Формални Езици и Езикови Процесори
ТУ, кат. КС, летен семестър 2012

Лекция 19

Тема:

Семантика на ПЕ
(Атрибутни Граматики)

Съдържание:

- Семантика на ПЕВН
- Статична Семантика
  - Атрибутни граматики
  - Еволюция на концепцията КСГ
  - Наследени атрибути
  - Синтезирани атрибути
  - Примери
- Динамична Семантика

Семантичен Анализ

- Функция: да се определи значението /смисъла/ на първичната програма
- (A+ B) * (C + D)
- Операндите дефинирани/декларирани ли са?
- Еднакъв тип ли са операндите? Ако не, компилаторът ще ги уеднакви
- Имат ли стойност операндите?

Семантичен Анализ

Пример:

pos = init + rate * 60;

Семантичен Анализ

Извършват се проверки като следните:

- Дефинирани/декларирани операнди
- Присвоени стойности на операнди
- Type checking, т.е операндите в изразите да имат допустим тип.
- Включване на операции за конвертиране на тип в случай на операнди с различен тип като предния пример rate(real), 60(integer)

Семантичен Анализ

Отчетени са операнди с различен тип:

rate (real) 60 (integer)

Уеднаквяване на типове се постига с добавяне на възел в дървото на разбора

Семантична routine *inttoreal()* служи като операция, която явно преобразува цели стойности в реални стойности

Илюстрация – модифицирано дърво на разбора

КСГ: не решават проблема

КСГ описват синтаксиса на ПЕ.

НО: има конструкти в ПЕ, които е трудно или невъзможно да се опишат с КСГ и тяхната класическа Chomsky/BNF нотация.

Следват три примера:

Пример 1

- Грешки от съвместимост на типовете. В Java, реална /floating point/ стойност не може да се присвои на цяла /integer type/ променлива, въпреки че обратното се допуска.
- Няма начин горното условие да се опише с КСГ правило.

Пример 2

- Няма начин да се опише в BNF нотация общото правило в ПЕ: всички променливи да се дефинират преди да се ползват.

Пример 3

- Няма начин да се опише в BNF нотация правилото в PL/1 и ADA, ако оператор end of a subroutine /подпрограма/ е следван от име, то трябва да съответства на името на подпрограмата.

Статична Семантика

Трите примера илюстрират правила на ПЕВН, които са извън синтаксиса и се наричат статични семантични правила.

Статична семантика: Нарича се СТАТИЧНА, т.к анализът на този тип спецификации и ограничения може да се проведе по време на компилация.

Въведение в Атрибутни Граматики

Едно средство за разрешаване статични семантчни проблеми в ПЕ е известно като Атрибутни Граматики АГ.

АГ са въведени от D.Knuth през 1986.

АГ са формален подход за описание и проверка коректността на статичните семантични правила в една програма.

Атрибутни Граматики - основи

1. АГ са КСГ с придадени атрибути **attributes**, attribute computation functions и predicate functions:

Attributes са променливи в периода на компилация. Атрибутите са асоциират с възли от дървото на разбора (възлите са именовани с терминални или нетерминални символи).

Атрибутни Граматики - основи

2. АГ са КСГ с придадени атрибути attributes, **attribute computation functions** и predicate functions: :

Attribute computation functions (още semantic functions) се асоциират с правилата на граматиката и задават начина на обработка на атрибутите.

Атрибутни Граматики - основи

3. АГ са КСГ с придадени атрибути attributes, attribute computation functions и **predicate functions**:

Predicate functions се асоциират с правилата на граматиката и служат за изразяване на някои синтактични и семантични правила на програмните/формалните/ езици.

Наследени атрибути

1. АГ като КСГ с наследени атрибути:

Базират се на низходящ /top-down/ процес на изграждане на дървото на разбора от корена към листата /короната/.

Наследени атрибути

Атрибутите на символите в дясната страна на продукция се изчисляват като функция (зависят от) атрибута(ите) на нетерминала в лявата страна на продукцията.

$X \rightarrow Y1\ Y2\ \ldots\ YN$

$X_a \rightarrow Y1_{f1(a)}\ Y2_{f2(a)}\ \ldots\ YN_{fn(a)}$

Синтезирани атрибути

2. АГ като КСГ със синтезирани атрибути:

Базират се на възходящ /bottom-up/ процес на изграждане на дървото на разбора от листата /короната/ към корена.

Синтезирани атрибути

Атрибутът(ите) на левия нетерминал в правилото се изчислява като функция (зависи от) атрибтите на символите в дясната страна на продукцията.

$X \rightarrow Y1\ Y2\ \ldots\ YN$

$X_{f(a1, a2,..,an)} \rightarrow Y1_{a1}\ Y2_{a2}\ \ldots\ YN_{an}$

Примери

ТС само с наследени атрибути: разбор на АИ и превод в инструкции на АЕ.

Наследени атрибути на основата на атрибут/променлива *p* се ползват за позоваване на регистри с общо предназначение за процесор с абстрактна регистрова архитектура.

АИ КСГ/ТС без атрибути

1. E → T	T
2. E → E + T	E T add Regx, Regy
3. T → F	F
4. T → T * F	T F mul Regx, Regy
5. F → a	move Regx, a
6. F → (E)	E

АИ КСГ/ТС с атрибути

1. $E_p \rightarrow T_p$	T_p
2. $E_p \rightarrow E_p + T_{p+1}$	$E_p\ T_{p+1}$ add STR(p),STR(p+1)
3. $T_p \rightarrow F_p$	F_p
4. $T_p \rightarrow T_p * F_{p+1}$	$T_p\ F_{p+1}$ mul STR(p),STR(p+1)
5. $F_p \rightarrow a$	move STR(p),a
6. $F_p \rightarrow (E_p)$	E_p

Демо

Разбор и генериране на код за: a + a * a

Демо

Разбор и генериране на код за: a + a * a

ОПЗ: aaa*+

Target code:

```
move Reg0,a
move Reg1,a
move Reg2,a
mul Reg1,Reg2
add Reg0,Reg1
```

Демо

Разбор и генериране на код за: a * (a + a)

ОПЗ: aaa+*

Target code:

```
move Reg0,a
move Reg1,a
move Reg2,a
add Reg1,Reg2
mul Reg0,Reg1
```

Демо

Разбор и генериране на код за: (a + a) * a

ОПЗ: aa+a*

Target code:

```
move Reg0,a
move Reg1,a
add Reg0,Reg1
move Reg1,a
mul Reg0,Reg1
```

Примери

КСГ само със синтезирани атрибути:

АИ граматика с операнди десетични константи

АГ със синтезирани атрибути позволява едновременно с разбора на израза да се проведе и интерпретация /пресмятане/ на израза.

АИ КСГ без атрибути

$$
\begin{aligned}
E &\rightarrow T \mid E + T \\
T &\rightarrow F \mid T * F \\
F &\rightarrow N \mid (E) \\
N &\rightarrow D \mid N\ D \\
D &\rightarrow 0 \mid 1 \mid \ldots \mid 9
\end{aligned}
$$

АИ КСГ без атрибути

E → T

E → E + T

T → F

T → T * F

F → N

F → (E)

N → D

N → N D

D → 0

D → 1

…

D → 9

АИ КСГ със синтезирани атрибути

$E_v \rightarrow T_v$

$E_{v1+v2} \rightarrow E_{v1} + T_{v2}$

$T_v \rightarrow F_v$

$T_{v1*v2} \rightarrow T_{v1} * F_{v2}$

$F_v \rightarrow N_v$

$F_v \rightarrow (E_v)$

$N_v \rightarrow D_v$

$N_{10*v1+v2} \rightarrow N_{v1}\ D_{v2}$

$D_0 \rightarrow 0_0$

$D_1 \rightarrow 1_1$

…

$D_9 \rightarrow 9_9$

Демо

Разбор и пресмятане на: 6 *(3 + 15)

Примери

КСГ с наследени и синтезирани атрибути

Динамична Семантика

Динамична Семантика

- Динамичната семантика се отнася до смисъла на изрази, оператори и дори цели програми expressions, statements and program units.
- Описанието на динамична семантка е по-трудно от описанието на синтаксис и статична семантика.
- Защо динамична семантика? Програмистите следва да знаят какво точно правят expressions, statements и program units. Не винаги English обясненията в ръководствата за ПЕ са прецизни и изчерпателни.

Динамична Семантика

- Класификация:
  - Операционна Семантика;
  - Аксиоматична Семантика;
  - Денотационна Семантика.

ДС: Операционна Семантика

- Основна идея: да се опише смисълът на една програма чрез изпълнение на операторите й на компютър, реален или виртуален.
- Промените в статуса на компютъра при изпълнение на даден оператор определят значението/смисъла на този оператор.
- Статусът на компютъра включва: съдържание на всички регистри и клетки на ОП.

ДС: Аксиоматична Семантика

- Основна идея да се докаже коректността на програми. Тези доказателства за коректност свидетелстват, че програмата извършва всички обработки съгласно спецификацията.
- При доказателство, всеки оператор се загражда с логически изрази, описващи ограничения свързани със съдържанието на програмни променливи.
- Логическите изрази, вместо пълния статус на абстрактния компютър, както е при ОС, служат да специфицират значението/смисъла на всеки оператор.

ДС: Денотационна Семантика

- Основна идея: За всяка езикова единица entity се определят математически обект и функция, която съпоставя инстанции на езиковата структура с инстанции на математическия обект.
- Доколкото математическите обекти са строго дефинирани, те представят точно значението на съответните езкови структури. Има строги подходи за манипулиране на математически обекти, но не и за езикови конструкти.
- Трудността е избор на матем обекти и функции.
- Нарича се DENOTATIONAL, т.к матем обектите денотират denote значението/смисъла на съответните синтактични единици /entities/.

Благодаря
За
Вниманието