УДК 004.4'4

ТРАНСЛЯТОР С ЯЗЫКА ОПИСАНИЯ АГРЕГАТИВНЫХ СИСТЕМ НА ЯЗЫК ВЫСОКОГО УРОВНЯ

Д. С. Наумов, Ф. А. Данилкин

Тульский государственный университет

Поступила в редакцию 03.12.2010 г.

Аннотация. Рассмотрена грамматика разработанного языка описания агрегативных систем. Приведено описание проектирования транслятора с разработанного языка на язык высокого уровня.
Ключевые слова: грамматика языка, транслятор, сканер, синтаксический анализатор, семантический анализатор.
Annotation. Aggregated systems definition language grammar is examined. Developed language to higher-level language translator designing is described.
Key words: language grammar, scanner, parser, semantic analyzer.

ВВЕДЕНИЕ

Анализ сложных, трудноформализуемых, многокомпонентных систем, таких как системы имитационного моделирования ликвидации последствий чрезвычайных ситуаций, проведения широкомасштабных тактических операций и др., указывает на необходимость применения универсального подхода к их построению. Одним из таких подходов является описание модели в виде агрегативной системы (A-схемы).

Для удобства построения программных моделей на основе A-схем необходимо иметь базовые инструменты их реализации. Ввиду того, что сложность, назначение и контекст использования моделей могут значительно варьироваться, должен быть разработан механизм, позволяющий создавать базовые унифицированные модели, которые могут дополняться и конкретизироваться в зависимости от поставленных задач.

Такой механизм может быть реализован в виде транслятора, позволяющего генерировать программные заготовки на языке высокого уровня по определенному описанию системы. Для описания агрегативной системы был разработан специальный язык.

МЕТОДИКА ЭКСПЕРИМЕНТА

Для задания грамматики языка необходимо охарактеризовать основные элементы агрегативной системы.

При агрегативном описании производится разбиение системы на подсистемы (агрегаты) с сохранением всех внутренних и внешних коммуникационных связей. В общем случае агрегат A-схемы определяется следующими множествами: $X = \left\{x_i \middle| i = 0...N_X - 1\right\}$ – множество входных сигналов агрегата; $Y = \left\{y_i \middle| i = 0...N_Y - 1\right\}$ – множество выходных сигналов агрегата; $T = \left\{t_i \middle| i = 0...N_T - 1\right\}$ – множество моментов времени; $Z = \left\{z_i \middle| i = 0...N_Z - 1\right\}$ – множество внутренних состояний агрегата; $H = \left\{h_i \middle| i = 0...N_H - 1\right\}$ – множество внутренних параметров агрегата [1, 2, 3].

Коммуникационные функции характеризуют обмен информацией между агрегатами внутри агрегативной системы и между агрегативной системой и внешней средой.

В соответствии с приведенным описанием был предложен следующий синтаксис языка описания агрегативных систем:

COUNT = <число>
/AGREGATE <идентификатор> {
X = [<список идентификаторов>]
Y = [<список идентификаторов>]
H = [<список идентификаторов>]
Z = [<список идентификаторов>]
F = [<список идентификаторов>]}/
/LINK <идентификатор> {
FROM <список контактов> TO <список контактов>}/

Приведенный синтаксис языка описания агрегативных систем подразумевает наличие трех основных частей:

1. Общее описание системы.

2. Описание агрегатов.

3. Описание коммуникационных функций.

Общее описание системы подразумевает указание количества агрегатов.

Описание агрегатов включает следующие элементы:

1. Определение входных сигналов.

2. Определение выходных сигналов.

3. Определение внутренних параметров.

4. Определение состояний.

5. Определение реализуемых функций.

Описание коммуникационных функций подразумевает определение внутренних связей агрегативной системы, т.е. определяет, какие выходы каких агрегатов связываются с какими входами каких агрегатов.

В качестве идентификаторов можно использовать любые последовательности символов, которые удовлетворяют следующим ограничениям:

1. Идентификатор может состоять из букв латинского алфавита, цифр, знака подчеркивания; никакие другие символы в идентификаторе недопустимы.

2. Идентификатор не может начинаться с цифры.

3. Идентификатор не может совпадать ни с одним из зарезервированных слов.

4. Длина идентификатора может быть произвольной.

Список идентификаторов представляет собой последовательность идентификаторов, разделенных между собой запятыми.

Список контактов представляет собой разделенную запятыми последовательность синтаксических единиц вида: agr1(x1), где agr1 – название агрегата, x1 – контакт агрегата (входной или выходной).

В разработанный язык для типового обозначения некоторых элементов введено 10 ключевых слов: COUNT, AGREGATE, LINK, FROM, TO, X, Y, H, Z, F.

Грамматика в соответствии с нормальной формой Бэкуса (НФБ) имеет вид [4]:

<программа> ::= <заголовок> <список агрегатов> <список связей>

<заголовок> ::= COUNT = <число>

<список агрегатов> ::= <агрегат> | <список агрегатов> <агрегат>

<список связей> ::= <связь> | <список связей> <связь>

<агрегат> ::= AGREGATE <идентификатор> {<список полей>}

<связь> ::= LINK <идентификатор> {FROM <список контактов> TO <список контактов>}

<список полей> ::= <поле> | < список полей >; <поле>

<поле> ::= <тип> = {<список идентификаторов>}

<тип> ::= X | Y | H | Z | F

<список контактов> ::= <контакт> | <список контактов>, <контакт>

<контакт> ::= <идентификатор> (<идентификатор >)

<список идентификаторов> ::= <идентификатор> | <список идентификаторов>, <идентификатор>

<идентификатор> ::= <буква> | <идентификатор> <буква> | <идентификатор> <цифра>

<число> ::= <цифра> | <число> <цифра>

<буква> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | A | B | C | D | E | F | G | H | I | J | K | L | M | O | P | Q | R | S | T | U | V | W | X | Y | Z |

<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Для приведения грамматики к единообразному виду (для упрощения работы синтаксического и семантического анализаторов) были введены следующие обозначения:

<программа> – Pr, <заголовок> – He, <список агрегатов> – La, <список связей> – Ll, <агрегат> – Ag, <связь> – Li, <список полей> – Sn, <поле> – No, <тип> – Ty, <список контактов> – Sc, <список идентификаторов> – Si, <контакт> – Co, <идентификатор> – id, <число> – di.

Транслятор условно может быть разделен на четыре части.

Первая часть – это лексический анализатор (или сканер). Задачей сканера является обнаружение и выделение из исходного текста основных символов языка, к которым можно отнести идентификаторы, служебные слова, целые числа, разделители. Существует ряд причин, по которым отделение лексического анализатора от других этапов транслятора является предпочтительным. Одной из причин является то, что при описании лексем можно использовать простой тип грамматики, что позволяет построить эффективный алгоритм сканера.

Грамматика сканера является регулярной. Каждое правило этой грамматики имеет вид

U ::= *T* или *U* ::= *VT*, где *T* - терминал, а *U, V* – нетерминалы:

id::= <буква> | id <буква> | id <цифра>

<число>::= <цифра> | <число> <цифра>

<разделитель>::= = | (|) | + | - | * | / | “ | ‘ | Пробел | Перевод на новую строку | Возврат каретки

<буква>::=a| b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | r | s | t | q | u | w | x | y | z | A | B | C | D | E | F | G | H | I | K | L | M | N | O | P | R | S | T | Q | U | W | X | Y | Z

<цифра>::=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Сканер, диаграмма которого представлена на рисунке 1, реализован в виде конечного автомата, выполняющего отдельный проход, на котором выполняется полный лексический анализ исходной программы и который выдаёт синтаксическому анализатору таблицу, содержащую исходную программу в форме внутренних символов. Состояния автомата определяются следующим образом: s_0 – начальное состояние сканера, s_1 – идентификатор, s_2 – число, s_3 – разделитель, s_4 – комментарий, s_5 – ошибка.

Результатом работы такого сканера является образ всей программы в виде набора лексем. Символы попадают в один из следующих классов:

1. Идентификаторы.
2. Целые числа.
3. Разделители.

Сканер также должен выделить ключевые слова. Они рассматриваются как известное предопределенное подмножество идентификаторов. После того как из исходной программы был выделен идентификатор, осуществляется проверка, которая определяет, является ли он ключевым словом.

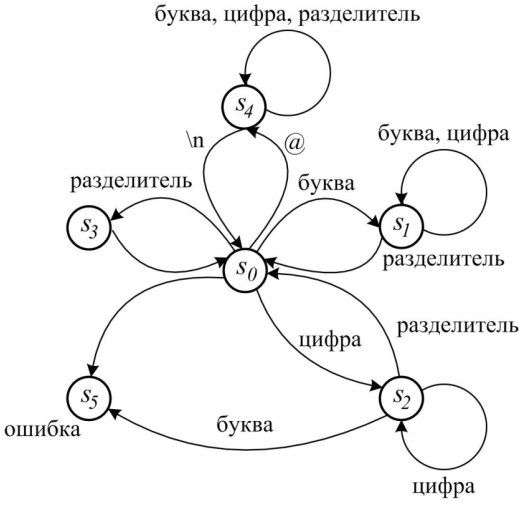

Рис. 1. Диаграмма состояний сканера

Вторая часть транслятора – синтаксический анализатор. Его задачей является полный синтаксический контроль исходной программы, и разбиение её на составные части, соответствующие конструкциям исходного языка.

Синтаксический анализатор реализует метод восходящего разбора – расширенное предшествование. Практическое применение метода простого предшествования затруднено ввиду встречающихся неоднозначных отношений предшествования. Существующие алгоритмы, позволяющие устранять неоднозначность, как правило, не гарантируют единственности правых частей правил модифицированной грамматики. Модифицированная грамматика значительно расширяется и теряет наглядность. По этой причине был использован метод расширенного предшествования (1,2)(2,1).

В соответствии с выбранным типом предшествования редуцирование основы производится при нахождении отношения >. Символы переписываются в стек до нахождения отношения между ними <. Это и будет составлять редуцируемую основу. По ней ищется соответствующее правило в массиве грамматики, выделяется правая часть и помещается вместо редуцируемой основы.

При неоднозначности отношений для выделения левого конца основы было заготовлено множество левых троек LT. Символы левых троек обозначим $S_1S_2S_3$. Эти тройки такие, что S_2S_3 начинают правую часть какого-либо правила, а символ S_1 образует с S_2 конфликтную пару.

Тогда при разборе, если символы в приводимой строке $S_nS_{n+1}S_{n+2}$ равны символам одной из троек $S_1S_2S_3$, то справедливо отношение предшествования <. В противном случае берется отношение =.

$$LT = \{\text{"He La AGREGATE "},$$
$$\text{"La Ll LINK","FROM Sc ,",}$$
$$\text{"TO Sc ,","\{ Sn ;","\{ Si ,"}\}.$$

Для построения матрицы предшествования были составлены множество $L(U)$ – множество самых левых символов относительно нетерминального символа U, $R(U)$ – множество самых правых символов относительно нетерминального символа U. Указанные множества приведены в таблице 1.

Параллельно с синтаксическим анализатором работает семантический анализатор, конт-

Таблица 1

Множества самых левых и самых правых символов относительно нетерминального символа

U	$L(U)$	$R(U)$
Pr	#	#
He	COUNT	di
La	Ag, La, AGREGATE	Ag, }
Ll	Li, Ll, LINK	Li, }
Ag	AGREGATE	}
Li	LINK	}
Sn	No, Sn, Ty, X, Y, H, Z, F	No, }
No	TY, X, Y, H, Z, F	}
Ty	X, Y, H, Z, F	X, Y, H, Z, F
Sc	Co, Sc, id	Co,)
Co	id	)
Si	id, Si	id

ролирующий программу с точки зрения семантики. Когда синтаксический анализатор узнаёт конструкцию исходного языка, он вызывает семантический анализатор, который контролирует данную конструкцию с точки зрения семантики и затем запоминает информацию о ней в таблице символов и в таблице констант.

После получения внутренней формы представления исходной программы происходит обращение к методам, реализующим генерацию интерфейса и реализаций классов и функций агрегативной системы на языке C++.

ЗАКЛЮЧЕНИЕ

В ходе исследований был разработан язык описания агрегативных систем. Для перевода программы с языка описания агрегативных систем на язык высокого уровня C++ был разработан транслятор, состоящий из лексического, синтаксического и семантического анализаторов. Разработанное программное обеспечение позволяет генерировать интерфейс и реализацию базовых классов и функций на языке C++, описывающих агрегативную систему.

Рассмотренный транслятор может найти применение при составлении программных моделей на основе *A*-схем для систем имитационного моделирования проведения широкомасштабных тактических операций, ликвидации последствий чрезвычайных ситуаций и др.

СПИСОК ЛИТЕРАТУРЫ

1. *Советов Б.Я.* Моделирование систем: Учеб. для вузов. 4-е изд. – М.: Высш. шк., 2005. – 343с.: ил.

2. *Сирота А.А.* Компьютерное моделирование и оценка эффективности сложных систем. – М.: Техносфера, 2006. – 280с.: ил.

3. *Бусленко Н.П.* Моделирование сложных систем. – М.: Наука, 1968. – 356 с.: ил.

4. *Грис Д.* Конструирование компиляторов для цифровых вычислительных машин. – М.: Мир, 1975. – 543с.

Наумов Дмитрий Сергеевич – аспирант, кафедра ЭВМ, Тульский государственный университет, Тел.+79202736451. E-mail: nds.tula@gmail.com

Данилкин Федор Алексеевич – доктор технических наук, профессор, кафедра ЭВМ, Тульский государственный университет, Тел. +79207432949.

Naumov Dmitriy Sergeyevich – Postgraduate, Tula state university, Computers department. Tel.+79202736451. E-mail: nds.tula@gmail.com

Danilkin Fedor Alekseyevich – Doctor of technical science, Professor, Tula state university, Computers department. Tel. +79207432949.