*Модел. и анализ информ. систем.* Т.**14**, №**4** (2007) 20–27

УДК 519.68

# Верификация синхронно-автоматных программ

Кубасов С.В.
*Ярославский государственный университет*
*e-mail: kubasovsv@univ.uniyar.ac.ru*

*получена 22 октября 2007*

**Аннотация**

Предлагается синхронная модель автоматной программы. Разработана методика верификации синхронно-автоматных программ. Некоторые свойства проверяются автоматически. Есть возможность проверки пользовательских свойств. Применение этой методики позволит выявить большое число ошибок, допускаемых в процессе разработки.

## 1. Введение

Разработка систем логического управления долгое время остается актуальной задачей. Со временем такие системы все более усложняются и все острее встает проблема корректности разработанных программ изначальным требованиям. В работе [1] была предложена синхронно-автоматная модель для программ логического управления, основанная на switch-технологии. В этой статье подводятся промежуточные итоги разработки модели и средств ее верификации.

В настоящее время полностью оформилась синхронно-автоматная модель. По сравнению с [1] она претерпела небольшие изменения. Предлагаются средства автоматической и полуавтоматической проверки модели. Есть возможность проверки пользовательских свойств.

## 2. Модель синхронно-автоматной программы

Синхронно-автоматная модель описывает синхронную систему. Основные компоненты модели — это синхронные автоматы и синхронные сети. Каждый из них в отдельности сам представляет синхронную систему. Изоморфизм представления достигается за счет унифицированного интерфейса.

### 2.1. Интерфейс синхронной системы

Интерфейс $I = (X, Y)$ синхронной системы определяется парой, состоящей из набора входных сигналов $X = \{x_1, \ldots, x_m\}$ $(m \geqslant 0)$ и выходных сигналов $Y = \{y_1, \ldots, y_k\}$ $(k \geqslant 0)$. Как можно видеть, оба списка сигналов могут быть пустыми. Каждый сигнал может принимать значения 0 и 1.

### 2.2. Синхронный автомат

Интерфейс $I = (X, Y)$ может быть реализован синхронным автоматом, который определяется следующим образом:

$$A = (Q, q_1, X, Y, \Phi).$$

Здесь

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

— множество внутренних состояний автомата, причем $n \geqslant 1$. $q_1$ — начальное состояние автомата.

Из множества входных сигналов строится входной алфавит автомата $L$:

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

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 состояние и не генерирует выходные сигналы.

### 2.3. Синхронная сеть

Интерфейс $I = (X, Y)$ может быть реализован синхронной сетью, которая представляется следующим образом:

$$N = (\mathcal{I}, X, Y, G).$$

Синхронная сеть строится из нескольких синхронных систем. Чем они представлены — не имеет значения, используется только интерфейс. Поэтому набор синхронных систем определяется только их интерфейсами: $\mathcal{I} = \{ I_1, \ldots, I_n \}$, где $n \geqslant 1$.

Входные и выходные сигналы синхронной сети, а также ее составных частей, определяемых интерфейсами, соединяются друг с другом, образуя группы сигналов: $G = \{G_1, \ldots, G_f\}$, где $f \geqslant 0$. Количество групп может быть любым, в том числе нулевым. Все сигналы можно разделить на ведущие и ведомые. К первым относятся входные сигналы интерфейса сети и выходные сигналы ее компонентов, ко вторым — все остальные. В каждую группу должен входить ровно один ведущий сигнал и не менее одного ведомого. Любой из сигналов может входить не более чем в одну группу в пределах данной сети, но может участвовать в двух разных группах, относящихся к разным сетям.

### 2.4. Модель в целом

Для моделирования простой синхронной системы достаточно одного автомата. Иначе приходится задействовать механизм группировки автоматов с помощью сетей. В любом случае существует компонент, описывающий всю систему — главный объект модели. В случае модели из одного автомата он же является главным элементом модели. Для модели, содержащей сети, главным элементом модели будет сеть, которая явным или косвенным образом включает в себя все остальные компоненты.

Заметим, что ни одна модель не может обойтись без автоматов. Они выполняют основную работу. Благодаря сетям мы можем произвольным образом группировать существующие блоки (построенные синхронные системы). Одна и та же синхронная система может быть многократно использована в пределах модели.

## 3. Верификация программы

Описание синхронно-автоматной модели хранится в файле в специально разработанном XML-формате. При помощи автоматизированного скрипта XML-файл преобразуется в esterel-программу [3, 2]. При этом структура модели сохраняется. Синхронные автоматы и синхронные сети преобразуются в модули языка esterel. Интерфейс синхронно-автоматной системы соответствует интерфейсу модуля esterel. Обозначения элементов, где это возможно, не изменяются.

Задача процедуры верификации заключается в выявлении разнообразных ошибок, которые могут возникнуть на этапе разработки. Можно выделить три типа проверок. Каждая из них имеет дело со своим классом ошибок.

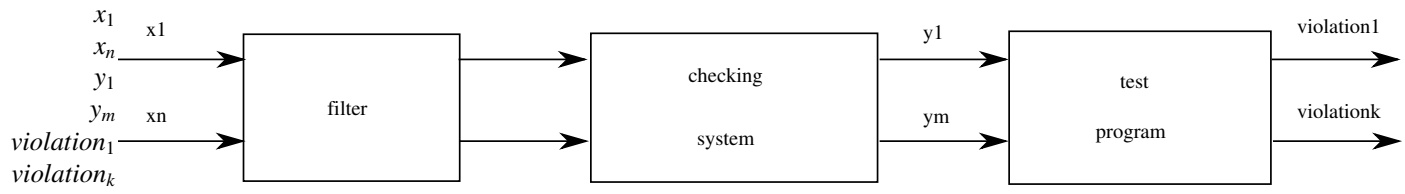


Рис. 1. Проверка пользовательских свойств синхронно-автоматной модели

### 3.1. Формат модели

В процессе преобразования модели в esterel-программу выполняются разнообразные проверки. В основном они касаются формата модели, принятых обозначений и т.п. Например, в текущей реализации требуется, чтобы имя автомата состояло из заглавной буквы A и следующего за ним номера. Или другой пример: в группу сигналов синхронной сети должен входить ровно один ведущий сигнал.

### 3.2. Синхронность

К синхронным свойствам модели относятся детерминированность и единственность генерации каждого сигнала. Свойство детерминированности означает, что при любом наборе входных данных значения выходных сигналов определяются однозначно. Свойство единственности генерации сигнала означает, что в каждом инстенте любой сигнал либо отсутствует, либо генерируется ровно одним оператором. Выполнение двух операторов, генерирующих один и тот же сигнал, является недопустимым. Выполнение свойства единственности генерации каждого сигнала обеспечивается алгоритмом преобразования синхронно-автоматной модели в программу на языке esterel, тем не менее оно все равно проверяется. Средства проверки встроены в компилятор языка esterel.

К ошибкам нарушения синхронности можно отнести сообщение о существовании циклической зависимости между сигналами, которая делает невозможным однозначное определение значения выходных сигналов. Такое может произойти только при неаккуратном использовании циклических зависимостей в синхронных сетях.

### 3.3. Пользовательские свойства

Проверка пользовательских свойств является наиболее привлекательной частью всей верификации, т.к. она позволяет проконтролировать критические характеристики разрабатываемой системы.

Пользовательское свойство описывается специальной программой на языке esterel, которая подсоединяется к верифицируемой системе (рис. 1). Выходы верифицируемой системы $y_1, \ldots, y_m$ становятся входами программы проверки. Выходы программы проверки $violation_1, \ldots, violation_k$ состоят из одного или нескольких сигналов, которые генерируются в случае нарушения свойств. Задача верификатора — определить, существует ли последовательность наборов входных сигналов $x_1, \ldots, x_n$, приводящая к генерации тревожного сигнала. Если тревожный сигнал никогда не генерируется, то проверяемое свойство выполнено.

Верификатор Xeve [4] компании Inria создан для проверки программ на языке esterel. Его работа состоит в том, чтобы отслеживать поведение системы при всех возможных историях входных данных. Для выбранного выходного сигнала верификатор может установить, что данный сигнал когда-либо генерируется или всегда отсутствует. Эта возможность используется для проверки пользовательских свойств. Дополнительно Xeve может решать задачу минимизации программы.

Обычно верификация проводится на всех возможных комбинациях входных данных, но можно ввести ограничения. Пишется еще одна вспомогательная программа-фильтр, вставляемая перед проверяемой системой. Программа фильтр имеет такой же набор сигналов, как проверяемая система. Все недопустимые комбинации сигналов отсеиваются за счет преобразования их в допустимую комбинацию.

Синхронные системы имеют память, поэтому важна не только конкретная раскладка сигналов по входам в данном инстенте, но и последовательность таких раскладах во времени. Этот аспект усложняет написание программы-фильтра.

# 4. Пример синхронно-автоматной программы — часы-будильник

## 4.1. Описание задачи

В качестве примера для демонстрации возможностей синхронно-автоматного программирования были выбраны часы с будильником. Подобные модели часов широко распространены и, вероятно, знакомы читателю. Внешний вид устройства схематично показан на рис. 2. Центральное место занимает жидкокристаллический экран, служащий для отображения времени и других значков. Правый верхний угол отдан динамику. С его помощью часы издают различные мелодии, подают звуковые сигналы, а в некоторых моделях даже сообщают текущее время. Нижнюю часть панели занимают кнопки “Mode”, “Hour”, “Min”, служащие для выбора режима, часа и минуты соответственно. Впрочем, последние две кнопки могут играть разную роль в зависимости от режима.

В рассматриваемом примере устройство обладает такими возможностями: отображение текущего времени, установка времени звонка, выбор мелодии звонка (одна из трех), включение/выключение звонка, установка/отмена режима ежечасного оповещения.

Заметим, что программа моделирует только процесс настройки будильника, задачи отсчета времени, включение звонка и ежечасных оповещений не рассматриваются.

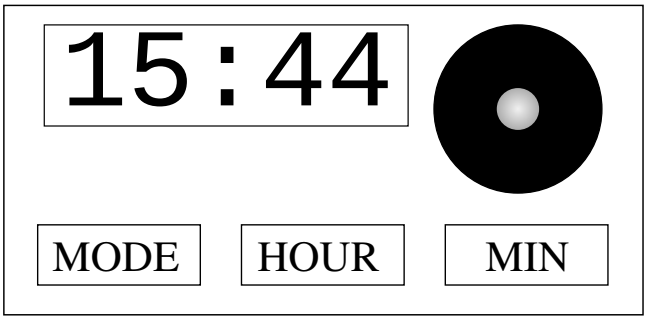


Рис. 2. Внешний вид часов с будильником

## 4.2. Реализация

Программа часов с будильником состоит из четырех автоматов и одной синхронной сети. Синхронная сеть Net1 представляет всю систему в целом, это главный объект модели. Автоматы A1–A4 каждый решают свою подзадачу. Они соединяются в рамках синхронной сети Net1.

Входные сигналы сети Net1 состоят из сигналов трех кнопок и одного сигнала таймера. Выходные сигналы сети Net1 выполняют разнообразные функции: вкл./выкл. таймера, включение мелодии будильника, переключения режимов отображения, установка текущего времени и времени будильника. Сама сеть, по сути, ничего не делает, она только соединяет свои компоненты друг с другом, некоторые сигналы выводятся на интерфейс.

Автомат A1 реализует основную функциональность программы. Он имеет четыре состояния: “время”, “время с установленным будильником”, “настройка текущего времени”, “настройка будильника”. Входные сигналы такие же, как у сети Net1, но взаимоисключающие. В каждый инстент может прийти не более одного входного сигнала. Такое решение было принято для упрощения логики работы устройства. Большинство случаев, когда одновременно приходят несколько сигналов, относятся к ошибочным действиям пользователя. Например, одновременно было нажато две кнопки. Чтобы не усложнять алгоритм управления, специально для фильтрации сигналов был создан отдельный модуль A2.

Выходные сигналы A1 разнообразны, см. таблицу 1. Их особенность в том, что для переключения различных режимов используются кратковременные сигналы, т.е. сигналы, которые длятся один инстент. Например, если нам надо переключить дисплей в режим мигания, мы посылаем соответствующий сигнал, а не поддерживаем этот сигнал все время, пока требуется, чтобы дисплей мигал. Интерфейс Net1 предполагает противоположное поведение. Специально для решения этой задачи был разработан модуль A4.

Внутреннее устройство автомата A1 должно быть ясно из рисунка 3.

Таблица 1. Интерфейсные сигналы и внутренние состояния автомата A1

| | |
|---|---|
| q1 | Часы |
| q2 | Часы + звонок включен |
| q3 | Установка текущего времени |
| q4 | Установка времени звонка |
| x1 | Кнопка “mode” |
| x2 | Кнопка “minute” |
| x3 | Кнопка “hour” |
| x4 | Сигнал таймера |
| y1 | Включить таймер ожидания |
| y2 | Выключить таймер ожидания |
| y3 | Вкл./выкл. режим мигания дисплея |
| y4 | Переключить на следующую мелодию звонка и проиграть ее |
| y5 | Вкл./выкл. ежечасные оповещения |
| y6 | Вкл./выкл. звонок |
| y7 | Текущее время + 1 минута |
| y8 | Текущее время + 1 час |
| y9 | Время звонка + 1 минута |
| y10 | Время звонка + 1 час |
| y11 | Переключить: показывать часы/звонок |

Таблица 2. Интерфейсные сигналы и внутренние состояния автомата A2

| | |
|---|---|
| q1 | Единственное состояние |
| x1 | Кнопка “mode” |
| x2 | Кнопка “minute” |
| x3 | Кнопка “hour” |
| x4 | Сигнал таймера |
| y1 | Кнопка “mode” |
| y2 | Кнопка “minute” |
| y3 | Кнопка “hour” |
| y4 | Сигнал таймера |

Таблица 3. Интерфейсные сигналы и внутренние состояния автомата A3

| | |
|---|---|
| q1 | Звонок 1 |
| q2 | Звонок 2 |
| q3 | Звонок 3 |
| x1 | Переключить на следующую мелодию звонка и проиграть ее |
| y1 | Проиграть звонок 1 |
| y2 | Проиграть звонок 2 |
| y3 | Проиграть звонок 3 |

Таблица 4. Интерфейсные сигналы и внутренние состояния автомата A4

| | |
|---|---|
| q1 | Выключено |
| q2 | Включено |
| x1 | Переключить состояние |
| y1 | Сигнал состояния |

Таблица 5. Интерфейсные сигналы синхронной сети Net1

| | |
|---|---|
| x1 | Кнопка “mode” |
| x2 | Кнопка “minute” |
| x3 | Кнопка “hour” |
| x4 | Сигнал таймера |
| y1 | Включить таймер ожидания |
| y2 | Выключить таймер ожидания |
| y3 | Проиграть звонок 1 |
| y4 | Проиграть звонок 2 |
| y5 | Проиграть звонок 3 |
| y6 | Ежечасные оповещения |
| y7 | Звонок (вкл./выкл.) |
| y8 | Текущее время + 1 минута |
| y9 | Текущее время + 1 час |
| y10 | Время звонка + 1 минута |
| y11 | Время звонка + 1 час |
| y12 | Показывать часы/звонок |
| y13 | Режим мигания дисплея (вкл./выкл.) |

Задача автомата A2 в фильтрации входных сигналов автомата A1. Выходные сигналы A2 соответствуют входным сигналам A1, а входные сигналы A2 — входным сигналам Net1. A1 требует, чтобы входные сигналы были взаимоисключающими. Наивысший приоритет имеет сигнал от таймера. Он пропускается всегда, подавляя все остальные сигналы. Если сигнала от таймера нет, пропускаются сигналы кнопок, но только в том случае, если нажата только одна из кнопок. Если нажато несколько кнопок одновременно (сигнала от таймера нет), все сигналы отфильтровываются.

Автомат A3 хранит текущую мелодию звонка, а также запускает мелодию на проигрывание при смене состояния. В реальном будильнике нам обязательно бы потребовался сигнал для проигрывания текущей мелодии, генерируемый в тот момент, когда подошло время звонка. В модели такой сигнал не предусмотрен, т.к. все, что нам требуется, — это выбор и хранение мелодии звонка. Поэтому проигрывание связано с переключением состояния.

Автомат A4 используется для реализации простейшего переключателя на два положения. У него есть один входной и один выходной сигнал. Выходной сигнал сохраняет свое значение из инстента в инстент, пока не придет переключающий входной сигнал. Входной сигнал изменяет значение выходного сигнала на противоположное и фиксирует это значение вплоть до следующего переключения.

Автомат A4 был создан для согласования отдельных сигналов интерфейса автомата A1 с сигналами интерфейса синхронной сети Net1. Синхронная сеть Net1 использует несколько экземпляров автомата A4, таким образом демонстрируется многократное использование разработанного кода.

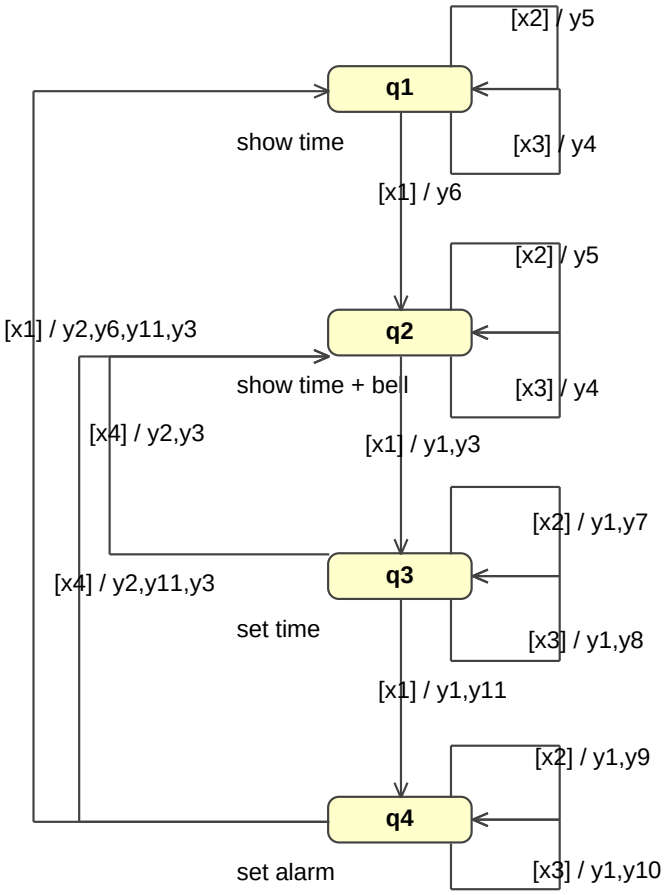


Рис. 3. Диаграмма автомата A1

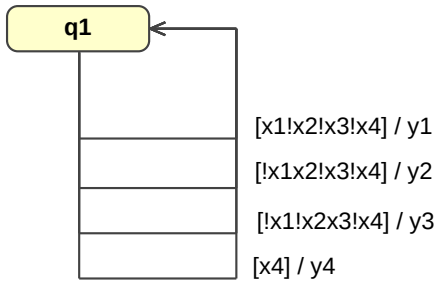


Рис. 4. Диаграмма автомата A2

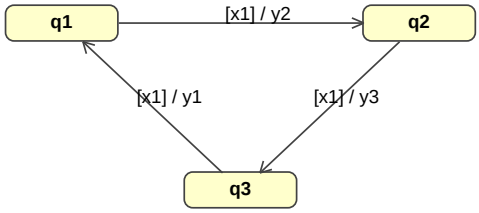


Рис. 5. Диаграмма автомата A3

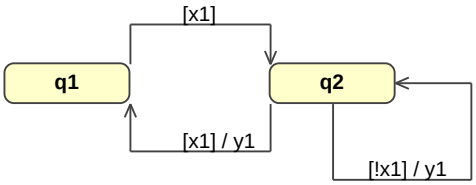


Рис. 6. Диаграмма автомата A4

Работа программы должна быть ясна из приведенных диаграмм и таблиц описания.

## 4.3. Проверка пользовательских свойств

Программа моделирования часов с будильником содержит несколько допущений, выполнение которых неочевидно из кода программы. Очень удачно, что для их проверки мы можем воспользоваться специально разработанными средствами верификации синхронно–автоматных программ. Можно рассмотреть такие свойства.

- Сигналы запуска и остановки таймера никогда не генерируются в одном инстенте.
- Входные сигналы автомата A1 являются взаимоисключающими.
- Сигналы проигрывания мелодии являются взаимоисключающими.
- Значок установленного звонка показывается только в состоянии q1 автомата A1.
- Таймер выключен, только когда автомат A1 находится в состояниях q1 и q2.
- В состояниях q1, q2, q3 автомата A1 дисплей показывает текущее время, в состоянии q4 — время будильника.

В окончательной версии программы текущее состояние автомата A1, как и любые другие сигналы, не относящиеся к интерфейсным, недоступно. Однако специально для проверки пользовательских свойств разработан особый режим генерации программы на языке esterel, в котором экспортируются все внутренние сигналы. Используя этот режим, мы можем проверить перечисленные выше свойства.

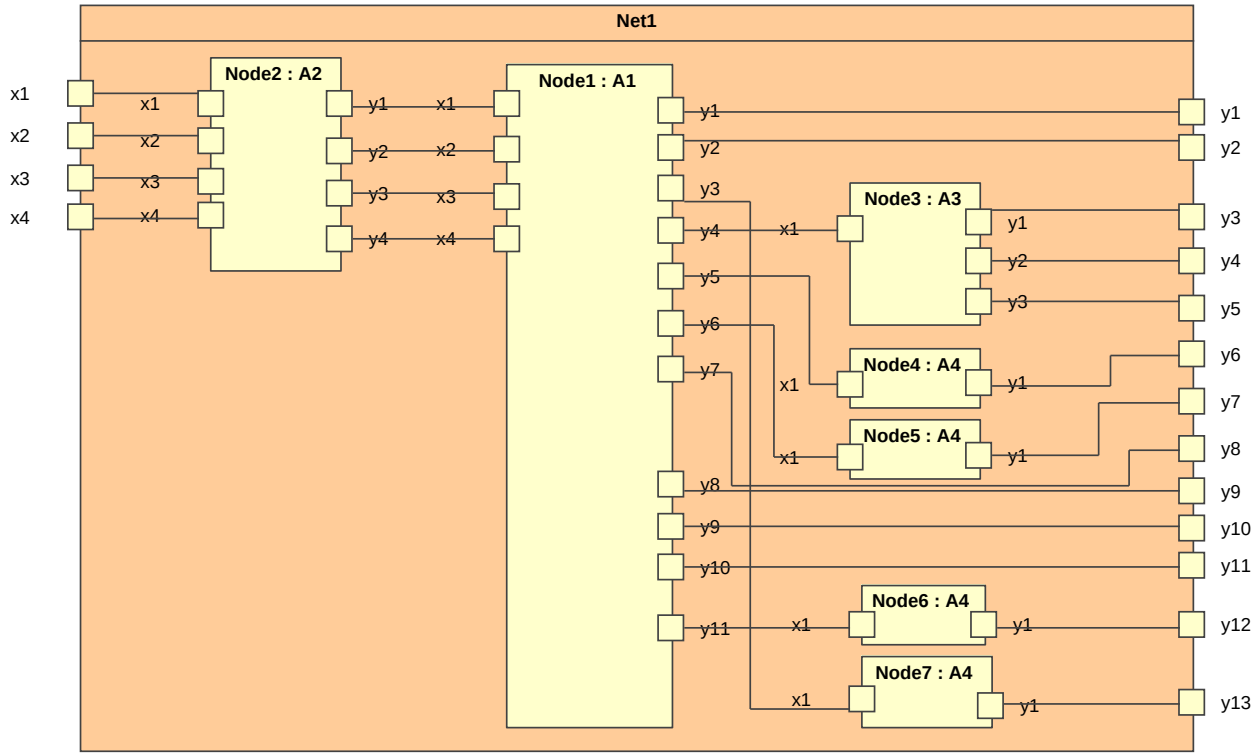


Рис. 7. Диаграмма синхронной сети Net1

Каждое свойство выражается отдельным модулем на языке esterel. Обычно достаточно 5–10 строк кода. Для примера рассмотрим свойство, связанное с проигрыванием мелодии (рис. 8).

Первая строка — комментарий. Во второй строке указывается имя модуля: *PlayRingSigsAlternativeViolation*. Строки 3–4 определяют интерфейс модуля. Входные сигналы x1–x4 играют роль локальных переменных. Они будут подсоединены к выходным сигналам y3–y5 синхронной сети Net1 соответственно. Выходной сигнал *VIOLATED* генерируется в случае нарушения свойства. Условие генерации сигнала *VIOLATED* записано в строке 6. Мы просто перебираем все возможные пары сигналов. Оператор *present* заключен в бесконечный цикл *loop* (строки 5–10). Проверка выполняется в каждом инстенте.

Проверка остальных свойств выглядит примерно так же.

```
 1: % Сигналы проигрывания мелодии являются взаимоисключающими
 2: module PlayRingSigsAlternativeViolation:
 3: input x1, x2, x3;
 4: output VIOLATED;
 5:   loop
 6:     present [x1 and x2 or x1 and x3 or x2 and x3] then
 7:       emit VIOLATED;
 8:     end present;
 9:     pause;
10:   end loop;
11: end module
```

Рис. 8. Пример модуля esterel, выражающего пользовательское свойство

## 5. Заключение

Разработанная методика позволяет строить автоматные программы, подчиняющиеся четкой формальной модели. Корректность модели проверяется на нескольких уровнях. Есть возможность определения и проверки пользовательских свойств.

## Список литературы

1. Кубасов, С.В. Синхронная модель автоматной программы / С.В. Кубасов, В.А. Соколов // Моделирование и анализ информационных систем. — 2007. — Т.14, №1. — С. 11–18.

2. 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.

3. 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.

4. Bouali, A. Xeve: an Esterel Verification Environment : Version v1_3 : Rapport technique №0214 / Amar Bouali ; Inria, Institut National de Recherche en Informatique et en Automatique. — France, 1997. — 23 pages. — ISSN 0249–0803.

## Verification of Synchronous-automaton Programs

Kubasov S.V.

This article presents a synchronous model of the automaton program. A technique of verification of synchronous-automaton programs has been developed. Some properties of the model are checked automatically. There is an ability of verifying user-defined properties. This technique helps to discover errors often made during the design process.