**УДК 004.4’242**

# ВЕРИФИКАЦИЯ АВТОМАТНЫХ МОДЕЛЕЙ МЕТОДОМ РЕДУЦИРОВАННОГО ГРАФА ПЕРЕХОДОВ

**С. Э. Вельдер, А. А. Шалыто**
(Санкт-Петербургский государственный университет информационных технологий, механики и оптики)

**Ключевые слова**: верификация моделей, темпоральные логики, Model checking, автоматное программирование.

В статье рассматриваются способы преобразования программ, разработанных на основе автоматного подхода, в модели Крипке, предназначенные для проверки свойств, относящихся к поведению системы. Эти свойства задаются формулами темпоральной логики. Предложен эффективный метод такого преобразования и формулировки указанных свойств, который позволяет строить небольшие модели Крипке и достаточно быстро проводить их проверку.

## Введение

В данной работе предлагается метод верификации моделей (*Model checking*) применительно к автоматным программам. В этом контексте исследуются особенности структуры (модели) Крипке. Причина, по которой этим вопросам уделяется внимание, заключается в преимуществах автоматных программ перед остальными в смысле верификации, так как алгоритмы верификации автоматных программ могут оперировать с моделью Крипке, заданной в явном виде, в то время как для традиционных программ процесс построения модели требует дополнительного исследования. Можно выделить следующие ключевые этапы верификации автоматных программ: преобразование автоматной модели в модель Крипке и построение требований к модели (свойства модели); собственно процесс верификации (отработки алгоритмов на полученных моделях); построение подтверждающих трасс (контрпримеров) в модели Крипке, а также представление контрпримеров, записанных в терминах модели Крипке, в виде путей в исходной автоматной модели. Алгоритмы верификации и построения контрпримеров в модели Крипке изложены в работах [1–5]. Генерация модели Крипке основано на построении редуцированного графа переходов, структура которого рассматривается в данной работе.

## Общие положения

В этом разделе описывается метод генерации множества атомарных предложений автоматной программы и преобразования автомата с булевыми входными переменными в модель Крипке. Метод состоит в редукции полного графа переходов с внесением тесных отрицаний внутрь атомарной формулы. Приводится пример записи требований к программе. Требования выражаются в темпоральной логике *CTL*.

Моделью Крипке (также *CTL*-моделью) для данного множества атомарных предложений *AP* будем считать тройку $\mathcal{M} = (S, R, Label)$, где

- $S$ — непустое множество состояний;
- $\rightarrow \subseteq S \times S$ — тотальное отношение на $S$, называемое отношением переходов. Для него должна выполняться формула $\forall s \in S \; \exists s' \in S \mid (s, s') \in \rightarrow$ (свойство тотальности). Оно сопоставляет каждому элементу (состоянию) $s \in S$ *непустое* множество его состояний-последователей.
- $Label \subseteq S \times AP$ — помечающее отношение, сопоставляющее каждому состоянию $s \in S$ множество атомарных предложений, истинных в $s$.

Иногда можно потребовать, чтобы в модели Крипке было задано непустое множество начальных состояний $S_0 \subseteq S$ или даже одно начальное состояние $s \in S$.

### Построение модели Крипке по автоматной модели

Рассмотрим программу, модель которой задается системой автоматов, взаимодействующих по вложенности. Необходимо преобразовать эту модель в единую модель Крипке, целиком описывающую поведение данной системы.

Прежде всего, для множества автоматов выполняется топологическая сортировка по отношению вложенности. Данное отношение не должно содержать циклов, иначе полученная модель имела бы бесконечный размер. Модель Крипке строится индуктивно для каждого автомата системы, причем автоматы обрабатываются в порядке, который был сформирован топологической сортировкой. Такой порядок означает, что перед обработкой внешних автоматов вложенные в них уже будут обработаны (для них будет подготовлена модель Крипке).

Итак, опишем алгоритм построения модели Крипке для одиночного автомата в предположении, что модели Крипке для всех его вложенных автоматов уже построены. Будем пользоваться терминологией, введенной в разд. 2.2.1.1 отчета по III этапу темы [6].

В методе редукции графа переходов множество *AP* задается следующим образом: $\{Y_1, Y_2, \ldots\} \cup \{e_1, e_2, \ldots\} \cup \{x_1, x_2, \ldots\} \cup \{!x_1, !x_2, \ldots\} \cup \{z_1, z_2, \ldots\} \cup$ {InState, InEvent, InAction} $\cup$ *Names*.

Здесь $\{Y_1, Y_2, \ldots\}$ – множество наименований всех состояний автоматов, $\{e_1, e_2, \ldots\}$ – событий, $\{x_1, x_2, \ldots\}$ – входных воздействий, а $\{z_1, z_2, \ldots\}$ – выходных воздействий. *Names* – множество наименований самих автоматов, а `InState`, `InEvent` и `InAction` – управляющие автомарные предложения, предназначенные для того, чтобы было удобно различать позиции, построенные из состояний, событий и выходных воздействий.

Модель Крипке будем строить по частям: вначале построим те ее части, которые соответствуют состояниям автомата (при этом потребуется обработать выходные воздействия и автоматы, вложенные в эти состояния), а потом добавим туда информацию о переходах. На первом шаге положим множество *S* равным множеству состояний исходного автомата, и для каждого состояния `s` добавим в отношение *Label* две пометки: (`s`, `s`) и (`s`, `InState`).

После этого для каждого состояния `s` необходимо выполнить следующую операцию. Пусть `s` содержит выходные воздействия $z_{s[1]}, \ldots, z_{s[u]}$, которые выполняются при входе в `s`. Добавим в модель $u$ состояний $\{r_1, \ldots, r_u\}$ и $u$ переходов $r_1 \to r_2$, …, $r_{u-1} \to r_u$, $r_u \to s$, в отношение *Label* добавим пометки $(r_k, z_{s[k]})$, $(r_k, \mathit{InAction})$ для всех $k$ от 1 до $u$. При добавлении ребер в модель на следующих этапах, каждую дугу, направленную в `s`, будем перенаправлять в $r_1$.

Пример такого преобразования приведен на рис. 1.

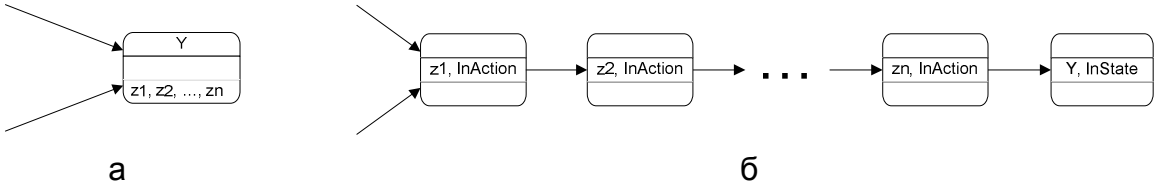



Рис. 1. Состояние с выходными воздействиями до преобразования (а) и после его (б)

Назовем эту операцию разделением выходных переменных и состояний. Теперь следует отделить состояния от вложенных в них автоматов.

Пусть в состояние Y внешнего автомата вложены автоматы $A_{Y,1}$, $A_{Y,2}$, …, $A_{Y,v}$ (для них модели Крипке уже построены по индуктивному предположению). В модели Крипке для внешнего автомата (которая еще строится) уже присутствует позиция, соответствующая состоянию Y. Для каждого из автоматов $A_{Y,1}$, $A_{Y,2}$, …, $A_{Y,v}$ добавим его модель Крипке к строящейся модели Крипке внешнего автомата. При добавлении требуется скопировать во внешнюю модель Крипке все состояния, переходы и пометки внутренней модели Крипке. Для каждого состояния внутренней модели Крипке создадим уникальное, соответствующее только ему, состояние внешней модели Крипке. Во внешнее отношение переходов $\rightarrow$ добавим переходы между теми состояниями, которые соответствовали всем переходам между состояниями внутренней модели. Аналогично поступим и с помечающим отношением *Label*.

После того, как внутренние модели Крипке будут скопированы во внешнюю, добавим в нее также $v$ переходов: для каждого $i$ от 1 до $v-1$ добавим в отношение $\rightarrow$ переход из терминальной позиции модели Крипке для автомата $A_{Y,i}$ в стартовую позицию модели Крипке для автомата $A_{Y,i+1}$, и один переход из терминальной позиции модели Крипке для автомата $A_{Y,v}$ в позицию, соответствующую состоянию Y. Если до этого в позицию Y вели какие-либо дуги, то все они перенаправляются в стартовую позицию модели Крипке для автомата $A_{Y,1}$.

Пример такого преобразования можно приведем на рис. 2.

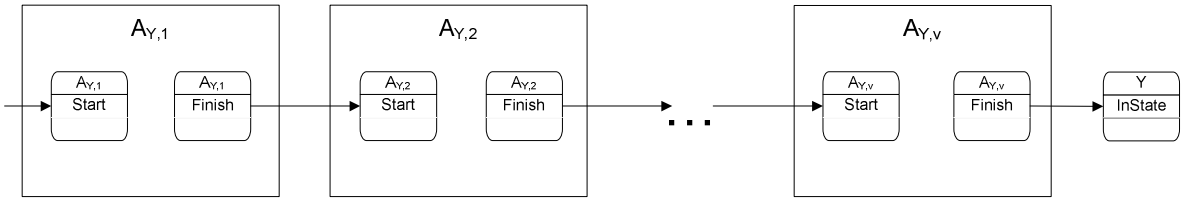



Рис. 2. Обработка автоматов, вложенных в состояние

Обратим внимание, что в различные состояния автомата *A* могут быть вложены различные «копии» одного и того же автомата *B*. В этом случае для каждой копии автомата создается своя модель Крипке (все эти модели изоморфны друг другу), и перенаправление дуг выполняется для нее. Размер полученной модели Крипке (число ее состояний) будет ограничен сверху произведением размеров моделей Крипке для каждого автомата в отдельности (без учета вложенных).

На данном этапе заканчивается обработка состояний автомата. Поэтому перейдем к описанию того, как обрабатываются переходы.

Рассмотрим множество следующих символов: $\{x_1, !x_1; x_2, !x_2; x_3, !x_3; \ldots\}$. Можно сказать, что это множество всех литералов, составленных из входных переменных. Следует различать смысл знаков $\neg$ и !. Первый из них означает выполнение операции логического отрицания, а второй интерпретируется просто как символ (часть строки $!x_i$).

Тогда для каждого ребра r исходного автомата, ведущего из состояния p в состояние q с пометкой $e_i \,\&\, h_{j[1]} \,\&\, h_{j[2]} \,\&\, h_{j[3]} \,\&\, \ldots\, h_{j[m]} \,/\, z_{i[1]}, \ldots, z_{i[n]}$, где либо $h_{j[j^*]} = x_{j[j^*]}$, либо $h_{j[j^*]} = !x_{j[j^*]}$ (это значит, что $h_{j[j^*]}$ есть либо входная переменная, либо ее отрицание), добавим в модель $n+1$ состояние $\{r_e, r_1, \ldots, r_n\}$, $n+2$ перехода: $p \rightarrow r_e$, $r_e \rightarrow r_1$, $r_1 \rightarrow r_2$, …, $r_{n-1} \rightarrow r_n$, $r_n \rightarrow q$, а в отношение *Label* добавим пометки $(r_e, e_i)$, $(r_e, \mathtt{InEvent})$, $(r_k, z_{i[k]})$, $(r_k, \mathtt{InAction})$ для всех $k$ от 1 до $n$, а также пометки $(r_e, h_{j[1]})$, $(r_e, h_{j[2]})$, …, $(r_e, h_{j[m]})$.

Пример такого преобразования отражен на рис. 3.

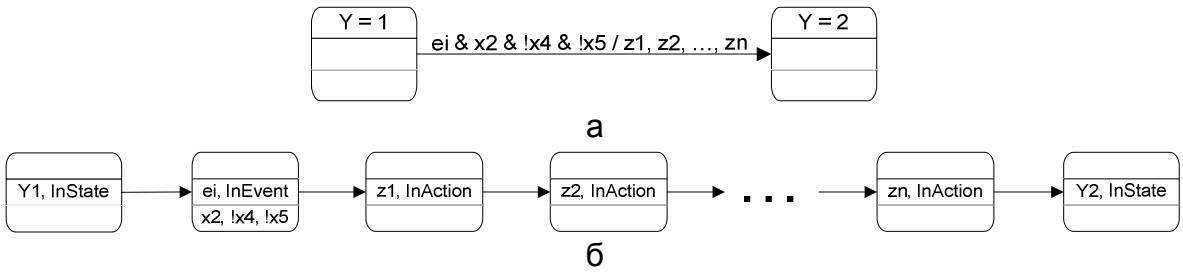



Рис. 3. Переход между состояниями
до преобразования по методу редукции (а) и после него (б)

Из этого рисунка видно, что для тех состояний, которые были построены из событий, в множество атомарных предложений были добавлены входные переменные в том виде, в котором они записаны на переходах автомата (вместе с отрицаниями, если имеются).

Далее для каждой внешней позиции полученной модели (внешней будем называть любую позицию, за исключением тех, которые были скопированы при обработке вложенных автоматов) добавим в отношение *Label* пометку с атомарным предложением (элементом множества *Names*), соответствующим имени обрабатываемого автомата (того, для которого строится модель и которому это состояние принадлежит). Эти действия предназначены для того, чтобы в формулах темпоральной логики можно было различать какой именно из автоматов системы выполняется на данном участке пути.

Если на переходах каких-либо автоматов указаны условия на состояния других автоматов, то эти переходы следует добавлять в модель Крипке только если данные условия соблюдаются. Так как автоматы вложены друг в друга, то следует проверять условия на переходах внутренних автоматов, описывающие поведение внешних автоматов. Пример построения модели Крипке по системе взаимодействующих автоматов приведен на рис. 4.

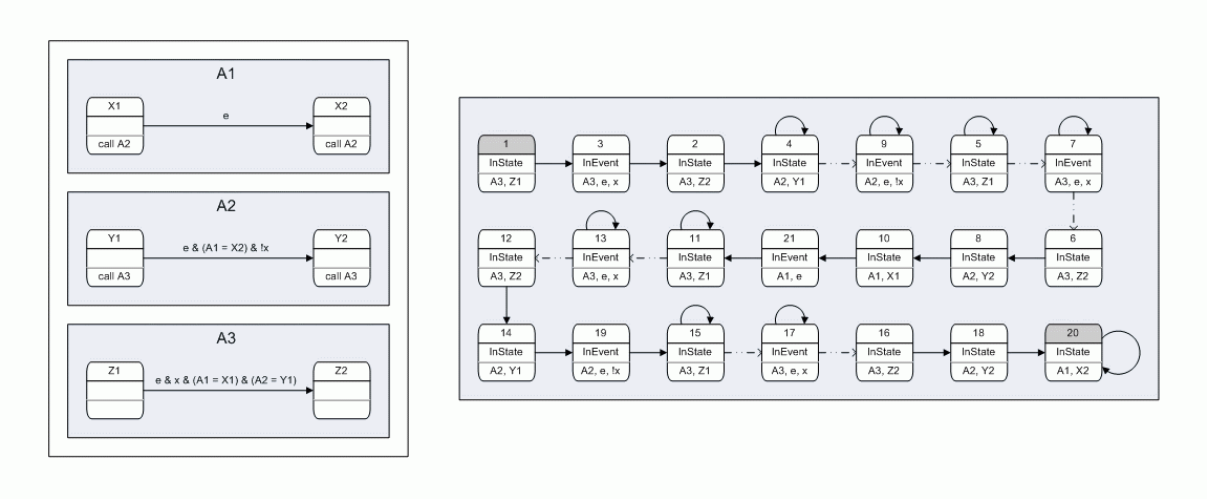



Рис. 4. Система автоматов, взаимодействующих по вложенности, с условиями на состояниях внешних автоматов и модель Крипке для этой системы

Рассмотрим пример работы этого алгоритма для системы, эмулирующей работу банкомата [6, 7]. Система состоит из двух автоматов `AClient` и `AServer`, причем `AServer` вложен в `AClient`. Схема связей для соответствующей автоматной модели изображена на рис. 5, а графы переходов автоматов `AServer` и `AClient` – соответственно на рис. 6 и 7.

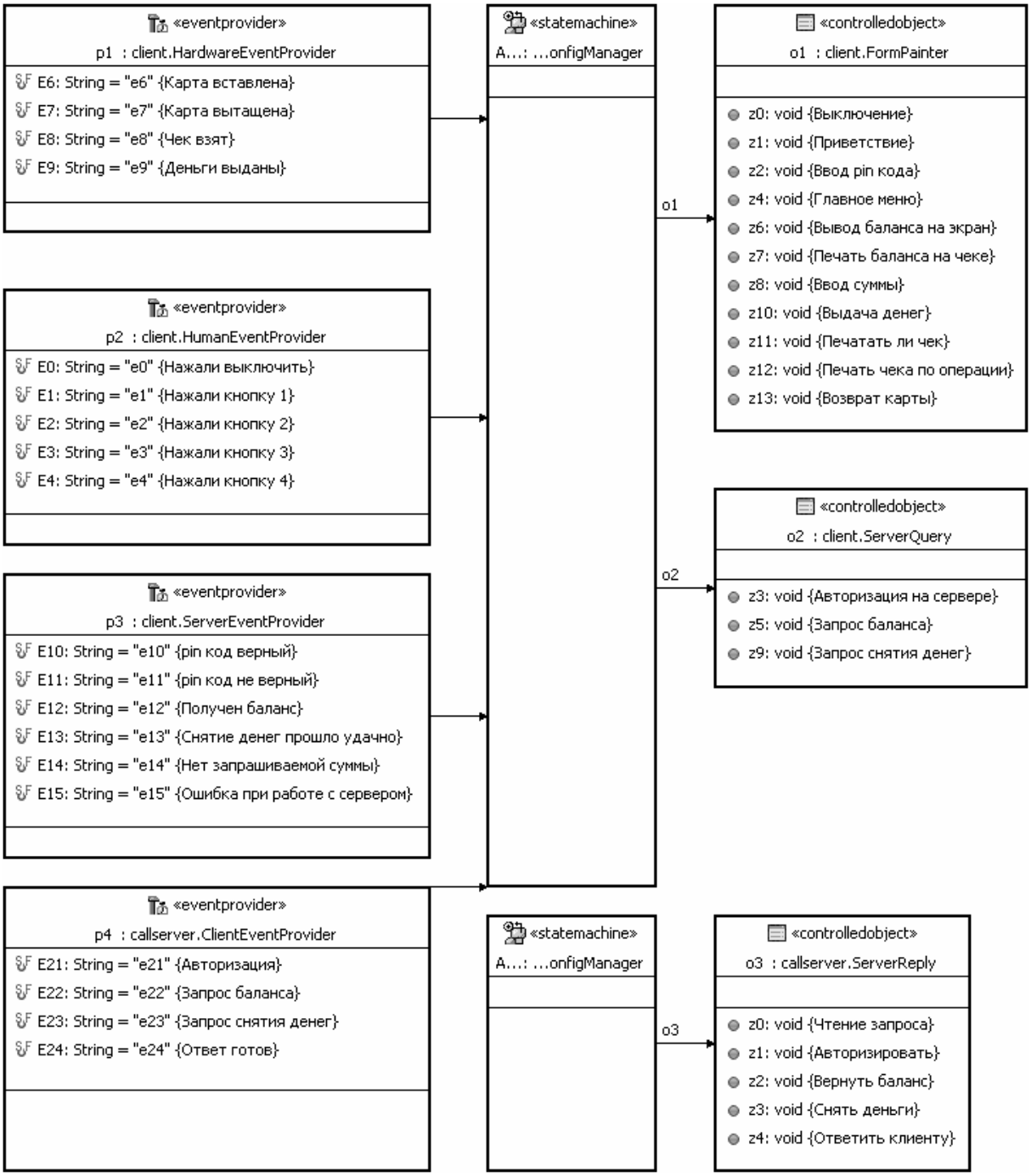



Рис. 5. Схема связей автоматной модели

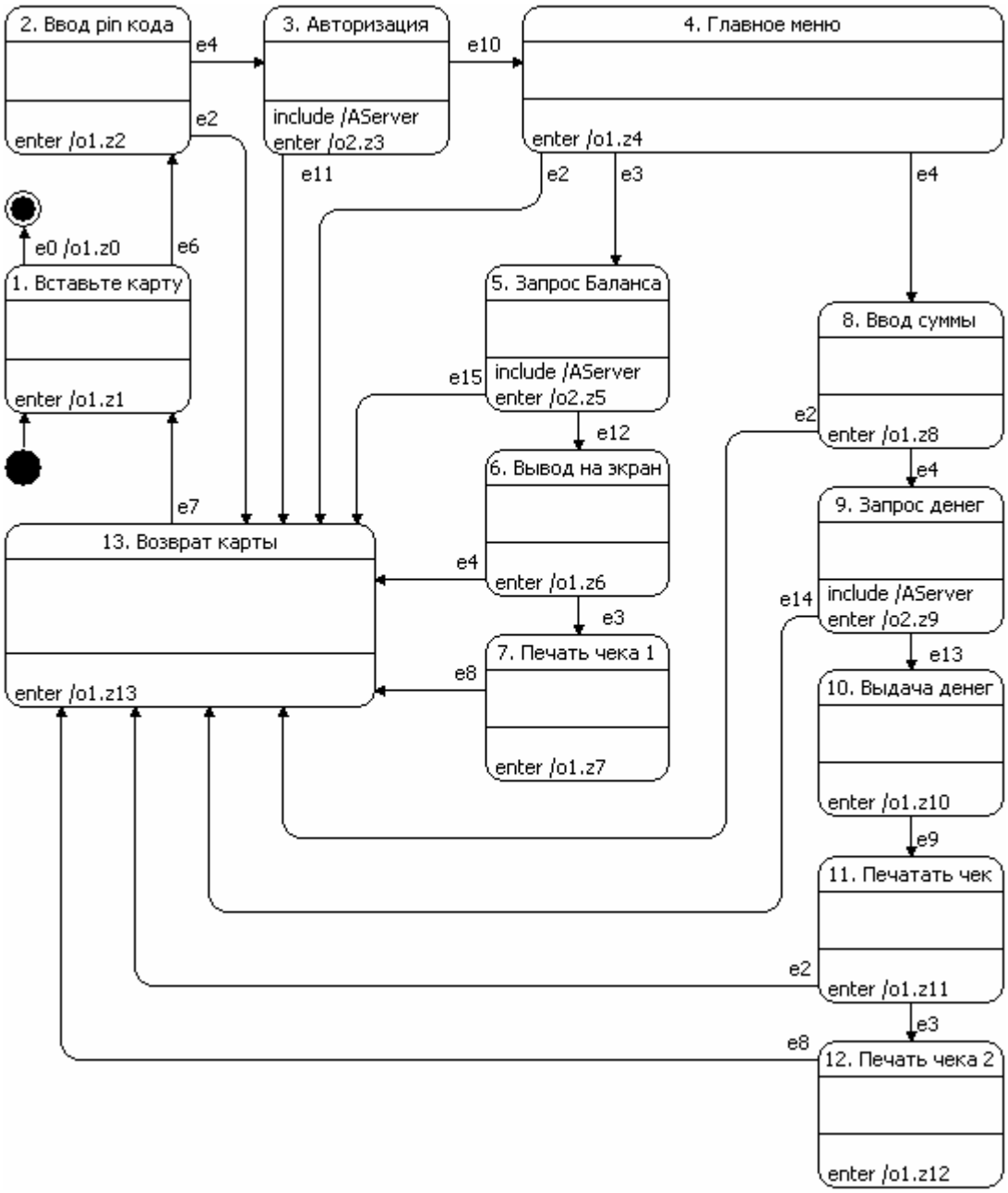



Рис. 6. Граф переходов автомата `AClient`

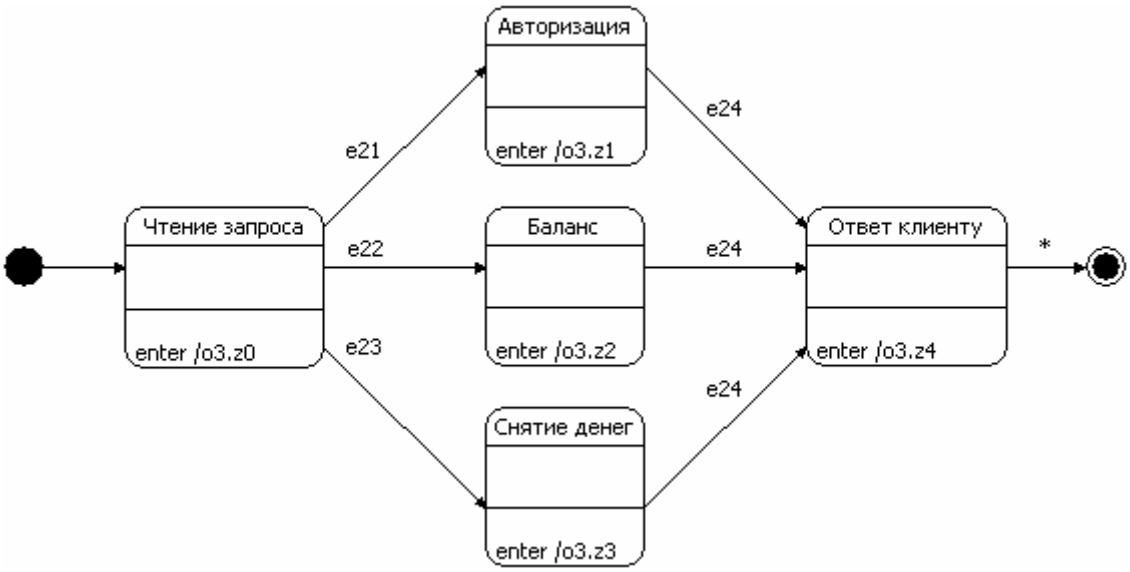



Рис. 7. Граф переходов автомата `AServer`

Для этой системы модель Крипке имеет большой размер. Поэтому она отображена на рис. 8 в упрощенном виде: позиции модели Крипке не выделяются каждая в свой блок, но видны особенности обработки вложенных автоматов.

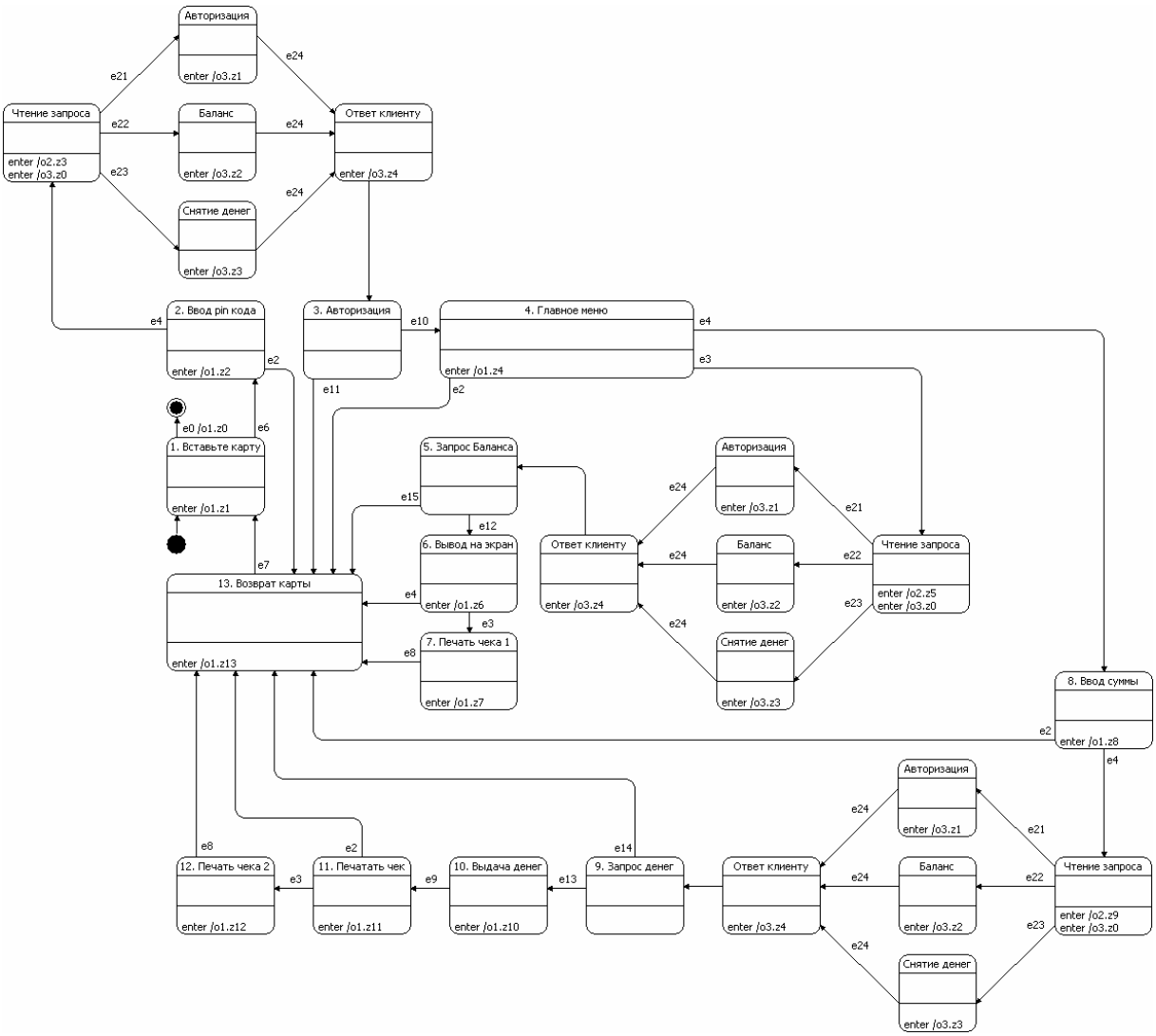



Рис. 8. Редукция графа переходов для модели банкомата

## Проверка *CTL*-формулы

Разберем построение и интерпретацию *CTL*-формул для редуцированных моделей.

*CTL*-семантика в данном методе будет немного отличаться от общепринятой: перед тем, как выполнять верификацию *CTL*-формулы, ее следует привести к определенному («каноническому») виду. Вначале в ней необходимо удалить все парные отрицания (путем замены подформул вида $\neg\neg f$ на $f$). После этого все входные воздействия, которые присутствуют в формуле без отрицания, требуется предварить двумя отрицаниями: одно из них синтаксическое, другое логическое (это значит, что необходимо заменить литералы вида $x_i$ на формулы $\neg!x_i$). Только после этих модификаций результирующую формулу можно верифицировать методами, предназначенными для языка *CTL*. Причина такого обращения с литералами заключается в следующем: требуется обеспечить, чтобы любая ссылка на несущественную переменную, которая упомянута в *CTL*-формуле, давала истинный результат (несущественными переменными на данном переходе называются те входные переменные исходного автомата, значение которых не проверяется на этом переходе).

Рассмотрим пример для модели банкомата. Проверим *CTL*-формулу: $e14 \rightarrow \neg \mathbf{E}[\neg o3.z0 \, \mathbf{U} \, y10]$. Она означает: если произошло событие `e14`, то невозможно попасть в состояние `10`, минуя позицию `o3.z0`. Эта формула верна во всех позициях модели. Все пути из позиции `e14` в состояние `10` дважды «проходят» через автомат `AServer`: в первый раз – попав в состояние `3`, а второй – попав в состояние `9` автомата `AClient`. Пример такого пути (не проходящего дважды через одно состояние одного и того же экземпляра автомата) выделен на рис. 9. Позиция `e14`, являющаяся стартовой для данного пути, выделена на рисунке жирным шрифтом.

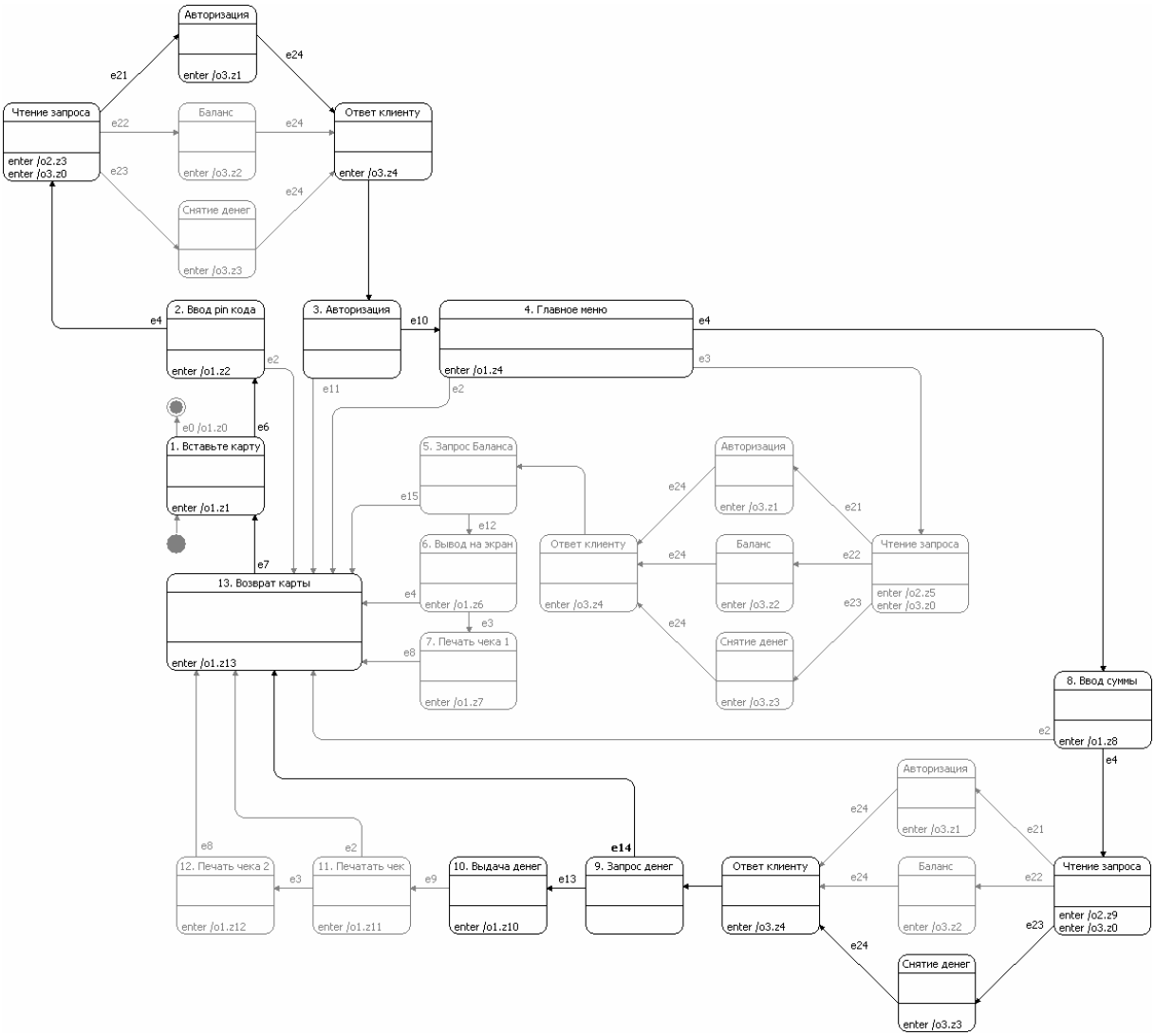



Рис. 9. Пример пути в модели Крипке для банкомата

Таким образом, в методе редукции графа переходов была видоизменена семантика *CTL*. Рассмотренная схема преобразовывала исходную формулу, построенную для новой семантики *CTL*, в новую формулу, для которой применима общепринятая семантика языка *CTL*. Использование такой схемы подходит для многих формул.

Путь в модели Крипке для банкомата, отображенный на рис. 9, можно показать и в терминах исходной системы автоматов (рис. 10, 11).

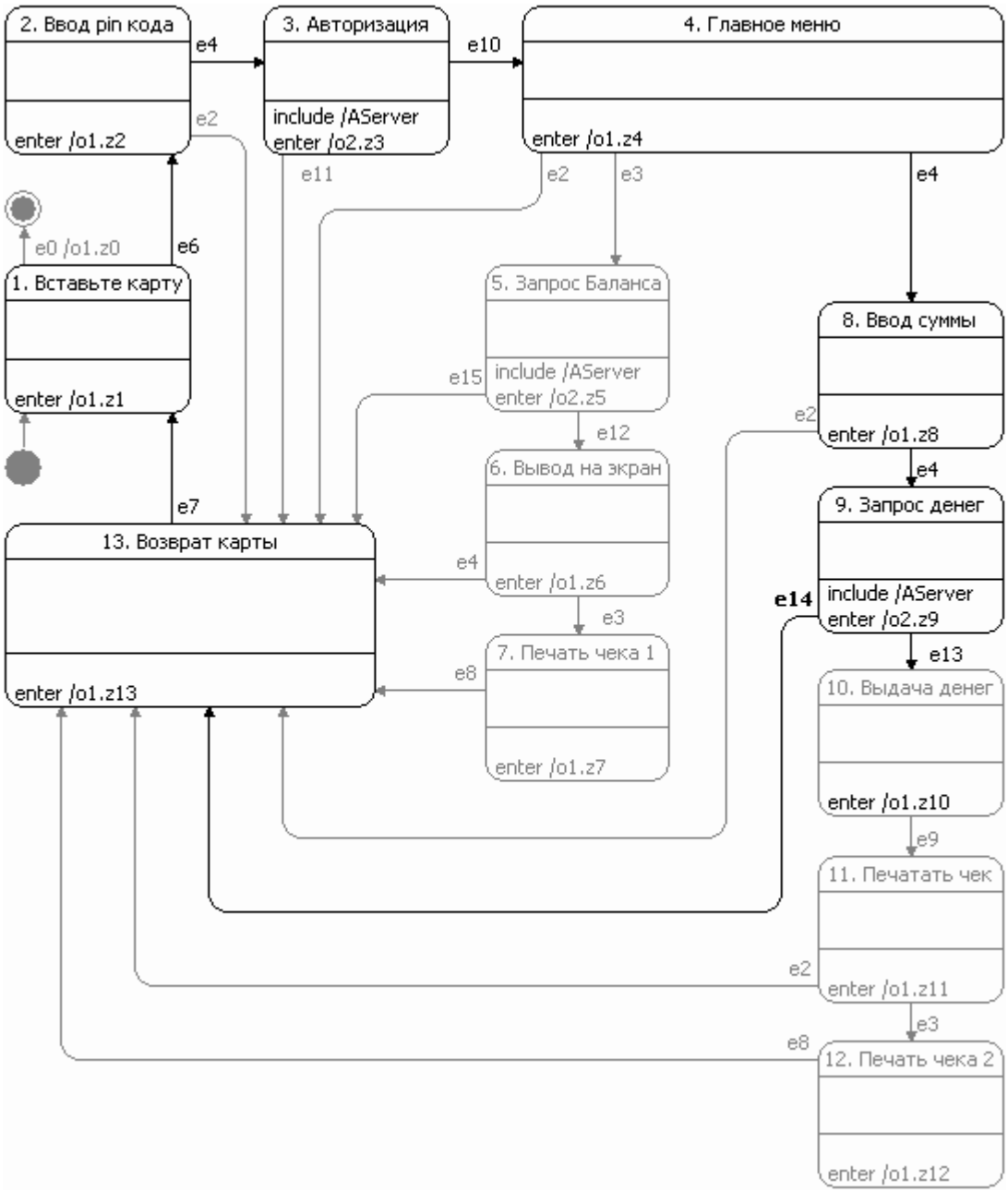
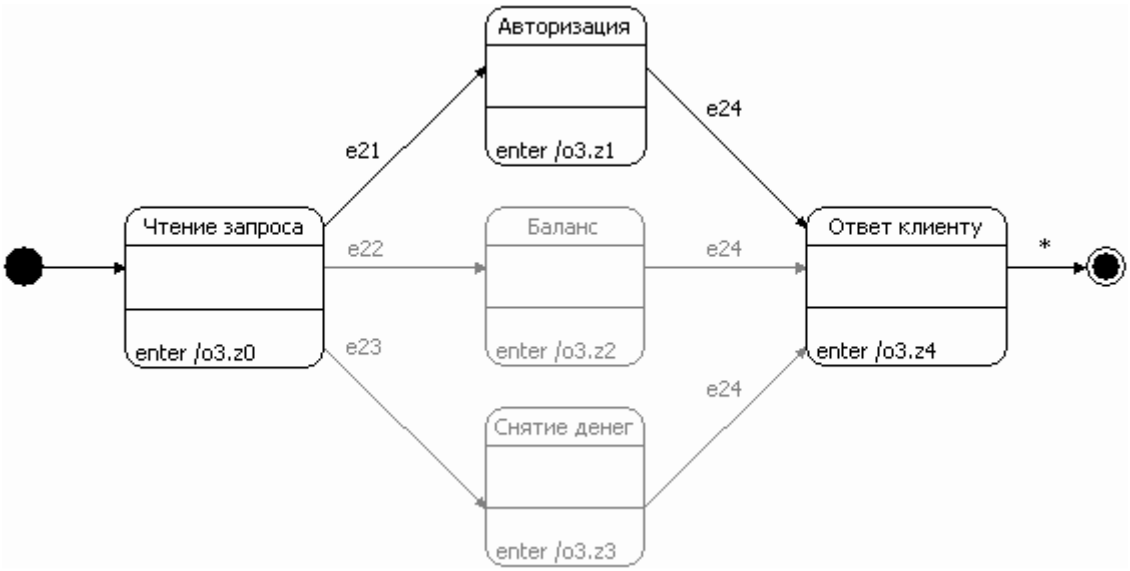
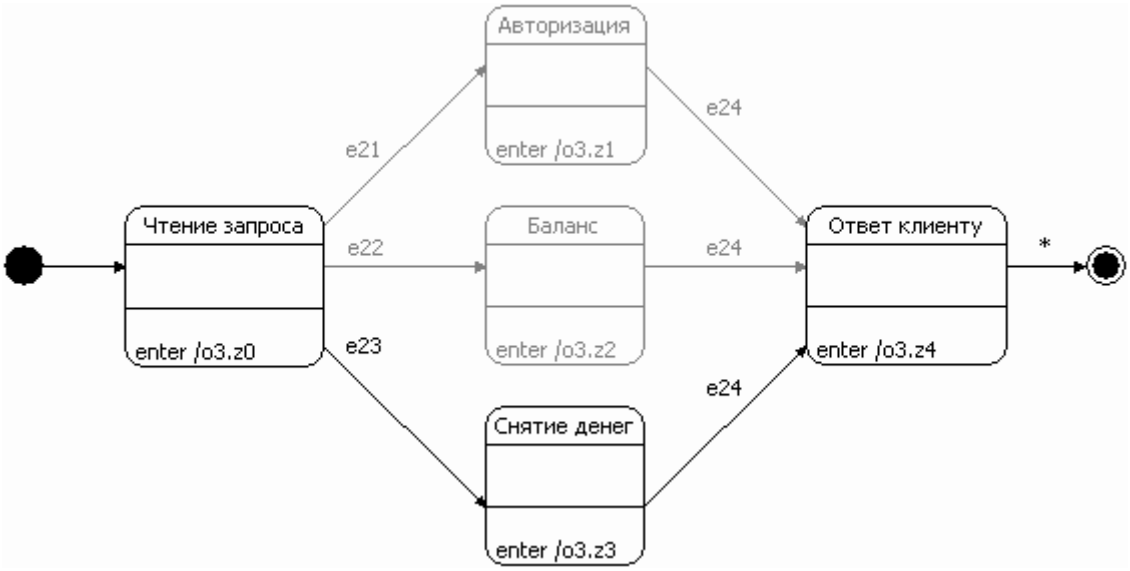



Рис. 10. Участок пути в автомате `AClient`




а




б

Рис. 11. Участок пути в автомате `AServer` при первом прохождении (а) и при втором прохождении(б)

**Преобразование сценария для модели Крипке в сценарий для автомата**

После моделирования и верификации требуется обработать результаты проверки модели. Программы, написанные с помощью традиционных методов, имеют достаточно сложные модели, и проводить анализ путей прямо на модели Крипке неэффективно.

При интерактивном моделировании совместно с исполнением и визуализацией автомата [8, 9] желательно автоматизировать процесс представления путей в модели Крипке путями в автомате.

После того, как отработала программа-верификатор, необходимо определить выполнимость формул, которые формируют *спецификацию*, на заданных участках автомата. Среди этих участков могут быть состояния, события, выходные воздействия.

Сценарий для любой подформулы спецификации представляет собой путь в модели Крипке, иллюстрирующий справедливость или ошибочность данной

подформулы. Задача состоит в том, чтобы сценарий, представленный программой для модели Крипке, был представлен в исходном автомате.

Для описанного в настоящей работе метода операция переноса путей из модели Крипке в автомат выполняется однозначно. Действительно, состояния модели, содержащие атомарное предложение Y =… или вспомогательное атомарное предложение *InState*, однозначно преобразуются в соответствующие им состояния автомата. Путь между любыми двумя соседними состояниями всегда представляет собой «змейку» из события и выходных воздействий. Любая из этих промежуточных позиций однозначно определяет то главное состояние автомата, из которого эта «змейка» исходит. Из атомарных предложений, которыми помечены состояния «змейки», однозначно восстанавливаются события. Значения существенных входных переменных (тех, которые записаны на переходе) и список несущественных переменных определяется оттуда же (в методе редукции). Последовательным проходом по полученному пути восстанавливается информация о выполнимости литералов, соответствующих выходным воздействиям, очередности этих литералов и о том, как попасть в данное состояние.

## Заключение

В работе предложен метод редукции графа переходов для представления системы автоматов структурами Крипке. Этот метод реализован программно. Программа позволяет строить трассы, которые подтверждают заданные формулы, начинающиеся с квантора существования пути (или опровергают отрицания этих формул). Рассмотренные примеры показывают, что предложенный метод позволяет убеждаться в корректности модели, находить ошибки в случае некорректных формул и находить ошибки в неверной модели. Ряд тестов проводился для намеренно измененной модели, для того, чтобы продемонстрировать эффективно работы методы.

## Литература


1. Emerson E. A., Clarke E. M. Using branching time temporal logic to synthesize synchronization skeletons / Science of Computer Programming 2. 1982, pp. 241–266.
2. Clarke E. M., Emerson E. A. Synthesis of synchronisation skeletons for branching time logic //Logic of Programs. LNCS 131. 1981, pp. 52–71.
3. Лифшиц Ю. Верификация программ и темпоральные логики. Лекция №3 курса «Современные задачи теоретической информатики». – СПбГУ ИТМО, 2005. – Режим доступа: http://logic.pdmi.ras.ru/~yura/modern/03modernnote.pdf
4. Лифшиц Ю. Символьная верификация программ. Лекция № 4 курса «Современные задачи теоретической информатики». – СПбГУ ИТМО, 2005. – Режим доступа: http://logic.pdmi.ras.ru/~yura/modern/04modernnote.pdf
5. Вельдер С. Э., Шалыто А. А. О верификации автоматных программ на основе метода Model Checking // Информационно-управляющие системы. – 2007. – № 3. – С. 27–38.
6. Разработка технологии верификации управляющих программ со сложным поведением, построенным на основе автоматного подхода. Отчеты по НИР, выполняемая по государственному контракту № 02.514.11.4048 от 18.05.2007.
Режим доступа: http://is.ifmo.ru/verification/
7. Козлов В. А., Комалева О. А. Моделирование работы банкомата. – СПбГУ ИТМО, 2006. – Режим доступа: http://is.ifmo.ru/unimod-projects/bankomat/
8. Сайт проекта UniMod. – Режим доступа: http://unimod.sf.net
9. Сайт eVelopers Corporation. – Режим доступа: http://www.evelopers.com

# MODEL CHECKING AUTOMATA-BASED PROGRAMS USING REDUCED TRANSITION GRAPH CONSTRUCTION

**Sergey Velder (velder@rain.ifmo.ru), Anatoly Shalyto (shalyto@mail.ifmo.ru)**

**Keywords**: model checking, temporal logic, automata-based programming

This article is concentrated on techniques of converting automata-based program models to Kripke models designed for checking properties related to system behavior. The definition of these properties if considered by means of temporal logic formulas. We propose an efficient technique of such converting and property stating that allows construction of small Kripke models and sufficiently fast checking such models.