Семантика языков программирования

Определение языка программирования должно иметь как минимум две части: синтаксис и семантику.

✔ Синтаксис задаётся формально контекстно – свободными грамматиками.

✔ Семантика чаще всего определяется неформально, например смысл оператора

while B do C

объясняют так: «Для вычисления этого оператора нужно вычислять оператор C до тех пор, пока значение выражения B истинно».

✔ В этом курсе мы рассмотрим методы формального задания семантики языков программирования.

Зачем нужна формальная семантика?

- ✔Чтобы точно знать возможности языка программирования.
- ✔ Чтобы доказывать корректность программы, а не экспериментировать с компилятором.
- ✔ Чтобы убедиться, что компилятор работает корректно.
- ✔ Для облегчения переносимости компилятора на различные платформы.

Эквивалентные преобразования программы

Зная, что

if true then C1 else C2

делает тоже самое, что и

C1

можно упростить программу.

Используя формальную семантику можно доказывать эквивалентность и более сложных фрагментов программы.

Эквивалентны ли фрагменты программы?

```
begin
   C1 ;
   if B
    then C2
    else C3
end
```

```
if B
  then
    begin
      C1 ; C2
    end
  else
    begin
      C1 ; C3
    end
```

А эти?

```
begin
  if B
    then C2
    else C3 ;
  C1
end
```

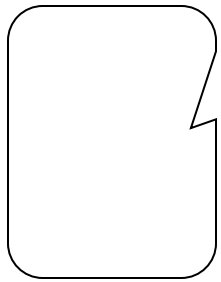
```
if B
  then
    begin
      C2 ; C1
    end
  else
    begin
      C3 ; C1
    end
```

Абстрактный синтаксис языка арифметических выражений

1) Синтаксические категории

Типичные предста - вители

e ∈ Exp
op ∈ Op
n ∈ Num



Синтакси - ческие категории

2) Определения

op ::= + | - | * | div
e ::= n | e′ op e″

Методы определения семантики

- Конкретная операционная семантика
- Естественная семантика
- Вычислительная (структурно – операционная) семантика
- Денотационная семантика

Конкретнтая операционная семантика языка Exp

```
topostfix(N,S,[N|S]) :- number(N).
topostfix(E,S,R) :-
        E =.. [Op,A,B],
        member(Op,[+,-,*,/]),
        topostfix(A,[Op|S],S1),
        topostfix(B,S1,R).

calc([],[R],R).
calc([N|Cs],S,R) :-
        number(N),
        calc(Cs,[N|S],R).
calc([Op|Cs],[N1,N2|S],R) :-
        member(Op,[+,-,*,/]),
        E =.. [Op,N1,N2],
        N is E,
        calc(Cs,[N|S],R).
```

Естественная семантика

Это аксиоматическая система, определяющая смысл каждой конструкции языка в виде вычисляемого ею значения.

Определим семантику языка арифметических выражений.

Для этого понадобится отношение

$$\Rightarrow : \mathbf{Exp} \rightarrow \mathbf{Num},$$

отображающее множество арифметических выражений на множество чисел.

Оно определяется индуктивно:

Правило 1: Для каждой числовой константы **n**, $\mathbf{n} \Rightarrow \mathbf{n}$.

Правило 2: Если $\mathbf{e} \Rightarrow \mathbf{v}$ и $\mathbf{e'} \Rightarrow \mathbf{v'}$, то $\mathbf{e\ op\ e'} \Rightarrow \mathbf{Ap\ (op,\ v,\ v')}$.

Правила, определяющие естественную семантику языка арифметических выражений

- ✔ Правило CR

$$\frac{}{n \Rightarrow n}$$

- ✔ Правило OpR

$$\frac{e \Rightarrow v \quad e' \Rightarrow v'}{e\ op\ e' \Rightarrow Ap(op, v, v')}$$

Вычисление значений арифметических выражений

Пусть нужно вычислить значение выражения

$$3 * 4 + 8 \text{ div } (4 - 2).$$

Это сумма, и применение правила OpR приведёт к

$$\frac{3 * 4 \Rightarrow v \qquad 8 \text{ div } (4 - 2) \Rightarrow v'}{3 * 4 + 8 \text{ div } (4 - 2) \Rightarrow \text{Ap}(+_{\text{NUM}}, v, v')}$$

Для вычисления применим ещё два раза правила OpR:

$$\frac{3 \Rightarrow v' \qquad 4 \Rightarrow v''}{3 * 4 \Rightarrow \text{Ap}(*_{\text{NUM}}, v', v'')}$$

$$\frac{8 \Rightarrow v' \qquad (4 - 2) \Rightarrow v''}{8 \text{ div } (4 - 2) \Rightarrow \text{Ap}(/_{\text{NUM}}, v', v'')}$$

Вычисление значения арифметических выражений (продолжение)

В конце концов, получив численную константу применим правила CR:

$$\frac{}{3 \Rightarrow 3} \qquad \frac{}{4 \Rightarrow 4} \qquad \frac{}{8 \Rightarrow 8} \qquad \frac{}{2 \Rightarrow 2}$$

Выполнив подстановку значений промежуточным переменным, получим окончательный результат.

Рассмотренная нами процедура поиска результата напоминает нам работу пролог - машины, только мы не фиксировали порядок применения правил.

Реализация естественной семантики языка Exp

```
eval(N,N) :- number(N).

eval(E,R) :-
        E =.. [Op,E1,E2],
        member(Op,[+,-,*,/]),
        eval(E1,R1),
        eval(E2,R2),
        Ee =.. [Op,R1,R2],
        R is Ee.

test(V) :-
        eval(2*3+4-6/2, V).
```

Индукция

Свойства семантики языка программирования можно доказывать по индукции.

Метод математической индукции:

Чтобы доказать свойство P(x) всех натуральных чисел, нужно доказать два отдельных утверждения:

1) Истинность P(0). Это база индукции.

2) То, что из истинности P(k) следует истинность P(k+1) . Это индуктивный шаг.

Почему?

- Потому, что эти два утверждения определяют рекурсивный процесс, который проверит истинность свойства P(x) для всех натуральных чисел.

Пример

Пусть нужно доказать, что сумма первых **n** натуральных чисел равна **n * (n+1) div 2**, то есть

0 + 1 + … + n = n * (n+1) div 2.

Это свойство всех натуральных чисел.

Итак, для доказательства **P(n)**, для всех **n ∈ N** нужно показать, что:

1) **0 = 0 * (0+1) div 2.** Для этого достаточно просто выполнить арифметические действия.

2) Из истинности

(1) **0 + 1 + … + n = n * (n+1) div 2**

вывести

(2) 0 + 1 + … + n + (n+1) = (n+1) * (n+2) div 2.

Прибавим к обоим частям истинного равенства (1) **(n+1)** получим:

0 + 1 + … + n + (n+1) = n * (n+1) div 2 + (n+1).

Далее выполнив преобразование правой части получим:

n * (n+1) div 2 + (n+1) = {умножим и поделим (n+1) на 2 }

n * (n+1) div 2 + 2 * (n+1) div 2 = {сложим дроби}

(n * (n+1) + 2 * (n+1)) div 2 = {вынесем (n+1) за скобку}

(n+1) * (n+2) div 2.

Структурная индукция

Метод математической индукции применим к натуральным числам потому, что их множество определяется по индукции:

0 ∈ N

Если n ∈ N, то и **n+1 ∈ N.**

Доказательство по индукции можно строить и для других множеств, заданных по индукции. Например, возьмем множество списков натуральных чисел. Обозначим через [] – пустой список, а через : - операцию построения списка из головы и хвоста. Наше множество Lists(**N**) можно определить так.

[] ∈ Lists(**N**)

Если l ∈ Lists(**N**), а **n ∈ N**, то **n:l** ∈ Lists(**N**) **.**

В форме правил:

$$\frac{}{[] \in \mathrm{Lists}(\mathbf{N})}$$

$$\frac{l \in \mathrm{Lists}(\mathbf{N}) \qquad n \in \mathbf{N}}{\mathbf{n{:}l} \in \mathrm{Lists}(\mathbf{N})}$$

Закон «`map` после `(++)`»

Для всех списков **`xs`**, **`ys`** и функций **`f`** выполняется равенство:

`map f (xs++ys) = map f xs ++ map f ys` ,

при условии что правильно определены типы.

xs	map f (xs++ys)	map f xs ++ map f ys
[]	map f ([]++ys) = (опр. (++)) map f ys	map f [] ++ map f ys = (опр. map) [] ++ map f ys = (опр. (++)) map f ys
x:xs	map f ((x:xs)++ys) = (опр. (++)) map f (x:(xs++ys)) = (опр. map) f x :map f (xs++ys)	map f (x:xs) ++ map f ys = (опр. map) (f x :map f xs) ++ map f ys = (опр. (++)) f x:map f xs++map f ys

Теорема. Отношение ⇒ для языка Exp является функцией

Для всех выражений **e ∈ Exp** справедливо, что если **e ⇒ v** и **e ⇒ v′**, то **v = v′**. Это **P(e)**.

Для доказательства применим структурную индукцию.

Нужно доказать:

1) P(n) для всех чисел **n**.

2) При условии истинности **P(e)** и **P(e′)** доказать **P(e op e′)**.

Первый случай

Если $\mathbf{n} \Rightarrow \mathbf{v}$, а для вычисления $\mathbf{v}$ мы могли использовать только правило CR то $\mathbf{n} = \mathbf{v}$.

Если $\mathbf{n} \Rightarrow \mathbf{v}'$, то из тех же соображений получим $\mathbf{n} = \mathbf{v}'$.

Из $\mathbf{n} = \mathbf{v}$ и $\mathbf{n} = \mathbf{v}'$ следует, что $\mathbf{v} = \mathbf{v}'$.

□

Второй случай

Если $e' \ op \ e'' \Rightarrow v$, а для вычисления v мы могли использовать только правило OpR , значит $e' \Rightarrow m'$, $e'' \Rightarrow m''$ а $v = Ap(op_{Num}, m', m'')$, где m', m'' - некоторые числа.

Если $e' \ op \ e'' \Rightarrow v'$, а для вычисления v' мы могли использовать только правило OpR , значит $e' \Rightarrow k'$, $e'' \Rightarrow k''$ а $v' = Ap(op_{Num}, k', k'')$, где k', k'' - некоторые числа.

Из $P(e')$ и $P(e'')$ получим, что $m'=k'$, $m''=k''$.

А поскольку op_{Num} - функция, получим $v=v'$. □