟨*logical-var*⟩}`.` ⟨*guarded-exp*⟩
| ⟨*exp*⟩`;` ⟨*guarded-exp*⟩
| ⟨*logical-var*⟩ = ⟨*exp*⟩`;` ⟨*guarded-exp*⟩
| ⟨*exp*⟩ = ⟨*exp*⟩`;` ⟨*guarded-exp*⟩
| ⟨*guarded-exp*⟩ | ⟨*guarded-exp*⟩
| `one`({⟨*guarded-exp*⟩})
| `all`({⟨*guarded-exp*⟩})

⟨*logical-var*⟩ ::= a fresh name (cannot be `lam`- or `E`-bound in this scope).

NR: I'm skeptical about some of the forms of guarded expression:

- Choice doesn't look flexible enough, e.g., to express this:

$$\exists x.(x < 0 | x > 0) \to 17$$

- Unconvinced about `one` and `all` here. Let's see some examples.

Add patterns, value constructors (application), choice, one, all.

2 Refinement ordering on environments

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

3 Forms of Judgement for V^-:

Metavariables	
$v,\ v'$	value
eq	equation
`?t`	a temporarily-stuck equation *NR: Needs to record* what *is stuck*
fail	failure
$\rho,\ \hat{\rho}$	environment: $name \to \mathcal{V}_\perp$
$\mathcal{T}$	Context of all temporarily stuck equations
e	An expression
ge	A guarded expression

Equations

An *equation* is either solved to produce bindings that extend an environemnt, gets temporarily stuck, or fails.

$$\langle \rho, eq \rangle \rightarrowtail \hat{\rho} \quad \text{(EQUATIONSUCCESS)}$$

$$\langle \rho, eq \rangle \rightarrowtail \texttt{?t} \quad \text{(EQUATIONTEMPSTUCK)}$$

$$\langle \rho, eq \rangle \rightarrowtail \textbf{fail} \quad \text{(EQUATIONFAIL)}$$

Success only refines the environment; that is, when $\langle \rho, e \rangle \rightarrowtail \rho'$, we expect $\rho \subseteq \rho'$.

Expressions

An expression either evaluates to produce a value or fails. *NR: Don't we want a* sequence *of values in the success case? And if we have a sequence, do we need* ***fail****? Or will the empty sequence suffice?*

$$\langle \rho, e \rangle \Downarrow v \quad \text{(EVALSUCC)}$$

$$\langle \rho, e \rangle \Downarrow \textbf{fail} \quad \text{(EVALFAIL)}$$

$$\langle \rho, \mathcal{T}, ge \rangle \Downarrow v \quad \text{(EVALGESUCC)}$$

$$\langle \rho, \mathcal{T}, ge \rangle \Downarrow \textbf{fail} \quad \text{(EVALGEFAIL)}$$

Temporarily stuck equations

$$\langle \rho, \texttt{?t}; \mathcal{T} \rangle \rightarrowtail \langle \hat{\rho}, \mathcal{T} \rangle \quad \text{(CTXTORHO)}$$

NR: I don't buy it. The token `?t` *won't be enough to enable a rule to compute* $\hat{\rho}$*.*

4 Lists

Ideas:

- Use $\cdot$ as an infix, associative concatenation operator.
- Extend it to single values in the customary way.
- Write the empty sequence as ε.

NR: Could it be worth writing laws here?

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

NR: Semicolon for list concatenation is not going to work here. I suggest concatenating with a small dot, as in "$eq \cdot \mathcal{T}$." As the notation stands, it makes my brain hurt. I'm not going to try to follow it.

$$\text{(GE-CTX-STUCK)}\ \frac{\langle \rho, eq\rangle \rightarrowtail \texttt{?t} \qquad \langle \rho, \texttt{?t}; \mathcal{T}\rangle \rightarrowtail \langle \hat{\rho}, \mathcal{T}\rangle \qquad \langle \rho, \mathcal{T}, ge\rangle \Downarrow v}{\langle \rho, eq; \mathcal{T}, ge\rangle \Downarrow v}$$

NR: The notation below has multiple horizontal lines. That makes it a derivation, not a rule. I'm having trouble figuring out what's being said here.

$$\text{(GE-EQ-SUCC)}\ \frac{\dfrac{\langle \rho, eq\rangle \rightarrowtail \texttt{?t} \qquad \langle \rho, \texttt{?t}; \mathcal{T}\rangle \rightarrowtail \langle \hat{\rho}, \mathcal{T}\rangle \qquad \langle \rho, \mathcal{T}, ge\rangle \Downarrow v}{\langle \rho, eq; \mathcal{T}, ge\rangle \Downarrow v}\ \text{(GE-CTX-STUCK)}}{\langle \rho, \mathcal{T}, eq; ge\rangle \Downarrow v}$$

NR: If you're trying to pluck an equation out of a list of things, try "$\mathcal{T} \cdot eq \cdot \mathcal{T}'$."

$$\text{(GE-CTX-SUCC)}\ \frac{\langle \rho, eq\rangle \rightarrowtail \hat{\rho} \qquad \langle \hat{\rho}, \mathcal{T}, ge\rangle \Downarrow v}{\langle \rho, eq; \mathcal{T}, ge\rangle \Downarrow v}$$

$$\begin{array}{l}\text{(GE-CTX-SUCC)}\\ \text{(GE-EQ-SUCC)}\end{array}\ \frac{\dfrac{\langle \rho, eq\rangle \rightarrowtail \hat{\rho} \qquad \langle \hat{\rho}, \mathcal{T}, ge\rangle \Downarrow v}{\langle \rho, eq; \mathcal{T}, ge\rangle \Downarrow v}\ [Right]}{\langle \rho, \mathcal{T}, eq; ge\rangle \Downarrow v}$$

$$\text{(GE-CTX-FAIL)}\ \frac{\langle \rho, eq\rangle \rightarrowtail \textbf{fail}}{\langle \rho, eq; \mathcal{T}, ge\rangle \Downarrow \textbf{fail}}$$

$$\text{(GE-EQ-FAIL)}\ \frac{\dfrac{\langle \rho, eq\rangle \rightarrowtail \textbf{fail}}{\langle \rho, eq; \mathcal{T}, ge\rangle \Downarrow \textbf{fail}}\ \text{(GE-CTX-FAIL)}}{\langle \rho, \mathcal{T}, eq; ge\rangle \Downarrow \textbf{fail}}$$

$$\text{(GE-EXP-FAIL)}\ \frac{\langle \rho, e\rangle \Downarrow \textbf{fail}}{\langle \rho, \mathcal{T}, e; ge\rangle \Downarrow \textbf{fail}}$$

$$\text{(GE-EQ-SUCC)}\ \frac{\langle \rho, e\rangle \Downarrow v' \qquad \langle \rho, \mathcal{T}, ge\rangle \Downarrow v}{\langle \rho, \mathcal{T}, e; ge\rangle \Downarrow v}$$