Proiectarea Semanticii Limbajului JavaScript-PLP

Secțiunea C - Documentație Tehnică

December 30, 2025

1 Modelul Semanticii Operaționale

Semantica limbajului este definită utilizând **Semantica Operațională Big-Step**. Deoarece în acest limbaj atribuirea este o expresie, evaluarea oricărei expresii poate modifica starea programului [cite: 730, 749-750].

1.1 Starea Programului (Environment)

Starea programului, notată cu σ, este o funcție care mapează identificatorii la valorile lor:

$$\sigma \in \text{Env} = \text{Var} \to \text{Value} \cup \{\perp\}$$

unde Value = {Int, Float, Bool, Char, String, Unit}[cite: 729].

2 Semantica Expresiilor

Evaluarea unei expresii e în starea σ produce o valoare v și o stare nouă σ' (datorită posibilelor atribuiri):

$$\langle e, \sigma \rangle \Downarrow \langle v, \sigma' \rangle$$

2.1 Reguli pentru Expresii

- **Constante:** $\langle c, \sigma \rangle \Downarrow \langle c, \sigma \rangle$
- **Variabile:** $\frac{\sigma(x)=v}{\langle x,\sigma\rangle \Downarrow \langle v,\sigma\rangle}$
- **Atribuire:** $\frac{\langle e,\sigma\rangle \Downarrow \langle v,\sigma'\rangle}{\langle x=e,\sigma\rangle \Downarrow \langle v,\sigma'[x \leftarrow v]\rangle}$ (returnează valoarea și actualizează starea)[cite: 731, 750].
- **Operații Binare:** $\frac{\langle e_1,\sigma\rangle \Downarrow \langle v_1,\sigma''\rangle \quad \langle e_2,\sigma''\rangle \Downarrow \langle v_2,\sigma'\rangle \quad v=v_1 \oplus v_2}{\langle e_1 \oplus e_2,\sigma\rangle \Downarrow \langle v,\sigma'\rangle}$
  Notă: e_2 se evaluează în starea σ'' produsă de e_1.[cite: 730].
- **Operații Unare (NOT, Neg):** $\frac{\langle e,\sigma\rangle \Downarrow \langle v_1,\sigma'\rangle \quad v=\ominus v_1}{\langle \ominus e,\sigma\rangle \Downarrow \langle v,\sigma'\rangle}$ [cite: 764-765].

3 Semantica Instrucțiunilor

Instrucțiunile transformă starea σ. Introducem o stare specială $Exit(v)$ pentru a gestiona instrucțiunea `return` [cite: 733-734].

3.1 Reguli de Control

- **Skip:** $\langle ;, \sigma \rangle \to \sigma$[cite: 773].
- **Declarație (Let):** $\frac{\langle e,\sigma\rangle \Downarrow \langle v,\sigma'\rangle}{\langle \text{let } x=e,\sigma\rangle \to \sigma'[x \leftarrow v]}$ sau $\langle \text{let } x, \sigma \rangle \to \sigma[x \leftarrow \text{Unit}]$ [cite: 767-768].
- **Secvențiere (Bloc):** $\frac{\langle s_1,\sigma\rangle \to \sigma'' \quad \sigma'' \text{ nu este } Exit \quad \langle s_2,\sigma''\rangle \to \sigma'}{\langle \{s_1,s_2\},\sigma\rangle \to \sigma'}$
- **Condițional (If-Else):**

$$\frac{\langle e, \sigma \rangle \Downarrow \langle \text{true}, \sigma'' \rangle \quad \langle s_1, \sigma'' \rangle \to \sigma'}{\langle \text{if } (e) \ s_1 \text{ else } s_2, \sigma \rangle \to \sigma'} \qquad \frac{\langle e, \sigma \rangle \Downarrow \langle \text{false}, \sigma'' \rangle \quad \langle s_2, \sigma'' \rangle \to \sigma'}{\langle \text{if } (e) \ s_1 \text{ else } s_2, \sigma \rangle \to \sigma'}$$

  [cite: 733, 770].

- **Repetiție (While):**

$$\frac{\langle e, \sigma \rangle \Downarrow \langle \text{false}, \sigma' \rangle}{\langle \text{while } (e) \ s, \sigma \rangle \to \sigma'} \qquad \frac{\langle e, \sigma \rangle \Downarrow \langle \text{true}, \sigma'' \rangle \quad \langle s, \sigma'' \rangle \to \sigma''' \quad \langle \text{while } (e) \ s, \sigma''' \rangle \to \sigma'}{\langle \text{while } (e) \ s, \sigma \rangle \to \sigma'}$$

  [cite: 733, 772].

4 Semantica Instrucțiunii Return

Instrucțiunea `return` oprește execuția imediată a blocului curent și propagă valoarea[cite: 734, 766]:

$$\frac{\langle e, \sigma \rangle \Downarrow \langle v, \sigma' \rangle}{\langle \text{return } e, \sigma \rangle \to Exit(v)}$$

Dacă un bloc întâlnește o stare $Exit(v)$, toate instrucțiunile următoare sunt sărite.

5 Sistemul de Tipizare

Limbajul utilizează **Strong Typing la runtime**. Orice operație între tipuri incompatibile (ex: adunarea unui `Int` cu un `String`) va opri execuția cu o eroare de tip `RuntimeError`, fără a încerca conversia automată a valorilor[cite: 735].