Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

ПОСТРОЕНИЕ ЛЕКСИЧЕСКОГО И СИНТАКСИЧЕСКОГО АНАЛИЗАТОРОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Шевчука Александра Олеговича

Научный руководитель
зав. кафедрой, к.ф.-м.н. _____ С. В. Миронов

Заведующий кафедрой
к.ф.-м.н., доцент _____ С. В. Миронов

Саратов 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В связи с регулярной модернизацией элементарной базы и с появлением новых прикладных задач для компьютерной техники вопросы построения компиляторов остаются актуальными. Этапы разбора программы при компиляции — лексический и синтаксический анализы, являются базовыми этапами компиляции. Качественное проведение разбора является необходимым условием для успешного проведения дальнейших этапов компиляции: сборки и оптимизации целевого кода.

В настоящей работе ставится задача систематизировать информацию по построению лексического и синтаксического анализаторов. В ходе работы необходимо описать процесс построения анализаторов и привести пример такого построения для учебного языка программирования.

В первой главе рассматриваются теоретические основы компиляции. Во второй главе более подробно рассматривается лексический анализ, а так же метод построения лексического анализатора с помощью конечных автоматов. В третьей главе более подробно рассматривается синтаксический анализ, а так же метод построения синтаксический анализатора с использованием системы `ANTLR`. В четвертой главе описан процесс реализации лексического и синтаксического анализаторов для учебного языка программирования `Cool`.

1 Компилятор

1.1 Структура компилятора

Компилятор — это программа, которая считывает текст программы, написанный на одном языке — исходном, и транслирует его в эквивалентный текст на другом языке — целевом. Этот процесс трансляции называется компиляцией. [1]

Процесс компиляции разделяется на две части: анализ и синтез. Анализ разбивает исходную программу на составные части и накладывает на них грамматическую структуру. Затем он использует эту структуру для создания промежуточного представления исходной программы. Анализ также собирает информацию об исходной программе и сохраняет ее в структуре данных, именуемой таблицей символов, которая передается вместе с промежуточным представлением синтезу. Синтез строит требуемую целевую программу на основе промежуточного представления и информации из таблицы символов.

1.2 Фазы компилятора

Если рассмотреть процесс компиляции более детально, можно увидеть, что он представляет собой последовательность фаз, каждая из которых преобразует одно из представлений исходной программы в другое. Таким образом процесс компиляции можно разбить на несколько этапов:

- лексический анализ;
- синтаксический (грамматический) анализ;
- семантический анализ;
- оптимизация;
- генерация кода.

На практике некоторые фазы могут быть сгруппированы вместе. Современные компиляторы разбиваются на две части, называемыми препроцессором и постпроцессором. Первая часть включает лексический и синтаксический анализы и генерирует дерево, представляющее синтаксическую структуру исходного текста. Это дерево хранится в основной памяти и образует интерфейс для второй части, которая выполняет генерацию кода.

2 Лексический анализатор

2.1 Роль в процессе компиляции

Лексический анализатор — это часть компилятора, которая читает исходную программу и выделяет в ее тексте лексемы входного языка. То есть результатом работы лексического анализатора является перечень всех найденных в тексте исходной программы лексем.

Поскольку лексический анализатор представляет собой первую фазу компилятора, его основная задача состоит в чтении входных символов исходной программы, их группировании в лексемы и вывод последовательностей токенов для всех лексем исходной программы. Поток токенов пересылается синтаксическому анализатору для разбора. [2]

Так как лексический анализатор является частью компилятора, которая читает исходный текст, он может заодно выполнять и некоторые другие действия, помимо идентификации лексем. Например отбрасывание комментариев и пробельных символов, а так же синхронизация сообщений об ошибках, генерируемых компилятором, с исходной программой.

2.2 Разделение на лексический и синтаксический анализ

Имеется ряд причин, по которым фаза анализа компиляции разделяется на лексический и синтаксический анализы: [3]

1. Упрощение разработки;
2. Увеличивается эффективность компилятора;
3. Увеличивается портируемость компилятора.

2.3 Токены, шаблоны, лексемы

При рассмотрении лексического анализа используются три связанных, но различных термина:

- токен;
- шаблон;
- лексема.

Токен представляет собой пару, состоящую из имени токена и необязательного атрибута. Имя токена — это абстрактный символ, представляющий тип лексической единицы. Имена токенов являются входными символами, обрабатываемыми синтаксическим анализатором. [4]

Шаблон (pattern) — это описание вида, который может принимать лексема токена.

Лексема представляет собой последовательность символов исходной программы, которая соответствует шаблону токена и идентифицируется лексическим анализатором как экземпляр токена.

Исторически сложилось, что для описания лексических свойств чаще всего используются регулярные выражения. Регулярные выражения состоят из констант и операторов, которые определяют множества строк и множества операций на них соответственно. Каждый шаблон соответствует множеству строк, так что регулярные выражения можно рассматривать как имена таких множеств. [5]

2.4 Построение с использованием конечных автоматов

Одним из способов проектирования лексического анализатора, является использование конечных автоматов. Детерминированный конечный автомат (ДКА) — пятерка (Q, V, δ , q_0, F), где

- Q — конечное множество состояний;
- V — конечное множество входных символов;
- δ — функция переходов, аргументами которой являются текущее состояние и входной символ, а значением — новое состояние;
- q_0 — начальное состояние;
- F — множество допускающих, заключительных состояний.

Автомат начинает работу в начальном состоянии, далее последовательно считывает по одному символу входного слова (цепочки входных символов). Считанный символ переводит автомат в новое состояние в соответствии с функцией переходов. Читая входную цепочку символов x и делая переходы из состояния в состояние, автомат после прочтения последнего символа входного слова окажется в некотором состоянии $q*$. Если это состояние является заключительным, то есть принадлежит множеству F, то говорят, что автомат допустил слово x. [6]

Метод построения лексического анализатора с использованием конечных автоматов состоит в определении множества отдельных автоматов для каждой из лексем. Поскольку одно и то же слово может соответствовать нескольких автоматам, то вводится понятие приоритета для каждого автомата. Если одно и тоже слово допускается несколькими автоматами, то отдается предпочтение

автомату с большим приоритетом. Последовательность символов для лексического анализа подается на вход каждому автомату. Для каждого автомата определяется длина максимального префикса входной последовательности которая принимается этим автоматом. Из всех автоматов выбираются те которые допускают максимальный по длине префикс, и из выбранных автоматов выбирается тот, у которого наиболее высокий приоритет. Выбранный автомат и определяет ту лексему которой соответствует данный префикс. На следующем шаге этот префикс удаляется из строки для лексического анализа, и алгоритм повторяется для оставшейся строки до тех пор пока строка не станет пустой.

3 Синтаксический анализатор

3.1 Контекстно-свободная грамматика

Контекстно-свободная грамматика или просто грамматика используется для определения синтаксиса языка программирования. Грамматика естественным образом описывает иерархическую структуру множества конструкций языка программирования.

3.2 Формальное определение контекстно-свободной грамматики

Контекстно-свободная грамматика состоит из терминалов, нетерминалов, стартового символа и продукций.

Терминалы представляют собой базовые символы, из которых формируются строки. Терминалы являются первыми компонентами токенов, возвращаемых лексическим анализатором.

Нетерминалы представляют собой синтаксические переменные, которые обозначают множества строк. Нетерминалы налагают на язык иерархическую структуру, облегчающую синтаксический анализ и трансляцию.

Стартовым символом считается один из нетерминалов грамматики, и множество строк, которые он обозначает, является языком, определяемым грамматикой.

Продукции грамматики определяют способ, которым терминалы и нетерминалы могут объединяться для создания строк.

3.3 Форма Бэкуса - Наура

Форма Бэкуса — Наура (далее БНФ) — формальная система описания синтаксиса, в которой одни синтаксические категории последовательно определяются через другие категории. БНФ используется для описания контекстно — свободных формальных грамматик. Существует расширенная форма, отличающаяся лишь более ёмкими конструкциями. [7]

3.4 Роль синтаксического анализатора

Синтаксический анализатор получает строку токенов от лексического анализатора и проверяет, может ли эта строка имен токенов порождаться грамматикой исходного языка. В случае корректной программы синтаксический анализатор строит дерево разбора и передает его следующей части компилятора для дальнейшей обработки. [7]

3.5 Дерево разбора

Дерево разбора может рассматриваться как графическое представление порождения, из которого удалена информация о порядке замещения нетерминалов.

3.6 Построение синтаксических анализаторов

Одним из способов построения синтаксических анализаторов является использование синтаксических генераторов. Существуют достаточно большое количество подобных специализированных инструментов, позволяющих автоматизировать процесс построения синтаксических анализаторов. В дипломной работе рассматривается способ построения с использованием генератора ANTLR.

3.7 Синтаксический генератор ANTLR

ANTLR (ANother Tool for Language Recognition) — генератор нисходящих анализаторов для формальных языков. Используется для разработки компиляторов, интерпретаторов и трансляторов. ANTLR состоит из двух основных частей:

— Генератора анализаторов — приложения, которое получает на вход описание грамматики в БНФ форме и генерирует код для лексического и синтаксического анализатора;

— Runtime — библиотеки, которая используется для создания конечной программы.

ANTLR автоматически генерирует лексический и синтаксический анализатор на основе описанной контекстно-свободной грамматике в БНФ форме. Для удобства генератор ANTLR имеет графический интерфейс для построения дерева разбора.

3.8 Описание грамматики

Грамматики для ANTLR четвертой версии описывается в файле с расширением .g4. Описание терминальных символов начинается с большой буквы, нетерминальных — с маленькой. Грамматика начинается с задания имени грамматики после ключевого слова grammar. Имя грамматики обязательно должно совпадать с именем файла, в котором грамматика описана.

4 Лексический и синтаксический анализаторы для языка Cool

4.1 Описание языка Cool

Cool представляет собой небольшой язык программирования, использующийся в учебных целях. Данный язык является функциональным строготипизированным, то есть проверка типов происходит статически на этапе компиляции. Программа написанная на языке Cool представляет собой набор классов. Классы состоят из атрибутов и функций (feature). [8]

4.2 Лексический анализатор для языка Cool

В процессе выполнения дипломной работы был создан лексический анализатор для языка Cool. Для разработки приложения использовался язык Python3, а так же операционная система ubuntu 14.04. На основе лексем языка Cool была написана программа, представляющая собой лексический анализатор использующий автоматный подход. На вход программе, передается файл с кодом программы, написанной на языке Cool. После чего лексический анализатор обрабатывает входные данные, разбивает программу на токены, и выводит их в виде результата.

4.3 Синтаксический анализатор для языка Cool

В ходе дипломной работы был создан синтаксический анализатор с помощью генератора ANTLR. Была описана грамматика языка Cool на основе выделенных продукций: program, classDef, featureList, feature, formalList, formal, expr, not, relation, addition, multiplication, isvoid, neg, dot, term, invokeExprs, letExprs, localOrFieldInit, caseExpr, typeName. С помощью ANTLR созданы лексический и синтаксический анализаторы. С использованием графического интерфейса генератора продемонстрировано дерево вывода для программы написанной на языке Cool.

ЗАКЛЮЧЕНИЕ

В ходе работы описан процесс построения лексического и синтаксического анализаторов. Для учебного языка программирования `Cool` был разработан лексический анализатор на базе подхода, использующего конечные автоматы. Рассмотрен процесс построения анализаторов с помощью системы `ANTLR`. Приведен пример построения анализаторов с использованием системы `ANTLR` для учебного языка программирования `Cool`.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Хантер, Р.* Проектирование и конструирование компиляторов / Р. Хантер. — Москва: ФиС, 1984.

2 *Мозговой, М.* Классика programмирования: алгоритмы, языки, автоматы, компиляторы. Практический подход. / М. Мозговой. — Санкт-Петербург: Наука и техника, 2006.

3 *Льюис, Ф.* Теоретические основы проектирования компиляторов. / Ф. Льюис, Д. Розенкранц. — Москва: Мир, 1979.

4 Лексический анализ [Электронный ресурс]. — URL: http://www.ict.edu.ru/ft/005128/ch5.pdf (дата обращения: 30.02.2016). Загл. с экр. Яз. рус.

5 Лексические анализаторы [Электронный ресурс]. — URL: http://storage.piter.com/upload/contents/978549807153/978549807153_p.pdf (дата обращения: 28.02.2016). Загл. с экр. Яз. рус.

6 *Хопкрофт, Д.* Введение в теорию автоматов, языков и вычислений. / Д. Хопкрофт, Р. Мотвани. — Москва: Вильямс, 2002.

7 *Ахо, А.* Теория синтаксического анализа, перевода и компиляции. / А. Ахо, Д. Ульман. — Москва: Мир, 1978.

8 The Cool Reference Manual [Электронный ресурс]. — URL: https://theory.stanford.edu/~aiken/software/cool/cool-manual.pdf (дата обращения: 7.03.2016). Загл. с экр. Яз. англ.