СРАВНИТЕЛЬНЫЙ АНАЛИЗ ГЕНЕРАТОРОВ–КОМПИЛЯТОРОВ LLgen И ACCENT

С. Б. Калачева

Санкт-Петербургский институт информатики и автоматизации РАН
199178, Санкт-Петербург, 14-я линия ВО, д.39

<svet@iias.spb.su>

УДК 681.3
С. Б. Калачева. **Сравнительный анализ генераторов–компиляторов LLgen и ACCENT** // Труды СПИИРАН, Вып. 2, т. 2. — СПб.: Наука, 2005.
Аннотация. *Проводится анализ генераторов компиляторов LLgen и ACCENT на основе тестирования по заданным примерам для вывода об их применимости. Содержатся таблицы характеристик мало известных генераторов компиляторов и их Интернет-адреса.* — Библ. 4 назв.

UDC 681.3
S. B. Kalacheva. **Comparative Analysis of Compilers of Compilers LLgen and ACCENT**// SPIIRAS Proceedings. Issue 2, vol. 2. — SPb.: Nauka, 2005.
Abstract. *Analysis of compilers of compilers LLgen and ACCENT is conducted on the base of testing on the settings examples for conclusion about their applicability. The tables of characters of the little known compilers of compilers are contained and their Internet-addresses are listed*. — Bibl. 4 items.

1.Введение

В настоящее время известно множество различных генераторов, компиляторов и лексических анализаторов. Вопрос рационального выбора конкретного программного обеспечения остается открытым. Обычно в первую очередь руководствуются двумя аспектами: во-первых, выбором по операционной среде, во-вторых, выбором по языку программирования.

Сейчас наиболее распространен генератор лексических анализаторов Lex и генератор синтаксических анализаторов Yacc или их свободно распространяемая версия в рамках проекта GNU Bison/flex. Версии их имеются для всех имеющихся платформ. Существует множество клонов этих генераторов.

В данной статье мы остановимся на других генераторах компиляторов и лексических анализаторов. В табл. 1 приведены названия и их краткая характеристика. В табл. 2 указаны адреса их сайтов в Internet. Для дальнейшего исследования и сравнения нами выбраны генераторы компиляторов LLgen и ACCENT. Оба они работают на платформе Unix, написаны на языке Си и генерируют компиляторы на языке Си. Тексты программ можно получить с их сайтов в Интернет, они снабжены подробной документацией. Этим они выгодно отличаются, скажем, от генератора компиляторов CoSY, содержащем на сайте только рекламу продукта, а размеры текстов для компиляции программ таковы, что они, в отличие от мощного генератора компиляторов Cocktail, помещаются на одну дискету. Оба они поддерживают РБНФ (РБНФ — Расширенная Форма Бэкуса-Наура с операцией итерации * от одного повторения и операцией + от нуля).

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Таблица 1. Характеристики некоторых генераторов компиляторов

Название	Характеристика	Авторы (страна, язык реализации)
CoSY	Создает качественные компиляторы для множества языков программирования и архитектуры процессоров	ACE компания (Нидерланды,C++)
Cocktail	Набор генераторов почти для всех фаз работы компилятора	the German National Research Center for Information Technology (Германия,C)
Gentle	Интегрированная система, перекрывающая весь спектр задач по конструированию компиляторов	F. W.Schroer (Германия, Gentle, Modula2)
PRECC	Бесконечно просматривающий вперед генератор компиляторов для КЗ грамматик	P.Breuer, J. Bowen (Англия, PRECC, C)
ACCENT	Генератор компиляторов для целого класса КС языков	F. W.Schroer (Германия,C)
AnaGram	LARP(1) генератор компиляторов для синтаксически управляемого разбора, запускается на Win32 платформах	Parsifal Software (США, C)
Eli	Eli начинает со структурного анализа (решаемого с помощью средств типа LEX и YACC), через анализ имен, типов и значений к сохранению структур данных трансляции и получения выходного текста	University of Colorado Boulder (США) University of Paderborn (Германия) Macquarie University,(Австралия)
ALE	Attribute-Logic Engine обьединяет синтаксический анализ структуры фразы и логическое программирование в ограничениях с типизированными структурами признаков	B. Carpenter, G. Penn (США,C)
Llgen	Инструмент для генерации эффективного анализатора на основе рекурсивного спуска	C.Jacobs (Нидерланды, C)
DEPOT4	Генератор нисходящих распознавателей, поддерживающий описание в стиле схожем с синтаксически управляемой схемой перевода	J.Lampe (Германия, Java)
LISA	Генерирует таблично-управляемые лексические анализаторы и LL(1) синтаксические анализаторы по регулярным выражениям и БНФ	University of Maribor (Словения, C++)
COCO	Coco/R генерирует анализаторы, работающие по методу рекурсивного спуска	H. Mossenbock (Австрия, Modula-2)
COGENCEE	Генератор компиляторов для Delphi, разработанный на основе Coco	Cocolsoft Computer Solution (Австрия, Pascal)
JavaCC	Генератор компиляторов и лексических анализаторов для Java	S.Sankar, S.Viswanadha (США, Java)
Gray	На входе — грамматики в расширенной БНФ, получаем Forth код компилятора рекурсивным спуском	M.A.Ertl (Австрия, Forth)

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Таблица 2. Интернет-адреса указанных генераторов компиляторов

Название	Ссылка в Internet (http, home page)
CoSY	<http://www.ace.nl/products/cosy.htm>
Cocktail	<http://www.first.gmd.de/cocktail>
Gentle	<http://www.first.gmd.de/gentle>
PRECC	<http://archive.comlab.ox.ac.uk/redo/precc>
ACCENT	<http://accent.compilertools.net/index.html>
AnaGram	<http://www.parsifalsoft.com/>
Eli	<http://www.cs.colorado.edu/~eliuser/>
ALE	<http://www.cs.toronto.edu/~gpenn/ale.html>
LLgen	<http://www.cs.vu.nl/~ceriel/LLgen.html>
DEPOT4	<http://www.math.tu-dresden.de/wir/depot4/Depot4.html>
LISA	<http://marcel.uni-mb.si/nikolaj/sigplan.htm#A1.R>
COCO	<http://www.scifac.ru.ac.za/coco/>
COGENCEE	<http://www.cocolsoft.com.au/cogen/cogen.htm>
JavaCC	<http://www.cs.albany.edu/~sreeni/JavaCC.html>
Gray	<ftp://server.complang.tuwien.ac.at/pub/forth/gray4.zip>

2. Краткое описание рассматриваемых систем

Система LLgen была разработана под руководством доктора Сериела Джакобса (Ceriel Jacobs) в университете Вридже (Vrije) в Амстердаме в 1985 году. <http://www.cs.vu.nl/~ceriel/LLgen.html>.

LLgen — это инструмент для создания анализатора на языке Си на основе рекурсивного спуска из ELL(1) грамматики. Допускается неоднозначная грамматика.

ELL(1) — расширенная (Extended) LL(1) грамматика,

1) ELL(1) грамматика содержит `term N` (`N` представляет собой положительное целое).
2) ELL(1) грамматика содержит `term *`, что означает пусто или более повторений `term`, а `term *N` (`N` представляет собой положительное целое) означает `N` или меньше (включая 0) повторений `term`.
3) ELL(1) грамматика содержит `term +`, что означает один или более повторений `term`.
4) ELL(1) грамматика содержит квадратные скобки [и], используемые для группирования.
5) ELL(1) грамматика содержит `term?`, что означает 0 или 1 появление `term`.

Пользователь системой должен знать язык Си, так как, по крайней мере, подпрограмму main ему придется писать самому. Кроме того, в грамматике в фигурных скобках можно вставлять тексты на языке Си. Они перейдут в программу. Сам LLgen также написан на языке Си на платформе Unix.

ACCENT — компилятор компиляторов, охватывающий весь класс контекстно-свободных грамматик. Работает также с неоднозначными грамматиками. Поддерживает РБНФ. ACCENT был разработан доктором Фридрихом Вильгельмом Шроером (Friedrich Wilhelm Schroer) в Германском национальном исследовательском

Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

центре информационных технологий в 1999 году. <http://accent.compilertools.net/index.html>. Использовался алгоритм J. Earley. ACCENT написан на языке Си для платформы Unix. и генерирует компиляторы на языке Си.

3.Тестирование генераторов–компиляторов

Рассмотрим работу генераторов–компиляторов — LLgen и ACCENT. Для тестирования на вход будем посылать две грамматики — G_1 и G_2, представленные в табл. 3. G_1 — LR грамматика, G_2 - LL(1) грамматика. При выборе этих грамматик в качестве тестовых примеров мы руководствовались следующими соображениями: во-первых, это различные типы грамматик и, во-вторых, эти фрагменты достаточно распространены в языках программирования.

Таблица 3. Грамматики G_1 и G_2

G_1:

```
1. Program : [ | '-' ] Expression ;
2. Expression : Decimal_number |
     Expression '+' Expression |
     Expression '-' Expression |
     Expression '*' Expression |
     Expression '/' Expression |
     '(' [ | '-' ] Expression ')';
3. Decimal_number :Unsigned_integer|
    Unsigned_integer Decimal_fraction ;
4. Decimal_fraction :
     '.'Unsigned_integer;
5. Unsigned_integer : Digit+ ' ' ;
6. Digit : '0'|'1'|'2'|'3'|'4'|'5'|
           '6'|'7'|'8'|'9';
```

G_2:

```
1. Program: Polish_Expression ;
2. Polish_Expression:
      Unsigned integer |
      '+' Polish_Expression
      Polish_Expression |
      '-' Polish_Expression
      Polish_Expression |
      '*' Polish_Expression
      Polish_Expression;
3. Unsigned_integer: Digit+ ' ';
4. Digit:'0'|'1'|'2'|'3'|'4'|'5'
         |'6'|'7'|'8'|'9';
```

Обе грамматики записаны в РБНФ. Тестирование проводилось на платформе Linux версии 2.4, компилятор Си, проект GNU.

Грамматика LLgen, предваряется токеном %start, после чего следует входная точка программы и стартовый символ через запятую.

Исходная грамматика G_1 такова (номера строк поставлены для удобства анализа).

```
01  %start LLparse,Program;
02  Program: [ | '-' ] Expression ;
03  Expression: Decimal_number |
04       Expression '+' Expression |
05       Expression '-' Expression  |
06       Expression '*' Expression |
07       Expression '/' Expression |
08       '(' [ | '-' ] Expression ')';
09  Decimal_number: Unsigned_integer |
              Unsigned_integer Decimal_fraction ;
10  Decimal_fraction: '.' Unsigned_integer;
```


Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

```
11  Unsigned_integer: Digit+ ' ' ;
12  Digit: '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9';
```

Результатом тестирования грамматики G_1 является:

```
"G1" line 3  "Alternation conflict"
"G1" line 3  "Alternation conflict"
"G1" line 4  "Alternation conflict"
"G1" line 5  "Alternation conflict"
"G1" line 6  "Alternation conflict"
"G1" line 9  "Alternation conflict"
"G1" line 9  "Alternation conflict"
Segmentation fault
```

При тестировании зафиксировано 7 ошибок `"Alternation conflict"` на строки с номерами 3, 3, 4, 5, 6, 9, 9. Эта ошибка возникает, когда несколько правил начинаются с одного и того же символа. В этом случае необходимо эквивалентное преобразование, например: выносим Unsigned_integer и Expression за скобку. Вводим в текст тестовой грамматики новый нетерминал, обозначенный как Op.

Эквивалентная преобразованная грамматика G_{11} выглядит так:

```
01  %start LLparse,Program;
02  Program: [ | '-' ] Expression ;
03  Expression: Expression Op Expression |
04                          Decimal_number|
05          '(' [ | '-' ] Expression ')';
06  Op : '+' | '-' | '*' | '/' ;
07  Decimal_number: Unsigned_integer
                  [ | Decimal_fraction ] ;
08  Decimal_fraction: '.' Unsigned_integer;
09  Unsigned_integer: Digit+ ' ' ;
10  Digit: '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9';
```

Результат тестирования:

```
"G11" line 4  "Alternation conflict"
"G11" line 4  "Alternation conflict"
Segmentation fault
```

При тестировании выявлены две ошибки `"Alternation conflict"` на строке 4. Они возникли, из-за того, что при разборе как `Expression`, так и `Decimal_number` начинаются с `Digit`.

В правиле в строках 3—5 виден случай левой рекурсии. От нее избавляются введением еще одного нетерминала.

В теории это выглядит так:

```
A : A 'a' A| 'b';
```

порождают последовательности: `b, bab, babab, bababab, babababab` и т. д. Заменяем правило для `A` эквивалентным образом на:

```
A : 'b' [ | 'a' 'b' A1] ;
```


Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

```
A1: [ |'a' A ] ;
```
Порождающиеся последовательности остаются те же: `b, bab, babab, bababab, babababab` и т. д.

Докажем это методом математической индукции.

Последовательности `b` и `bab` порождаются тривиально. Пусть `A` — ba_nb_n , тогда `A1` — aba_nb_n, `A` — $baba_nb_n$ (или $ba_{n+1}b_{n+1}$) или $bababa_nb_n$ (или $ba_{n+2}b_{n+2}$), $(n>1)$. Все индексы получены (заметим, что полученная грамматика неоднозначная).

Для устранения ошибок вводим в грамматику G_{11} дополнительный нетерминал `Expression1`. Получаем грамматику G_{12}.

```
01  %start LLparse,Program;
02  Program: [ | '-' ] Expression ;
03  Expression:  [Decimal_number | '('
                              [ | '-' ] Expression ')']
04              [  |  op [Decimal_number | '(' [ | '-' ]
                              Expression ')'] Expression1] ;
05  Expression1: [  | op Expression] ;
06  Op: '+' | '-' | '*' | '/' ;
07  Decimal_number: Unsigned_integer
                    [ |  Decimal_fraction ] ;
08  Decimal_fraction: '.' Unsigned_integer;
09  Unsigned_integer: Digit+ ' ' ;
10  Digit: '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9';
```

В результате тестирования грамматика принята анализатором; сгенерированы файлы анализатора. Если необходимо получить компилятор, строки на языке Си вставляются в фигурных скобках после соответствующих правил грамматики.

Тестирование показало, что для грамматики G_2 преобразования не нужны.

РБНФ несколько отличается в системе ACCENT, а именно отсутствует `term+` и при использовании `term * term` следует заключать в круглые скобки. Строку 10 пришлось записать в виде:

```
Unsigned_integer:Digit (Digit)*;
```

При этом грамматика G_1 будет выглядеть следующим образом.

```
01  Program : [ | '-' ] Expression ;
02  Expression : Decimal_number |
03       Expression '+' Expression |
04       Expression '-' Expression  |
05       Expression '*' Expression |
06       Expression '/' Expression |
07       '(' [ | '-' ] Expression ')';
08  Decimal_number :Unsigned_integer | Unsigned_integer
                    Decimal_fraction ;
09  Decimal_fraction : '.' Unsigned_integer;
10  Unsigned_integer:Digit (Digit)*
11  Digit : '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9';
```


Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Результат тестирования: грамматика принята анализатором; сгенерированы файлы анализатора. Если необходимо получить компилятор, строки на языке Си вставляются в фигурных скобках после соответствующих правил грамматики. Требуется также файл с перечислением терминальных символов, им должны быть сопоставлены тексты на Си в фигурных скобках.

Тестирование показало, что для грамматики G_2 преобразования не нужны.

4. Выводы

I. По генератору-компилятору LLgen
   1. LLgen занимает 4 К байт.
   2. Система вследствие ограниченности типа грамматики может использоваться в учебных целях. Это и практика в эквивалентных преобразованиях грамматики: система обладает развитой диагностикой, включая синтаксические ошибки и диагностику левой рекурсии. Особое внимание стоит обратить на две ошибки: `"Repetition conflict"`, — когда система не может понять, сколько раз применять правило в случаях операции итерации, и встречавшуюся нам диагностику `"Alternation conflict"`, — когда несколько правил начинается одинаково и система не может установить, какое из них применить.
   3. Также полезно использовать LLgen в качестве компилятора, добавляя для каждого правила соответствующий код на языке Си в фигурных скобках. Этот текст перейдет в программу анализатора.

II. По генератору-компилятору ACCENT
   1. ACCENT занимает 117К байт на диске.
   2. Диагностика слабо развита, удалось получить только сообщение: “Illegal token”.
   3. Среди недостатков следует указать то, что получившийся анализатор работает до встретившейся ошибки.
   4. Система, похоже, действительно охватывает весь класс контекстно-свободных грамматик и может быть использована для построения компиляторов.

Литература

[1] *Ахо А., Ульман, Дж.* Теория синтаксического анализа, перевода и компиляции. Том 1, М: Мир, 1978 — 612 с.

[2] *Jacobs C. J. H.,* Some Topics in Parser Generation, <http://www.cs.vu.nl/~ceriel/LLgen.html>

[3] *Earley J.* An Efficient Context-Free Parsing Algorithm. Communications of the ACM, 13, N 2, 1970 — 125 p.

[4] *Levis P. M, Staerns R. E.* Syntax Directed Transduction. Journal of ACM, V.15,N3 ,1986 — 134 p.