Лабораторная работа №1 “Использование конечного автомата для построения лексического анализатора”

1 Цель работы

Целью проведения лабораторной работы является:

* закрепление теоретических знаний в области построения лексических анализаторов;
* получение начальных навыков описания языков в виде регулярных выражений;
* приобретение опыта построения лексических анализаторов с использованием конечных автоматов;
* получение навыков реализации конечных автоматов.

2 Построение лексического анализатора.

Основная задача лексического анализа – разбить входной текст, состоящий из последовательности одиночных символов, на последовательность слов (лексем). Все лексемы делятся на классы. Примерами таких классов являются числа, идентификаторы, строки. Отдельно выделяют ключевые слова. Как правило, ключевые слова – это некоторое ограниченное подмножество идентификаторов.

С точки зрения дальнейших этапов работы компилятора, лексический анализатор выдает информацию двух видов: класс лексемы и значение лексемы. Ключевые слова распознаются либо явным выделением из входной последовательности, либо сначала выделяется идентификатор, а затем делается проверка на принадлежность его множеству ключевых слов.

Лексический анализатор может работать или как самостоятельная фаза трансляции, или как подпрограмма, работающая по принципу “дай лексему”. В первом случае выходом лексического анализатора является файл лексем, во втором лексема выдается при каждом обращении к лексическому анализатору.

Работа лексического анализатора описывается и реализуется формализмом конечных автоматов. Однако непосредственное описание конечного автомата неудобно. Поэтому для описания лексических анализаторов применяют другие подходы, например формализм регулярных выражений.

Будем создавать лексический анализатор по следующей схеме:

* описание распознаваемого языка в форме регулярных выражений;
* преобразование регулярных выражений в недетерминированный конечный автомат;
* преобразование недетерминированного конечного автомата к детерминированному;
* минимизация конечного автомата.

Рассмотрим построение лексического анализатора на примере распознавания буквы азбуки Морзе. Буква состоит из последовательности точек и тире, после каждого такого символа должен следовать пробел. Буква заканчивается пробелом. Буква может быть пустой, то есть состоять из одного символа пробела.

2.1 Описание распознаваемого языка в форме регулярного

выражения.

Определим, что является регулярным выражением:

1 Пустая строка ε является регулярным выражением.

2 Если элемент а принадлежит алфавиту описываемого языка, то а – регулярное выражение.

3 Если r и s регулярные выражения, обозначающие языки L(r) и L(s), то регулярными являются следующие выражения:

а) r | s – объединение языков, то есть утверждается, что элемент должен принадлежать или языку L(r) или языку L(s);

б) r s – конкатенация языков, то есть утверждается, что за элементом языка L(r) следует элемент языка L(s);

в) r\* - замыкание Клини языка, то есть ноль или более символов языка L(r);

г) (r) – регулярное выражение в скобках

Кроме того, для описания языков используются операция r+, означающая один или более символов языка r, и операция r?, означающая ноль или один символ языка r.

Опишем букву азбуки Морзе в форме регулярного выражения. Сначала рассмотрим базовый случай. Элементы точка, тире и пробел принадлежат алфавиту распознаваемого языка, следовательно, существуют регулярные выражения “точка”, “тире” и “пробел” (по правилу 2). За каждой точкой или тире следует пробел, следовательно, существуют регулярные выражения “точка пробел” и “тире пробел” (по правилу 3б). В букве может встречаться либо точка с пробелом, либо тире с пробелом, следовательно, имеет место регулярное выражение “точка пробел | тире пробел” (по правилу 3а). Такая последовательность может встретиться в букве множество раз, а может и не встретиться вообще, то есть регулярное выражение имеет вид “(точка пробел | тире пробел)\*” (по правилам 3г и 3в). Буква должна заканчиваться пробелом, то есть регулярное выражение для распознавания буквы азбуки Морзе выглядит следующим образом

(точка пробел | тире пробел)\* пробел

2.2 Преобразование регулярного выражения в недетерминированный конечный автомат

Конечный автомат это пятерка (S, Σ, δ, S0, F)

S - конечное множество состояний.

Σ - конечное множество допустимых входных сигналов.

δ - функция переходов. Она отражает множество Sх(Σ∪{ε}) в множество состояний недетерминированного конечного автомата. Для детерминированного автомата функция переходов отражает множество SхΣ во множество состояний автомата. Другими словами, в зависимости от состояния и входного символа, δ определяет новое состояние автомата.

S0 - начальное состояние конечного автомата, S0 ∈ S.

F - множество конечных состояний автомата, F ∈ S.

Работа конечного автомата представляет собой последовательность шагов. Шаг определяется состоянием автомата и входным символом. Сам шаг состоит в изменении состояния автомата и считывании следующего символа входной последовательности.

Существуют следующие правила преобразования регулярных выражений в конечный автомат.

1 Регулярное выражение “ε” преобразуется в автомат из двух состояний и ε -перехода между ними (рисунок 1).

ε

Рисунок 1. – Автомат для ε-перехода

2 Регулярное выражение из одного символа “а” преобразуется в конечный автомат из двух состояний и перехода между ними по входному сигналу а (рисунок 2).

а

Рисунок 2. – Автомат для перехода по символу а

3 Пусть есть регулярное выражение rs и уже построены конечные автоматы для выражения r и выражения s. Тогда два автомата соединяются последовательно. На рисунке 3 представлены исходные автоматы для языков r и s. На рисунке автомат для распознавания конкатенации этих языков.

Автомат для r Автомат для s

а

b

Рисунок 3. – Исходные автоматы

а

b

Рисунок 4. – Автомат для конкатенации языков

4 Пусть есть регулярное выражение r | s и уже построены конечные автоматы для выражения r и выражения s (рисунок 3). Тогда в результирующем автомате должна быть альтернатива выполнения одного из двух автоматов. То есть автомат для выражения r | s при автоматах для r и s с рисунка 3 имеет вид, представленный на рисунке 5.

b

а

ε

ε

ε

ε

Рисунок 5. – Автомат для объединения языков

5 Пусть есть регулярное выражение r\* при построенном конечном автомате r. В этом случае вводятся два новых состояния для возможности обхода автомата выражения r, а также вводится ε-переход между конечным и начальным состояниями для возможности многократного повторения автомата r. Если для регулярного выражения r построен автомат аналогичный рисунку 3, то регулярному выражению r\* соответствует конечный автомат, представленный на рисунке 6.

а

ε

ε

ε

ε

Рисунок 6. – Автомат для замыкания Клини языка

Построим конечный автомат для регулярного выражения

(точка пробел | тире пробел)\* пробел

Сначала построим конечные автоматы для регулярных выражение “точка”, “тире” и “пробел” по правилу 2. Эти автоматы представлены на рисунке 7.

тире

точка

пробел

Рисунок 7

Автомат для регулярных выражений “точка пробел” и “тире пробел” строим конечные автоматы по правилу 3 (рисунок 8).

тире

пробел

точка

пробел

Рисунок 8

Регулярное выражение “точка пробел | тире пробел” преобразуем по правилу 4. Соответствующий автомат изображен на рисунке 9.

точка

пробел

тире

пробел

ε

ε

ε

ε

Рисунок 9

Теперь строим конечный автомат для регулярного выражения (точка пробел | тире пробел)\*, используя правило 5 (рисунок 10).

точка

пробел

тире

пробел

ε

ε

ε

ε

ε

ε

ε

ε

Рисунок 10

Применяя правило 3 для построения конечного автомата для полного регулярного выражения, получим следующий автомат для распознавания буквы азбуки Морзе, представленный на рисунке 11.

точка

пробел

тире

пробел

ε

ε

ε

ε

ε

ε

ε

ε

пробел

0

1

2

3

4

5

6

7

8

10

9

Рисунок 11. – Недетерминированный конечный автомат

2.3 Преобразование недетерминированного конечного автомата к детерминированному

Работа конечного автомата заключается в нахождении пути от начального состояния в конечное, по которому можно прочитать лексему. Лексема принимается, если такой путь найден. Недетерминированный конечный автомат включает ε-переходы и переходы из одного состояния по одному и тому же входному символу в разные состояния. Поэтому при работе недетерминированного автомата возможны откаты в процессе поиска пути, вследствие этого недетерминированный автомат работает медленно. Для каждого недетерминированного конечного автомата можно построить детерминированный конечный автомат.

Рассмотрим алгоритм преобразования. Входом алгоритма является недетерминированный автомат N, выходом – детерминированный автомат D состоящий из множества состояний Dstates и множества переходов Dtrans.

Введем обозначения.

S - состояние из N;

T - множество состояний из N. При работе алгоритма такие множества становятся состояниями D.

Пусть реализованы следующие функции.

εclosure(S). Эта функция строит ε-замыкание состояния S, то есть множество состояний в которые можно перейти из S по ε-переходам, это множество включает также само состояние S;

εclosure(T). Функция строит ε-замыкание множества состояний T, то есть множество состояний, достижимых из множества состояний T через ε-переходы. Это множество включает и состояния, принадлежащие T.

Move(T, a). Функция строит множество состояний, в которые можно перейти из Т по входному символу а.

Алгоритм состоит в следующем.

Dstates:=εclose(S0);

Пока в Dstates хотя бы одно множество Т непомеченно выполнить

begin

пометить множество Т

Для каждого входного символа а выполнить

begin

U:=εclosure(Move(T, a));

Если U ∉ Dstates

то Dstates:=Dstates+U;

Dtrans[T, a]:=U;

end;

end;

Рассмотрим преобразование недетерминированного автомата с рисунка 11 к детерминированному.

Определим εclosure(S0). Через ε-переходы из нулевого стартового состояния можно попасть в состояния 1,2,5,9. Таким образом, в Dstates заносим множество {0,1, 2, 5, 9}.

На первой итерации цикла помечаем множество T={0,1, 2, 5, 9}.

Для входного символа "точка" U={3}, так как по этому символу из состояний множества {0,1, 2, 5, 9} можно перейти только в третье состояние, а из него ε-переходов нет. Dstates={{0,1, 2, 5, 9}',{3}}. Помещаем множество {3} в таблицу переходов в качестве нового состояния при текущем состоянии {0,1, 2, 5, 9} и входном символе “точка”, то есть Dtrans[{0,1, 2, 5, 9}, точка]:={3}. Для входного сигнала "тире" U={6}. Для символа "пробел" U={10}. После первой итерации внешнего цикла конечный автомат имеет вид Dstates={{0,1, 2, 5, 9}', {3}, {6}, {10}}, Dtrans =

|  |  |  |  |
| --- | --- | --- | --- |
|  | точка | тире | пробел |
| {0,1, 2, 5, 9} | {3} | {6} | {10} |

На второй итерации рассматриваем Т={3}, так как это первое непомеченное состояние в Dstate. Из этого множества возможен переход только по символу "пробел" в состояние 4. Из состояния 4 существуют ε-переходы в состояния 8, 9, 1, 2, 5. После выполнения второй итерации автомат имеет вид. Dstate={{0,1, 2, 5, 9}',{3}', {6}, {10}, {4, 8, 9, 1, 2, 5}}, Dtrans=

|  |  |  |  |
| --- | --- | --- | --- |
|  | точка | тире | пробел |
| {0,1, 2, 5, 9} | {3} | {6} | {10} |
| {3} | - | - | {4, 8, 9, 1, 2, 5} |

Рассмотрение множества {6} добавит в автомат множество {7, 8, 9, 1, 2, 5} и соответствующий переход. Из множества {10} нет никаких переходов, так как оно конечное, поэтому четвертая итерация цикла ничего к графу не добавит. После этого конечный автомат будет иметь вид

Dstates={{0,1, 2, 5, 9}',{3}', {6}', {10}', {4, 8, 9, 1, 2, 5}, {7, 8, 9, 1, 2, 5}}

Dtrans=

|  |  |  |  |
| --- | --- | --- | --- |
|  | Точка | тире | пробел |
| {0,1, 2, 5, 9} | {3} | {6} | {10} |
| {3} | - | - | {4, 8, 9, 1, 2, 5} |
| {6} | - | - | {7, 8, 9, 1, 2, 5} |
| {10} | - | - |  |

Из множества {4, 8, 9, 1, 2, 5} по символу "пробел" можно перейти в множество {10}, по символу "тире" – в множество {6}, а по символу "точка" – в состояние {3}. Таким образом, рассмотрение множества {4, 8, 9, 1, 2, 5} не приведет к добавлению к автомату новых состояний, но добавит соответствующие переходы. Аналогичные рассуждения можно провести для множества {7, 8, 9, 1, 2, 5}. После рассмотрения этих двух состояний в Dstates не останется непомеченных множеств, следовательно, алгоритм завершит работу. При этом Dtrans имеет вид

|  |  |  |  |
| --- | --- | --- | --- |
|  | Точка | тире | пробел |
| {0,1, 2, 5, 9} | {3} | {6} | {10} |
| {3} | - | - | {4, 8, 9, 1, 2, 5} |
| {6} | - | - | {7, 8, 9, 1, 2, 5} |
| {10} | - | - |  |
| {4, 8, 9, 1, 2, 5} | {3} | {6} | {10} |
| {7, 8, 9, 1, 2, 5} | {3} | {6} | {10} |

Таким образом, конечный автомат будет иметь вид, представленный на рисунке 12.

S0={0,1, 2, 5, 9}

S1={3}

S2={6}

S3={4, 8, 9, 1, 2, 5}

S4={7, 8, 9, 1, 2, 5}

S5={10}

точка

тире

пробел

пробел

пробел

точка

тире

точка

тире

пробел

пробел

2.4 Минимизация конечного автомата

При минимизации конечного автомата все состояния делятся на две группы: финальные и нефинальные. Затем выполняется следующая процедура. Рассматриваем группу {S1, S2, …, Sk} и некоторый входной символ а. Если по этому входному символу есть переходы в разные группы, то исходную группу разбивают на части. Принадлежность Si к той или иной части зависит от того, в какую группу существует переход из состояния Si по входному символу а. Эта процедура продолжается до тех пор, пока не останется ни одной группы, которую можно было бы разбить по какому-либо входному сигналу.

Минимизируем конечный автомат, приведенный на рисунке 12. Разбиваем все состояния на две группы. В первую войдет конечное состояние S5, а во вторую все неконечные состояния, то есть группа имеет вид {S0, S1, S2, S3, S4}.

Первую группу разделить не возможно, так как она состоит из единственного состояния. Рассмотрим входной сигнал “точка” и вторую группу. По этому входному символу есть переходы только из состояний S0, S3, S4 в состояние S1, то есть для всех состояний группы по входному символу “точка” осуществляется переход в одну и ту же группу. Следовательно, входной сигнал “точка” не разбивает группу. Аналогичные рассуждения можно провести и для входного сигнала “тире”. Теперь рассмотрим входной символ “пробел”. По этому сигналу из состояний S1 и S2 осуществляется переход в состояния той же группы, а из состояний S0, S3, S4 – в состояние S5, то есть в другую группу. Таким образом, группу {S0, S1, S2, S3, S4} делим на две {S0, S3, S4} и {S1, S2}. Разбить получившиеся группы нельзя ни по какому входному сигналу. Таким образом, минимизированный конечный автомат будет иметь три состояния. Рассмотрим переходы между состояниями. Как видно по рисунку 12 из множества состояний {S0, S3, S4} существует переход в множество состояний {S1, S2} по символам “точка” и “тире”, а по символу “пробел” - переход в множество {S5}. Из множества {S1, S2} можно выполнить переход в множество состояний {S0, S3, S4} по символу “пробел”. Из состояния S5 переходов нет. Минимизированный конечный автомат для распознавания буквы азбуки Морзе представлен на рисунке 13.

0

1

2

тире

точка

пробел

пробел

Рисунок 13. – Минимизированный конечный автомат

2.5 Конечный автомат для распознавания нескольких лексем

Распознаваемые языки включают более чем одну лексему. Рассмотрим на примере как должен быть построен конечный автомат для такого языка.

Пусть язык включает два класса лесем.

Первый описывается регулярным выражением (цифра)+пробел. Для распознавания этой лексемы построен конечный автомат, приведенный на рисунке14.

0

2

1

цифра

пробел

цифра

Рисунок 14. – Автомат для распознавания лексемы (цифра)+пробел

Второй класс лексем описывается выражением буква(цифра)\*пробел и распознается автоматом, представленным на рисунке 15.

0

2

1

буква

пробел

цифра

пробел

Рисунок 15. – Автомат для распознавания лексемы буква(цифра)\*пробел

Результирующий автомат для распознавания двух лексем представлен на рисунке 16. В таком автомате получается два конечных состояния. В зависимости от того, в каком финальном состоянии оказался автомат определяется класс распознанной лексемы.

0

2

1

буква

пробел

цифра

пробел

4

3

цифра

пробел

цифра

Рисунок 16

2.6 Распознавание ключевых слов

Как уже упоминалось, ключевые слова могут распознаваться явным образом, то есть для них строится собственный конечный автомат.

Рассмотрим, как распознать язык, состоящий либо из идентификатора, заканчивающегося пробелом, либо из служебного слова if, за которым также следует пробел. Для этого в качестве входных символов необходимо отдельно выделить символ “i”, символ “f”, все символы без “i” и “f”, а также символ “пробел”. Детерминированный автомат для такого языка представлен на рисунке 17.

без “i” и “f”

0

1

2

4

i

f

пробел

3

5

f

i

без “i” и “f”

все входные сигналы, кроме сигнала “пробел”

пробел

Рисунок 17

В приведенном автомате финальное состояние 3 соответствует распознанной лексеме класса “служебное слово”, а финальное состояние 5 – лексеме класса “идентификатор”.

Этот же язык может распознаваться с использованием недетерминированного автомата. Тогда из состояния 0 в состояние 4 можно попасть еще и по входному символу “i”. Кроме того, будет отсутствовать переход из состояния 2 в состояние 4. Работать этот автомат будет так. Если пришел сигнал “i”, то автомат перейдет в состояние 1. Если в состоянии 1 или 2 на входе автомата появиться неожидаемый сигнал, то произойдет откат до состояния 0 и работа автомата продолжится переходом в состояние 4.

Второй вариант распознавания ключевых слов не подразумевает построения специальных автоматов для ключевых слов. В этом случае анализатор должен иметь таблицу, содержащую ключевые слова языка. После распознавания идентификатора лексический анализатор сравнивает значение лексемы с элементами таблицы ключевых слов. Если было найдено совпадение, то лексема относится к классу “ключевое слово”, иначе – к классу “идентификатор”.

3 Реализация лексического анализатора в виде конечного автомата

Рассмотрим реализацию в виде конечного автомата лексического анализатора построенного ранее.

Определим тип “входной сигнал”. Этот тип должен включать нижеприведенные элементы.

* “Точка” (dot), “тире” (dash) и “пробел” (space) для отражения входных сигналов конечного автомата.
* “Другой символ” (other). Этот элемент необходим для отражения всех символов, не входящих в алфавит азбуки Морзе, но появляющихся на входе конечного автомата.
* “Конец последовательности” (endf). Этот элемент необходим, так как автомат должен знать, когда заканчивается последовательность, которую нужно распознать как букву или как ошибочную последовательность.

type

input\_signal = (dot, dash, space, other, endf)

Определяем возможные состояния конечного автомата. Согласно графу это будут состояния S0, S1, S2, а также специальное состояние S\_error, которое будет соответствовать ошибке, возникшей при разборе.

type

state = (S0, S1, S2, S\_error)

Исходя из рисунка 13, определим функцию переходов конечного автомата. В ошибочное состояние автомат переводит любой, неожидаемый сигнал. Сигнал “Другой символ” переводит автомат в ошибочное состояние из любого состояния автомата.

const

next\_state: array [dot..other, S0..S1] of state =

((S1, S\_error),

(S1, S\_error),

(S2, S0),

(S\_error, S\_error));

В рассматриваемом примере только один класс лексем, но обычно автомат распознает лексемы нескольких классов, поэтому введем тип “класс лексемы”. В данной реализации этот тип будет включать только одно значение - “буква”.

type lexeme\_class=(letter)

Реализуем работу конечного автомата, считая что глобальная переменная Entry : string содержит входную последовательность. Переменные cur\_state и cur\_input необходимы для хранения текущего состояния автомата и входного символа. Глобальная переменная lex\_val будет содержать значение распознаваемой лексемы. В начале работы автомата эта переменная инициализируется пустой строкой, а формирование значение лексемы будет производиться при распознавании символа входной последовательности. Приведенная ниже функция возвращает класс распознанной лексемы. Работа функции заключается в переходе по состояниям конечного автомата до тех пор, пока не будет достигнуто конечное состояние или входная последовательность не разобрана полностью. Если автомат перешел в ошибочное состояние, то в системе возникает исключительная ситуация и пользователь получает сообщение об ошибке. Такое же сообщение пользователь получит, если просмотрена вся входная последовательность, а автомат не находится в конечном состоянии.

function state\_machine : lexeme\_class;

var cur\_state:state; cur\_input:input\_signal;

begin

lex\_val:=''; cur\_state:=s0; cur\_input:=recognize;

while (cur\_state<>S2) and (cur\_input<>endf) do

begin

cur\_state:=next\_state[cur\_input, cur\_state];

if cur\_state=S\_error

then raise exception.create(‘Лексическая ошибка');

if cur\_state<> s2 then cur\_input:=recognize;

end;

if (cur\_state <> S2) and (cur\_input=endf)

then raise exception.create('Лексическая ошибка')

else result:=letter;

end;

При реализации конечного автомата была использована функция recognize, которая возвращает следующий символ входной последовательности. Эта функция необходима, так как последовательность имеет строковый тип, а для входных символов автомата задан свой тип input\_signal. Кроме того, функция формирует значение лексемы.

function recognize:input\_signal;

begin

if entry=''

then result:=endf

else

begin

case entry[1] of

'.': result:=dot;

'-': result:=dash;

' ': result:=space;

else result:=other

end;

lex\_val:=lex\_val+entry[1];

entry:=copy(entry, 2,length(entry));

end;

end;

Теперь рассмотрим текст основной программы. Она будет состоять в вызове подпрограммы лексического анализатора до тех пор, пока не будет разобрана вся входная последовательность. В зависимости от класса распознанной лексемы основная программа выдает пользователю сообщение, включающее класс и значение лексемы. Будем считать, что входная последовательность вводится пользователем через объект класса TEdit.

begin

entry:=edit1.text;

while (entry<>'') do

begin

case state\_machine of

letter:showmessage('Буква'+lex\_val);

end;

end;

end;

3.4 Контрольные вопросы

1 В чем состоит основная задача лексического анализа?

2 Какую информацию должен выдавать лексический анализатор?

3 Что такое ключевые слова и как они выделяются из входной последовательности?

4 Что такое регулярное выражение?

5 Каким образом описывается конечный автомат?

6 Чем определяется и в чем состоит шаг работы конечного автомата?

7 Каким образом можно преобразовать к недетерминированному конечному автомату регулярные выражения r+ и r?.

3.5 Задание

Создать лексический анализатор, выделяющий из входной последовательности следующие лексемы:

* "идентификатор" - последовательность букв и цифр, начинающаяся с буквы;
* "число" - последовательность цифр, целая часть от дробной отделяется точкой или запятой, причем дробная часть может отсутствовать;
* "операция сравнения" - строки "==" или "!=";
* "знак операции" - символы + или – ;
* ключевые слова if и switch.

Лексемы отделяются друг от друга пробелом или начинаются с новой строки.

Для лексемы “число” построить недерминированный автомат, затем его детерминировать и минимизировать. Для остальных лексем эти действия можно не выполнять.

Пример

входная последовательность

3434334 eree wew3

3434.4 + :=

Результат

число 3434334

идентификатор eree

идентификатор wew3

число 3434.4

знак операции +

операция сравнения ==

При выполнении лабораторной работы можно использовать допущение, что входная последовательность не превышает 255 символов.

Написание регулярных выражений и построение автоматов выполняется студентом при подготовке к лабораторной работе. Реализация конечного автомата производится в ходе аудиторного занятия.

6 Содержание отчета

Отчет должен содержать:

* Регулярные выражения, описывающие распознаваемые лексемы.
* Недетерминированный, детерминированный и минимизированный автоматы для распознавания лексемы “число”.
* Конечный автомат для распознавания всех лексем, перечисленных в задании.

7 Защита лабораторной работы

При защите лабораторной работы студент демонстрирует работу программы и объясняет, как задан и как работает построенный конечный автомат, то есть каким образом автомат меняет состояния для различных видов входной последовательности.

Кроме того, студент должен пояснить, исходя из каких положений задания, были написаны регулярные выражения, а также наиболее важные моменты построения детерминированного конечного автомата.