6. Nofal M., Fouad K.M. Developing Web-Based Semantic Expert Systems // International Journal of Computer Science (2014), vol. 11 (1), pp. 103-110.
7. Ruiz-Mezcua, B., Garcia-Crespo, A., Lopez-Cuadrado, J. & Gonzalez-Carrasco, I. (2011). An expert system development tool for non AI experts // Expert Systems with Applications (2011), vol.38, pp. 597–609.
8. Голенков В.В., Гулякина Н.А. Принципы построения массовой семантической технологии компонентного проектирования интеллектуальных систем // Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2011): материалы Междунар. научн.-техн.конф. Минск, 10-12 февраля 2011 г.) – Минск: БГУИР, 2011. – С. 21-58.
9. Чернецки К., Айзенекер У. Порождающее программирование: методы, инструменты, применение / Пер. с англ. – СПб: Питер, 2005. – 736 с.
10. Грибачев К.Г. Delphi и Model Driven Architecture. Разработка приложений баз данных. – СПб: Питер, 2004. – 352 с.
11. Frankel D. Model Driven Architecture: Applying MDA to Enterprise Computing. – New York: Wiley, 2003, 352 p.
12. OMG Model Driven Architecture. URL: http://www.omg.org/mda/ (дата обращения: 10.07.2015)
13. De Miguel M., Jourdan J., Salicki S. Practical experiences in the application of MDA // LNCS. – 2002. – Vol. 2460. – Pp. 128-139.
14. Personal Knowledge Base Designer. URL: http://www.knowledge-core.ru/index.php?p=pkbd (дата обращения: 12.11.2015).
15. Дородных Н.О., Юрин А.Ю. Использование диаграмм классов UML для формирования продукционных баз знаний // Программная инженерия. – 2015. – №4. – С. 3-9.

УДК 004.042 004.415.22

РЕАЛИЗАЦИЯ АВТОМАТНОГО ПОДХОДА К ПРОГРАММИРОВАНИЮ НА ОСНОВЕ ОА-ПАРАДИГМЫ[11]

Салибекян Сергей Михайлович, к.т.н., доцент Национального исследовательского университета «Высшая школа экономики», Московский институт электроники и математики, Москва, salibek@yandex.ru

Введение

Автоматный подход к организации вычислительного процесса нашел широкое применение как в области трансляции искусственных языков (абстрактный автомат) [1], так и в области логического управления техническими объектами (структурный автомат). Автоматная парадигма может быть реализована как схемотехнически, так и программно на базе других парадигм вычислительного процесса. Достаточно много работ посвящено реализации автоматного подхода на базе императивной парадигмы вычислительного процесса [2]. В частности, существует автоматное программирование (или программирование с явным выделением состояний) [3,4]. Однако авторы незаслуженно обходят вниманием реализацию автоматного подхода на базе вычислительной системы (ВС), работающей по принципу dataflow (управлением вычислениями с помощью потока данных). Задача нашего исследования – заполнить этот пробел.

В парадигме dataflow [5] вычислительный процесс представляет собой передачу операндов, оформленных в виде токенов (совокупность операнда и его служебной информации) между исполнительными (вычислительными) устройствами. Автоматный подход предполагает наличие нескольких состояний и переход ВС в процессе вычислений из одного состояния в другое. Первый подход асинхронный (хотя встречается и синхронный

[11] Статья рекомендована к опубликованию в журнале "Прикладная информатика"

вариант dataflow) второй – синхронный, т.к. ВС работает по тактам (в конце каждого такта происходит переход из одного состояния в другое). Эти парадигмы строят вычислительный процесс по совершенно разным принципам, однако их удалось объединить на базе объектно-атрибутного (ОА) подхода к организации вычислительного процесса и структур данных, относящегося к парадигме dataflow [6,7].

1. Функциональное устройство как автомат

Следует отметить, что автоматная парадигма лежит в основе ОА-подхода. Так, ОА-ВС представляет собой совокупность функциональных устройств (ФУ), обменивающихся между собой данными, оформленными в виде милликоманд (МК). МК представляет собой простейший токен, состоящий из двух полей: данные и атрибут. По атрибуту ФУ идентифицирует данные и при необходимости сохраняет их в своем контексте (контекст – совокупность внутренних регистров ФУ, определяющих его внутреннее состояние). По приходе МК к ФУ оно обрабатывает ее: записывает в контекст, или производит вычисления, или выдачу их результата другим ФУ. Другими словами, при приходе МК ФУ меняет свое состояние – таким образом, ФУ представляет собой автомат, правда, не автомат в классическом понимании (автомат с конечным или счетным множеством состояний), а автомат с пространством состояний. В качестве аналогов можно привести гибридные автоматы [8], в частности, модели Тавернини, Бэка-Гукенхеймера-Майерса и Нерода-Кона. Описание математической модели ФУ как автомата приведено в [9]). ФУ могут быть реализованы как аппаратно, так и программно. В целях упрощения структуры ОА-ВС и удобства его проектирования и программирования ФУ следует делать как можно более простыми. Однако автоматные модели могут быть и достаточно сложными. Для преодоления такой проблемы необходимо, чтобы автомат был реализован на базе нескольких ФУ. Такой автомат назовем ОА-автоматом. И наше исследование посвящено разработке методики организации такого вычислительного процесса: необходимые типы ФУ, информационные связи между ФУ, синхронизация вычислений, способ ввода/вывода данных и т.д. Как известно, автоматы бывают различных типов: абстрактный и структурный, детерминированный (ДА) и недетермированный (НДА). Также находит применение автоматная декомпозиция: вызываемый и вложенный автомат. О реализации всех перечисленных типов ФУ согласно ОА-подходу и пойдет речь ниже.

2. Детерминированный автомат

Основной принцип, по которому работает ОА-автомат, следующий: каждому состоянию ОА-автомата ставится в соответствие так называемое ФУ-состояние, которое принимает входные данные (сигналы) от ФУ-источника данных, обрабатывает их и передает управление другому (другим) ФУ. Например, реализация абстрактного ДА на основе ОА-архитектуры описана в [6]. В качестве ФУ-источника в данном случае используется ФУ, выполняющее лексический анализ текста. Т.к. лексический анализ достаточно прост, его реализацию удобно было инкапсулировать в ФУ-источник. Все остальные стадии анализа языка (такой подход в большинстве случаев применяется для распознания языков программирования): синтаксический, семантический, синтеза строки на другом языке – выполняются на базе ФУ типа FUList (контроллер ОА-списка), выполняющие роль ФУ-состояний. Такое ФУ контролирует ОА-список, каждая линия которого содержит описание лексем, по которым следует производить переход в другое состояние, и подпрограмму, которая запускается в том случае, когда лексема, выданная источником данных, подпадает под описание, находящееся в линии ОА-списка (линией называется информационная капсула (ИК), входящая в ОА-список). Подпрограмма описывает действия, совершаемые при переходе в другое состояние, в том числе и команды перенастройки ФУ-источника с помощью специальной МК. В контекст ФУ-источника (ФУ лексического разбора) входит

регистр, где помещается адрес ФУ-состояния, куда будет оправлена МК с очередной лексемой на следующем также работы ОА-автомата. Магазинная память организуется с помощью ОА-списка, который используется в роли стека, хранящего ИК, который могут быть использованы во время анализа текста.

ОА-ДА был использован для реализации компилятора ОА-языка, который входит в состав ОА-среды программирования и моделирования. И следующим этапом стала разработка ОА-НДА.

3. Недетерминированный автомат

В недетерминированном автомате одновременно могут быть активными сразу несколько состояний. И если формально ДА без стека определяется как пятерка (Σ, Q, S, F, P), где Σ – множество входных символов, Q – множество состояний автомата, S –начальное состояние автомата ($S \in Q$), F – множество конечных состояний, P – множество правил перехода в другое состояние в виде: $(q_i,a_i) \rightarrow (q_j)$, где q_i — текущее состояние автомата, a_i — текущий символ на входной ленте, q_j – новое состояние автомата. НДА от ДА отличается следующим: во-первых, у него может быть несколько начальных состояний, т.е. S – это множество, элементы которого принадлежат множеству состояний автомата ($S \subseteq Q$). Во время работы НДА может пребывать сразу в нескольких состояниях и за один такт его работы может активизироваться несколько правил перехода.

На базе императивной парадигмы вычислительного процесса реализация НДА достаточно сложна, поэтому НДА стараются привести к ДА. Причиной тому, что ДА и императивная ВС так хорошо сочетаются, является то, что в императивной парадигме и в ДА ВС в один момент времени выполняется только в одном состоянии (для императивной парадигмы состояние ВС можно связать с ячейкой памяти, из которой выбирается исполняемая команда. Для реализации же НДА, где активны несколько состояний, более подходит dataflow-парадигма, т.к. она обладает врожденным параллелизмом: в ней активными могут быть сразу несколько активных устройств. Следует заметить, что в последнее время к параллельным вычислениям стали приспосабливать и императивную парадигму, однако приспособление это искусственно: например, выделяются вычислительные нити, каждая из которых, по сути, представляет собой императивный вычислительный процесс (синхронизация вычислений между вычислительными нитями весьма сложно и неестественно). Для реализации НДА будем использовать ОА-подход.

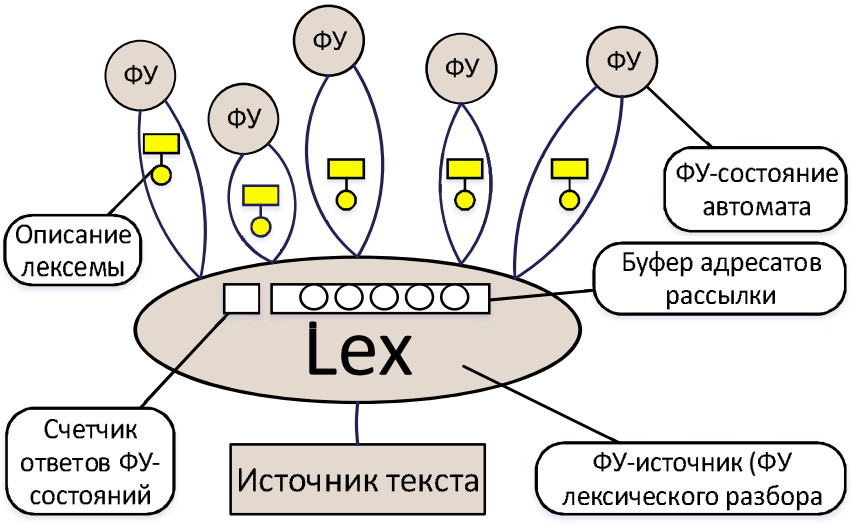

Рис. 1 – ОА-недетерминированный автомат

Итак, ВС из [10] для реализации абстрактного НДА требует некоторых доработок. Во-первых, ФУ-источник данных (он реализует лексический разбор исходного текста и

передачу лексем для ФУ-состояний) требует наличия буфера адресов активных ФУ-состояний, для которых производится рассылка лексем (буфер рассылки). Перед началом анализа текста в него записывается адреса ФУ-состояний, задающих начальное состояние автомата. Во-вторых, требуется синхронизация вычислений, т.е. ФУ-источник должен принять решение о том, когда начинать следующий такт работы. Такт продолжается до тех пор, пока все ФУ-состояния не закончат обработку присланных им данных. Для определения момента конца такта используется следующий механизм. В контекст ФУ вводится счетчик рассылок, в который перед началом такта работы ОА-автомата записывается количество адресатов рассылки данных. ФУ-состояние по окончанию обработки данных с помощью МК ReceiverMkSet производит пересылку на ФУ-источник адреса ФУ-состояния, которое должно быть активным на следующем такте работы ОА-автомата (т.е. которому ФУ-источник должно будет переслать лексему). По приходе этой МК ФУ-источник уменьшает счетчик на 1. Достижение счетчиком значения 0 является признаком необходимости начала нового такта работы ОА-автомата. Однако из одного состояния НДА может происходить переходы сразу в несколько состояний на следующем такте. Поэтому понадобилось ввести еще одну МК с мнемоникой ReceiverMkAdd, которая устанавливает адрес активного ФУ-состояния, однако не вызывает уменьшения счетчика рассылок на 1. Таким образом, ФУ-состояние производит пересылку адресов активных состояний с помощью МК ReceiverMkAdd, а последний адрес пересылается с помощью МК ReceiverMkAdd. Однако в НДА встречается ситуация, когда происходит переход из нескольких состояний в одно, в результате чего в буфер рассылок могут попасть несколько одинаковых адресов. Поэтому ФУ-источник организует проверку дублирования адресов ФУ в буфере рассылки.

Таким образом, работа ФУ-источника в течение одного такта ОА-НДА состоит из двух фаз: раздача данных ФУ-состояниям, прием МК ReceiverMkAdd и ReceiverMkSet от ФУ-состояний и запись в буфер адресов ФУ-состояний, который будут активными на следующем такте. На следующий такт происходит рассылка очередной порции данных (лексемы для абстрактного автомата) для ФУ-состояний, адреса которых были накоплены в буфере рассылки на предыдущем такте, и накопление адресов для рассылки наследующем такте и т.д.

4. ОА-структурный автомат

Структурный автомат, в отличие от автомата абстрактного, работает не с символами, а с входными сигналами (т.е. поток входных данных структурирован). Такой автомат используется в реактивных системах, т.е. системах, работающих в реальном времени (например, системах управления техническими объектами). Сигналы могут быть любой природы: двоичное, целое или дробное число, символ, аналоговый сигнал. Сигналы обычно обозначаются с помощью так называемых переменных, имеющих определенную мнемонику (обозначение). Например, X, Y, Z и т.п. Для формализации описания работы структурного автомата введем понятие **переходного предиката** p_{ij} – это предикат, приписанный к дуге графа состояний структурного автомата, идущей от вершины с номер i к вершине с номером j. Аргументами предиката является сигналы. Пусть $P=\{p_{ij}\}$, где $i,j=1\ldots N$ (N-число переходных предикатов в структурном автомате) – множество переходных предикатов. Автомат переходит из состояния i в состояние j в том случае, когда переходной предикат p_{ij} возвращает true («правда»). Возьмем формальное определение структурного автомата из [4]. Это – шестерка $\{X, Q, Y, y_0, \varphi, \delta\}$, где X – множество входных сигналов, которые отражают состояние объекта управления (X_E) внешней среды (X_O), Y – множество выходных сигналов, воздействующих на объект управления, Z – множество внутренних состояний автомата, y_0 – начальное состояние автомата (для ДА, $y_0 \in Q$, φ – функция выходных воздействий (φ состоит из двух компонент: φ' – функция выходных воздействий в состояниях (φ': $Q \rightarrow Y$) для автомата Мура, φ'') – функция выходных воздействий на переходах (φ'': $X \times Q \rightarrow Y$), δ –

функция переходов (X x $Q \rightarrow Q$). Благодаря введенному нами понятию переходного предиката можно формализовать функцию перехода δ: это множество переходных предикатов $P \subseteq \{p_{ij}\}$. Правила перехода можно описать с помощью матрицы смежности графа состояний автомата, где в ячейках, обозначающих смежные вершины, будут расположены не знаки «1», а переходные предикаты. Процесс работы структурного автомата, таким образом, будет представлять собой последовательность активации переходных предикатов и соответствующих им переходов из одного состояния автомата в другое.

Теперь перейдем к описанию реализации структурного автомата на базе ОА-подхода. Для того, чтобы сделать автомат полностью работающим по ОА-принципу, все входные и выходные сигналы будут приниматься и вырабатываться ФУ-источниками и ФУ-вывода (ФУ-коллектор); а внутри ОА-автомата все данные между ФУ будут передаваться с помощью МК: т.е. считанный ФУ-источником внешний входной сигнал представляется в виде ИП (атрибут служит идентификатором сигнала, а в нагрузку помещается значение сигнала), ИП с помощью МК передается на одно или несколько ФУ-состояний в зависимости от того, детерминированным или недетерминированным является ОА-автомат. В отличие от ОА-абстрактного автомата, в структурном автомате могут присутствовать несколько ФУ-источников, независимо друг от друга передающих данные на ФУ-состояния. ФУ состояния, в свою очередь, нуждаются в доработке по сравнению с абстрактными автоматами. Выходные сигналы выдаются в результате выполнения переходных подпрограмм и оформляются в виде ИП, которые передаются на ФУ-коллекторы, преобразующие их во внешние выходные сигналы.

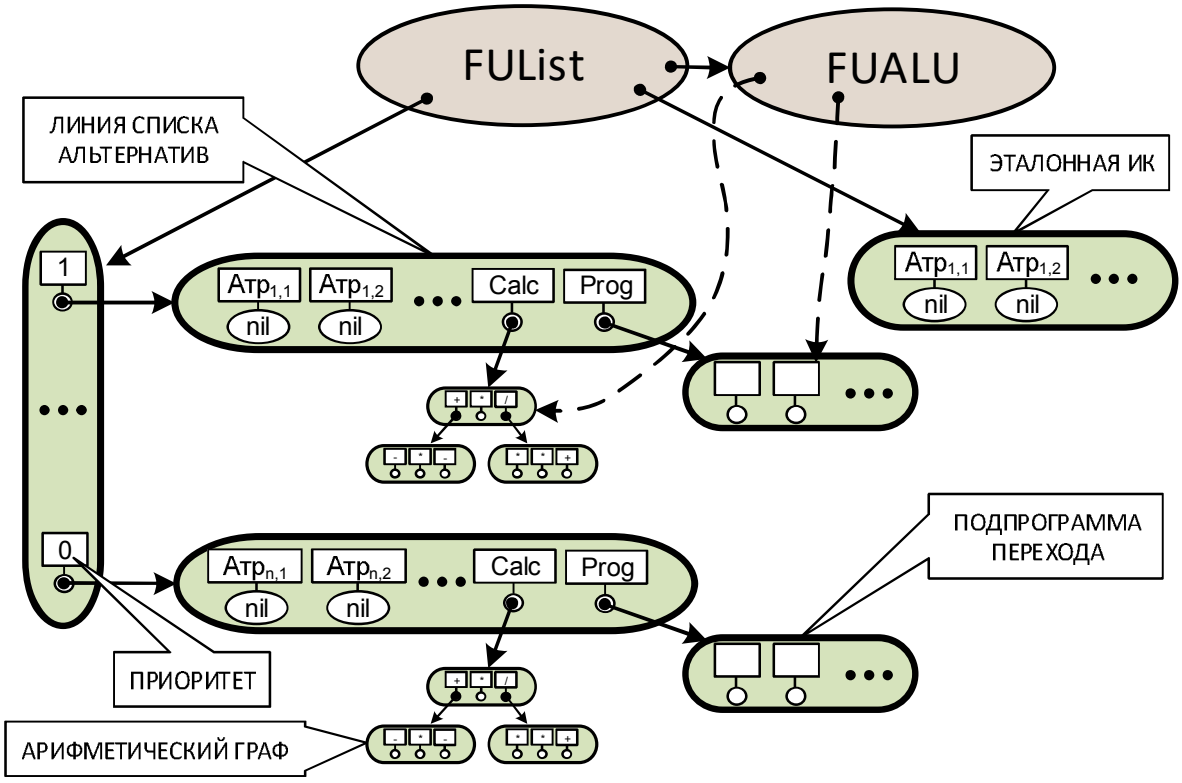

Рис. 2 – ФУ-состояние структурного ОА-автомата

Изменения потребовали и FUList, выполняющие роль ФУ-состояний. Во-первых, в систему последовательно приходит несколько сигналов, и в большинстве случаев переходные предикаты имеют несколько операндов. Следовательно, операнды (т.е. ИП с их описанием) необходимо накапливать. Такое накопление происходит в эталонной капсуле. Очередная пришедшая к ФУ-состоянию ИП описания сигнала помещается в эталонную ИК,

и далее производится равнение этой ИК с линиями (ИК) ОА-писка (рис. 2). В линии списка, контролируемого FUList, описываются переходной предикат и переходная программа. FUList передает значения сигналов из эталонной капсулы в переходные предикаты, описанные в линиях списка. Если один или несколько переходных предикатов выдают true, то происходит активизация соответствующей переходной подпрограммы.

Опишем формат линии ОА-списка структурного автомата. В ИК, входящей в список, расположены ИП с атрибутами сигналов, которые являются аргументами переходного предиката. Нагрузки этих ИП содержат пустой указатель (nil). В линии также находятся ИП с атрибутами Calc и Prog, содержащие в нагрузках, соответственно, указатели на ОА-арифметический граф, предназначенный для вычисления переходного предиката, и на переходную программу. ОА-арифметический граф несколько напоминает арифметическое дерево, используемое для внутреннего представления арифметических выражений в компиляторе. В этом ОА-графе помещаются МК для ФУ АЛУ (арифметико-логическое устройство). Ссылка на ФУ АЛУ, которое будет осуществлять вычисление переходного предиката, находится в контексте FUList. Если атрибуты ИП из эталонной капсулы совпадают с атрибутами из линии списка, FUList передает ссылку на ОА-арифметический граф на FUALU, который производит вычисление переходного предиката. Запускаемая в случае истинности переходного предиката переходная программа осуществляет перенастройку ФУ ОА-автомата и выдачу ИП с выходными сигналами. После этого происходит сброс эталонных ИП во всех ФУ-состояниях, и начинается новый цикл работы ОА-автомата.

В случае, если возникнет ситуация, когда переходной предикат будет истинным сразу для нескольких переходов из одного состояния, можно выбрать всего один переход, исходя из наибольшего приоритета, который можно установить в атрибутах линий списка (рис. 2).

5. Автоматная декомпозиция

Также следует рассказать и об автоматной декомпозиции, которая позволяет существенно упростить проектирование структурного автомата. В [4] выделяется два вида декомпозиции: вложенный и вызываемый автоматы. Для них мы разработали следующее формальное определение. Итак, вложенный автомат сопоставляется с вершиной host-автомата (главного автомата) и может быть описан шестеркой: AI={Σ, QI, QH, S, PI, PH}, где Σ — множество входных символов (сигналов), QI – множество состояний вложенного автомата, QH – множество состояний host-графа, S – начальное состояние вложенного автомата, PI – множество правил перехода между состояниями вложенного автомата ($PI \subseteq \Sigma x QI x QI$), где x – декартово произведение; PH – множество правил перехода в состояния host-графа. Для того, чтобы привязать вложенный автомат к вершине host-автомата, введем функцию, отображающую вершины host-графа на множество вложенных автоматов FI: $QH \rightarrow \{AI\}$, где $\{AI\}$ – множество вложенных автоматов. FI является сюръекцией, т. е. с одной вершиной из множества QH может быть ассоциировано сразу несколько вложенных автоматов. Для того, чтобы хранить текущие состояния вложенных автоматов, введем множество состояний вложенных автоматов host-автомата QIT, а также функцию $QITF$: $QIT \rightarrow QI$, отображающую текущее состояние вложенных автоматов. Функция является биекцией. Если host-автомат попадает в одно из состояний, с которым ассоциируется один или несколько вложенных автоматов, то управление на один шаг автомата передается вложенному автомату, и новое состояние host-автомата определяется вложенными автоматами.

Вызываемый автомат можно описать с помощью следующего формализма: $CA=\{\Sigma, Q, h, s, P\}$, где Σ — множество входных символов (сигналов), Q – множество состояний вызываемого автомата, h – состояние host-автомата ($h \in H$, где H – множество состояний host-

автомата), в которое происходит возвращение из вызываемого автомата (h устанавливается host-автоматом), s – начальное состояние вызываемого автомата, P – правила перехода вызываемого автомата. Возврат из вызываемого автомата происходит по правилу вида: (α_i, q_i) $\rightarrow$ h, где α_i – переходной предикат.

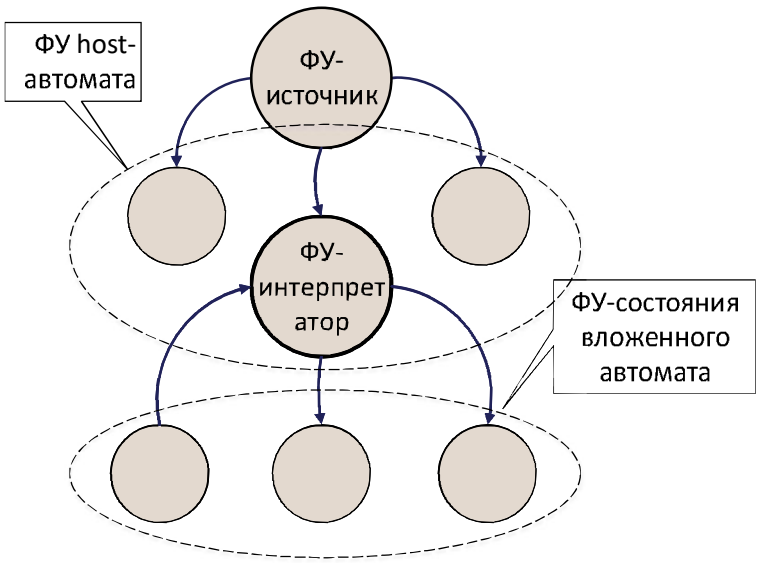

Рис. 3 – Вложенный ОА-автомат

Реализация вложенного автомата по ОА-принципу основывается на применении ФУ типа FUInterpretator (интерпретатор). Он преобразует выходящий поток МК в другой поток. В нашем случае он будет выполнять роль ФУ-состояния host-автомата. Интерпретатор содержит адрес активного ФУ-состояния вложенного автомата. При приходе МК со входными данными он перенаправляет ее на ФУ-состояние вложенного автомата, которое, в свою очередь, выполняет переходную подпрограмму и записывает в интерпретатор адрес ФУ-состояния, вложенного автомата, также при необходимости записывается адрес нового активного ФУ в ФУ-источник host-автомата (рис. 3).

Вложенный автомат представляет собой подмножество ФУ-состояний с выделенным начальным состоянием. При вызове вложенного автомата переходная программа host-автомата записывает в адрес возврата в переходные программы конечных состояний вызываемого автомата – таким образом, если вложенный автомат попадает в одно из конечных состояний, он переходит в следующее состояние host-автомата.

Заключение

В результате проведенного исследования были достигнуты следующие результаты. Во-первых, дано формально определение структурного, вложенного и вызываемого автоматов. Во-вторых, были предложены методики реализации НДА, структурного, вложенного и вызываемого автоматов на базе ОА-подхода, включающие способ организации вычислительной системы, принцип синхронизации вычислений, ввода и вывода данных. Методика может найти применение как при программной, так при аппаратной реализации автоматной модели.

Литература

1. Ахо и др. Компиляторы: принципы, технологии и инструментарий, 2-е изд. : Пер. с англ. – М. : 000 "И.Д. Вильяме",2008.-1184 с.
2. Непейвода Н. Н. Стили и методы программирования. М.: Интернет-университет информационных технологий, 2005.

3. Шалыто А. А. Программная реализация управляющих автоматов // Судостроительная промышленность. Сер. «Автоматика и телемеханика». 1991. Вып. 13, с. 41, 42. http://is.ifmo.ru/works/switch_prr/
4. Поликарпова Н. И., Шалыто А. А. Автоматное программирование. 2008. — 167 с.
5. Jurij Silk, Borut Robic and Theo Ungerer «Asynchrony in parallel computing: From dataflow to multithreading» Institut Jozef Stefan, Technical Report CDS-97-4, September 1997.
6. Салибекян С.М., Панфилов П.Б. ОА-архитектура – новый подход к созданию объектных систем // Объектные системы – 2011: материалы III Международной научно-практической конференции (Ростов-на-Дону 10-12 мая 2011 г.) / Под общ. ред. П.П. Олейника. – Ростов-на-Дону, 2011. – С. 73-79 URL: http://objectsystems.ru/files/Object_Systems_2011_Proceedings.pdf
7. Салибекян С.М., Панфилов П.Б. Объектно-атрибутный подход к построению интеллектуальных систем // Нейрокомпьютеры: разработки и применение. 2011, №11 – с. 9-17.
8. Колесов Ю. Б., Сениченков Ю. Б. Моделирование систем. Динамические и гибридные системы. Учебное пособие. — СПб.: БХВ-Петербург, 2012. — 224 с.
9. Салибекян С.М., Панфилов П.Б. Формализация dataflow-модели вычислительного процесса. // Объектные системы – 2013: материал VII Международной научно-практической конференции (Ростов-на-Дону, 10-12 мая 2013 г.) / Под общ. ред. П.П. Олейника. – Ростов-на-Дону: ШИ (ф) ЮРГТУ (НПИ), 2013. – С. 87-93 URL: http://objectsystems.ru/files/2012/Object_Systems_2013_Proceedings.pdf
10. Салибекян С.М., Панфилов П.Б. Анализ языка с помощью объектно-атрибутной архитектуры вычислительной системы // Программная инженерия. №1, 2013 – стр. 9-16.

УДК 004.4’6

К ВОПРОСУ РАЗРАБОТКИ ИМИТАЦИОННОЙ МОДЕЛИ КОМПЛЕКСА ЭЛЕКТРОСЕТЕВОГО ОБОРУДОВАНИЯ ПЕРЕДАЮЩЕЙ ЭНЕРГОСИСТЕМЫ

Трусов Роман Евгеньевич, студент, Комсомольский-на-Амуре государственный технический университет, Россия, г. Комсомольск-на-Амуре, kreezzzfaqer@gmail.com
Горькавый Михаил Александрович, канд. тех. наук, доцент, Комсомольский-на-Амуре государственный технический университет, Россия, г. Комсомольск-на-Амуре, idpo@knastu.ru

Актуальность работы

Разрабатываемой имитационной модели комплекса электросетевого оборудования предстоит качественно оценивать потери электроэнергии в передающей энергосистеме, исходя из состояния текущего оборудования и влияния внешних факторов на него. Таким образом, используя полученные результаты и, зная, при каких условиях они были получены, станет возможным создание управленческого воздействия на саму систему с целью изменения состояния оборудования для повышения эффективности передающей энергосистемы. Внедрение такой системы в работу передающих энергетических компаний в перспективе сократит потери электроэнергии, возникающие из-за неоптимального использования или несвоевременного ремонта стационарного оборудования. Сокращение потерь затронет ранее открытый вопрос об энергосбережении [1], [2].

Цель работы

Основной целью работы являетс сформировать концептуальную модель комплекса электросетевого оборудования, основанную на классификации стационарного оборудования передающей энергосистемы.

Создание имитационной модели проходит в несколько этапов. В данной работе рассматриваются этапы классификации оборудования и разработка концептуальной модели.