Московский авиационный институт
(национальный исследовательский университет)
Факультет «Прикладная математика и физика»

Курсовой проект по курсу
«Системное программное обеспечение»

на тему “Вычисление арифметического выражения с помощью атрибутного ДМП-процессора”

Студент: Полей-Добронравова Амелия Вадимовна
Группа: М80-307Б-18
Руководитель: Семёнов А. С.

Оценка:
Дата:

Москва 2020

Построение Атрибутной транслирующей грамматики

Входными символами грамматики являются символы множества {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, /, *}.

Операционные символы {+}, {*}, {-}, {/},{i}. Все операционные символы кроме {i} имеют по три атрибута, значениями которых являются значения левого операнда, правого операнда и результата соответственно. Операционный символ {i} обозначает операцию формирования из последовательности символов числа и имеет один атрибут: его значение. Результатом работы (посчитанный ответ) является самый правый (третий) атрибут последнего выведенного операционного символа.

Нетерминальные символы V={S, E, T, M, P, R, Q}. Начальный символ S и символ E не имеют атрибутов.
T имеет один синтезированный атрибут b, M имеет один унаследованный атрибут a, значение которого наследуется от b. В некоторых случаях a выступает синтезированным атрибутом, получающим свое значение от результата сложения или вычитания.
Нетерминал P имеет один синтезируемый атрибут x, в котором хранится значение считанного числа. Операционный символ {i} передаёт P значение атрибута.
Нетерминал R отвечает за подсчитывание последовательности умножения или деления. Он имеет три атрибута: y, k, t.
Нетерминал Q имеет два трибута: v, g. Он нужен для подсчета одной операции умножения или деления.

Терминальные символы T = {i, +, -, *, /}.

Правила транслирующей грамматики

1) S -> E
2) E -> $T_b M_a$

   a <- b

3) M_a -> $+T_b M_a \{+\}_{s,p,h}$

   s <-a

   p<- b

   h<- s+p

a<- y

a <- h

a <- g

4) M_a -> $-T_bM_a\{-\}_{l,m,w}$

w <- l-m

m <-b

l<-a

a <- w

a<- y

a <- g

5) M_a -> ε
6) T_b -> $P_xR_{y,k,t}$

t <- x

b <- k

b <- v

b <- g

b <- y

7) $R_{y,k,t}$ -> $*P_xQ_{v,\ g}\{*\}_{o,q,u}$

u <- o * q

o <- t

q <- y

k <- y

g <- k

q <- x

v <- u

8) $R_{y,k,t}$ -> $/P_xQ_{v,g}\{/\}_{z,c,j}$

j <- z / c

z <- t

c <- y

k <- y

g <- k

c <- x

v <- j

9) $R_{y,k}$ -> ε

y <- k

k <- t

10) P_x -> $i_f\{i\}_f$

x <- f

11) M_a -> ε

m<-a

s <-a

a <- h

a <- w

12) $Q_{v,g}$ -> $R_{y,k}$

t <- v

Управляющая таблица для транслирующей грамматики

(в ячейках записаны номера применяемых правил)

	i	+	-	*	/	ε

S	1					
E	2					
M		3	4	5	5	11
T	6					
R		9	9	7	8	9
P	10					
Q	12	12	12	12	12	12
i	ВЫБРОС					
+		ВЫБРОС				
-			ВЫБРОС			
*				ВЫБРОС		
/					ВЫБРОС	
⊥						ДОПУСК
{i}	ВЫДАЧА ({i})	ВЫДАЧА ({i})	ВЫДАЧА ({i})	ВЫДАЧА ({i})	ВЫДАЧА ({i})	ВЫДАЧА ({i})
{+}	ВЫДАЧА ({+})	ВЫДАЧА ({+})	ВЫДАЧА ({+})	ВЫДАЧА ({+})	ВЫДАЧА ({+})	ВЫДАЧА ({+})
{-}	ВЫДАЧА ({-})	ВЫДАЧА ({-})	ВЫДАЧА ({-})	ВЫДАЧА ({-})	ВЫДАЧА ({-})	ВЫДАЧА ({-})
{*}	ВЫДАЧА ({*})	ВЫДАЧА ({*})	ВЫДАЧА ({*})	ВЫДАЧА ({*})	ВЫДАЧА ({*})	ВЫДАЧА ({*})
{/}	ВЫДАЧА ({/})	ВЫДАЧА ({/})	ВЫДАЧА ({/})	ВЫДАЧА ({/})	ВЫДАЧА ({/})	ВЫДАЧА ({/})

Начальное содержимое магазина — S⊥

Каждая операция ВЫДАЧА выдаёт на выходную ленту имя операционного символа и все его атрибуты.

Пример работы ДМП-процессора

$(2+4/2, S\perp, \varepsilon)$ |-

$(i_2+4/2, S\perp, \varepsilon)$ |-

$(i_2+4/2, E\perp, \varepsilon)$ |-

$(i_2+4/2, TM\perp, \varepsilon)$ |-

$(i_2+4/2, PRM\perp, \varepsilon)$ |-

$(i_2+4/2, i_2\ \{i\}_2R_2M\perp, \varepsilon)$ |-

$(+4/2, \{i\}_2R_2M\perp, \varepsilon)$ |-

$(+4/2, R_2M\perp, \{i\}_2)$ |-

$(+4/2, M_2\perp, \{i\}_2)$ |-

$(+4/2, +TM\{+\}_2\perp, \{i\}_2)$ |-

$(4/2, TM\{+\}_2\perp, \{i\}_2)$ |-

$(i_4/2, TM\{+\}_2\perp, \{i\}_2)$ |-

$(i_4/2, PRM\{+\}_2\perp, \{i\}_2)$ |-

$(i_4/2, i_4\{i\}_4R_4M\{+\}_2\perp, \{i\}_2)$ |-

$(/2, \{i\}_4R_4M\{+\}_2\perp, \{i\}_2)$ |-

$(/2, R_4M\{+\}_2\perp, \{i\}_2\{i\}_4)$ |-

$(/2, /PQ\{/\}_4M\{+\}_2\perp, \{i\}_2\{i\}_4)$ |-

$(2, PQ\{/\}_4M\{+\}_2\perp, \{i\}_2\{i\}_4)$ |-

$(i_2, PQ\{/\}_4M\{+\}_2\perp, \{i\}_2\{i\}_4)$ |-

$(i_2, i_2\{i\}_2Q_2\{/\}_{4,2}M\{+\}_2\perp, \{i\}_2\{i\}_4)$ |-

$(\varepsilon, \{i\}_2Q_2\{/\}_{4,2}M\{+\}_2\perp, \{i\}_2\{i\}_4)$ |-

$(\varepsilon, R_2\{/\}_{4,2}M\{+\}_2\perp, \{i\}_2\{i\}_4\{i\}_2)$ |-

$(\varepsilon,\{/\}_{4,2,2}M\{+\}_2\perp, \{i\}_2\{i\}_4\{i\}_2)$ |-

$(\varepsilon,M\{+\}_{2,2}\perp, \{i\}_2\{i\}_4\{i\}_2\{/\}_{4,2,2})$ |-

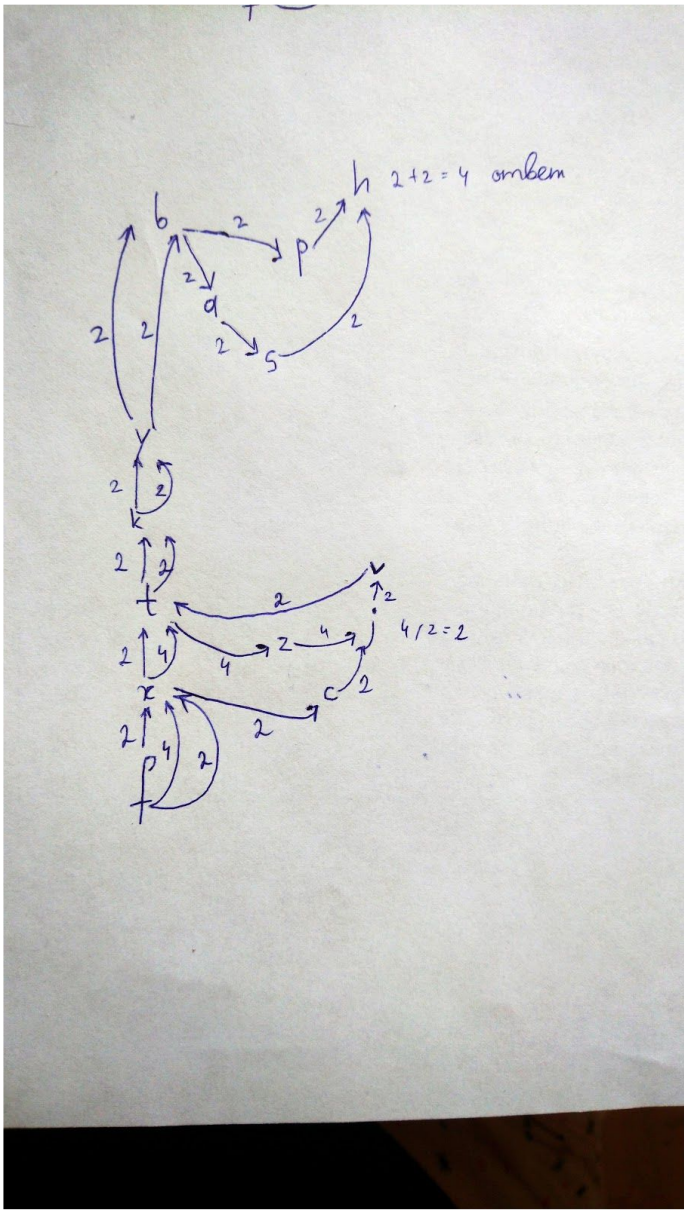
$(\varepsilon,\{+\}_{2,2,4}\perp, \{i\}_2\{i\}_4\{i\}_2\{/\}_{4,2,2})$ |-

$(\varepsilon,\perp, \{i\}_2\{i\}_4\{i\}_2\{/\}_{4,2,2}\{+\}_{2,2,4})$ |-

$(\varepsilon, \perp, \{i\}_2\{i\}_4\{i\}_2\{/\}_{4,2,2}\{+\}_{2,2,4}) \vdash$

ДОПУСК

Тот же пример через дерево связи атрибутов:



Стрелки вверх обозначают синтезирование, вниз - наследование.

Реализация

Передача Атрибутному ДМП-процессору грамматики:

```
ArrayList mag_symbols = new ArrayList() { new at_mag("M", "a"), new
at_mag("Q", "v","g"), new at_mag("S"), new at_mag("E"), new at_mag("S"), new
```

```
at_mag("T", "b"), new at_mag("R", "y", "k", "t"), new at_mag("P", "x"), new
at_mag("i"), new at_mag("+"), new at_mag("-"), new at_mag("*"), new
at_mag("/") };

AT_Translator at_translator = new AT_Translator(new ArrayList()  { "q0",
"q1", "qf"}, new ArrayList() { "1", "2", "3", "4", "5", "6", "7", "8", "9",
"0", "-", "+", "*", "/" }, mag_symbols, "q0","S", new ArrayList() { "qf" },
new ArrayList() { new at_mag("+", "s", "p", "h"), new at_mag("-", "l", "m",
"w"), new at_mag("*", "o", "q", "u"), new at_mag("/", "z", "c", "j"), new
at_mag("i", "f") });

at_translator.addDeltaRule("q0", new ArrayList() { "1", "2", "3", "4", "5",
"6", "7", "8", "9", "0" },"S", new ArrayList() { "q1" }, new ArrayList() {
"E" }, new ArrayList() { "" });

at_translator.addDeltaRule("q1", new ArrayList() { "1", "2", "3", "4", "5",
"6", "7", "8", "9", "0" }, "E", new ArrayList() { "q1" }, new ArrayList() {
"T","M" }, new ArrayList() { "" });
at_translator.addATRule(2,"a", new ArrayList() {"b"});

at_translator.addDeltaRule("q1", new ArrayList() { "+" }, "M", new
ArrayList() { "q1" }, new ArrayList() { "+", "T", "M" }, new ArrayList() {
"+" });
at_translator.addATRule(3, "h", new ArrayList() { "s", "+", "p" });
at_translator.addATRule(3, "s", new ArrayList() { "a" });
at_translator.addATRule(3, "p", new ArrayList() { "b" });
at_translator.addATRule(3, "a", new ArrayList() { "h" });
at_translator.addATRule(3, "a", new ArrayList() { "y" });
at_translator.addATRule(3, "a", new ArrayList() { "g" });

at_translator.addDeltaRule("q1", new ArrayList() { "-" }, "M", new
ArrayList() { "q1" }, new ArrayList() { "-", "T", "M" }, new ArrayList() {
"-" });
at_translator.addATRule(4, "w", new ArrayList() { "l", "-", "m" });
at_translator.addATRule(4, "l", new ArrayList() { "a" });
at_translator.addATRule(4, "m", new ArrayList() { "b" });
at_translator.addATRule(4, "a", new ArrayList() { "w" });
at_translator.addATRule(4, "a", new ArrayList() { "y" });
at_translator.addATRule(4, "a", new ArrayList() { "g" });

at_translator.addDeltaRule("q1", new ArrayList() { "*", "/" }, "M", new
ArrayList() { "q1" }, new ArrayList() { "" }, new ArrayList() { "" });

at_translator.addDeltaRule("q1", new ArrayList() { "1", "2", "3", "4", "5",
"6", "7", "8", "9", "0" }, "T", new ArrayList() { "q1" }, new ArrayList() {
"P", "R" }, new ArrayList() { "" });
at_translator.addATRule(6, "t", new ArrayList() { "x" });
at_translator.addATRule(6, "b", new ArrayList() { "v" });
at_translator.addATRule(6, "b", new ArrayList() { "g" });
at_translator.addATRule(6, "b", new ArrayList() { "k" });
at_translator.addATRule(6, "b", new ArrayList() { "y" });

at_translator.addDeltaRule("q1", new ArrayList() { "*" }, "R", new
ArrayList() { "q1" }, new ArrayList() { "*", "P", "Q" }, new ArrayList() {
"*" });
at_translator.addATRule(7, "u", new ArrayList() { "o", "*", "q" });
at_translator.addATRule(7, "o", new ArrayList() { "t" });
at_translator.addATRule(7, "q", new ArrayList() { "y" });
```

```
at_translator.addATRule(7, "k", new ArrayList() { "y" });
at_translator.addATRule(7, "g", new ArrayList() { "k" });
at_translator.addATRule(7, "q", new ArrayList() { "x" });
at_translator.addATRule(7, "v", new ArrayList() { "u" });

at_translator.addDeltaRule("q1", new ArrayList() { "/" }, "R", new
ArrayList() { "q1" }, new ArrayList() { "/", "P", "Q" }, new ArrayList() {
"/" });
at_translator.addATRule(8, "j", new ArrayList() { "z", "/", "c" });
at_translator.addATRule(8, "z", new ArrayList() { "t" });
at_translator.addATRule(8, "c", new ArrayList() { "y" });
at_translator.addATRule(8, "k", new ArrayList() { "y" });
at_translator.addATRule(8, "g", new ArrayList() { "k" });
at_translator.addATRule(8, "c", new ArrayList() { "x" });
at_translator.addATRule(8, "v", new ArrayList() { "j" });

at_translator.addDeltaRule("q1", new ArrayList() { "+", "-","e" }, "R", new
ArrayList() { "q1" }, new ArrayList() { "" }, new ArrayList() { "" });
at_translator.addATRule(9, "y", new ArrayList() { "k" });
at_translator.addATRule(9, "k", new ArrayList() { "t" });

at_translator.addDeltaRule("q1", new ArrayList() { "1", "2", "3", "4", "5",
"6", "7", "8", "9", "0" }, "P", new ArrayList() { "q1" }, new ArrayList() {
"i" }, new ArrayList() { "i" });
at_translator.addATRule(10, "x", new ArrayList() { "f" });

at_translator.addDeltaRule("q1", new ArrayList() { "e" }, "M", new
ArrayList() { "q1" }, new ArrayList() { "" }, new ArrayList() { "" });
at_translator.addATRule(11, "m", new ArrayList() { "a" });
at_translator.addATRule(11, "s", new ArrayList() { "a" });
at_translator.addATRule(11, "a", new ArrayList() { "h" });
at_translator.addATRule(11, "a", new ArrayList() { "w" });

at_translator.addDeltaRule("q1", new ArrayList() { "1", "2", "3", "4", "5",
"6", "7", "8", "9", "0", "-", "+", "*", "/", "e" }, "Q", new ArrayList() {
"q1" }, new ArrayList() { "R" }, new ArrayList() { "" });
at_translator.addATRule(12, "t", new ArrayList() { "v" });

                                Console.WriteLine("\nВведите строку для
обработки");
Console.WriteLine(at_translator.Execute(Console.ReadLine()).ToString());
```

Класс Атрибутного ДМП-процессора

public class atr - атрибут

public class at_mag - магазинный символ

public class at_rule - правило передачи атрибута по номеру дельтаправила

class AT_Translator : Translator - сам класс АТ-ДМП-процессора

public bool Is_operation_symbol(string name) -проверяет, существует ли операционный символ с именем name

public class DeltaQSigmaGammaSix1 : DeltaQSigmaGammaSix - дельтаправило, позволяющее передавать массив из нескольких символов, встречаемых во входной строке, при которых возможно правило, а не только один символ, как в *DeltaQSigmaGammaSix*

public DeltaQSigmaGammaSix1 FindDelta(string Q, string a) - поиск правила по состоянию

public int FindDeltaNumber(DeltaQSigmaGammaSix1 delta) - поиск номера дельтаправила

public AT_Translator(ArrayList Q, ArrayList Sigma, ArrayList Gamma, string Q0, string z0, ArrayList F, ArrayList operation_symbol) - конструктор АТ-ДМП-процессора

public void addATRule(int number, string left, ArrayList right) - добавить атрибутное правило с номером дельтаправила number

public at_mag findATop(string nam) - собрать операционный символ по имени nam, если в этой грамматике существует операционный символ с таким именем

public at_mag findATmag(string nam) - найти для вставки в магазин символ с нужным названием и вычислить значения его атрибутов, если возможно

public int getATval (string name) - ищет в стеке первое вхождение атрибута и берет его значение

public bool setATval(string name, int value, Stack<int> deltanumbers, int y) - ищет в магазине атрибут name куда можно присвоить значение value

public bool is_last_atr_in_rule(string nameatr, int number) - если это последний операционный символ правила, вернуть правду

public void calculate_atr3(at_mag op) - вычислить значение третьего атрибута операционного символа

public void whereAT(atr at, Stack<int> deltanumbers, bool ispeek) - ищем куда вставить значение атрибута и вставляем

public bool Members_in_magazine(int number) - проверяет остались ли в магазине члены правила

public DeltaQSigmaGammaSix1 FindDelta(string currState, string Left, string st) - ищет правило перехода

Работа ДМП-процессора осуществляется в функции Execute.