Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. Вып. 53. Автоматное программирование. 2008, с. 123–137.

УДК 004.4’242

МЕТОДЫ ВЕРИФИКАЦИИ МОДЕЛЕЙ АВТОМАТНЫХ ПРОГРАММ

С. Э. Вельдер, А. А. Шалыто

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. Предложены несколько методов такого преобразования и способов формулировки свойств.

Введение

В данной статье рассматриваются вопросы верификации на моделях (*Model checking*) применительно к автоматным программам. В их контексте исследуется специфика структуры (модели) Крипке для автоматных программ. Причина, по которой этим вопросам уделяется внимание, заключается в преимуществах автоматных программ перед остальными в смысле верификации. Действительно, алгоритмы верификации автоматных программ могут оперировать с моделью Крипке, заданной в явном виде. В работе выделены ключевые этапы верификации автоматных программ: преобразованиие автоматной модели в модель Крипке и построение требований к 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. Предполагается, что собственно алгоритм верификации и построения контрпримеров в модели Крипке заранее выбран (такие алгоритмы изложены, например, в работах [1–5]).

Построение модели Крипке по автоматной модели

Рассмотрим несколько методов для генерации множества атомарных предложений модели Крипке, соответствующей автоматной программе, и преобразования автомата с булевыми входными переменными в эту модель. Для каждого метода приводится пример записи требований к программе. Требования выражаются на языке темпоральной логики *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 (*позиций*);
- $\rightarrow \subseteq S \times S$ — тотальное отношение на S, называемое отношением переходов. Свойство тотальности можно записать логической формулой $\forall s \in S \; \exists s' \in S \mid (s, s') \in \rightarrow$. Это отношение сопоставляет каждому состоянию *непустое* множество его состояний-последователей.
- $Label \subseteq S \times AP$ — помечающее отношение, которое сопоставляет 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 $s_0 \in S$.

Кратко опишем особенности методов преобразования автомата в модель Крипке.

Метод атомарных переходов требует «стереть» у автомата все события и переменные и оставить только состояния. Этот метод позволяет проверять не очень много свойств, но для простых свойств он достаточно эффективен. Он не требует преобразования контрпримера из модели Крипке в автомат, так как на переходах нет промежуточных состояний (в отличие от всех остальных методов).

Метод установки состояний на событиях и выходных воздействиях требует стереть у автомата только входные воздействия. Каждый переход разбивается на блоки, соответствующие событиям и выходным воздействиям, которые расположены друг за другом. Данный метод позволяет упоминать события и выходные воздействия в формулах темпоральной логики.

Метод полного графа переходов требует построить для автомата модель, в которой для каждого события будет сформирована полная система переходов. Полученная модель будет иметь большой размер, но в ней можно будет проверять любые темпоральные свойства.

Метод редуцированного графа переходов требует разложить переходы на атомарные блоки из событий и выходных воздействий, а входные воздействия должны быть особым способом в них сохранены. Из подробного описания в соответствующем разделе следует, что *метод установки состояний на событиях и выходных воздействиях* является его упрощением. Метод редуцированного графа переходов строит небольшие модели (линейного размера по отношению к размеру исходного автомата) и позволяет проверять практически все свойства, которые можно сформулировать относительно состояний, событий и воздействий (входных и выходных). Он является лучшим по соотношению эффективность-выразительность.

Для удобства и эффективности в последних трёх методах каждое состояние модели Крипке помечено одним из трёх «управляющих» атомарных предложений: *InState*, *InEvent*, *InAction* – для состояний модели, построенных, соответственно, из *состояний*, *событий*, *выходных воздействий* исходного автомата. Это сделано для того, чтобы при записи формулы в темпоральной логике можно было различать тип исследуемого состояния.

Во всех методах множество стартовых состояний модели Крипке состоит из одного элемента – стартового состояния исходного автомата.

Существуют и альтернативные способы получения модели Крипке, например, методы дублирования состояний [6–9].

Методы конвертации автомата в модель Крипке и формулировки проверяемых свойств демонстрируются на трёх примерах:

1. Автомат `ATrig`, эмулирующий работу *R*-триггера (его граф переходов изображен на рис. 1).
2. Автомат `ARemote`, эмулирующий универсальный инфракрасный пульт для бытовой техники [10]. Его схема связей и граф переходов изображены, соответственно, на рис. 2 и 3. Этот автомат изображён в упрощённой форме. Чтобы строго задать его структуру, необходимо рёбра, на которых записана логическая дизъюнкция разбить на несколько рёбер, каждое из которых дизъюнкции не содержит (эти рёбра будут соответствовать операндам дизъюнкции на исходном ребре).
3. Автомат `AElevator`, управляющий дверьми лифта (рис. 4).

В рамках исследований по государственному контракту [11] разработанные методы демонстрировались на примере системы, эмулирующей работу банкомата [12].

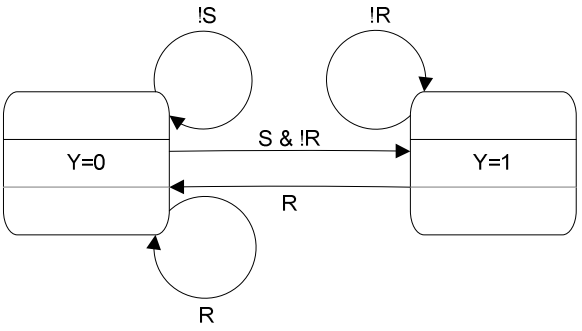


Рис. 1. Граф переходов автомата ATrig

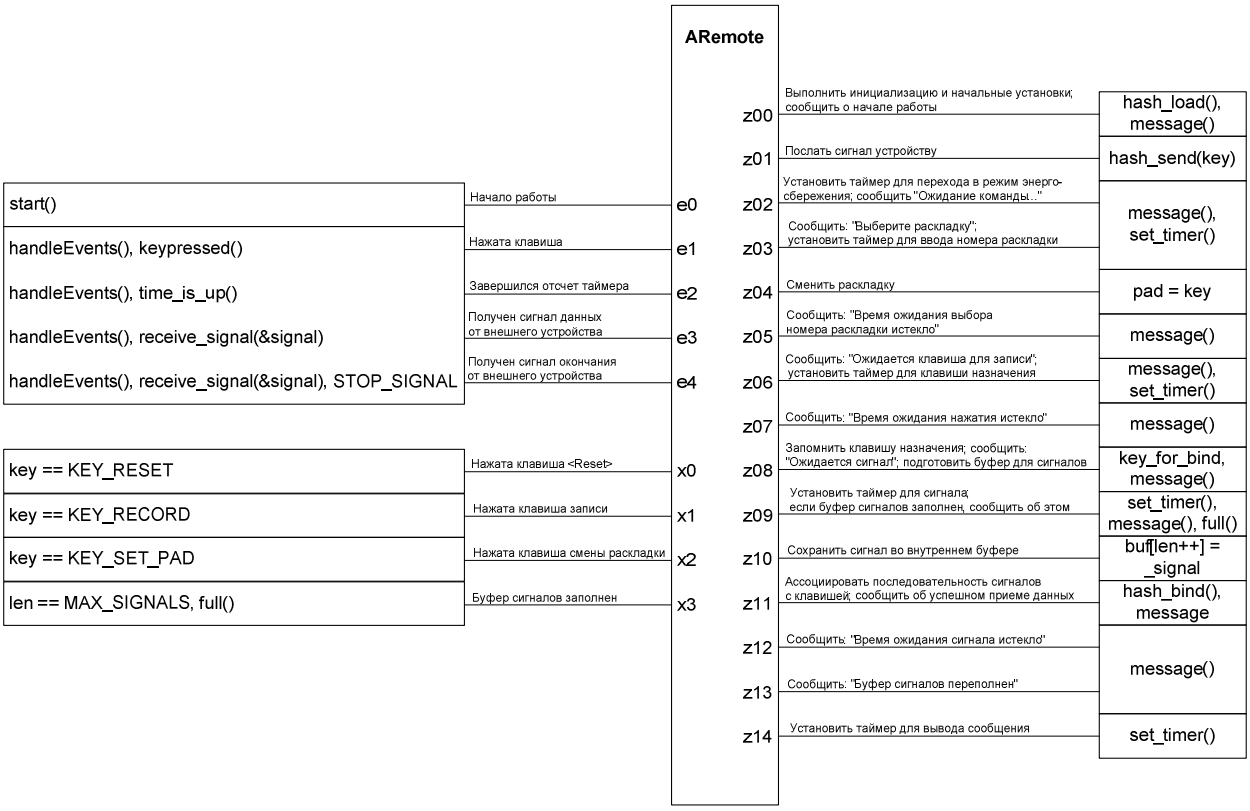


Рис. 2. Схема связей автомата ARemote

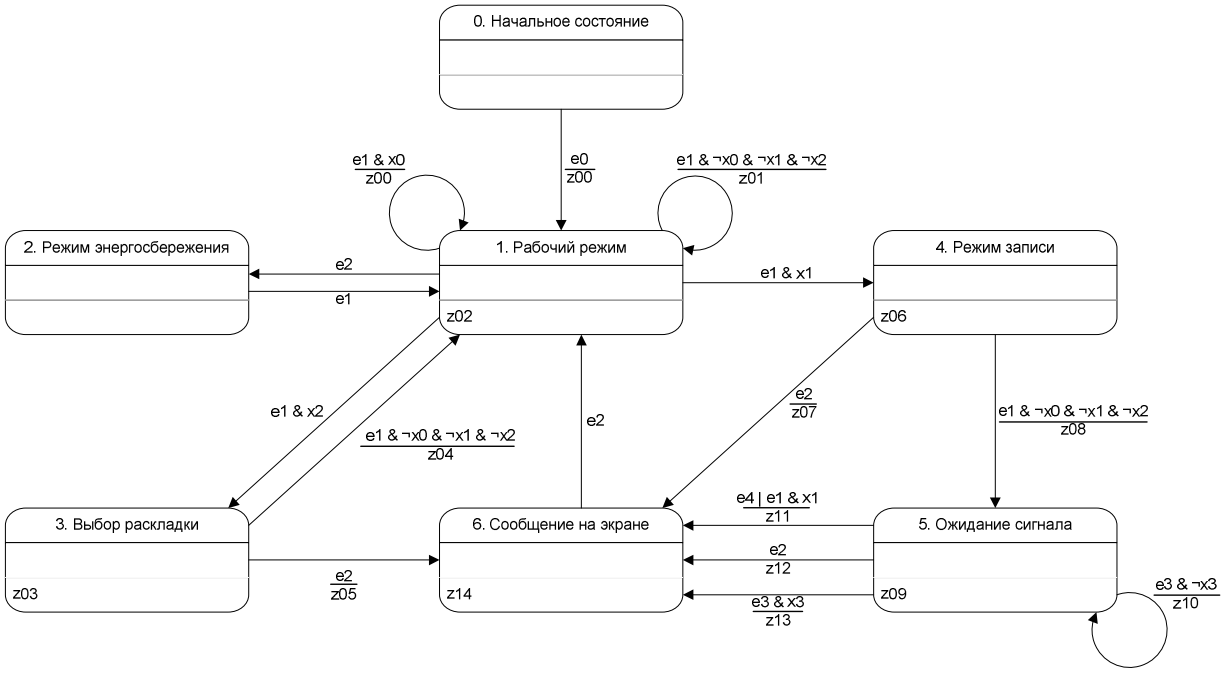


Рис. 3. Граф переходов автомата `ARemote`

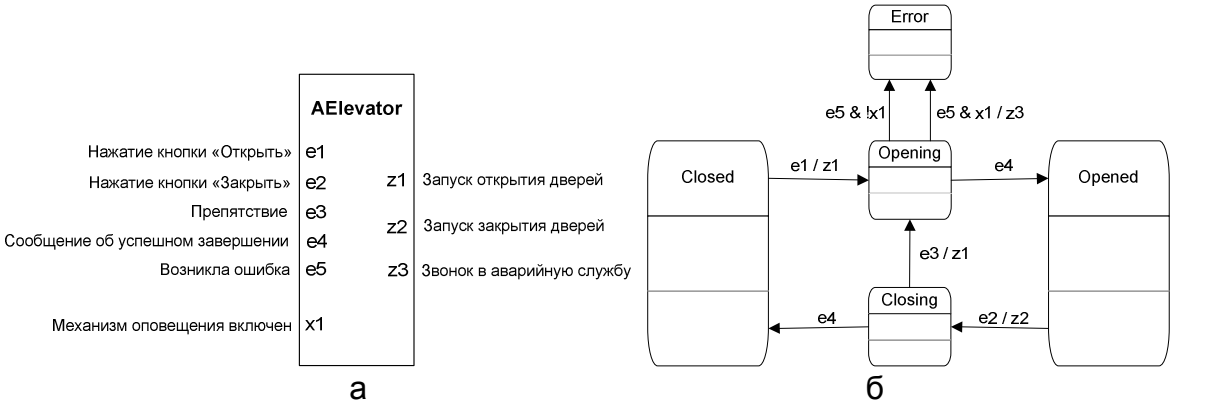


Рис. 4. Схема связей (а) и граф переходов (б) автомата `AElevator`

Во всех четырёх методах вложенные автоматы считаются частями исходного. При построении модели Крипке это означает, что если в состояние *s* автомата *A* вложен автомат *B*, то все рёбра модели Крипке, входящие в позицию, соответствующую состоянию *s* модели *A*, следует перенаправить в стартовое состояние модели *B*, а у всех рёбер, исходящих из состояния *s* модели *A* следует изменить начало на терминальную позицию модели *B*. 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 создаётся своя модель Крипке (все эти модели изоморфны друг другу) и перенаправление рёбер выполняется для неё.

После построения модели Крипке для автоматной модели выполним следующие действия. Добавим во множество атомарных предложений *AP* имена всех автоматов системы (множество этих имён обозначим через *Names*). Далее для каждой позиции полученной модели добавим в отношение *Label* пометку с атомарным предложением (элементом множества *Names*), соответствующим имени автомата, 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.

Метод атомарных переходов

Множества атомарных предложений *AP* и состояний *S* в модели Крипке, построенной по данному методу совпадают с множеством состояний исходного автомата, отношение переходов $\rightarrow$ совпадает с отношением переходов исходного автомата, а помечающее отношение *Label* является тождественным (*Label* = *Id*). Это означает, что каждое состояние помечено единственной переменной, которая заведена специально для него, а все метки на переходах «удаляются» (рис. 5). Общий размер такой модели линеен по отношению к размеру автомата.

а б

Рис. 5. Переход между состояниями до преобразования (а) и после него (б)

Модели Крипке, полученные по этому методу для автоматов `ARemote` и `AElevator`, изображены, соответственно, на рис. 6 и 7.

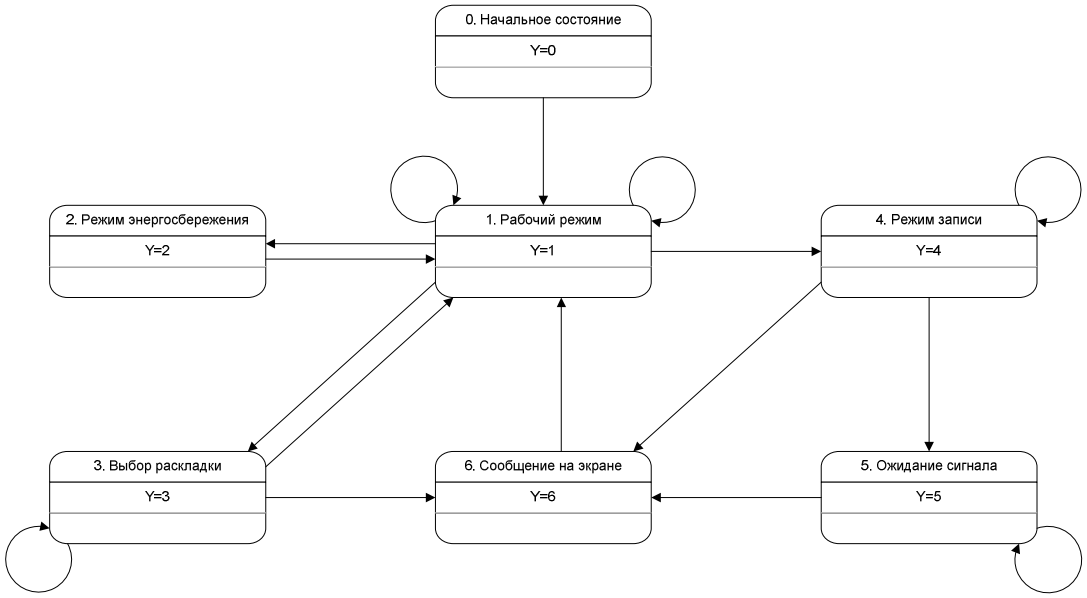


Рис. 6. Сокращенная модель без событий и входных воздействий для автомата `ARemote`

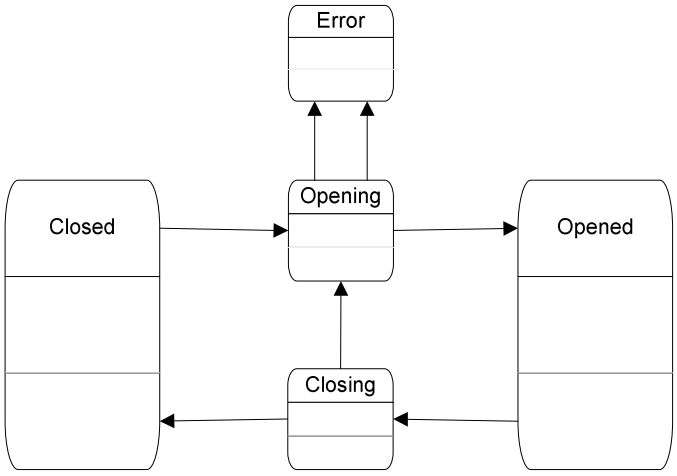


Рис. 7. Сокращенная модель без событий и входных воздействий для автомата `AElevator`

Приведём теперь примеры *CTL*-формул, для которых верификация позволяет проверять выполнимость на модели.

Для автомата `ARemote` можно взять формулу $\neg\mathbf{E}[\neg(\mathtt{Y=6})\ \mathbf{U}\ (\mathtt{Y=1})]$. Она утверждает, что в состояние `1` нельзя попасть, минуя состояние `6` (нельзя попасть в рабочий режим, минуя сообщение на экране). Эта формула выполняется в состояниях `4`, `5` и `6` модели Крипке и исходного автомата и только в них.

Для автомата `AElevator` проверим формулу $\mathbf{AG}(\mathtt{Closing} \rightarrow \mathbf{AX}\ \mathtt{Closed})$, которая означает, что если дверь начала закрываться, то на следующем шаге она обязательно закроется. Эта формула неверна во всех состояниях модели и автомата. Например, последовательность состояний (`Closing`, `Opening`) является опровергающей (контрпримером) для этой формулы.

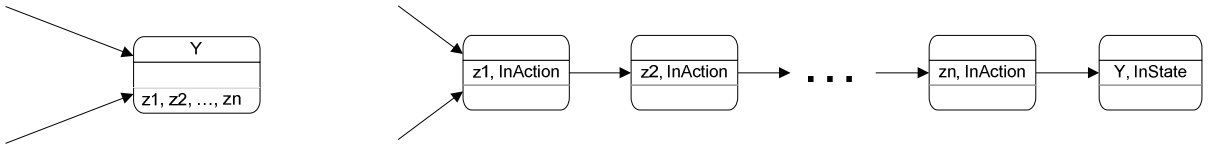
Особенностью данного метода является то, что комбинаторные свойства у модели не отличаются от таковых у исходного автомата. Следовательно, структура Крипке для композиции автоматов не отличается от композиции структур Крипке отдельных автоматов. Сведения о композиции структур Крипке также изложены в работе [13].

Метод установки состояний на событиях и выходных воздействиях

В данном методе множество атомарных предложений *AP* равно $\{\mathtt{Y}_1, \mathtt{Y}_2, \ldots\} \cup \{\mathtt{e}_1, \mathtt{e}_2, \ldots\} \cup \{\mathtt{z}_1, \mathtt{z}_2, \ldots\} \cup \{\mathit{InState}, \mathit{InEvent}, \mathit{InAction}\}$. 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` выполняем следующую операцию. Пусть `s` содержит выходные воздействия $\mathtt{z}_{\mathtt{s}[1]}, \ldots, \mathtt{z}_{\mathtt{s}[n]}$, которые выполняются при входе в `s`. Добавим в модель *n* состояний: $\{\mathtt{r}_1, \ldots, \mathtt{r}_n\}$ и *n* переходов: $\mathtt{r}_1 \rightarrow \mathtt{r}_2, \ldots, \mathtt{r}_{n-1} \rightarrow \mathtt{r}_n, \mathtt{r}_n \rightarrow \mathtt{s}$, в отношение *Label* добавим пометки $(\mathtt{r}_k, \mathtt{z}_{\mathtt{s}[k]})$, $(\mathtt{r}_k, \mathit{InAction})$ для всех *k* от 1 до *n*. Далее, при добавлении рёбер в модель на следующих этапах, каждое ребро, идущее в `s`, будем перенаправлять в $\mathtt{r}_1$.

Пример такого преобразования проиллюстрирован на рис. 8.

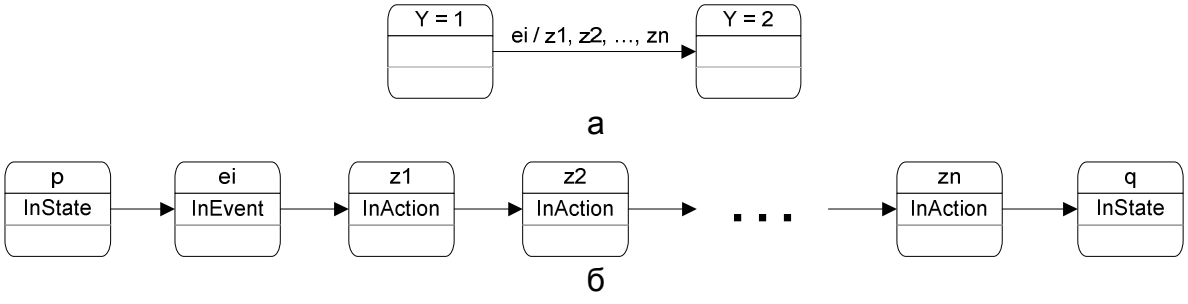


а б

Рис. 8. Состояние с выходными воздействиями до преобразования (а) и после (б)

Эту операцию назовём разделением выходных переменных и состояний. Она будет выполняться также в третьем и четвёртом методе, описанным в следующих разделах.

Далее для каждого ребра `r` исходного автомата с пометкой $(\mathtt{e}_\mathtt{i}\ \&\ \ldots\ /\ \mathtt{z}_{\mathtt{i}[1]}, \ldots, \mathtt{z}_{\mathtt{i}[n]})$, ведущего из состояния `p` в состояние `q`, добавим в модель $n + 1$ состояние $\{\mathtt{r}_\mathtt{e}, \mathtt{r}_1, \ldots, \mathtt{r}_n\}$, $n + 2$ перехода: $\mathtt{p} \rightarrow \mathtt{r}_\mathtt{e}$, $\mathtt{r}_\mathtt{e} \rightarrow \mathtt{r}_1$, $\mathtt{r}_1 \rightarrow \mathtt{r}_2, \ldots, \mathtt{r}_{n-1} \rightarrow \mathtt{r}_n, \mathtt{r}_n \rightarrow \mathtt{q}$, а в отношение *Label* добавим пометки $(\mathtt{r}_\mathtt{e}, \mathtt{e}_\mathtt{i})$, $(\mathtt{r}_\mathtt{e}, \mathit{InEvent})$, $(\mathtt{r}_\mathtt{k}, \mathtt{z}_{\mathtt{i}[k]})$, $(\mathtt{r}_\mathtt{k}, \mathit{InAction})$ для всех *k* от 1 до *n* (рис. 9).



а

б

Рис. 9. Переход между состояниями до преобразования по второму методу (а) и после него (б)

Модели Крипке, построенные рассматриваемым методом из автоматов `ARemote` и `AElevator`, изображены на рис. 10 и 11. Размер модели, полученной таким способом, линеен по поличеству вхождений переменных в условия на переходах.

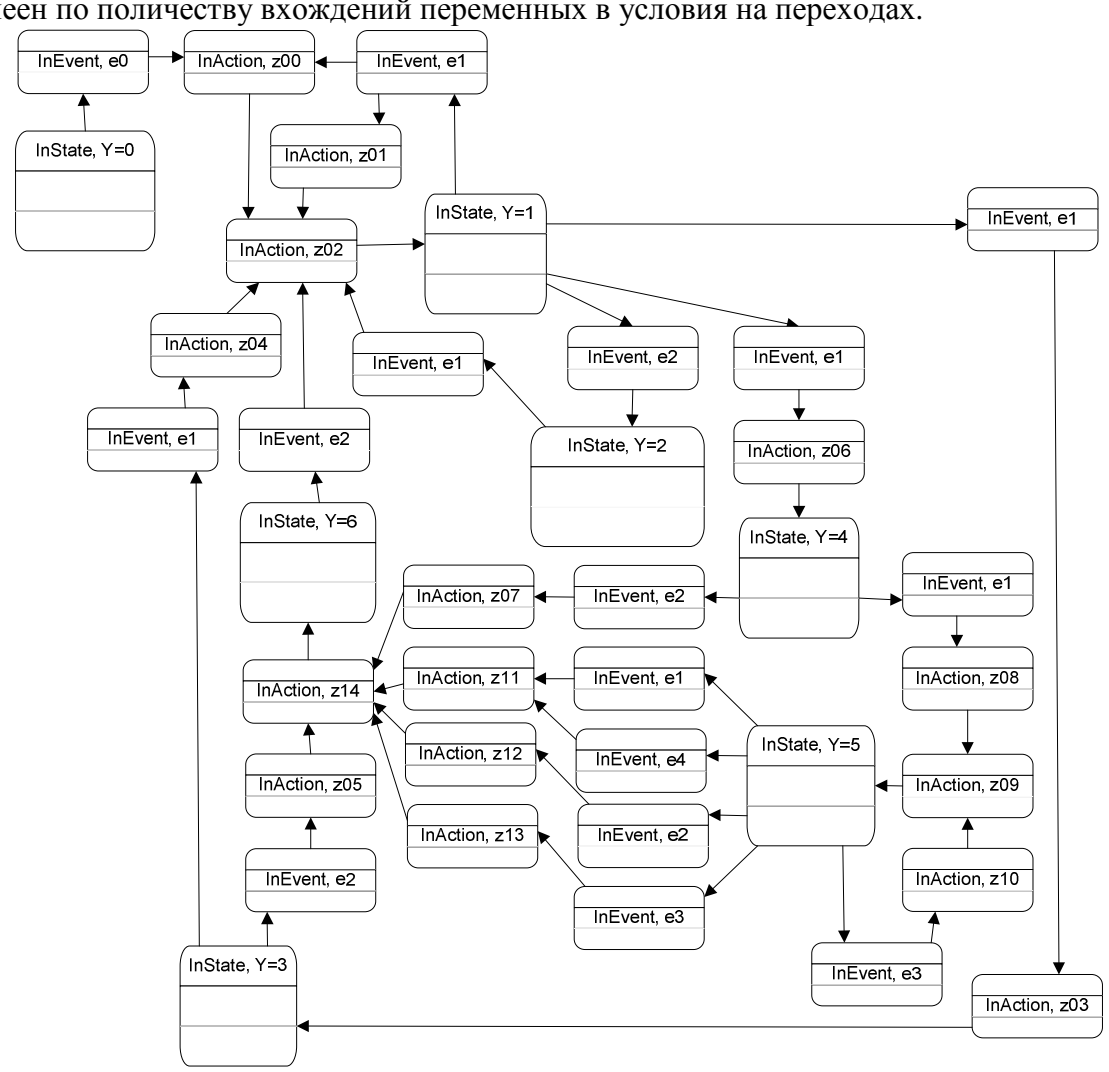


Рис. 10. Модель Крипке, полученная из автомата `ARemote` по второму методу

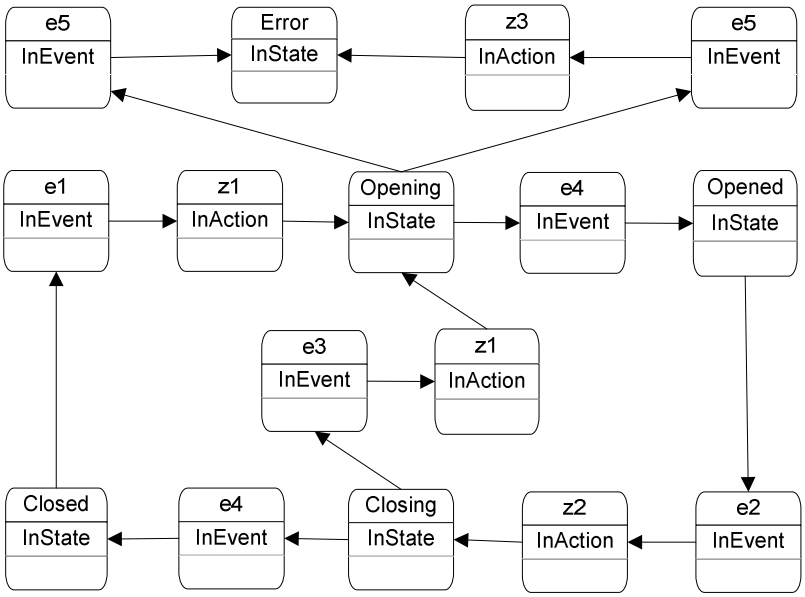


Рис. 11. Модель Крипке, полученная из автомата `AElevator` по второму методу

Приведём пример. Проверив выполнимость формулы

AG(*InState* → (¬*InAction*) **W** *InEvent*).

во всех состояниях модели Крипке для любого автомата, можно убедиться, что в каком бы состоянии автомат ни находился, выходное воздействие не может наступить раньше, чем произойдет некоторое событие.

Метод полного графа переходов

Метод, описанный в этом разделе, является наиболее выразительным в том смысле, что с его помощью можно верифицировать больше всего свойств.

В данном методе множество атомарных предложений *AP* равно $\{Y_1, Y_2, \ldots\} \cup \{e_1, e_2, \ldots\} \cup \{x_1, x_2, \ldots\} \cup \{z_1, z_2, \ldots\} \cup \{$*InState*, *InEvent*, *InAction*$\}$.

Как и в предыдущем случае, начиная с множества *S*, равного множеству состояний исходного автомата, добавим в отношение *Label* для каждого элемента $s \in S$ две пометки: (s, s) и (s, *InState*). После этого выполним разделение выходных переменных и состояний (как во втором методе) и приступим к добавлению в модель информации о переходах между состояниями исходного автомата. Для этого рассмотрим набор из всех булевых переменных в исходном автомате (состоящий из событий и выходных воздействий). Для каждого состояния p и каждой двоичной последовательности, которую можно присвоить переменным этого набора, введём понятие сценария, который должен произойти в этом состоянии при условии, что значением набора стала данная последовательность. Сценарий этот можно описать на естественном языке следующим образом: в состоянии p произошло событие e_i, при этом входные воздействия $x_{j[1]}, \ldots, x_{j[t]}$ (и только они) оказались истинными, после чего были вызваны выходные воздействия $z_{k[1]}, \ldots, z_{k[u]}$, и автомат перешёл в состояние q. Для каждого такого сценария (обозначим его буквой r) создадим $u + 1$ дополнительное состояние $\{r_e, r_1, \ldots, r_u\}$ и $u + 2$ перехода: $p \to r_e$, $r_e \to r_1$, $r_1 \to r_2$, …, $r_{u-1} \to r_u$, $r_u \to q$, а в отношение *Label* добавим пометки (r_e, e_i), (r_e, $x_{j[j^*]}$) (r_e, *InEvent*), (r_{k^*}, $z_{k[k^*]}$), (r_m, *InAction*) для всех j^* от 1 до t и k^* от 1 до n.

Таким образом, перебор сценариев будет эквивалентен перебору двоичных наборов, которые могут быть присвоены входным воздействиям, и такой перебор требуется выполнить для каждого состояния. Количество состояний полученной модели ограничено снизу числом

количество_состояний_в_исходном_автомате × (количество_двоичных_наборов + 1),

а число соответствующих двоичных наборов есть $2^{\text{количество\_входных\_переменных}}$.

Пример построения модели Крипке по автомату ATrig с помощью данного метода приведён на рис. 12.

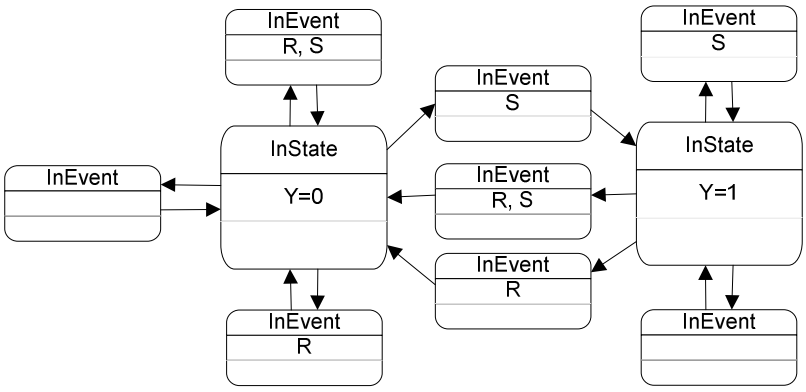


Рис. 12. Полный граф переходов для автомата ATrig

Теперь можно проверить, например, свойство $\neg(S \vee R) \rightarrow \mathbf{EX\ EX}\ (Y{=}1)$, которое утверждает, что если оба управляющих сигнала отсутствуют, то, преодолев переход, на следующем шаге можно оказаться в состоянии 1. Это свойство выполняется во всех состояниях, за исключением тех, которые помечены только служебным словом *InEvent*.

Метод редуцированного графа переходов

В данном методе множество *AP* равно $\{Y_1, Y_2, \ldots\} \cup \{e_1, e_2, \ldots\} \cup \{x_1, x_2, \ldots\} \cup \{!x_1, !x_2, \ldots\} \cup \{z_1, z_2, \ldots\} \cup \{\mathit{InState}, \mathit{InEvent}, \mathit{InAction}\}$.

Начинаем, как и в предыдущих методах, с того, что присвоим переменной *S* множество состояний исходного автомата и для каждого состояния $s \in S$ добавим в отношение *Label* две пометки: (s, s) и $(s, \mathit{InState})$. После этого выполним разделение выходных переменных и состояний, аналогичное тому, которое описано во втором методе (установки состояний на событиях и выходных воздействиях).

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 ¬ 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 *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 отражён на рис. 13.

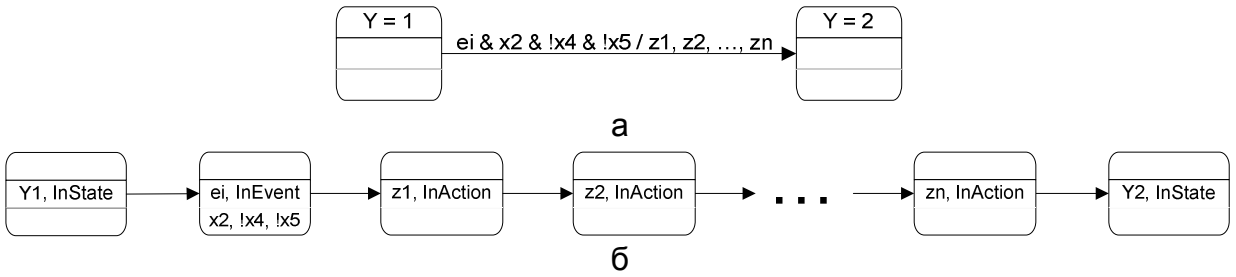


Рис. 13. Переход между состояниями до преобразования по методу редукции (а) и после него (б)

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

На рис. 14, 15 и 16 показаны модели Крипке, построенные с помощью редукции графа переходов из автоматов `ARemote`, `AElevator`, `ATrig` соответственно.

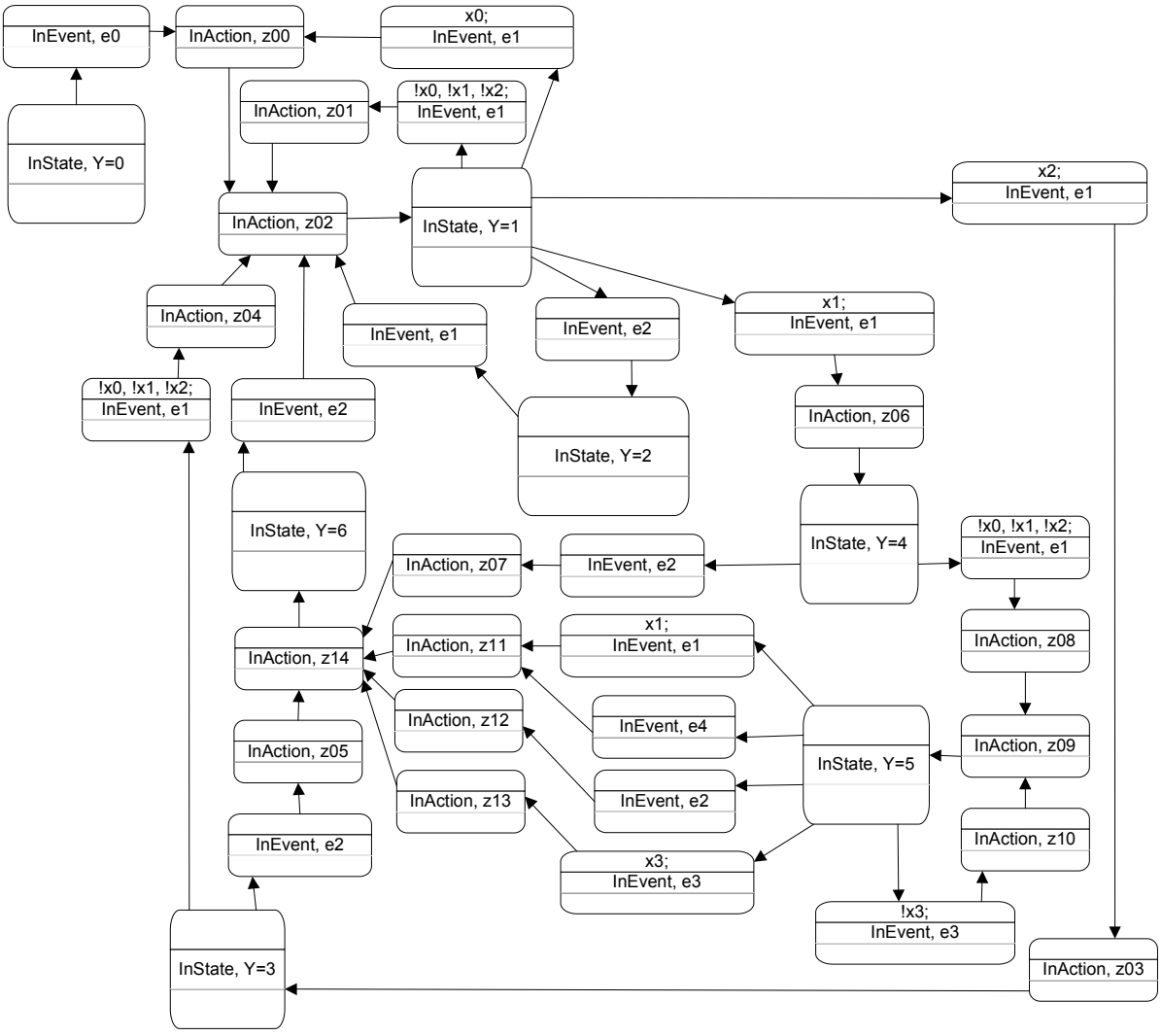


Рис. 14. Редукция графа переходов для автомата `ARemote`

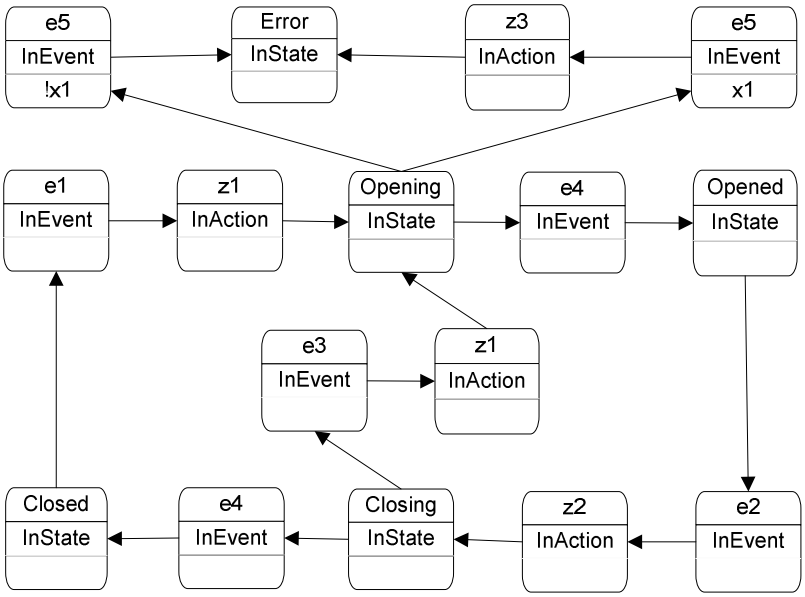


Рис. 15. Редукция графа переходов для автомата `AElevator`

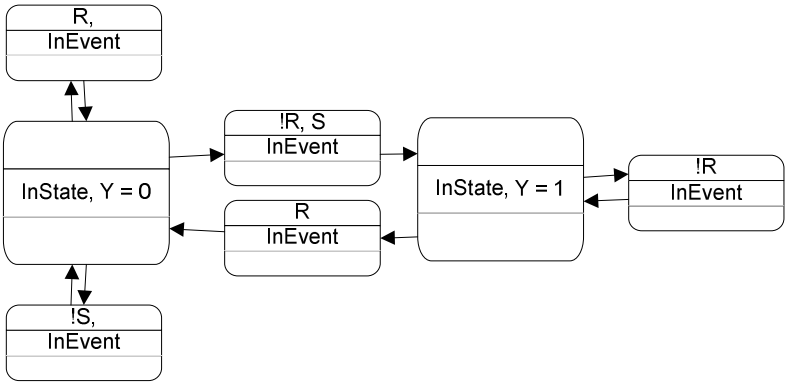


Рис. 16. Редукция графа переходов для автомата `ATrig`

Размер получившейся модели линеен, так же, как и при установке состояний на событиях и выходных воздействиях.

Теперь разберём 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, её следует привести к определённому («каноническому») виду. Вначале в ней нужно удалить все парные отрицания (путём замены подформул вида $\neg\neg f$ на f). После этого все входные воздействия, которые присутствуют в формуле без отрицания, необходимо предварить двумя отрицаниями: одно из них синтаксическое, другое логическое (это значит, что нужно 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).

Приведём пример для автомата `ARemote`. Пусть требуется проверить свойство: «существует способ попасть в рабочий режим с заполненным буфером сигналов». В терминах языка *CTL* с исходной семантикой данное свойство может быть записано следующим образом: $\mathbf{E}[x3\ \mathbf{U}\ (Y{=}1)]$.

Эта формула не выполняется в состоянии `Y=5` (рис. 14). На это, правда, и не стоило рассчитывать. Преобразуем формулу согласно нашему методу: **E**[¬`!x3` **U** (`Y=1`)].

В формулу языка *CTL* было внесено отрицание другое атомарного предложения, являющегося «синтаксическим отрицанием» исходного. Преобразованная формула уже верна для состояния `Y=5`.

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*.

Метод редуцированного графа переходов можно рассматривать как упрощение метода полной системы переходов, которое заключается в том, что двоичные наборы значений несущественных переменных отождествляются. Примеры верификации *CTL*-формул этим методом показывают, что данный подход не снижает существенно способность модели описывать поведение системы по сравнению с предыдущим методом. Использование такой схемы подходит для многих формул.

Преобразование сценария для модели Крипке в сценарий для автомата

После выполнения моделирования и верификации требуется обработать результаты проверки модели. Программы, написанные традиционными подходами, имеют достаточно сложные модели, и проводить проводить анализ путей прямо на модели Крипке неэффективно. При интерактивном моделировании совместно с исполнением и визуализацией автомата [14, 15] процесс представления путей в модели Крипке путями в автомате желательно автоматизировать.

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 описанных в настоящей работе методов, 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 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.

Рассмотрим пример для автомата `ARemote`. Пусть для состояния `3` выполняется верификация формулы ¬**E**[¬(`Y=6`) **U** (`Y=1`)] (в состояние `1` нельзя попасть, минуя состояние `6`). Эта формула в состоянии `3` не выполняется. Верификатор сгенерировал (кратчайший и единственный в данном случае) контрпример, который на рис. 17 выделен серым цветом. Это конечный путь, любое бесконечное продолжение которого удовлетворяет формуле ¬**E**[¬(`Y=6`) **U** (`Y=1`)].

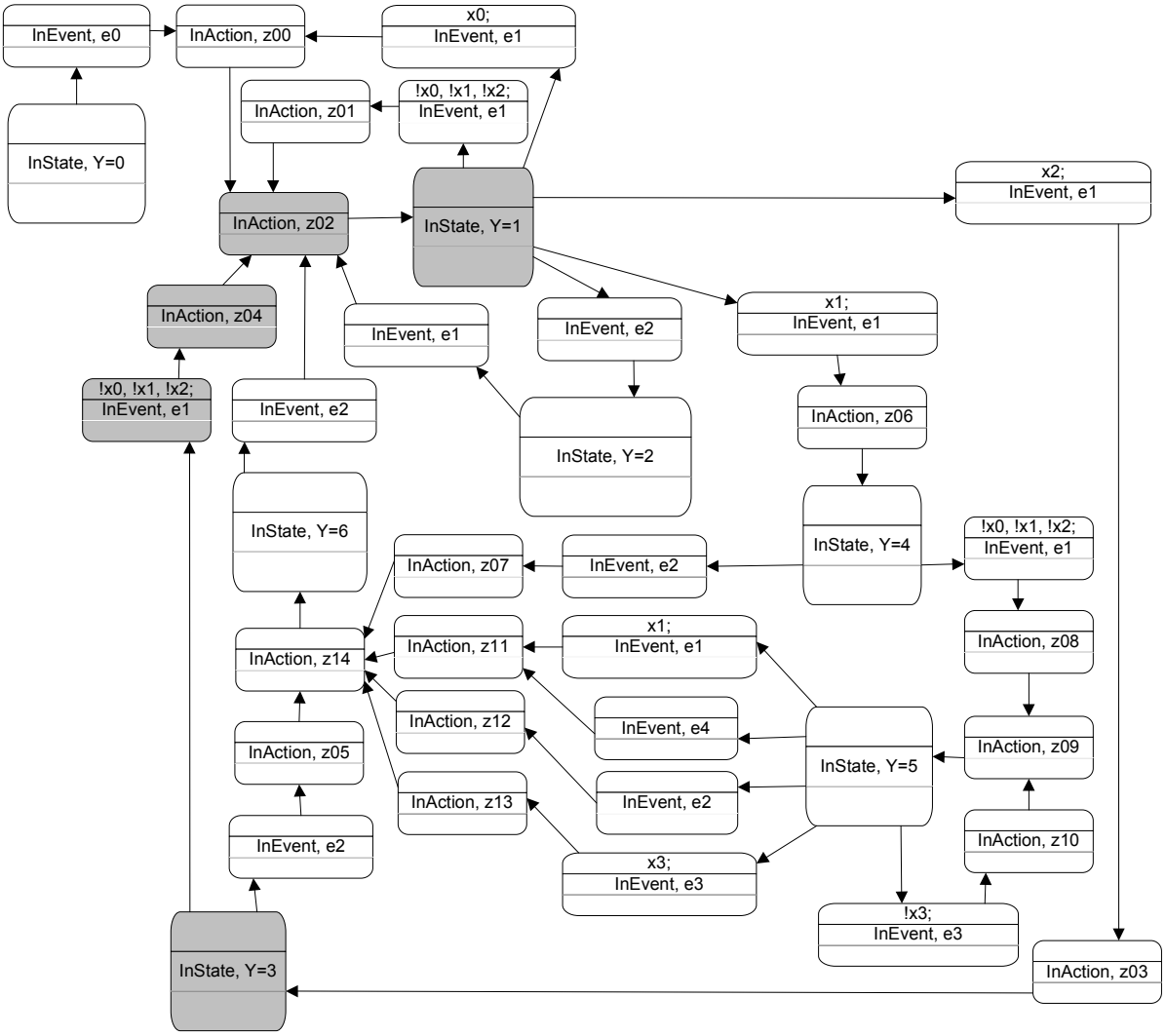


Рис. 17. Путь в модели Крипке

Этот же путь, но представленный в исходном автомате, можно увидеть на рис. 18. Промежуточная информация о переходе между состояниями выделена жирностью ребра и его метки.

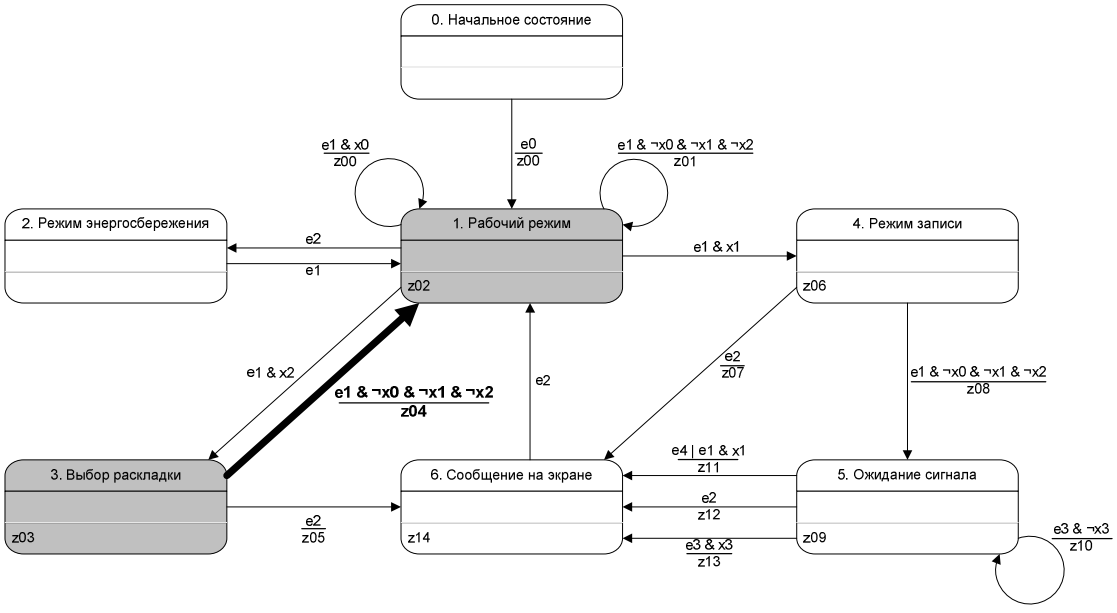


Рис. 18. Путь в исходном автомате `ARemote`

Заключение

В работе были предложены методы для представления автоматной модели структурами Крипке. В рамках исследований по государственному контракту [11] четвёртый из разработанных методов (метод редукции графа переходов) был реализован программно. Также были приведены примеры темпоральных формул, которые можно верифицировать данным методом. Программа позволяет строить трассы, которые подтверждают заданные формулы, начинающиеся с квантора существования пути (или опровергают отрицания этих формул). Рассмотренные примеры показывают, что запрограммированный алгоритм позволяет убеждаться в корректности модели, находить ошибки в случае некорректных формул и находить ошибки в моделях (ряд тестов проводился для намеренно изменённой модели, чтобы проверить работу алгоритма в соответствующих ситуациях).

Литература

1. *Emerson E. A., Clarke E. M.* Using branching time temporal logic to synthesize synchronisation skeletons // Science of Computer Programming 2: 241-266, 1982.
2. *Clarke E. M., Emerson E. A.* Synthesis of synchronisation skeletons for branching time logic / Logic of Programs, LNCS 131, pp. 52–71, 1981.
3. *Лифшиц Ю.* Верификация программ и темпоральные логики. Лекция №3 курса «Современные задачи теоретической информатики». СПбГУ ИТМО, 2005. 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. 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. *Roux C., Encrenaz E.* CTL May Be Ambiguous when Model Checking Moore Machines. UPMC – LIP6 – ASIM, CHARME, 2003. http://sed.free.fr/cr/charme2003-presentation.pdf
7. Finite state machine. http://en.wikipedia.org/wiki/Finite_state_machine
8. Mealy machine. http://en.wikipedia.org/wiki/Mealy_machine
9. Moore machine. http://en.wikipedia.org/wiki/Moore_machine
10. *Вельдер С. Э., Бедный Ю. Д.* Универсальный инфракрасный пульт для бытовой техники. Курсовая работа, СПбГУ ИТМО, 2005. http://is.ifmo.ru/projects/irrc/
11. Разработка технологии верификации управляющих программ со сложным поведением, построенным на основе автоматного подхода. НИР, выполняемая по гос. контракту № 02.514.11.4048 от 18.05.2007.
12. *В. А. Козлов, О. А. Комалёва.* Моделирование работы банкомата. СПбГУ ИТМО, 2006. http://is.ifmo.ru/unimod-projects/bankomat/
13. Margaria T. Model Structures. Service Engineering – SS 06. https://www.cs.uni-potsdam.de/sse/teaching/ss06/sveg/ps/2-ServEng-Model-Structures.pdf
14. Сайт проекта UniMod. http://unimod.sf.net
15. Сайт eVelopers Corporation. http://www.evelopers.com