ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Костюк Ю. Л.

ТЕОРИЯ АВТОМАТОВ И ФОРМАЛЬНЫХ ЯЗЫКОВ

Лекция 5

Грамматика со сравнениями

Результат операции сравнения (=, ≠, <, >, ≤, ≥) – истина (true) или ложь (false). Во входной цепочке символов операция ≠ может кодироваться, например, как <>, операция ≤ в виде <=, операция ≥ в виде >=.

Порождающие правила для сравнения двух выражений (например, операциями < и >):

$C \rightarrow S<SZ \mid S>SZ$

После преобразования к нормальной форме Грейбах и факторизации:

$C \rightarrow (S)VUD \mid aHVUD \mid kVUD \mid +GVUD \mid -GVUD$

$D \rightarrow <SZ \mid >SZ$

Семантические действия генератора ОПС для нетерминала *C*:

□□□□□□ | *a*□□□□ | *k*□□□ | □□□□□ | □□–'□□

а для нетерминала *D* соответственно:

□□< | □□>

Грамматика с условными операторами

Порождающие правила, задающие условные операторы в полной и сокращенной форме (служебные слова **if**, **then**, **else** – терминалы):

$A \rightarrow$ **if** C **then** AEZ

$E \rightarrow$ **else** $A \mid \lambda$

Семантические действия генератора ОПС при порождении условных операторов нетерминалом A:

□□**1**□□**3**

Семантические действия для нетерминала E:

2□

Числа **1, 2, 3** в семантических действиях обозначают выполнение семантических программ, генерирующих в ОПС операнды-метки (номера элементов ОПС) и операции условного и безусловного перехода на метки.

Грамматика с циклами

Порождающее правило для задания оператора цикла

(служебные слова **while**, **do** – терминалы):

$A \rightarrow$ **while** C **do** AZ

Семантические действия генератора ОПС при порождении цикла:

4□1□5

Числа **1, 2, 3, 4, 5** в семантических действиях обозначают выполнение семантических программ, генерирующих в ОПС операнды-метки (номера элементов ОПС) и операции условного и безусловного перехода на эти метки.

Семантические программы используют счетчик k – номер очередного генерируемого элемента ОПС , а также еще один магазин – магазин меток.

Программа 1.

1. В магазин меток записывается k.
2. В ОПС записывается пустой элемент – место для будущей метки.
3. В ОПС записывается операция **jf** – переход при условии false.

Программа 2.

1. Через верхний элемент магазина меток, как ссылку на ранее заготовленное место для метки, записывается k + 2.
2. В магазин меток записывается k.
3. В ОПС записывается пустой элемент – место для будущей метки.
4. В ОПС записывается операция **j** – безусловный переход.

Программа 3.

1. Через верхний элемент магазина меток, как ссылку на ранее заготовленное место для метки, записывается k.

Программа 4.

1. В магазин меток записывается k.

Программа 5.

1. Через верхний элемент магазина меток, как ссылку на ранее заготовленное место для метки, записывается k + 2.
2. В ОПС записывается метка, значение для которой читается из магазина меток.
3. В ОПС записывается операция **j** – безусловный переход.

При исполнении ОПС метки, как и другие операнды, записываются в магазин интерпретатора.

Выполнение операции безусловного перехода **j**:

1) из магазина извлекается операнд (метка);
2) счетчику, который содержит текущий номер исполняемого элемента ОПС, присваивается значение метки (т.е. производится переход), в магазин ничего не записывается.

Выполнение операции условного перехода **jf**:

1) из магазина извлекается 1-й операнд (true или false)
2) из магазина извлекается 2-й операнд (метка);
3) если 1-й операнд false, то счетчику, который содержит текущий номер исполняемого элемента ОПС, присваивается значение 2-го операнда (т.е. производится переход),

   если 1-й операнд true, то перехода не производится,

   и в обоих случаях в магазин ничего не записывается.

Пример. На входе цепочка:

if a>b **then** a:=b **else** b:=a $\perp$

будет сгенерирована следующая ОПС:

a b > m_1 **jf** a b := m_2 **j** b a :=

↑ ↑

m_1 m_2

Вычисление ОПС при a = 5, b = 2.

Шаг	Магазин	Операция	a	b
1	a, b	>	5	2
2	true, m_1	**jf**	5	2
3	a, b	:=	5	2
4	m_2	**j**	2	2
5				

Сгенерированная ОПС:

a b > m_1 **jf** a b := m_2 **j** b a :=

↑ ↑

m_1 m_2

Вычисление ОПС при a = 2, b = 5.

Шаг	Магазин	Операция	a	b
1	a, b	>	5	2
2	false, m_1	**jf**	5	2
3	b, a	:=	5	2
4			5	5

Пример. На входе цепочка:

while a>b **do** a:=b ⊥

будет сгенерирована следующая ОПС:

a b > m_1 **jf** a b := m_0 **j**

↑ ↑

m_0 m_1

Вычисление ОПС при a = 5, b = 2.

Шаг	Магазин	Операция	a	b
1	a, b	>	5	2
2	true, m_1	**jf**	5	2
3	a, b	:=	5	2
4	m_0	**j**	2	2
5	a, b	>	2	2
6	false, m_1	**jf**	2	2

Грамматика с составными операторами

Порождающие правила для задания составного оператора, т.е. последовательности других операторов (точка с запятой и служебные слова **begin**, **end**,– терминалы):

$$A \rightarrow \textbf{begin}\ AQ\ \textbf{end}$$
$$Q \rightarrow ;AQ \mid \lambda$$

При этом семантические действия генератора ОПС для нетерминала A:

□□□□

а для нетерминала Q соответственно:

□□□

Грамматика с операторами ввода и вывода

Порождающие правила для задания стандартных операторов ввода и вывода (служебные слова **read**, **write** – терминалы):

$$A \rightarrow \textbf{read}\ (aH) \mid \textbf{write}\ (S)$$

Семантические действия генератора ОПС соответственно:

$$\square\square a\square\mathbf{r} \quad | \quad \square\square\square\mathbf{w}$$

r – операция чтения со стандартного устройства ввода в переменную, операнд должен быть ссылкой на переменную

w - вывод значения арифметического выражения в стандартное устройство вывода, операнд – числовое значение.

Пример. Задана входная цепочка:

begin read(a); **read**(M[a]); **write**(M[a]*a) **end**

будет сгенерирована ОПС:

a **r** M a **i** **r** M a **i** a * **w**

Вычисление этой ОПС:

Шаг	Магазин	Операция
1	a	**r**
2	M, a	**i**
3	M[a]	**r**
4	M, a	**i**
5	M[a], a	*
6	M[a]*a	**w**

Распределение памяти и описание переменных

Статический способ. Память распределяется в процессе трансляции. Для каждой переменной вычисляется ее адрес (размещение в памяти относительно начала выделенного участка). При использовании переменной подставляется ее адрес.

Этот способ используется для глобальных переменных, существующих от начала до самого конца выполнения программы. Если это массивы, то их длина должна быть задана константой.

Динамический способ. Распределение памяти – при выполнении программы по запросу. Например, для массива, после того, как вычислен его размер. Однако память для паспорта (описателя) массива может распределяться статическим способом.

Грамматика с описаниями переменных

Начальный нетерминал P определяет программу в целом:

$P \rightarrow$ **int** $R\ P$ | **int1** $R\ P$ | **int2** $R\ P$ | **begin** $A\ Q$ **end**

$R \rightarrow a\ M$

$M \rightarrow$, $a\ M$ | ;

Служебные слова **int**, **int1**, **int2** задают описания:

int - целых переменных, **int1** -одномерных массивов целых, **int2** двумерных массивов целых.

Семантические действия генератора ОПС соответственно

для нетерминала P:

11□□ | **12**□□ | **13**□□ | **14**□□**15**

для нетерминала R:

16□

для нетерминала R:

□**16**□ | □

Числа **11, 12, 13, 14, 15** задают семантические программы.

Семантические программы

Программа 11.

Переключение на заполнение таблицы переменных типа **int** (целочисленных), в таблице будут записываться имена переменных.

Программа 12.

Переключение на заполнение таблицы переменных типа одномерный массив **int1** (целочисленных), в таблице будут записываться имена массивов.

Программа 13.

Переключение на заполнение таблицы переменных типа двумерный массив **int2** (целочисленных), в таблице будут записываться имена массивов.

Семантические программы

Программа 14.

1. Завершение формирования таблиц переменных.
2. Генерация в ОПС операций выделения памяти блоками для каждого из типов переменных:

– для типа **int** – в виде массива целых, каждая переменная в нем занимает отдельный элемент и ей приписан номер;

– для типа **int1** и **int2** – в виде массива паспортов массивов, в них записывается размерность по одному или двум измерениям соответственно, а также ссылка на начало размещения массива в памяти .

3. Генерация в ОПС операций обнуления паспортов массивов для переменных типа **int1** и **int2**.

Семантические программы

Программа 15.

Генерация в ОПС операций освобождения всех выделенных в процессе выполнения программы блоков памяти.

Программа 16.

1. Проверка имени переменной, поступающей из входной цепочки, на совпадение с именами, ранее занесенными в таблицы. Если есть совпадение, то сигнализация об ошибке «повторное описание переменной».
2. Если ошибки нет, то добавление имени переменной в одну из таблиц переменных (на которую ранее было переключение).

Динамическое выделение памяти

Память скалярным переменным и паспортам массивов выделяется статически, а элементам массивов – динамически.

Для одно- и двумерным массивам определены стандартные операторы:

$A \rightarrow$ **mem1** (a, S) | **mem2** (a, S, S)

Семантические действия генератора ОПС соответственно:

□□ a □□ **m1** | □□ a □□□□ **m2**

m1 – запись в ОПС операции выделения памяти одномерному массиву.

m2 – запись в ОПС операции выделения памяти двумерному массиву.

Выполнение операций m1 и m2 при работе интерпретатора

m1 – требует двух операндов:

1) ссылка на паспорт массива,
2) количество элементов в массиве.

m2 – требует трёх операндов:

1) ссылка на паспорт массива,
2) количество строк в массиве,
3) количество и элементов в строках массива.

Выполняются следующие действия:

1) проверяется, не выделялась ли раньше память для переменной, указанной в первом операнде;
2) Если да, то ошибка, а если нет, то выделяется (по запросу к операционной системе) требуемая память, после чего в паспорт массива записывается ссылка (адрес) на нулевой элемент массива, а также размерность по одному или двум измерениям соответственно.

Грамматика с процедурами

К ранее описанной грамматике для начального нетерминала P добавляется еще один вариант порождения - описание процедуры:

$P \rightarrow$ **proc** a (O) I **;** P

$O \rightarrow$ **int** a W | **int1** a W | **int2** a W | λ

$W \rightarrow$, O | λ

$I \rightarrow$ **int** a **;** I | **int1** a **;** I | **int2** a **;** I | **begin** A Q **end**

Вызов процедуры определяется еще одним вариантом порождения нетерминала A:

$A \rightarrow a\,(L)$

$L \rightarrow S\,J \mid \lambda$

$J \rightarrow ,\, S\,J \mid \lambda$

Этот вариант порождения нетерминала A неразличим по первому терминальному символу с порождением оператора присваивания, поэтому требуется преобразование «факторизация».

Правила грамматики после факторизации

Вместо порождающих правил:

$$A \rightarrow aH := SZ \mid a\ (L)$$

получим правила грамматики LL(1):

$$A \rightarrow aB$$

$$B \rightarrow [SK := SZ \mid := SZ \mid (L)$$

Для остальных порождающих правил не требуются дополнительные преобразования.

Эти порождающие правила определяют процедуры с разным количеством параметров, в том числе и без параметров, но в описании и вызове после имени процедуры всегда должны быть круглые скобки.

В описании процедуры могут быть также описаны локальные переменные (в том числе массивы).

Подстановка параметров при вызове может быть как по значению (если подставляется выражение), так и по ссылке (если подставляется имя переменной или имя массива).

Распределение памяти в программе с процедурами

Интерпретатор, выполняя вычисления по ОПС, использует магазин для промежуточных величин, явно в программе не описанных. Память для переменных, описанных в главной программе и в процедурах выделяется в том же магазине.

Для каждого вызова процедуры (в том числе ***рекурсивной***) выделяется новый **блок памяти**, который освобождается при выходе из процедуры.

Порядок выделения памяти в магазине:

- блок глобальных скалярных переменных, паспортов и элементов для глобальных массивов;
- блок параметров и локальных переменных для вызова 1-й процедуры (процедура вызывается из главной программы);
- блок параметров и локальных переменных для вызова 2-й процедуры (если 2-я процедура вызывается из 1-й процедуры);
- . . .
- вершина магазина, где выполняются вычисления.

Управление блоком памяти

Интерпретатор, используя магазин для вычислений как в главной программе, так и после вызова процедуры, управляет текущими значениями следующих величин:

i - номер обрабатываемого элемента ОПС;

r - номер элемента ОПС, куда необходимо вернуться при завершении процедуры;

b - ссылка на начало блока памяти текущего вызова процедуры;

t - ссылка на вершину магазина (конец блока памяти).

Каждый элемент данных в магазине хранит содержимое и признак вида:

- числовое значение;
- ссылка на другой элемент данных в магазине (который может быть значением или ссылкой);
- ссылка на номер элемента ОПС, куда возможен переход.

Структура блока памяти для вызова процедуры

Блок начинает создаваться при выполнении оператора вызова.

Содержимое блока:

- номер элемента ОПС – начала вызываемой процедуры;
- ссылка на начало предыдущего блока памяти;
- номер элемента ОПС, куда необходимо вернуться при завершении процедуры, из которой выполняется текущий вызов;
- 1-й фактический параметр вызова процедуры;

  . . .
- последний фактический параметр вызова процедуры;
- локальные переменные, описанные внутри вызываемой процедуры, формируются после входа в процедуру.

Операции ОПС для вызова процедуры

Пусть оператор вызова процедуры **proc1** с *n* параметрами:

proc1(S1, S2, . . ., S*n*)

после трансляции в виде ОПС:

m1 pr S1 S2 . . . S*n* **call**

m1 – метка, номер элемента ОПС – начало процедуры;
pr – операция начало вызова процедуры;
S1 – ОПС для 1-го параметра;
S2 – ОПС для 2-го параметра;
. . .
S*n* – ОПС для 3-го параметра;
call – операция завершения вызова процедуры.

Эти операции создают в магазине начало блока памяти для вызова процедуры.
Последняя операция **call** совершает переход на метку **m1**.
Параметр – имя переменной подставляется по ссылке,
параметр – выражение подставляется по значению.

Операции ОПС при выполнении процедуры

После трансляции описания процедуры в ОПС первая операция должна быть:

d **bp**

d – размер памяти в магазине для размещения локальных переменных в процедуре;

bp – операция выделения памяти в текущем блоке.

Последняя операция в процедуре:

ret (*без параметров*)

ret – операция возврата из процедуры, аннулирование памяти для текущего блока. Восстанавливает управляющие величины *i*, *r*, *b*, *t* для блока, из которого был вызов процедуры.

Адрес операнда – фактического параметра или локальной переменной в ОПС вычисляется как сумма *b* - ссылки на начало блока и смещения относительно начала блока.

Обработка ошибок при трансляции и выполнении программы

Уровни возможных ошибок:

- синтаксическая ошибка при распознавании лексемы;
- синтаксическая ошибка при работе LL(1)-распознавателя;
- ошибка при работе семантических программ генератора ОПС;
- ошибка при работе интерпретатора во время выполнения ОПС.

Ошибки при работе лексического анализатора:

Лексический анализатор - процедура, вызываемая из ***синтаксического распознавателя – генератора ОПС***. Выделяет из входного текста очередную лексему – номер лексемы и значение.

Если в лексеме ошибка, то возвращается номер ошибки. Кроме того, отслеживается номер ***текущей строки*** текста, и номер ***текущего символа*** в строке.

Ошибки при работе LL(1)-распознавателя:

- если верхний символ в магазине – терминал, и при проверке на его совпадение с входной лексемой обнаружено несоответствие;
- если верхний символ в магазине – нетерминал, и входная лексема такова, что в таблице LL(1)-распознавателя на пересечении строки и столбца – ошибка.

Ошибки при работе семантических программ генератора ОПС:

- если при анализе описаний переменных обнаружено имя, совпадающее с уже имеющемся в таблице именем;
- если при анализе выражений обнаружено имя, не совпадающее ни с одним из имеющихся в таблице именем.

При обнаружении ошибки выдается сообщение о виде ошибки и месте ошибочного символа во входном тексте (номер ***текущей строки*** текста, и номер ***текущего символа*** в строке).

Если ошибок не обнаружено, то ОПС будет полностью сгенерирована, и её можно выполнять интерпретатором.

Ошибки при работе интерпретатора,

если невозможно выполнить очередную операцию:

- недопустимый тип операнда для данной операции;
- недопустимое значение арифметической операции (переполнение, деление на 0);
- индекс выходит за пределы границ массива.

Для диагностики в начале выполнения операции необходимо предусмотреть проверку операндов!

Для локализации ошибки можно при генерации ОПС для каждой операции в ОПС создать ссылку на входной текст (номер ***текущей строки*** текста, и номер ***текущего символа*** в строке).