Модел. и анализ информ. систем. Т.**14**, №**1** (2007) 11–18

УДК 519.68/.69

Синхронная модель автоматной программы

Кубасов С.В., Соколов В.А.
Ярославский государственный университет
150 000, Ярославль, Советская, 14

получена 13 декабря 2006

Аннотация

Предлагается модель автоматной программы, удовлетворяющая требованиям синхронной модели. Создание формальной модели автоматной программы предоставляет возможность применения технологий и инструментов верификации синхронных программ для проверки автоматных программ. В частности предполагается использование языка esterel с пакетом программ компании Esterel Technologies Inc. для построения верификатора.

1. Введение

С 90-х годов XX века в России развивается технология автоматного программирования (АП). Она предназначена для программирования управления ответственными проектами, в особенности связанными с реактивными системами. Технология предлагает использовать графы переходов в качестве языка спецификации. АП на данный момент не достаточно формализовано, что затрудняет построение инструментов для автоматической верификации программ. Данная работа предлагает формальную модель автоматной программы. Автоматная модель опирается на синхронное программирование. Это подход к построению реактивных систем, развивающийся на Западе. Синхронное программирование — зрелая технология, она может предложить инструменты для проверки моделей. Ниже более подробно рассматривается синхронный подход и автоматное программирование. В заключение приводится формальная модель автоматной программы.

2. Синхронный подход

Синхронное программирование разрабатывается для создания реактивных систем [1, 2, 3, 4, 5]. В синхронном подходе реактивную систему представляют синхронной моделью. В первом приближении это черный ящик с набором входных и выходных сигналов, имеющий внутреннее состояние. Синхронный подход накладывает ряд ограничений на среду функционирования системы. Дальше реактивная система чаще всего будет рассматриваться в условиях синхронной модели, поэтому, если не утверждается иначе, термин синхронная система означает реактивную систему, удовлетворяющую ограничениям синхронной модели. Ниже описываются основы синхронного подхода.

1. Взаимодействие с окружающей средой при помощи сигналов

Для взаимодействия с окружающий средой применяется единственный способ — сигналы. Входные сигналы устанавливаются окружающей средой и читаются системой. Выходные сигналы генерируются системой, читаются окружающей средой. Обычный сигнал имеет статус присутствия. Сигнал может либо присутствовать, либо отсутствовать. В дополнение каждый сигнал может нести значение определенного типа, например, целое число. Значение сигнала может изменяться только тогда, когда статус сигнала — „присутствует“. В остальных случаях значение сигнала не меняется, но оно всегда есть! Существуют также специальные типы сигналов: чистые сигналы и сенсоры. Первые не имеют значения, вторые — статуса. Значение сенсора может свободно меняться.

2. Временная модель

Синхронный подход использует дискретную временную модель. Каждый отдельный момент времени называется инстентом (от англ. instent). Работа системы в каждый инстент называется реакцией (от англ. reaction). Можно говорить: „реакция системы на входные сигналы“. Для системы время течет как строго упорядоченная последовательность инстентов. Реакция в каждый инстент заключается в вычислении

```
<Инициализация>
<Для каждого> <входного события> <повторять>
  <Вычислить выходные данные>
  <Обновить память>
<Конец цикла>
```

Рис. 1. Модели реактивной системы, управляемой событиями

```
<Инициализация>
<Для каждого> <срабатывания таймера> <повторять>
  <Считать входные данные>
  <Вычислить выходные данные>
  <Обновить память>
<Конец цикла>
```

Рис. 2. Модели реактивной системы, управляемой таймером

значений выходных сигналов текущего инстента и состояния системы в следующий инстент. Вычисление однозначно определяется входными сигналами и внутренним состоянием системы в текущий инстент. Реакции в разные инстенты непосредственно независимы друг от друга, но связаны историей, которая хранится как состояние системы.

До этого момента мы говорили только о логическом времени. В отношение реального времени возможны две разных реализации: управляемая событиями (см. рис. 1) или таймером (см. рис. 2).

3. Гипотеза нулевой задержки

Считается, что все действия в пределах инстента выполняются мгновенно и одновременно. Длительность любой реакции равна нулю. Каждый инстент атомарен, т.е. все действия, запланированные в данном инстенте, выполняются как единое целое.

4. Модель синхронной системы

Любая синхронная система может быть разбита на две взаимодействующие части (см. рис. 3): функциональную логику и память. Функциональная логика — это правила, по которым вычисляются значения выходных сигналов и состояние в следующий инстент. Вычисление однозначно для данных значений входных сигналов и состояния системы. Вычисление повторяется в каждый инстент, имеет нулевую длительность. Память имеет специальный регистровый тип. В пределах инстента сигналы от памяти остаются постоянными. Новое значение, определяемое в текущем инстенте, ячейки памяти получат только в следующем инстенте. Таким образом удается избежать мгновенно обратной связи между входом и выходом логического блока. Заметим, что логический блок не хранит внутри себя какого-либо состояния между инстентами. Его функция только вычислительная.

5. Структура системы

Крупные системы удобно строить из составных блоков. Синхронное программирование позволяет использовать этот подход. Каждый составной блок является подобием целой системы. У него есть входы

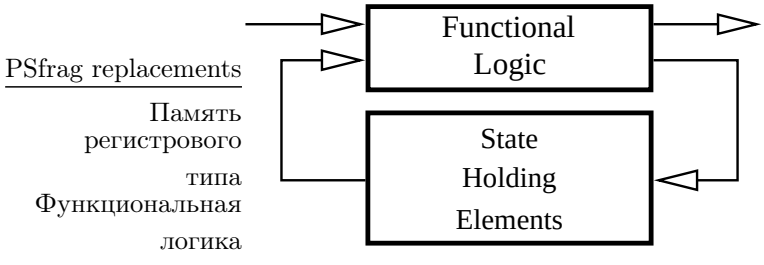

Рис. 3. Модель реактивной системы

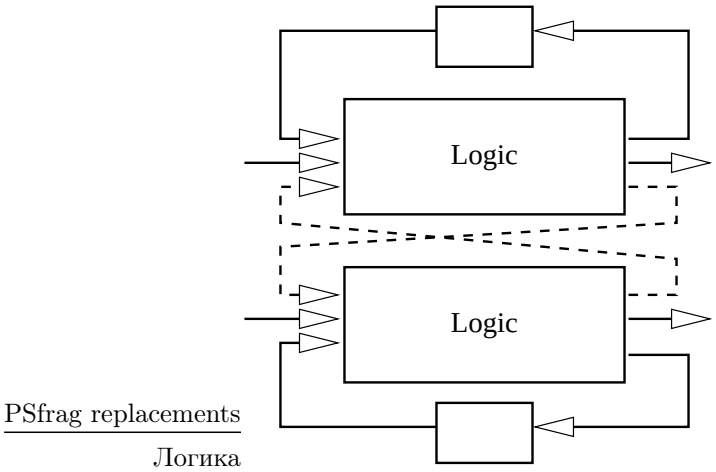

Рис. 4. Соединение двух синхронных блоков

и выходы, блок имеет состояние. Пусть X, Y и U — векторы входных, выходных сигналов и внутреннее состояние соответственно. Нижний индекс — номер блока, верхний индекс — номер инстента. f и g — функции от входных сигналов и внутреннего состояния. Тогда можно записать:

$$U^i_{n+1} = f(U^i_n, X^i_n)$$
$$Y^i_n = g(U^i_n, X^i_n).$$

Набор выходных сигналов в текущий инстент (n) и состояние блока в следующий инстент $(n + 1)$ вычисляется по набору входных сигналов и состоянию в текущий инстент. Это верно для любого блока (i) и любого инстента (n).

Различные блоки соединяются друг с другом посредством входов и выходов. Выходной сигнал одного блока может являться входным сигналом для одного или более других блоков. А значит, входной сигнал может быть общим для нескольких блоков. Пусть символ в скобках обозначает номер сигнала в векторе. Например, $Y(k)$ — k-й сигнал вектора выходных сигналов Y. Тогда:

$$Y^j_n(k) = X^i_n(l)$$

или

$$X^j_n(k) = X^i_n(l).$$

На рис. 4 показана композиция двух синхронных блоков. Пунктирными стрелками отмечены связи между логическими частями разных блоков. Здесь кроется проблема: возможно появление цикла. Таким образом, композиция синхронных блоков в общем случае не является синхронной системой. Существует несколько подходов к решению этой проблемы. В данной статье применяется подход, используемый языком esterel.

6. Язык Esterel

Язык Esterel относится к семейству синхронных языков [6, 7]. Это императивный язык, предназначенный в первую очередь для описания потока управления. Программа синхронизируется с едиными глобальными часами. Система всегда ведет себя функционально. Для любого допустимого состояния системы и значения входных сигналов однозначно определяется следующее состояние и выходные сигналы (формула (1)). Иначе говоря, каждая реакция является детерминированной. Семантическая проверка программы на языке esterel возлагается на компилятор.

$$\{\text{состояние, вх. сигналы}\} \longmapsto \{\text{след. состояние, вых. сигналы}\} \quad (1)$$

3. Автоматное программирование

Автоматное программирование является технологией [8]. Его следует рассматривать как набор рекомендаций к проектированию и кодированию программы. Разработчику дается значительная свобода в реализации. Допускается, и даже приветствуется, одновременное применение автоматного программирования и других технологий, например, объектно-ориентированной [12], процедурной и т.д. На сайте http://is.ifmo.ru представлено множество студенческих работ, демонстрирующих применение автоматного подхода. Можно видеть его успешное сочетание с другими технологиями. АП может быть свободно применено при программировании на любом языке, поддерживающем минимальные конструкции ветвления [11].

Для поддержки автоматного программирования создан проект Unimod [9, 10]. Он предлагает графический редактор автоматов, средства, способствующие написанию и отладке автоматных программ, набор классов и интерфейсов, образующих каркас программы. Разработчик по своему желанию может варьировать долю программы, закодированную с помощью автоматов, в отличие от части, написанной на универсальном языке Java. Однако полностью избежать программирования на языке java не удастся в сколь-нибудь сложной программе.

В чем же заключается основная идея автоматного программирования? АП рекомендует задуматься прежде всего о состояниях проектируемой системы [13]. Сначала мы выделяем все допустимые состояния, затем рассматриваем возможные переходы между ними. Утверждается, что таким образом удается избежать непредусмотренных ситуаций в поведении программы. Думать о состоянии естественно для человека. А значит, проектировщику легче создавать систему.

Гибкость автоматного программирования имеет обратную сторону. Проектируемая система не может быть описана полностью формально. Как бы ни были удобны и естественны для человека средства моделирования, рано или поздно возникает система такого размера, которая уже не может быть полностью охвачена взором проектировщика. Хотелось бы иметь средства для автоматической (с помощью компьютера) проверки требуемых свойств. Для решения этой задачи предлагается формализация автоматного программирования с погружением его в синхронный подход. Цена формализации есть потеря гибкости. В рассмотрении участвуют только автоматы. Весь код на языке java или любом другом языке отметается. Это не означает, что высокоуровневые языки невозможно будет использовать. Они могут применяться в окружении. Сущность подхода состоит в том, чтобы выделить часть программы, состоящую только из взаимодействующих автоматов, и проверять эту часть. Отверифицированная модель может быть использована, например, из java программы, но мы должны заранее определить и ограничить все возможные способы воздействия кода на java на нашу модель. Ниже будет показано, как это можно осуществить.

4. Синхронная модель автоматной программы

Автоматная программа реализует реактивную систему. Синхронный подход к автоматному программированию рассматривает реактивную систему в синхронной модели. Такую систему будем называть синхронной системой. Основной элемент синхронной модели автоматной программы — синхронный автомат. Простейшая синхронная автоматная модель состоит из одного автомата. Несколько синхронных автоматов можно объединить в синхронную сеть автоматов. Это более сложная синхронная модель. Правила построения синхронной автоматной сети позволяют строить сеть не только из автоматов, но и из других синхронных сетей. Такие сети будем называть просто синхронными сетями. Синхронная сеть также представляет синхронную модель.

1. Синхронный автомат

Определим синхронный автомат следующим образом:

$$A = (Q, q_0, X, Y, \Phi).$$

Здесь

$$Q = \{ q_0, q_1, \ldots, q_n \}$$

— множество внутренних состояний автомата, причем $n \geqslant 0$. q_0 — начальное состояние автомата.

$$X = \{ x_1, \ldots, x_m \}$$

— множество входных сигналов, причем $m \geqslant 0$. Если $m = 0$, то множество входных сигналов пусто. Каждый сигнал может принимать значения 0 и 1.

$$Y = \{y_1, \ldots, y_k\}$$

— множество выходных сигналов, причем $k \geqslant 0$. Если $k = 0$, то множество выходных сигналов пусто. Каждый сигнал может принимать значения 0 и 1.

Из множества входных сигналов строится входной алфавит автомата L:

$$L ::= 0 \mid 1 \mid x_i \mid L \wedge L \mid L \vee L \mid \neg L \mid (L).$$

Как можно видеть, алфавит L состоит из констант 0 и 1 (логические истина и ложь), всех входных сигналов и любой комбинации перечисленных элементов, соединенных с помощью логических операций конъюнкции, дизъюнкции и отрицания. Для однозначной интерпретации формул будем считать, что все логические операции имеют разный приоритет, убывающий в такой последовательности: $\neg$, $\wedge$, $\vee$. Скобки используются для переопределения приоритета.

Пусть также

$$R = \{y_{i_1}, \ldots, y_{i_l} \mid l \in \mathbb{N} \cup \{0\},\ y_{i_j} \in Y\}$$

— множество выходных реакций. Элементы этого множества — списки выходных сигналов. В списке каждый сигнал присутствует не более одного раза. Списки, отличающиеся только порядком сигналов, считаются одинаковыми. Допускается пустой список.

$$\Phi\colon Q \times L \to Q \times R$$

— функция переходов. Переход определяется начальным состоянием и логической формулой. Новое состояние может совпадать с начальным, т.е. переход является петлей. Между любыми двумя состояниями (в том числе совпадающими) может быть любое число переходов. Все переходы, выходящие из любого состояния, должны быть помечены ортогональными формулами, чтобы для любой комбинации входных сигналов нашлось не более одного перехода. Однако не обязательно, чтобы каждому набору входных сигналов соответствовал переход. Если перехода нет, по умолчанию считаем, что система не меняет состояния и не генерирует выходных сигналов.

2. Синхронная сеть автоматов

Пусть есть n синхронных автоматов:

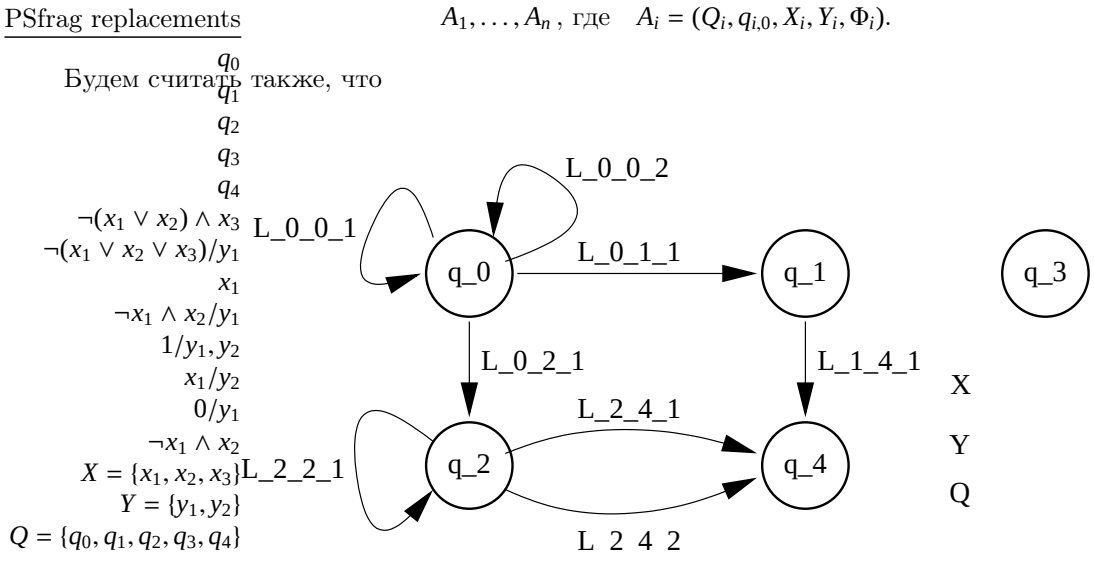

Рис. 5. Пример синхронного автомата

$$\bigcup_{i=1}^{n} X_i = X, \qquad \bigcup_{i=1}^{n} Y_i = Y.$$

Объединим автоматы в сеть. Для этого свяжем некоторые сигналы разных автоматов. Определим группы, состоящие из любого (но не нулевого) числа входных сигналов, и необязательного выходного сигнала. Обозначим группу связанных сигналов K:

$$K \in \{\, \tilde{X} \cup \tilde{Y} \mid \tilde{X} \subseteq X,\ \tilde{X} \neq \varnothing,\ \tilde{Y} \subseteq Y,\ |\tilde{Y}| \leqslant 1 \,\}.$$

Иначе говоря, группа K состоит из непустого подмножества входных сигналов X и одного необязательно выходного сигнала из множества Y. Множество всех групп обозначим $\overline{K}$. В частном случае множество $\overline{K}$ может быть пустым.

Наконец, определим внешний интерфейс синхронной сети. Выберем входные сигналы некоторых автоматов и назовем их входными сигналами системы автоматов $\overline{X}$. Если какой-то из входных сигналов входит в группу, то нужно брать всю группу, считая ее за один входной сигнал системы автоматов. Однако нельзя брать сигналы из группы, содержащей выходной сигнал. Иначе получилось бы, что вход системы автоматов самоопределяется системой. Выберем выходные сигналы некоторых автоматов и назовем их выходными сигналами группы автоматов $\overline{Y}$. Выходные сигналы автоматов, являющиеся выходом системы автоматов, могут относиться к группам. Множество $\overline{X}$, как и множество $\overline{Y}$, может быть пустым. Формально все эти условия можно записать следующим образом:

$$\overline{X} \subseteq X$$
$$\forall K \in \overline{K}\colon (K \cap \overline{X} \neq \varnothing \Rightarrow K \subseteq \overline{X})$$
$$\overline{Y} \subseteq Y$$

Добавим еще несколько условий. Состояние каждого входного сигнала каждого автомата может быть отнесено к одному из следующих случаев: 1) входит в группу, в которой есть выходной сигнал; 2) является входным сигналом сети автоматов (либо не входит ни в одну группу, либо входит в группу без выходного сигнала); 3) входит в группу без выходного сигнала, но не является входным сигналом сети автоматов; 4) ни к чему не присоединен. В последних двух случаях будем считать, что сигнал всегда равен нулю. Выходные сигналы также могут быть ни с чем не связанными. В этом случае значение сигнала определяется по общим правилам, но не используется — пропадает.

Условия построения сети автоматов допускают существование интересных частных случаев. Сеть может не иметь входных или выходных сигналов, а также ни тех ни других. Сеть автоматов может быть несвязной. В сеть автоматов могут включаться изолированные автоматы, т.е. такие автоматы, все входы и выходы которых ни к чему не присоединены. Правила построения сети однозначно определяют, как будет себя вести любая система. С практической точки зрения многие частные случаи не рациональны

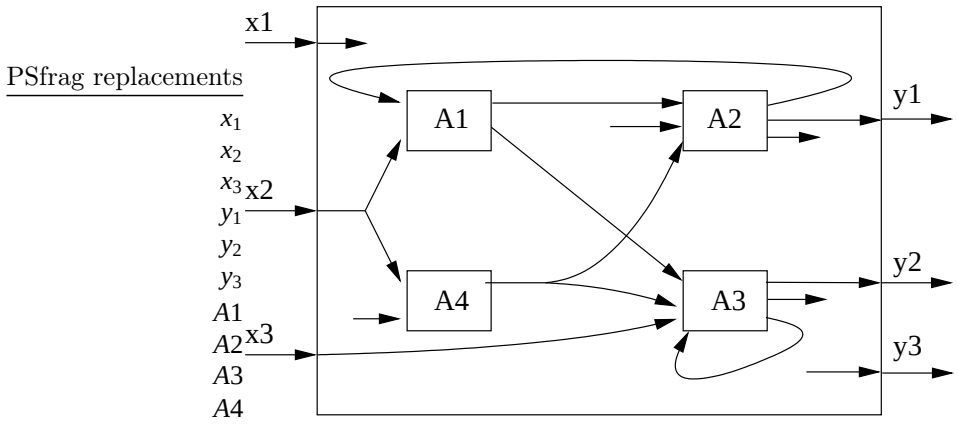

Рис. 6. Пример синхронной сети автоматов. Здесь $A1$, $A2$, $A3$, $A4$ — некоторые автоматы. Вся сеть имеет три входных и три выходных сигнала.

или не имеют смысла. От изолированных автоматов можно избавиться без потери каких-либо свойств. Сеть автоматов без входов и выходов тоже бесполезна. Однако задача упрощения сети выходит за рамки данной работы. Мы будем рассматривать любые сети и автоматы, подходящие под определение, вне зависимости от их практической ценности.

3. Синхронное поведение модели

Синхронная автоматная модель ведет себя как общая синхронная система. Вспомним основные идеи синхронного подхода и укажем, как они реализуются в синхронной автоматной модели. Взаимодействие с окружающий средой происходит с помощью сигналов (часть 1.), у системы есть входные и выходные сигналы. В модели синхронных автоматов используются только чистые сигналы, у них нет связанного значения. Дискретная временная модель (часть 2.). Все автоматы модели работают согласно единым глобальным часам. В каждый инстент на основании входных сигналов и текущего состояния вычисляются выходные сигналы и следующее состояние. Реальное время в модели не используется. Подходит любая из моделей рис. 1 и рис. 2. Гипотеза нулевой задержки (часть 3.). Реакция автоматов вычисляется мгновенно. Разбиение системы на две части (рис. 3) почти очевидно. Память регистрового типа — внутренние состояния автоматов. Особенность регистровой памяти обеспечивается соглашениями о выполнении модели. Логика кодируется в условиях переходов между состояниями, а также в соединениях подсистем (автоматов, сетей автоматов и т.д.). Проблема соединения двух (и более) синхронных подсистем (рис. 4) остается актуальной. Ее решение в детерминированном подходе языка esterel (формула (1)).

Сеть автоматов планируется для реализации на языке esterel. Компилятор и другие вспомогательные утилиты языка esterel будут обеспечивать проверку большинства условий, наложенных на систему.

5. Заключение

В настоящей работе предложена модель автоматной программы, удовлетворяющая требованиям синхронной модели. Синхронную модель автоматной программы можно использовать для построения верификатора автоматной программы. Проверка может быть выполнена средствами уже существующего верифицирующего компилятора языка esterel и др. утилитами.

Автоматная модель удобна для разработчика. Она позволяет разделить иерархическую декомпозицию системы на уровни, что становится важным при проектировании крупных систем.

Список литературы

1. Шопырин, Д. Г. Синхронное программирование / Д. Г. Шопырин, А. А. Шалыто // Информационно-управляющие системы. – 2004. – №3. – С. 35–42.

2. Benveniste, A. The Synchronous Languages 12 Years Later: Invited Paper / A. Benveniste, P. Caspi, S.A. Edwards, N. Halbwachs, P. le Gurnic, R. de Simone // Proceedings of the IEEE. – 2003, January. – vol. 91, no. 1. – pp. 64–83.

3. Andre, C. Representation and Analysis of Reactive Behaviors: A Synchronous Approach / C. Andre // Computational Engineering in Systems Applications (CESA), IMACS Multiconference. – Lille, France. – 1996, July. – pp. 19–29. – ISBN 2-9502908-7-6.

4. Barros, T. Formal specification and verification of distributed component systems: PhD thesis: defence date: 25.11.2005 / T. Barros; Universite de Nice; INRIA Sophia Antipolis. – France, 2005. – 158 p.

5. Halbwachs, N. Synchronous programming of reactive systems, a tutorial and commented bibliography / N. Halbwachs // Tenth International Conference on Computer-Aided Verification, CAV'98, Vancouver (B.C.), Canada: LNCS 1427, Springer Verlag, 1998, June. – vol. 1427. – pp. 1–16. – ISBN 3-540-64608-6.

6. Berry, G. The Esterel v5 Language Primer, version v5_91: Документация к программному обеспечению Esterel Compiler, version 5.91. Ecole des Mines de Paris (EMP), ARMINES, and INRIA / G. Berry and the Esterel Team. – 2000, 5 June.

7. Berry, G. The Esterel v5_91 System Manual: Документация к программному обеспечению Esterel Compiler, version 5.91. Ecole des Mines de Paris (EMP), ARMINES, and INRIA / G. Berry. – 2000, 5 June.

8. Кузнецов, Б. П. Психология автоматного программирования / Б. П. Кузнецов // BYTE / Россия. – 2000. – №11. – С. 22–29.

9. Гуров, В. С. UML. SWITCH-Технология. Eclipse / В. С. Гуров, М. А. Мазин, А. С. Нарвский, А. А. Шалыто // Информационно-управляющие системы. – 2005. – №6(13). – С. 12–17.

10. Гуров, В. С. Исполняемый UML из России / В. С. Гуров, А. С. Нарвский, А. А. Шалыто. // PC Week/RE. – 2005. – №26. – С. 18–19.

11. Шалыто, А. А. SWITCH-Технология — автоматный подход к созданию программного обеспечения «реактивных» систем / А. А. Шалыто, Н. И. Туккуль // Программирование. – 2001. – №5. – С. 45–62.

12. Наумов, А. С. Объектно-ориентированное программирование с явным выделением состояний: Бакалаврская работа / А. С. Наумов; СПбГУ ИТМО. – СПб., 2004. – 209 с.

13. Шалыто, А. А. Программирование с явным выделением состояний / А. А. Шалыто, Н. И. Туккель // Мир ПК. – 2001. – №8. – С. 116–121; №9. – С. 132–138.

Synchronous model of automaton program

Kubasov S.V., Sokolov V.A.

This article presents a model of automaton program that satisfies synchronous model requirements. A formal automaton program model lets to use an existing technologies and tools of synchronous programs verification for checking automaton programs. An Esterel language and an Esterel Technologies Inc. toolbox will be used to build a program verifier.