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：

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

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。

$[\![e]\!] : \text{state} \to \mathbb{Z}$ 是一个程序状态到整数的函数；
$[\![e]\!](s)$ 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：

- $[\![n]\!](s) = n$
- $[\![x]\!](s) = s(x)$
- $[\![e_1 + e_2]\!](s) = [\![e_1]\!](s) + [\![e_2]\!](s)$
- $[\![e_1 - e_2]\!](s) = [\![e_1]\!](s) - [\![e_2]\!](s)$
- $[\![e_1 * e_2]\!](s) = [\![e_1]\!](s) * [\![e_2]\!](s)$

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。

$$e_1 \equiv e_2 \quad \text{iff.} \quad \forall s.\ [\![e_1]\!](s) = [\![e_2]\!](s)$$

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

$$[\![e]\!](s) = [\![e]\!](s)$$

- 整数类型表达式行为等价“$\equiv$”具有对称性；

$$[\![e_1]\!](s) = [\![e_2]\!](s) \Rightarrow [\![e_2]\!](s) = [\![e_1]\!](s)$$

- 整数类型表达式行为等价“$\equiv$”具有传递性；

$$[\![e_1]\!](s) = [\![e_2]\!](s), [\![e_2]\!](s) = [\![e_3]\!](s) \Rightarrow [\![e_1]\!](s) = [\![e_3]\!](s)$$

- 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

$[\![e]\!] \subseteq \text{state}$
$s \in [\![e]\!]$ 当且仅当 e 在状态 s 下求值为真。

- $[\![\text{TRUE}]\!] = \text{state}$
- $[\![\text{FALSE}]\!] = \varnothing$
- $s \in [\![e_1 < e_2]\!]$ 当且仅当 $[\![e_1]\!](s) < [\![e_2]\!](s)$
- $[\![e_1 \&\& e_2]\!] = [\![e_1]\!] \cap [\![e_2]\!]$
- $[\![!e_1]\!] = \text{state} \backslash [\![e_1]\!]$

布尔表达式行为等价：

$$e_1 \equiv e_2 \quad \text{iff.} \quad \forall s.\ s \in [\![e_1]\!] \ \text{iff.}\ s \in [\![e_2]\!]$$

6 程序语句的指称语义定义

$[\![c]\!] \subseteq \text{state} \times \text{state}$
$(s_1, s_2) \in [\![c]\!]$ 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

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

6.2 空语句

$$[\![\text{skip}]\!] = \{(s, s) \mid s \in \text{state}\}$$

6.3 顺序执行语句

$$[\![c_1; c_2]\!] = [\![c_1]\!] \circ [\![c_2]\!] = \{(s_1, s_3) \mid (s_1, s_2) \in [\![c_1]\!], (s_2, s_3) \in [\![c_2]\!]\}$$

6.4 条件分支语句

定义 1：

$$\begin{aligned}[\![\text{if } (e) \text{ then } \{c_1\} \text{ else } \{c_2\}]\!] = \quad & \big(\{(s_1, s_2) \mid s_1 \in [\![e]\!]\} \cap [\![c_1]\!]\big) \cup \\ & \big(\{(s_1, s_2) \mid s_1 \notin [\![e]\!]\} \cap [\![c_2]\!]\big)\end{aligned}$$

定义 2：

$$[\![\text{if } (e) \text{ then } \{c_1\} \text{ else } \{c_2\}]\!] = \text{test_true}([\![e]\!]) \circ [\![c_1]\!] \cup \text{test_false}([\![e]\!]) \circ [\![c_2]\!]$$

其中，

$$\text{test_true}([\![e]\!]) = \{(s_1, s_2) \mid s_1 \in [\![e]\!], s_1 = s_2\}$$
$$\text{test_false}([\![e]\!]) = \{(s_1, s_2) \mid s_1 \notin [\![e]\!], s_1 = s_2\}。$$

6.5 循环语句语句

定义 1：

$$\text{iterLB}_0([\![e]\!], [\![c]\!]) = \text{test_false}([\![e]\!]);$$
$$\text{iterLB}_{n+1}([\![e]\!], [\![c]\!]) = \text{test_true}([\![e]\!]) \circ [\![c]\!] \circ \text{iterLB}_n([\![e]\!], [\![c]\!]);$$
$$[\![\text{while } (e) \text{ do } \{c\}]\!] = \bigcup_{n\in\mathbb{N}} \text{iterLB}_n([\![e]\!], [\![c]\!])。$$

定义 2：

$$\text{boundedLB}_0([\![e]\!], [\![c]\!]) = \varnothing$$
$$\text{boundedLB}_{n+1}([\![e]\!], [\![c]\!]) = \text{test_true}([\![e]\!]) \circ [\![c]\!] \circ \text{boundedLB}_n([\![e]\!], [\![c]\!]) \cup \text{test_false}([\![e]\!])$$
$$[\![\text{while } (e) \text{ do } \{c\}]\!] = \bigcup_{n\in\mathbb{N}} \text{boundedLB}_n([\![e]\!], [\![c]\!])$$