# РЕАЛИЗАЦИЯ ЛИНГВИСТИЧЕСКОГО ОБЕСПЕЧЕНИЯ АВТОМАТИЗАЦИИ РЕШЕНИЯ ЗАДАЧ УПРАВЛЕНИЯ: ЯЗЫК ГАММА

М.Ф. Степанов [1], А.М. Степанов [2], М.А. Пахомов [1], О.Н. Пименова [1], Л.С. Михайлова [3]

[1] Саратовский государственный технический университет имени Ю.А. Гагарина
[2] Институт проблем точной механики и управления РАН

Рассматриваются вопросы создания лингвистического обеспечения, предоставляющего пользователям системы ГАММА-3 возможности автоматического решения задач, порядок решения которых задается процедурно на проблемно-ориентированном языке.

**Ключевые слова:** теория управления, автоматизация решений, конечные автоматы

Разрабатываемое лингвистическое обеспечение автоматизации решения задач системы ГАММА-3 [1], [2] включает лексический анализатор, синтаксический анализатор, а также интерпретатор внутреннего представления входной программы. Лексический анализатор строится как конечный автомат. В синтаксическом анализаторе используется метод «сдвиг-свёртка». Синтаксическое дерево далее используется интерпретатором в качестве внутреннего представления исходной программы.

В связи с тем, что язык ГАММА является интерпретируемым языком, результаты синтаксического анализа могут непосредственно использоваться для интерпретации, т.е. выполнения предусмотренных пользовательской программой действий. Однако при этом эффективность собственно процесса интерпретации может оказаться весьма низкой. Критериями эффективности традиционно выступают время исполнения и объём используемой памяти. Поэтому целесообразно провести некоторую предварительную работу, которая в компиляторах выполняется на этапе генерации кода.

Чтобы компилятор мог построить код результирующей программы для синтаксической конструкции входного языка, часто используется метод, называемый синтаксически управляемым переводом — СУ-переводом [3].

Для полного представления о типе и структуре найденной и разобранной синтаксической конструкции входного языка, в принципе, достаточно знать последовательность номеров правил грамматики, примененных для ее построения. Однако форма представления этой достаточной информации может быть различной в зависимости, как от реализации самого компилятора, так и от фазы компиляции. Эта форма называется внутренним представлением программы (иногда используются также термины «промежуточное представление» и «промежуточная программа»).

Все внутренние представления программы обычно содержат в себе две принципиально различные вещи — операторы и операнды. Различия между формами внутреннего представления заключаются лишь в том, как операторы и операнды соединяются между собой. Также операторы и операнды должны отличаться друг от друга, вне зависимости от того, в каком порядке они встречаются во внутреннем представлении программы. За

различение операндов и операторов, как уже было сказано выше, отвечает разработчик компилятора, который руководствуется семантикой входного языка.
Известны следующие формы внутреннего представления программ:

- связные списочные структуры, представляющие синтаксические деревья;
- многоадресный код с явно именуемым результатом (тетрады);
- многоадресный код с неявно именуемым результатом (триады);
- обратная (постфиксная) польская запись операций;
- ассемблерный код или машинные команды.

Существуют три формы записи выражений — префиксная, инфиксная и постфиксная. При префиксной записи операция записывается перед своими операндами, при инфиксной — между операндами, а при постфиксной — после операндов. Общепринятая запись арифметических выражений является примером инфиксной записи. Запись математических функций и функций в языках программирования является префиксной (другие примеры префиксной записи — команды ассемблера, триады и тетрады).

Триады представляют собой запись операций в форме из трех составляющих: операции и двух операндов.

Например, триады могут иметь вид:

**<операция> (<операнд1>,<операнд2>).**

Особенностью триад является то, что один или оба операнда могут быть ссылками на другую триаду в том случае, если в качестве операнда данной триады выступает результат выполнения другой триады. Поэтому триады при записи последовательно нумеруют для удобства указания ссылок одних триад на другие (в реализации компилятора в качестве ссылок можно использовать не номера триад, а непосредственно ссылки в виде указателей — тогда при изменении нумерации и порядка следования триад менять ссылки не требуется).

Триады представляют собой линейную последовательность команд. При вычислении выражения, записанного в форме триад, они вычисляются одна за другой последовательно.

Каждая триада в последовательности вычисляется так:

- Операция, заданная триадой, выполняется над операндами, а если в качестве одного из операндов (или обоих операндов) выступает ссылка на другую триаду, то берется результат вычисления той триады.
- Результат вычисления триады нужно сохранять во временной памяти, так как он может быть затребован последующими триадами.
- Если какой-то из операндов в триаде отсутствует (например, если триада представляет собой унарную операцию), то он может быть опущен или заменен пустым операндом (в зависимости от принятой формы записи и ее реализации).
- Порядок вычисления триад, как и тетрад, может быть изменен, но только если допустить наличие триад, целенаправленно изменяющих этот порядок (например,

триады, вызывающие переход на несколько шагов вперед или назад при каком-то условии).

Триады представляют собой линейную последовательность, а потому для них несложно написать тривиальный алгоритм, который будет преобразовывать последовательность триад в последовательность команд результирующей программы (либо последовательность команд ассемблера). В этом их преимущество перед синтаксическими деревьями. Однако здесь требуется также и алгоритм, отвечающий за распределение памяти, необходимой для хранения промежуточных результатов вычисления, так как временные переменные для этой цели не используются. В этом отличие триад от тетрад.

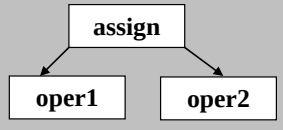
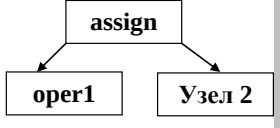
Так же как и тетрады, триады не зависят от архитектуры вычислительной системы, на которую ориентирована результирующая программа. Поэтому они представляют собой машинно-независимую форму внутреннего представления программы.

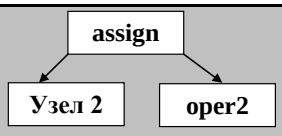
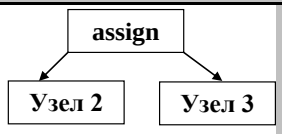
Триады требуют меньше памяти для своего представления, чем тетрады, они также явно отражают взаимосвязь операций между собой, что делает их применение удобным. Необходимость иметь алгоритм, отвечающий за распределение памяти для хранения промежуточных результатов, не является недостатком, так как удобно распределять результаты не только по доступным ячейкам временной памяти, но и по имеющимся регистрам процессора. Это дает определенные преимущества. Триады ближе к двухадресным машинным командам, чем тетрады, а именно эти команды более всего распространены в наборах команд большинства современных компьютеров.

Поэтому в качестве внутреннего представления программы на языке ГАММА используются триады.

В синтаксическом графе у каждой вершины, соответствующей некоторому нетерминальному символу имеются дочерние вершины, количество которых определяется применённым на этапе синтаксического анализа правилом подстановки грамматики. Поскольку их количество, по определению, конечно, то конечным будет и количество шаблонов последовательности триад, некоторые из которых приведены в таблице 1.

Табл.1. Преобразование типовых узлов дерева вывода в последовательность триад

| **Вид узла дерева** | **Результирующий код** | **Примечание** |
|---|---|---|
|  | i) assign (oper1,oper2) | **assign – тип триады**<br>**oper1, oper2 — операнды (листья дерева)** |
|  | i) Code (Узел 2, i)<br>i+j) assign (oper1,^i+j-1) | **Узел 2 — нижележащий узел дерева вывода**<br>**Code (Узел 2, i) -последовательность триад, порождаемая для Узла 2, начиная с триады с номером i,**<br>**j — количество триад, порождаемых для Узла 2** |

| | | |
|---|---|---|
|  | i) Code (Узел 2, i)<br>i+j) assign (^i+j-1, oper2) | **Узел 2 — нижележащий узел дерева вывода**<br>**Code (Узел 2, i) -последовательность триад, порождаемая для Узла 2, начиная с триады с номером i,**<br>**j — количество триад, порождаемых для Узла 2** |
|  | **i) Code (Узел 2, i)**<br>**i+j) Code (Узел 3,i+j)**<br>**i+j+k) assign (^i+j-1, ^i+j+k-1)** | **Узел 2, Узел 3 — нижележащие узлы дерева вывода**<br>**Code (Узел 2, i) -последовательность триад, порождаемая для Узла 2, начиная с триады с номером i,**<br>**j — количество триад, порождаемых для Узла 2**<br>**Code (Узел 3, i+j) -последовательность триад, порождаемая для Узла 3, начиная с триады с номером i+j,**<br>**k — количество триад, порождаемых для Узла 3** |

Укрупнённый алгоритм приведён на рисунке 1.


**Рис.1.** Укрупнённый алгоритм генерации триад оператора присваивания

При этом дочерние вершины дерева разбора, помеченные как «Узел 2» и «Узел 3» представляют собой корневые вершины поддеревьев разбора. К ним относятся, в частности: <формула>, <терм>, <элемент массива> и т.д. Поскольку они в дереве разбора могут встречаться многократно, то целесообразно, как это предусмотрено схемой СУ-перевода, осуществлять рекурсивный вызов генератора триад – функции makeTriadListOfSyntSymbol. При этом вначале осуществляется анализ правой части оператора присваивания, например, <формула>, для определения собственно присваиваемого значения, а затем – левой части, т.е. места памяти, в котором должно быть сохранено значение правой части оператора.

Проиллюстрируем работу приведенного алгоритма на примере разбора фрагмента программы на входном языке ГАММА (см. рисунки 2-4).

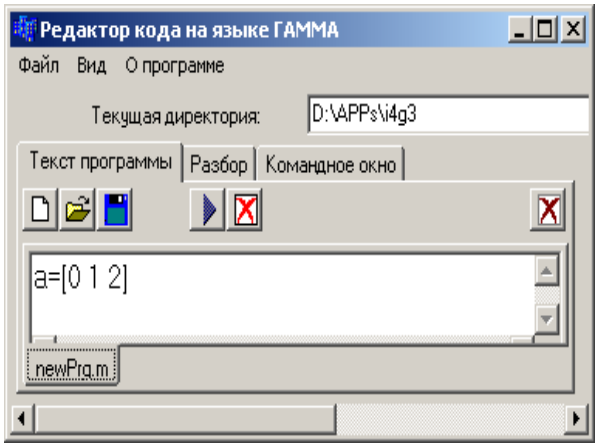



**Рис. 2.**

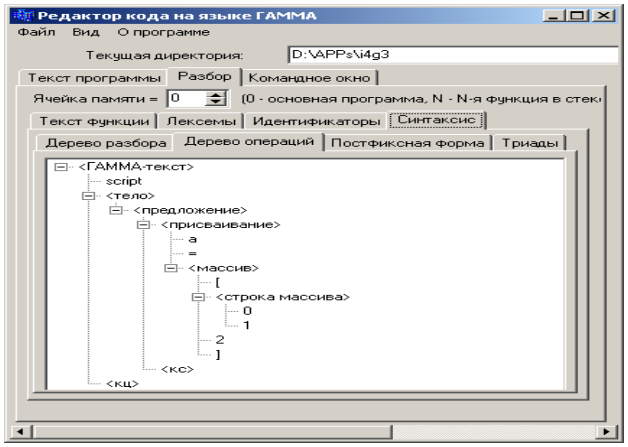



**Рис. 3.**

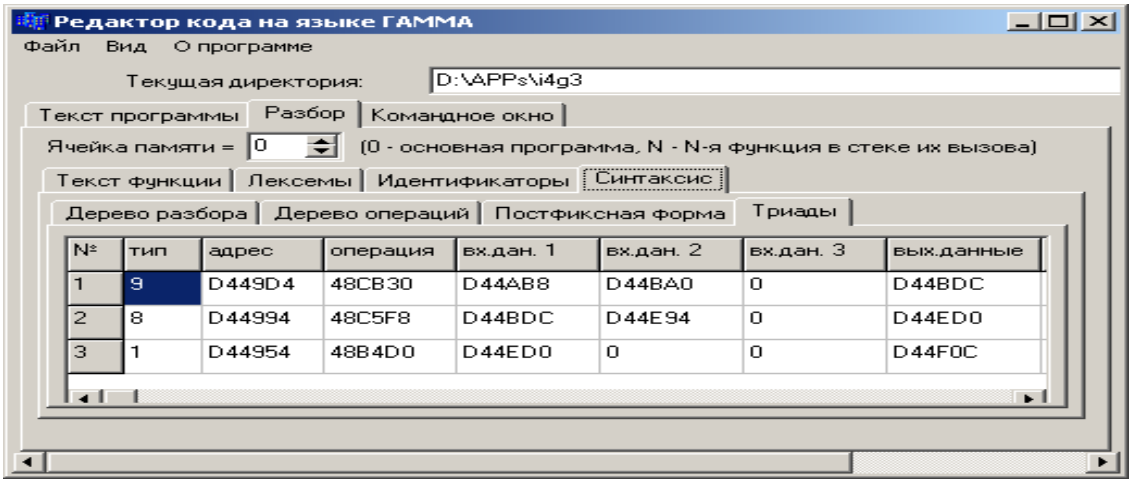



**Рис. 4.**

Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект 15-07-99684-а).

## Литература

1. Александров А.Г. Система ГАММА-3 и ее применение / А.Г.Александров, Л.С.Михайлова, М.Ф.Степанов // Автоматика и телемеханика, 2011, № 10. С. 19 – 27
2. Степанов М.Ф. О развитии концепции атоматического решения задач теории управления в системе ГАММА-3 / А.Г.Александров, Л.С.Михайлова, М.Ф.Степанов, Т.М.Брагин, А.М.Степанов // Мехатроника, автоматизация, управление. №9, 2011. С. 14 – 19
3. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. В 2-х томах / Пер. с англ. Под ред. В.М.Курочкина. – М.: Мир, 1978.