

Compilers

Author: Daria Shutina

Compilers

23-02-23

Interpreters

[A simple interpreter](#)

[A smarter interpreter](#)

[Example](#)

23-02-23

There are several ways to define semantics of the language:

- write a document with standards
- Write an interpreter and a compiler
- Use formal semantics. You have a start state, a finish state and a transition function

Let us define an expression E which is a variable, an integer or a binary operation:

$E := X \mid N \mid Binop \times E E$, where $X = \{a, b, c, \dots\}$, $\times = \{+, -, *, \backslash, \%, <, \dots\}$.

We can see any expression as a binary tree.

The semantics of the language is a total map:

$$\underbrace{[\mid \cdot \mid]}_{\text{semantics}} : \underbrace{E}_{\text{expressions}} \rightarrow \underbrace{D}_{\text{semantics domain}}$$

$D = (X \rightarrow Z) \rightarrow Z$ -- we got a variable, turn it into an integer, then turn the integer into the result integer.

Interpreters

We have the input and the output. An interpreter takes a program and its input as arguments, and returns what the program would return.

A simple interpreter

$$\sigma \in E$$

- $n \in N; \sigma \xrightarrow{n} n$
- $x \in X; \sigma \xrightarrow{x} \sigma(x)$
- $\sigma \xrightarrow{l \times_1 r} x \times_2 y$

A smarter interpreter

$$S = X \mid N \mid read \mid write \mid skip \mid Binop \mid \underbrace{S_1; S_2}_{\text{composition of expressions}}$$

In this case, `;` is a concatenation operator

$$[\mid \cdot \mid]_s : S \rightarrow D$$

The simplest idea for the output is a set of numbers.

- $c \xrightarrow{skip} c$
- $(\sigma, w) \xrightarrow{x := e} (\sigma[x \leftarrow [e]_\sigma], w)$
- $(\sigma, (i : is : o)) \xrightarrow{read(x)} (\sigma[x \leftarrow i], (is, o))$ ($i : is : o$ – i is head, is is tail)

We take the first number and assign it into x (that is what σ does).

- $(\sigma, (i, o)) \xrightarrow{write(e)} (\sigma, (i, o + +[e]_\sigma))$
- $\left. \begin{array}{l} c \xrightarrow{s_1} c' \\ c' \xrightarrow{s_2} c'' \end{array} \right\} \Rightarrow c \xrightarrow{s_1; s_2} c''$

Example

The diagram illustrates the execution of a program with memory and stack states. It shows the following steps:

- Initial state: $(\sigma, (\epsilon, \epsilon))$ where σ is a memory state with $x \mapsto 1$ and $y \mapsto 2$. The stack is empty.
- Execution of $z := x + y$: The stack becomes (ϵ, ϵ) with $z \mapsto 3$.
- Execution of $read(y)$: The stack becomes $(2, \epsilon)$ with $y \mapsto 2$.
- Execution of $S_4 = read(x)$: The stack becomes $(1, 2, \epsilon)$ with $x \mapsto 1$.
- Execution of $S_3; S_4$: The stack becomes $(1, 2, \epsilon)$ with $x \mapsto 1$.
- Execution of $S_1; S_2$: The stack becomes $(1, 2, \epsilon)$ with $x \mapsto 1$.