УДК 621.391

ТРАНСЛЯЦИЯ СЕТЕВЫХ МОДЕЛЕЙ ПРОТОКОЛОВ СИСТЕМ СВЯЗИ В ПРОГРАММНУЮ РЕАЛИЗАЦИЮ

Донченко А.А., Кисляков М.А.
Северо-Кавказский филиал Московского технического университета связи и информатики,
г. Ростов-на-Дону
E-mail: skf-mtuci@mail.ru

Аннотация. Рассматривается метод решения задачи синтаксически управляемой трансляции сетевых моделей протоколов в программную реализацию, обеспечивающий представление логической структуры сетевой модели протокола в терминах языка программирования.
Ключевые слова: трансляция, протокол, сети связи, программирование.

Постоянное развитие архитектуры вычислительных сетей, и цифровых систем связи (ЦСС) в частности, возросшая сложность сервиса и протоколов обуславливают актуальность теоретических исследований проблем, связанных с протоколами. На данный момент не существует общей стройной теории протоколов. Основной задачей при создании такой теории является проблема выбора формального аппарата, который стал бы ее основой. Этот аппарат должен позволять адекватно описывать элементы, структуру, свойства протокола, а также иметь развитую теорию, позволяющую обеспечить их верификацию, поддерживающую методы синтеза и реализации протоколов [1]. Тем самым обеспечить разработчиков инструментом, позволяющим полно и однозначно описать спецификацию разрабатываемых протоколов. В настоящее время в качестве такого аппарата широко применяются сети Петри и различные их модификации. В данной работе в качестве формального аппарата описания протоколов использованы сетевые модели EN [2], которые являются модификацией известного аппарата Е-сетей. Аппарат сетевых моделей EN ориентирован главным образом на описание систем и параллельных процессов, исследование их характеристик с помощью имитационного моделирования и решения задачи достижимости требуемой разметки. В отличие от классических сетей Петри, сетевые модели EN позволяют эффективно описывать не только динамику параллельных процессов, но и управление потоком данных, временные соотношения и процедуры преобразования данных в этих процессах.

В работе рассмотрено формальное описание протокола установления виртуального соединения перспективной ЦСС и методы трансляции данной спецификации в объектную программу на языке программирования высокого уровня, являющуюся конечной программной реализацией протокола.

Постановка задачи

Формально структура любой конечной сетевой модели EN представляется [3] набором:

$$EN = < P, T, F, M, G > \qquad (1.1)$$

где $P = \{p_1, p_2, ..., p_n, r_1, r_2, ..., r_k\}$ - конечное множество позиций, включающее подмножества простых позиций p и разрешающих позиций r;

$T = \{t_1, t_2, ..., t_l\}$ - конечное множество переходов;

$F \subseteq \{I:P \times T \to (0,1)\} \cup \{0:T \times P \to (0,1)\}$ - множество потоковых отношений, задающее для каждого перехода его входные и выходные позиции;

M:P→(0, 1, 2 ...) - функция разметки, определяющая маркировку позиций в форме неотрицательных целых чисел;

G:P→Y - функция состояния памяти сети, описывающая состояние векторов ячеек памяти сети для каждого варианта разметки, через вектора памяти позиций, занятых метками с определенными атрибутами.

Формально функционирование сетевой модели EN описывается выражениями смены маркировки (1.2) и изменения состояния памяти сети (1.3):

$$\text{для } \forall p \in P, M'(p) = M(p) - I(p,t) + Q(t,p) \quad (1.2)$$

$$\text{для } \forall p \in P, G'(p) = G(p) - I(p,t)A(p) + Q(t,p)\Pi(A) \quad (1.3)$$

где $\Pi = \{\rho_1, \rho_2, ..., \rho_i\}$ – процедуры преобразования атрибутов меток для каждого из *i* переходов сети;

A(*p*) – вектор атрибутов метки, находящейся в позиции *p*.

Сетевая модель EN произвольной сложности состоит из фиксированного набора конструкций, называемых примитивами или элементарными сетями *en(t)*, которые определяются [3] как совокупность, состоящая из перехода *t* и множества всех структурных связок, в которых участвует переход *t*.

en(t), описывающая переход t, определяется выражением:

$$en(t) = \langle X, Y, \Psi, \tau \rangle, \quad (1.4)$$

где: *X,Y* – конечные множества соответственно входных и выходных позиций перехода *t*;

Ψ– процедура функционирования перехода *t*;

τ – функция временной задержки;

Процедура Ψ определяется совокупностью функций *C, r₁, r₂* и *ρ,* или

$$\Psi = \{C, r_1, r_2, \rho\}, \quad (1.5)$$

где: *C* – необходимое условие срабатывания перехода *t*;

r₁, r₂ – функции входного и выходного, соответственно, выбора;

ρ – функция преобразования атрибутов меток элементарной сети.

Условие *C* задается в виде логического выражения, аргументами которого являются *B={b,* $\bar{b}$*}*, обозначающие занятость входных и выходных позиций метками.

Функции r_1 и r_2 записываются в виде условного выражения:

$$r_i(t) = \{\Pi_{i1} \to X_{i1}, \Pi_{i2} \to X_{i2}, \ldots, \Pi_{i1i} \to X_{i1i}\}, \; i=1,2, \quad (1.6)$$

где Π_{ij} – некоторый предикат;

X_{ij} – определенное подмножество входных (*i=1*) или выходных (*i=2*) позиций; *j=1,2,…,lᵢ*.

Значением функции $r_i(t)$ является подмножество $X_{i\,j}$, которому соответствует первый истинный предикат $\Pi_{i\,j}$, при просмотре условного выражения слева направо.

Значение предиката $\Pi_{i\;\;j}$ определяется в зависимости от разметки данной элементарной сети, а также от состояния памяти сети и атрибутов меток.

Функция *ρ* определяет преобразование памяти элементарной сети, которое выполняется в конце фазы активности сработавшего перехода:

$$\rho : [\Pi_1 \to (l_{11}, l_{12}, ..., l_{1k}); ...; \Pi_s (l_{s1}, l_{s2}, ..., l_{sm})], \quad (1.7)$$

где $l_{11},\ldots,l_{sm}$ – операции над описателями меток.

Задача трансляции, в общем виде, состоит в преобразовании текста, описанного на некотором входном языке $L_{вх}$, в объектную программу на выходном языке $L_{вых}$ [3]. Формально представляется:

$$Z = F(W), \quad (1.8)$$

где *W* - цепочка символов из алфавита *A* входного языка $L_{вх}$

Z - цепочка, составленная из символов алфавита *B* выходного языка $L_{вых}$

F – функция (набор правил) преобразования.

Исходные данные

Спецификация протокола, описанного на языке сетевых моделей EN, представляется в виде графического описания логической структуры и таблицы множества исходных данных (табл. 1), включающих определение исходного состояния сети.

Таблица множества исходных данных. Таблица 1.

	r_1	r_2	τ	A	ρ
t_1	M(p1)-1	M(p2)+1	40	5	$f(a_{1)}=a_1+1$
T	M(p2)-1	(a_4=1): M(p3)+1 ($a_4\neq1$): M(p4)+1	3	5	$f(a_{3)}=0$
. . .					
$M_0 = \{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$					
$G_0 = \{A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$					

Метод решения

Для разработки правил трансляции необходимо, в первую очередь, определить синтаксическую и семантическую структуру языка сетевых моделей EN. Для этого воспользуемся определениями из теории формальных языков [3].

Языком *L* называется множество цепочек конечной длинны в алфавите Σ.

Грамматикой G_L называется математическая система, определяющая язык *L*.

Формально грамматика описывается четверкой:

$$G = (N, T, P, S), \qquad (1.9)$$

где N – конечное множество нетерминальных символов (нетерминалов),

T – конечное множество терминальных символов (терминалов),

($N \cap T = \varnothing$. множества T и N – не пересекаются)

P – конечное не пустое множество правил (продукций), каждое из которых имеет вид $\alpha \to \beta$, где α и β - цепочки над объединенным алфавитом $N \cup T$.

S – выделенный элемент нетерминального словаря, $S \in N$.

Определение терминального словаря языка EN

Выделим элементарные символы (терминалы) составляющие язык EN $\in T$:

- p_i – элементы множества P, указывающие на определенную позицию;
- t_i – элементы множества T, указывающие на определенный переход;
- a_i - элементы множества A, указывающие на определенный атрибут метки;
- $M(p_i)$ – элементы множества M, определяющие наличие или отсутствие метки в позиции p_i;
- $A(p_i)$ – элементы множества A, определяющие вектор ячеек памяти позиции p_i;
- $f(a_i)$ – правила преобразования атрибутов меток;
- +,- – операции добавления и изъятия меток из позиций, а также преобразования атрибутов ячеек памяти;
- $>,<,=,\neq,\geq,\leq$ – операции сравнения.

Для выделения терминалов (лексем) при вводе текста, описывающего модель на исходном языке, необходимо определить правила формирования лексем из множества вводимых элементов. Эти правила будут применены на этапе лексического анализа входного языка.

Разработка лексических грамматик.

Определим набор элементов, формирующих терминальный словарь и правила лексического вывода терминалов языка EN:

1. <*N*> - число*;*

S→DS

D→'0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'

D→ε

где *D*–нетерминал, обозначающий цифры, из которых составлено число;

' ' – выделяют элементарный символ;

| - знак «либо» (альтернатива);

ε - «пустой» символ.

<*N*> имеет атрибут *N.atr,* определяющий значение числа.

2. <*U*> - операции сравнения;

S→ '>'|'<'|'='|'><'|'>='|'<='

3. <*Z*> - операции добавления/изъятия;

S→ '+'|'-'

Терминалы <*U*> и <*Z*> атрибутов не имеют

4. <*P*> - позиция;

S→ ' p' <*N*>

<*P*> имеет атрибут *P.atr,* определяющий номер позиции.

5. <*T*> - переход;

S→ ' t' <*N*>

<*T*> имеет атрибут *T.atr,* определяющий номер перехода.

6. <*AI*> - переход;

S→ 'a' <*N*>

<*AI*> имеет атрибут *AI.atr,* определяющий номер ячейки памяти (атрибута метки).

7. <*M*> - элемент разметки;

S→ 'M(' <*N*> *')' S*

<*M*> имеет атрибут *M.atr,* определяющий номер позиции на множестве разметки сети.

8. <*A*> - вектор ячеек памяти;

S→ 'A(' <*N*> *')' S*

<*A*> имеет атрибут *A.atr,* определяющий значение вектора ячеек памяти на множестве векторов ячеек памяти сети.

9. <*F*> - правило преобразования;

S→ ' f ' <*N*>

<*F*> имеет атрибут *F.atr,* определяющий номер правила преобразования.

Разработанные грамматики синтаксической структуры являются регулярными выражениями, а из работы [4] известно, что по регулярным выражениям можно построить конечный автомат, который разбирает язык, задаваемый этими выражениями. Диаграмма переходов такого автомата представлена на рис.1.

Разработка транслирующей грамматики

Для реализации выбран один из самых распространенных языков программирования – C++, являющийся на данный момент де факто стандартом системного программирования.

Структуру любой программы на языке программирования высокого уровня можно представить в общем виде как совокупность блоков:

<заголовок программы>

<инициализация начальных данных>

<структуры>
<функции>
<присвоение начальных значений>
<блок операторов (тело программы)>
<завершающие операторы>

Правила, определяемые транслирующей грамматикой, должны предписывать операции заполнения соответствующих блоков текста программы.

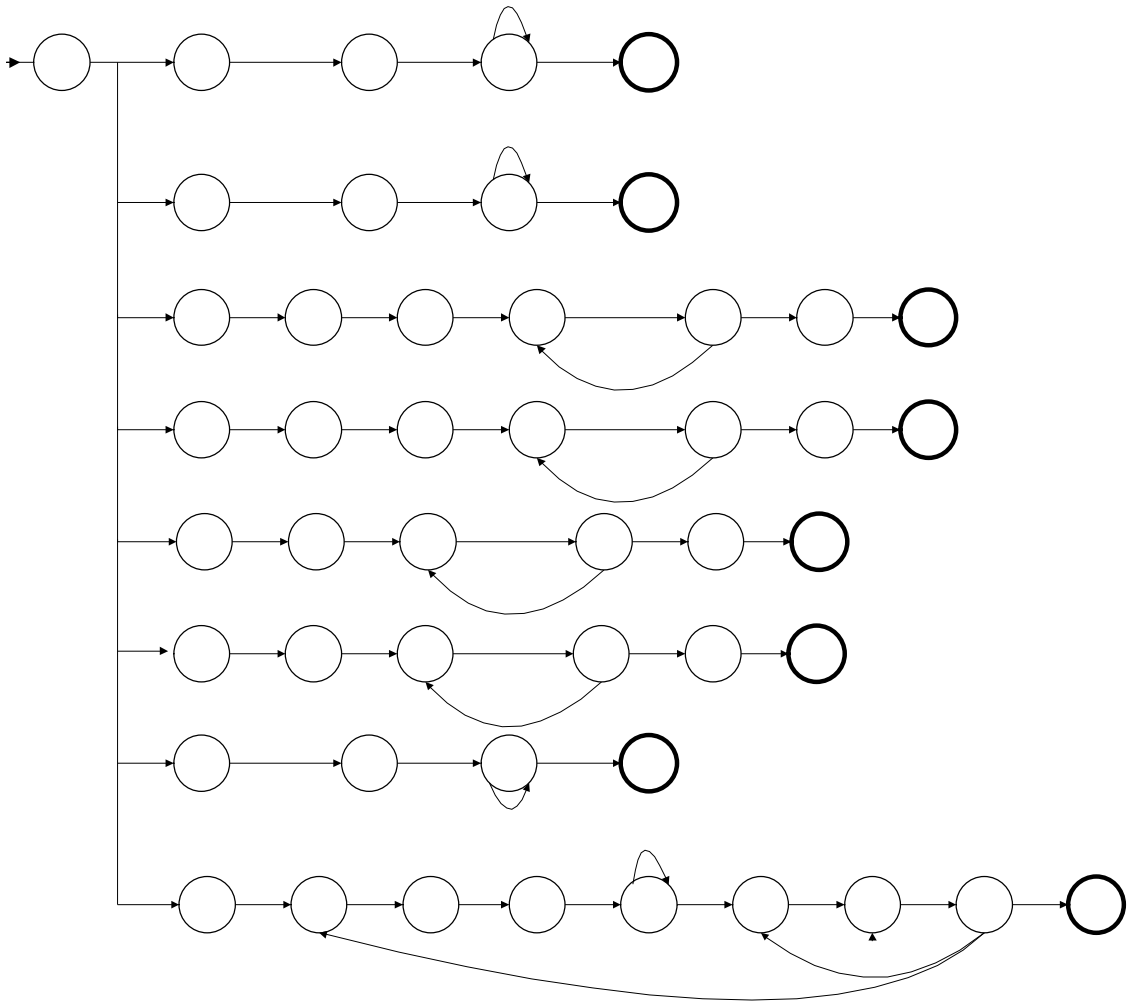

Рис. 1. Диаграмма переходов лексического анализатора языка сетевых моделей EN

Начальный символ грамматики определяет запись заголовка программы и завершающего оператора:

$S \rightarrow$ *void main () { <лексема> }*

<лексема> $\rightarrow$ <*P*>|<*T*>|<*A*>|<M_0>|<G_0>|<t_i>

<*P*>$\rightarrow$ *int Np = P.atr*

Инициализация переменной *Np* (количество позиций) и присвоение ей значения атрибута *P.atr*.

<*P*>$\rightarrow$ *int masM [P.atr]*

Инициализация массива *masM* для хранения данных разметки размерностью *P.atr*.

<*T*>$\rightarrow$ *int Nt = T.atr*

Инициализация переменной *Nt* (количество переходов) и присвоение ей значения атрибута *T.atr*.

<*A*>$\rightarrow$ *int Na = A.atr*

Инициализация переменной *Na* (емкость вектора ячеек памяти) и присвоение ей значения атрибута *A.atr.*

<A>→ int masA [A.atr] [P.atr]

Инициализация двумерного массива *masA* для хранения значений векторов ячеек памяти сети размерностью *P.atr.*

$<M_0> \rightarrow \{ masM [N.atr]=1 \mid <M_0> \}_{N \in P}$

Заполнение начальными значениями массива текущей разметки, значение *masM [i]=1* характеризует наличие метки в позиции p_i.

$<G_0> \rightarrow \{ masA [A.atr] [P.atr]= N.atr \mid <G_0> \}_{N \in P,\ ai \in P}$

Заполнение начальными значениями массива состояния векторов ячеек памяти, в соответствии со значениями *N.atr.*

<t_i>→ func_t T.atr () {

<задержка срабат. перехода>

<проверка условия>

<перемещение меток>

<преобразование векст. памяти> }

Инициализация функции *func_t<i>,* определяющей полный набор правил функционирования перехода t_i.

<t_i>→ struct_t T.atr () {

int <X> // входные позиции перехода

int <Y> // выходные позиции перехода

int <TR> // задержка срабат. перехода

struct_R<r_i> () {

int <RI> // условие срабат. перехода

int <PM> // изъять метку из

int <AM> } // добавить метку в

struct_RO<ρ_i> () {

int <ROI> // правило преобразования атрибутов метки

int <C1> // левая часть правила

int <C2> } } // правая часть правила

Инициализация структуры *struct_t<i>,* значения полей которой содержит данные полного описания перехода t_i.

<t_i>→ if <условие срабат. перехода> then func_t T.atr

Заполнение <тела программы>. Строка программы описывает вызов на выполнение функции *func_t<i>* при выполнении условия срабатывания перехода t_i.

Реализация модели формального описания в конкретной программной или программно-аппаратной среде, является неотъемлемой частью процесса создания коммуникационных протоколов сетей связи. Предложенный в статье метод и разработанные на его основе правила лексического анализа входного текста языка сетевых моделей EN и правила преобразования полученных лексем в текст программы на языке С++ составляют основу для создания транслятора спецификаций модельного представления протоколов в программную реализацию.

Литература

1. Аничкин, С.А. Протоколы информационно-вычислительных сетей. Справочник. / С.А. Аничкиин., С.А.Белов, А.В. Беренштейн и др -М.: Радио и связь, 1990.-504 с.
2. Донченко, А.А. Сетевой аппарат формального описания динамических процессов / А.А. Донченко, А.В. Езимов, А.В. Пряхин // Теория и техника радиосвязи: Науч.-техн. сб. ВНИИС ,- 2002. – Вып. 2. – С.81– 86.
3. Ахо, А. Теория синтаксического анализа, перевода и компиляции / А. Ахо, Д. Ульман -М.: Мир, 1978. – Т.1. – 612с.
4. Хопкрофт, Д. Введение в теорию автоматов, языков и вычислений / Д. Хопкрофт, Р. Мотвани, Д. Ульман -СПб.: Вильямс, 2002.