15 Апреля 2010

Инсерционное моделирование 2

Лекция 7

Основы функционального программирования

Синтаксис

Синтаксис выражений: операции, константы

C – множество **атомарных констант**

$T_{\Omega}(C)$ алгебра **константных выражений** Ω – сигнатура операций

V – множество **переменных**

$T_{\Omega}(C,V)$ – алгебра **термов** (**алгебраических выражений**)

Φ – множество **функциональных символов**

$T_{\Omega\cup\Phi}(C,V)$ – алгебра **функциональных термов**

(**функциональных выражений**)

Интерпретированные и **неинтерпретированные** функциональные символы

Пример

Целые числа, символы, арифметические операции+
бинарная операция ((),()) образования списков +
конкатенация списков conc((),()) +
if(()=(),(),()) + head, tail, …

Обобщения

Алгебраические системы,
типы, многоосновные алгебраические системы,
полиморфизм, функционалы высших степеней

Система функциональных определений

$$f_i(x_1, x_2, \ldots) = E_i(f, x),$$
$$i = 1, \ldots, n, x = (x_1, x_2, \ldots)$$
$$f_i \in \Phi, x_j \in V, E_i(f, x) \in T_{\Omega \cup \Phi}(C, V)$$

Функциональная программа

Система функциональных определений + функциональное выражение, значение которого нужно вычислить: $R[F]$ (среда и агент)

Денотационная семантика

Область D

Семантика константных выражений: $[[A]] \in D$

Семантика функций: $[[f_i]] \in D^{n_i} \to D$

Наименьшая неподвижная точка функционала

$$(f_1,\ldots, f_n) \mapsto (F_1(f),\ldots, F_n(f))$$

$$F_i(f) = \lambda x E_i(f, x)$$

$f_i(x_1, x_2,\ldots) = E_i(f, x)$,
$i = 1,\ldots n,\ x = (x_1, x_2,\ldots)$

Семантика константных выражений определяется правилами вычислений, а семантика алгебраических выражений, отношением тождественного равенства.

Операционная семантика

Подстановка: $E' := E[x_1 := E_1, x_2 := E_2, \ldots]$

Упрощение: $simpl(E)$,

$2 * 2 = 4, x + 0 = x, \ldots$

Переходы:

$$\frac{F = G[z := f_i(t_1, t_2, \ldots)],\ f_i(x_1, x_2, \ldots) = E_i \in R}{R[F] \to R[simpl(G[z := E_i[x_1 := t_1, x_2 := t_2, \ldots]])]}$$

$$R[F] \to R[simpl(F)]$$

Теорема

$$[[A_1]] = d_1, [[A_2]] = d_2, \ldots \Rightarrow$$

$$\Rightarrow [[f_i]](d_1, d_2, \ldots) = d \Leftrightarrow f_i(A_1, A_2, \ldots) \xrightarrow{*} B, [[B]] = d$$

Для доказательства используется теорема о наименьшей неподвижной точке:

$$f_i(x_1, x_2, \ldots) = \bigsqcup f_i^{(n)}(x)$$

$$f_i^{(0)}(x) = \perp$$

$$f_i^{(n+1)}(x) = E_i(f^{(n)}, x)$$

Многоуровневая семантика

$$\frac{F = G[z := S[E]], (R \cup S)[E] \to T[E']}{R[F] \to R[simpl(G[z := T[E']])]}$$

$$E \in T_{\Omega}(C,V) \to R[E] = simple(E)$$

$$let(R)in(E) = R[E]$$

Стратегии вычислений

- **Call-by-value (eager)** самое внутреннее (inermost)
- **Call-by name (lazy)** самое внешнее (outermost)
- **Parallel,…** все самые внешние (внутренние)

Эффективность и полнота

Пример

Операции /\, \/, ~, atom

Формулы представляются списками

(dis,x,y), (conj,x,y), (neg,x),(atom,x)

Функция nnf

nnf(x)=if(head(x)=dis)then nnf_dis(x) else nnf_not_dis(x),

nnf_dis(x)=(dis,nnf(head(tail(x)),head(tail(tail(x))))),

nnf_not_dis=if(head(x)=conj)then nnf_conj(x) else nnf_not_dc(x)

nnf_not_dc(x)=if(head(x)=neg)then prop_neg(tail(x))

……

Задачи

1. Завершить определение функции nnf
2. Какую семантику применять для вычисления этой функции и почему?

1. Переписать отношение переходов для семантики call-by-value
2. Переписать отношение переходов для семантики call-by-name

Параллельное функциональное программирование

Вызовы и развертывания

$$f_i(x_1, x_2, \ldots) = E_i(f, x),$$
$$i = 1, \ldots n,\ x = (x_1, x_2, \ldots)$$

$f_i(t_1, t_2, \ldots)$ вызов

$E_i[x_1 := t_1, x_2 := t_2, \ldots]$ развертывание вызова

Вхождение вызова в *E,* отличное от *E* – внутренний вызов

Операционная семантика
параллельного функционального программирования

$R = \{ f_i(x_1, x_2, \ldots) = E_i(f, x) \mid i = 1, \ldots n\}$

$$\frac{t = G[x_1 := G_1, x_2 := G_2, \ldots]}{R[t] \to R[simpl(G[x_1 := G_1', x_2 := G_2', \ldots])]}$$

$$\frac{t = G[x_1 := G_1, x_2 := G_2, \ldots], R[G_1] \to R[F_1], R[G_2] \to R[F_2], \ldots}{R[t] \to R[simpl(G[x_1 := F_1', x_2 := F_2', \ldots])]}$$

E **– вызов =>** E' **– развертывание,** G_i **– вызовы,**
$G_i \to F_i$ **– одношаговый переход для внутренних вызовов (рекурсивно)**