

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	∈	Expr
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

Конкретная операционная семантика языка Expr

- `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

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

отображающее множество арифметических выражений на множество чисел.

Оно определяется индуктивно:

Правило 1: Для каждой числовой константы n , $n \Rightarrow n$.

Правило 2: Если $e \Rightarrow v$ и $e' \Rightarrow v'$, то $e \text{ op } e' \Rightarrow \text{Ar} (\text{op}, v, v')$;

Правила, определяющие естественную семантику языка арифметических выражений

✓ Правило CR

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

✓ Правило OpR

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

Вычисление значений арифметических выражений

Пусть нужно вычислить значение выражения

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

Это сумма, и применение правила OpR приведёт к

$$\begin{array}{l} 3 * 4 \Rightarrow v \qquad 8 \text{ div } (4 - 2) \Rightarrow v' \\ \hline 3 * 4 + 8 \text{ div } (4 - 2) \Rightarrow \text{Ap}(+_{\text{NUM}}, v, v') \end{array}$$

Для вычисления применим ещё два раза правила OpR:

$$\begin{array}{l} 3 \Rightarrow v' \qquad 4 \Rightarrow v'' \\ \hline 3 * 4 \Rightarrow \text{Ap}(*_{\text{NUM}}, v', v'') \\ \\ 8 \Rightarrow v' \qquad (4 - 2) \Rightarrow v'' \\ \hline 8 \text{ div } (4 - 2) \Rightarrow \text{Ap}(/_{\text{NUM}}, v', v'') \end{array}$$

Вычисление значения арифметических выражений (продолжение)

В конце концов, получив численную константу применим правила CR:

$$\overline{\quad} \\ 3 \Rightarrow 3$$

$$\overline{\quad} \\ 4 \Rightarrow 4$$

$$\overline{\quad} \\ 8 \Rightarrow 8$$

$$\overline{\quad} \\ 2 \Rightarrow 2$$

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

$n * (n+1) \text{ div } 2$, то есть

$$0 + 1 + \dots + n = n * (n+1) \text{ div } 2.$$

Это свойство всех натуральных чисел.

Итак, для доказательства $P(n)$, для всех $n \in \mathbf{N}$ нужно показать, что:

1) $0 = 0 * (0+1) \text{ div } 2$. Для этого достаточно просто выполнить арифметические действия.

2) Из истинности

$$(1) \quad 0 + 1 + \dots + n = n * (n+1) \text{ div } 2$$

вывести

$$(2) \quad 0 + 1 + \dots + n + (n+1) = (n+1) * (n+2) \text{ div } 2.$$

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

$$0 + 1 + \dots + n + (n+1) = n * (n+1) \text{ div } 2 + (n+1).$$

Далее выполнив преобразование правой части получим:

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

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

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

$$(n+1) * (n+2) \text{ div } 2.$$

Структурная индукция

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

$$0 \in \mathbf{N}$$

Если $n \in \mathbf{N}$, то и $n+1 \in \mathbf{N}$.

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 $\text{Lists}(\mathbf{N})$ можно определить так.

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

Если $l \in \text{Lists}(\mathbf{N})$, а $n \in \mathbf{N}$, то $n:l \in \text{Lists}(\mathbf{N})$.

В форме правил:

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

$$\frac{l \in \text{Lists}(\mathbf{N}) \quad n \in \mathbf{N}}{n:l \in \text{Lists}(\mathbf{N})}$$

Закон « map после (++) »

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

$$\text{map } f \text{ (xs++ys)} = \text{map } f \text{ xs ++ map } f \text{ ys} ,$$

при условии что правильно определены типы.

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

Теорема. Отношение \Rightarrow для языка \mathbf{Expr} является функцией

Для всех выражений $e \in \mathbf{Expr}$ справедливо, что если $e \Rightarrow v$ и $e \Rightarrow v'$, 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 $P(e \text{ op } e')$.

Первый случай

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

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

Из $n = v$ и $n = v'$ следует, что $v = v'$.



Второй случай

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

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

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

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