

指称语义

1 一个极简的指令式程序语言：SimpleWhile

下面是 SimpleWhile 语言中程序表达式的语法。

```
EI :: = N | V | EI + EI | EI - EI | EI * EI
EB :: = TRUE | FALSE | EI < EI | EB && EB | ! EB
```

SimpleWhile 语言的程序语句包括空语句、变量赋值语句、顺序执行、if 语句与 while 语句。

```
C :: = SKIP |
      V = EI |
      C; C |
      if (EB) then { C } else { C } |
      while (EB) do { C }
```

2 SimpleWhile 整数类型表达式的指称语义

在极简的 SimpleWhile 语言中，整数类型表达式中只有整数常量、变量、加法、减法与乘法运算。

```
EI :: = N | V | EI + EI | EI - EI | EI * EI
```

我们约定其中整数变量的值、整数运算的结果都是没有范围限制的。基于这一约定，我们可以如下定义程序状态集合：

$$\text{state} \triangleq \text{var_name} \rightarrow \mathbb{Z}$$

进一步，整数类型表达式 e 的行为可以被定义为 e 在每个程序状态上的值。

$\llbracket e \rrbracket : \text{state} \rightarrow \mathbb{Z}$ 是一个程序状态到整数的函数；
 $\llbracket e \rrbracket(s)$ 表示表达式 e 在程序状态 s 上的求值结果。

基于这一设定，可以写出下面的具体定义：

- $\llbracket n \rrbracket(s) = n$
- $\llbracket x \rrbracket(s) = s(x)$
- $\llbracket e_1 + e_2 \rrbracket(s) = \llbracket e_1 \rrbracket(s) + \llbracket e_2 \rrbracket(s)$
- $\llbracket e_1 - e_2 \rrbracket(s) = \llbracket e_1 \rrbracket(s) - \llbracket e_2 \rrbracket(s)$
- $\llbracket e_1 * e_2 \rrbracket(s) = \llbracket e_1 \rrbracket(s) * \llbracket e_2 \rrbracket(s)$

其中 $s \in \text{state}$ 。

3 行为等价

基于整数类型表达式的语义定义 `eval_expr_int`，我们可以定义整数类型表达式之间的行为等价（亦称语义等价）：两个表达式 `e1` 与 `e2` 是等价的当且仅当它们在任何程序状态上的求值结果都相同。

$$e_1 \equiv e_2 \quad \text{iff.} \quad \forall s. \llbracket e_1 \rrbracket(s) = \llbracket e_2 \rrbracket(s)$$

下面罗列的都是整数类型表达式语义等价的例子。

对于任意整数类型表达式 e

- $0 + e \equiv e$
- $e + 0 \equiv e$
- $e - 0 \equiv e$
- $e * 0 \equiv 0$
- $0 * e \equiv 0$

对于任意整数 n, m, r

- 如果 $n + m = r$ ，那么 $n + m \equiv r$
- 如果 $n - m = r$ ，那么 $n - m \equiv r$
- 如果 $n * m = r$ ，那么 $n * m \equiv r$

对于任意整数类型表达式 e_1, e_2, e_3

- $(e_1 + e_2) + e_3 \equiv e_1 + (e_2 + e_3)$
- $(e_1 + e_2) - e_3 \equiv e_1 + (e_2 - e_3)$
- $(e_1 - e_2) + e_3 \equiv e_1 - (e_2 - e_3)$
- $(e_1 - e_2) - e_3 \equiv e_1 - (e_2 + e_3)$

4 行为等价的性质

整数类型表达式之间的行为等价符合下面几条重要的代数性质。

- 整数类型表达式行为等价“ $\equiv$ ”具有自反性

$$\llbracket e \rrbracket(s) = \llbracket e \rrbracket(s)$$

- 整数类型表达式行为等价“ $\equiv$ ”具有对称性；

$$\llbracket e_1 \rrbracket(s) = \llbracket e_2 \rrbracket(s) \Rightarrow \llbracket e_2 \rrbracket(s) = \llbracket e_1 \rrbracket(s)$$

- 整数类型表达式行为等价“ $\equiv$ ”具有传递性；

$$\llbracket e_1 \rrbracket(s) = \llbracket e_2 \rrbracket(s), \llbracket e_2 \rrbracket(s) = \llbracket e_3 \rrbracket(s) \Rightarrow \llbracket e_1 \rrbracket(s) = \llbracket e_3 \rrbracket(s)$$

- 整数类型表达式行为等价“ $\equiv$ ”是一个等价关系。

Congruence 性质

- 如果 $e_1 \equiv e'_1$ 并且 $e_2 \equiv e'_2$, 那么 $e_1 + e_2 \equiv e'_1 + e'_2$ 。
- 如果 $e_1 \equiv e'_1$ 并且 $e_2 \equiv e'_2$, 那么 $e_1 - e_2 \equiv e'_1 - e'_2$ 。
- 如果 $e_1 \equiv e'_1$ 并且 $e_2 \equiv e'_2$, 那么 $e_1 * e_2 \equiv e'_1 * e'_2$ 。

利用这些代数性质, 我们可以简明证明以下命题。

应用: 对于任意整数类型表达式 e ,

$$e + 1 + 2 \equiv e + (1 + 2) \equiv e + 3$$

5 布尔表达式的指称语义

$$\llbracket e \rrbracket \subseteq \text{state}$$

$s \in \llbracket e \rrbracket$ 当且仅当 e 在状态 s 下求值为真。

- $\llbracket \text{TRUE} \rrbracket = \text{state}$
- $\llbracket \text{FALSE} \rrbracket = \emptyset$
- $s \in \llbracket e_1 < e_2 \rrbracket$ 当且仅当 $\llbracket e_1 \rrbracket(s) < \llbracket e_2 \rrbracket(s)$
- $\llbracket e_1 \&\&e_2 \rrbracket = \llbracket e_1 \rrbracket \cap \llbracket e_2 \rrbracket$
- $\llbracket !e_1 \rrbracket = \text{state} \setminus \llbracket e_1 \rrbracket$

布尔表达式行为等价:

$$e_1 \equiv e_2 \quad \text{iff.} \quad \forall s. s \in \llbracket e_1 \rrbracket \quad \text{iff.} \quad s \in \llbracket e_2 \rrbracket$$

6 程序语句的指称语义定义

$$\llbracket c \rrbracket \subseteq \text{state} \times \text{state}$$

$(s_1, s_2) \in \llbracket c \rrbracket$ 当且仅当从 s_1 状态开始执行程序 c 会以程序状态 s_2 终止。

6.1 赋值语句

$$\llbracket x = e \rrbracket = \{(s_1, s_2) \mid s_2(x) = \llbracket e \rrbracket(s_1), \text{ for any } y \in \text{var_name}, \text{ if } x \neq y, s_1(y) = s_2(y)\}$$

6.2 空语句

$$\llbracket \text{skip} \rrbracket = \{(s, s) \mid s \in \text{state}\}$$

6.3 顺序执行语句

$$\llbracket c_1; c_2 \rrbracket = \llbracket c_1 \rrbracket \circ \llbracket c_2 \rrbracket = \{(s_1, s_3) \mid (s_1, s_2) \in \llbracket c_1 \rrbracket, (s_2, s_3) \in \llbracket c_2 \rrbracket\}$$

6.4 条件分支语句

定义 1:

$$\begin{aligned} \llbracket \text{if } (e) \text{ then } \{c_1\} \text{ else } \{c_2\} \rrbracket = & (\{(s_1, s_2) \mid s_1 \in \llbracket e \rrbracket\} \cap \llbracket c_1 \rrbracket) \cup \\ & (\{(s_1, s_2) \mid s_1 \notin \llbracket e \rrbracket\} \cap \llbracket c_2 \rrbracket) \end{aligned}$$

定义 2:

$$\llbracket \text{if } (e) \text{ then } \{c_1\} \text{ else } \{c_2\} \rrbracket = \text{test_true}(\llbracket e \rrbracket) \circ \llbracket c_1 \rrbracket \cup \text{test_false}(\llbracket e \rrbracket) \circ \llbracket c_2 \rrbracket$$

其中,

$$\begin{aligned} \text{test_true}(\llbracket e \rrbracket) &= \{(s_1, s_2) \mid s_1 \in \llbracket e \rrbracket, s_1 = s_2\} \\ \text{test_false}(\llbracket e \rrbracket) &= \{(s_1, s_2) \mid s_1 \notin \llbracket e \rrbracket, s_1 = s_2\}. \end{aligned}$$

6.5 循环语句语句

定义 1:

$$\begin{aligned} \text{iterLB}_0(\llbracket e \rrbracket, \llbracket c \rrbracket) &= \text{test_false}(\llbracket e \rrbracket); \\ \text{iterLB}_{n+1}(\llbracket e \rrbracket, \llbracket c \rrbracket) &= \text{test_true}(\llbracket e \rrbracket) \circ \llbracket c \rrbracket \circ \text{iterLB}_n(\llbracket e \rrbracket, \llbracket c \rrbracket); \\ \llbracket \text{while } (e) \text{ do } \{c\} \rrbracket &= \bigcup_{n \in \mathbb{N}} \text{iterLB}_n(\llbracket e \rrbracket, \llbracket c \rrbracket). \end{aligned}$$

定义 2:

$$\begin{aligned} \text{boundedLB}_0(\llbracket e \rrbracket, \llbracket c \rrbracket) &= \emptyset \\ \text{boundedLB}_{n+1}(\llbracket e \rrbracket, \llbracket c \rrbracket) &= \text{test_true}(\llbracket e \rrbracket) \circ \llbracket c \rrbracket \circ \text{boundedLB}_n(\llbracket e \rrbracket, \llbracket c \rrbracket) \cup \text{test_false}(\llbracket e \rrbracket) \\ \llbracket \text{while } (e) \text{ do } \{c\} \rrbracket &= \bigcup_{n \in \mathbb{N}} \text{boundedLB}_n(\llbracket e \rrbracket, \llbracket c \rrbracket) \end{aligned}$$