ПРОГРАММИРОВАНИЕ, *2008*, № *1*, *с.* 38-60

УДК 519.687.1

МОДЕЛИРОВАНИЕ, СПЕЦИФИКАЦИЯ И ВЕРИФИКАЦИЯ "АВТОМАТНЫХ" ПРОГРАММ

© **2008 г. Е. В. Кузьмин, В. А. Соколов**

Ярославский государственный университет им. П.Г. Демидова

150000 Ярославль, ул. Советская, 14

E-mail: egorkuz@uniyar.ac.ru, sokolov@uniyar.ac.ru

Поступила в редакцию

1. ВВЕДЕНИЕ

Статья посвящена описанию, спецификации и верификации моделей программ, построенных на основе автоматного подхода к программированию [1–5].

Технология автоматного программирования является современной отечественной разработкой, которая активно развивается и поддерживается рядом российских исследовательских групп. При автоматном подходе к проектированию и построению программ выделяются две части: системно независимая и системно зависимая. Первая часть реализует логику программы и задается системой взаимодействующих автоматов Мура-Мили. Проектирование каждого автомата состоит в создании по словесному описанию (декларации о намерениях) схемы связей, описывающей его интерфейс, и графа переходов, определяющего его поведение. По этим двум документам формально и изоморфно может быть построен модуль программы, соответствующий автомату (и затем реализована системно зависимая часть).

"Автоматное" программирование не зависит от платформы, операционной системы или языка программирования. Технология автоматного программирования является достаточно эффективной при построении программного обеспечения для "реактивных" систем и систем логического управления. Эта технология, не исключающая других методов построения программного обеспечения "без ошибок", существенно более конструктивна, так как позволяет начинать "борьбу с ошибками" еще на стадии алгоритмизации.

Интересным является тот факт, что к "автоматным" программам могут быть успешно применимы (в совокупности) все существующие методы анализа корректности (верификации) сложных систем: имитационное моделирование, тестирование, дедуктивный анализ и метод проверки модели.

Имитационное моделирование и тестирование предусматривают проведение испытаний перед тем, как приступить к производству системы. Имитационному моделированию подвергается абстрактная схема или прототип, тестирование применяется непосредственно к самому продукту. Для программного обеспечения имитационное моделирование и тестирование включают в себя ввод определенных исходных данных и наблюдение за соответствующими выходными результатами. Эти методы позволяют находить ошибки (особенно эффективно на ранних стадиях), но они не могут гарантировать полное отсутствие ошибок в программе или оценить, сколько их еще осталось.

Несмотря на то, что дедуктивный анализ [6] является очень трудоемким методом с сильной привязкой к семантике языка программирования и не может быть полностью автоматизирован и лишен контроля эксперта, он, тем не менее, при желании вполне может быть применим для "автоматных" программ после их полного построения непосредственно для проверки корректности процедур (написанных на некото-

ром языке высокого уровня), соответствующих выходным воздействиям или входным запросам. Каждое выходное воздействие выполняет свою, обычно небольшую, отдельную задачу, корректность реализации которой может быть проверена. Таким образом, можно говорить о том, что автоматная структура программы благоприятствует применению метода дедуктивного анализа. Но этот метод оказывается бесполезным при проверке логики программы.

С другой стороны, для проверки логики "автоматных" программ удобным является метод проверки модели [7]. При этом методе для программы строится формальная конечная модель (то есть с конечным числом состояний), а проверяемые свойства задаются с помощью формул темпоральной логики. Проверка выполнимости темпоральных формул, задающих свойства модели, происходит автоматическим образом. Важно отметить, что построение модели программы для автоматической верификации является весьма серьезной и сложной задачей. При построении модели для программы, написанной традиционным способом, возникает проблема адекватности этой программной модели исходной программе. Модель может не учитывать ряд программных свойств или порождать несуществующие свойства. Автоматный подход к программированию с точки зрения моделирования и анализа программных систем имеет ряд преимуществ по сравнению с традиционным подходом. При автоматном программировании не возникает проблемы адекватности модели, поскольку набор взаимодействующих автоматов, описывающий логику программы, уже является адекватной моделью, по которой формальным образом строится программный модуль. И это является бесспорным плюсом автоматной технологии. Более того, модель имеет конечное число состояний, что является необходимым на практике условием для успешной автоматической верификации, поскольку метод проверки на модели представляет собой переборный метод.

Одними из наиболее популярных темпоральных логик для спецификации и верификации свойств программных систем являются логика CTL (branching-time logic или computation tree logic) и логика линейного времени LTL (linear-time logic). Для целей верификации автоматных программ логика LTL заслуживает особого внимания, поскольку любая формула в рамках этой логики по сути представляет собой автомат Бюхи, описывающий (принимающий) бесконечные допустимые пути структуры Крипке, которая, в свою очередь, задает поведение (все возможные исполнения) проверяемой на корректность "автоматной" модели. Это позволяет при спецификации и верификации "автоматных" программ оперировать в основном таким простым понятием, как "автомат".

Таким образом, перечисленное выше позволяет говорить о возможности эффективного применения для верификации "автоматных" программ (для анализа корректности логики "автоматных" программ) метода проверки модели.

Существует ряд программных средств верификации абстрактных моделей, построенных в рамках некоторых формализмов (сети Петри, взаимодействующие асинхронные процессы, автоматы с реальным временем, гибридные автоматы и т.д.), базирующихся на общих подходах и технологиях. Среди этих средств верификации классическим методом проверки модели можно выделить, например, CPNTools [8], SPIN [9], SMV [10] и CADP [11], использующие в качестве языков спецификации формальных моделей, соответственно, раскрашенные сети Петри высокого уровня, язык Promela, синхронные процессы и язык LOTOS. С одной стороны, можно считать, что целью создания такого рода программных продуктов является, главным образом, отработка тех или иных теоретических и прикладных методов верификации моделей, но в то же время многие из них имеют ориентированный на конкретную область применения язык спецификации – например, на моделирование и анализ коммуникационных протоколов или разработку синхронных аппаратных логических схем. Однако, программных средств верификации, направленных непосредственно на поддержку технологии автоматного программирования, не существует, что не исключает, а скорее предполагает активное использование уже имеющихся (проверенных временем) разработок ведущих лабораторий мира, которые обычно долгое время сопровождаются тематическими конференциями (зачастую полностью

посвященными этим продуктам). Существующие средства-верификаторы в основном первоначально предполагают выделение модели из реальной программы, а затем задание этой модели в рамках некоторого абстрактного формализма. Далее специфицируются и проверяются свойства построенной формальной модели, поведение которой, вообще говоря, может значительно отличаться от оригинальной программы, в том числе и из-за особенностей используемого формализма. Цель исследований по теме данной статьи – создание программного комплекса, рассчитанного на проектирование программных продуктов с помощью автоматных моделей, из которых после соответствующего анализа на корректность происходит дальнейшее построение и порождение программы. Таким образом, в этом случае будет анализироваться адекватная автоматная модель, отвечающая за функциональную логику программы. По автоматной модели проверяются (относительно спецификации) функциональные требования, предъявляемые к программному обеспечению. Если после проверки автоматной модели в программе обнаружатся ошибки, то они, скорее всего, будут относиться к ошибкам деталей реализации и никак не будут касаться функциональной логики программы, что не потребует при исправлении этих ошибок глобальной перестройки (перепроектирования) всей программы.

В нашей статье рассматривается одна из возможных моделей построения автоматных программ – иерархическая модель. Для спецификации и верификации автоматных программ, построенных по этой модели, предлагается технология применения метода проверки модели (метода model checking). Исследуется возможность спецификации (с точки зрения простоты и адекватности восприятия) структурных и семантических свойств автоматных программ с помощью темпоральной логики. При этом акцент ставится в большей степени на семантике автоматных программ, нежели на структуре, в отличие от большого количества работ, где взаимодействующие автоматы (с различными расширениями) рассматриваются как некий абстрактный формализм, для которого, наряду с разрешимостью классических для теоретической информатики свойств, оценивается лишь трудоемкость метода проверки модели, связанная только со структурой автоматов и схемой их взаимодействия. Более того, важным моментом в настоящей статье является выбор обозреваемого поведения модели автоматных программ, так как именно от него зависят как классы (виды) проверяемых свойств, так и сама возможность адекватной спецификации. В целом, статья ориентирована на поддержку автоматного подхода к программированию "реактивных" систем и сложных систем логического управления.

2. ИЕРАРХИЧЕСКАЯ МОДЕЛЬ АВТОМАТНЫХ ПРОГРАММ ДЛЯ "РЕАКТИВНЫХ" СИСТЕМ И СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ

Рассмотрим иерархическую систему взаимодействующих детерминированных конечных автоматов

$$\mathrm{A} = (A_0, A_{11}, \ldots, A_{1k_1}, \ldots, A_{n1}, \ldots, A_{nk_n}),$$

где n и k_i $(1 \le i \le n)$ – натуральные числа. Автомат A_0 назовем основным, остальные автоматы – вложенными. Между всеми автоматами установлена иерархия по вложенности. Автомат A_{ij} может передавать управление стоящему ниже по иерархии автомату A_{i+1k}. В таком случае автомат A_{ij} называется главным, а A_{i+1k} – вложенным. У каждого вложенного автомата существует только один главный автомат, в который он вложен.

Система автоматов A рассматривается как реактивная система управления некоторым объектом. Система A получает от объекта управления события, характеризующие, например, изменение его состояния, а также сама запрашивает текущие параметры объекта, что также считается входным воздействием на систему A. В то же время система управления, реагируя на поступающую информацию, сама оказывает воздействия на объект управления.

Кроме описанного взаимодействия с "внешней средой", происходит аналогичное взаимодействие автоматов между собой внутри системы посредством передачи (со стороны главно-

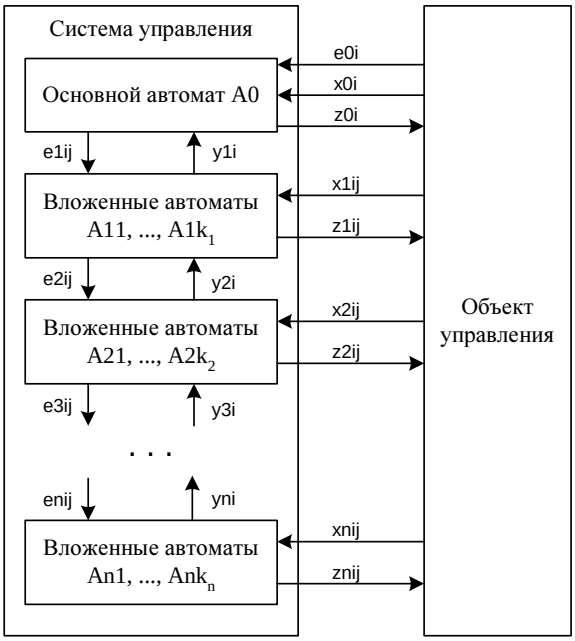

Рис. 1. Модель взаимодействия системы A с объектом управления.

го автомата) управления с некоторым событием вложенным автоматам и отслеживания их текущих состояний.

В рассматриваемой модели только основной автомат A_0 принимает события от объекта управления и реагирует на них; вложенные же автоматы обращаются к объекту управления только с запросами состояния его параметров (причем вложенный автомат может сделать запрос лишь в том случае, если главный автомат передал ему управление).

Описываемая модель взаимодействия системы управления A с объектом управления показана на рис. 1. На рисунке события обозначаются символом **e**, запросы к объекту управления (входные переменные) – **x**, текущее состояние некоторого автомата A_{ij} хранится в переменной $\mathbf{y}_{ij}$, а выходные воздействия обозначаются **z**.

Поведение любого автомата A системы A аналогично поведению всей системы в том смысле, что автомат A реагирует на произошедшие события и, в зависимости от своего состояния, состояния вложенных автоматов и состояния объекта управления (т. е. значений входных переменных), оказывает воздействие на объект управления или же на вложенные автоматы, передавая им управление с некоторым событием.

Обозначим $E_A = \{e_1, \ldots, e_k\}$ множество всех имен событий, на которые реагирует автомат A. Определим на множестве E_A переменную событий ϵ, в которую будет помещаться имя произошедшего события для автомата A. Текущее значение переменной ϵ будем обозначать $E_A(e)$.

Введем множество $X_A = \{x_1, \ldots, x_n\}$ запросов автомата A к объекту управления. Каждый запрос рассматривается как некоторый предикат, истинность которого зависит от состояния параметров объекта управления.

Также введем $Z_A = \{z_1, \ldots, z_r\}$ – множество выходных воздействий автомата A. Воздействия z_i подразделяются на две группы. Первая группа – это воздействия непосредственно на объект управления. Воздействия второго типа – передача управления вложенному автомату с некоторым событием (генерация события для вложенного автомата). В этом случае выходное воздействие z_i имеет вид $A'(e'_j)$, где A' – вложенный автомат, а e'_j – сгенерированное автоматом A событие для вложенного автомата A', которому передается управление для обработки этого события.

Тогда автомат A управляющей системы автоматов A представляет собой набор $(\Sigma, Q, q_0, E, X, Z, \delta)$, где

- $Q = \{q_0, q_1, \ldots, q_n\}$ – конечное множество состояний автомата A;
- q_0 – начальное состояние;
- $\Sigma = \{a_1, a_2, \ldots, a_k\}$ – конечный алфавит пометок дуг переходов;
- $\delta : Q \times \Sigma \to Q$ – функция переходов из одного состояния в другое.

Каждый переход срабатывает по определенному правилу. Прежде чем описывать правила переходов, введем еще ряд обозначений.

Для метки перехода $a \in \Sigma$ обозначим $E(a)$ событие, на которое автомат A реагирует при срабатывании перехода с меткой a.

Будем обозначать $X(a)$ множество запросов к объекту управления, истинность которых необходима для срабатывания перехода с пометкой a.

Пусть Z^* – множество конечных последовательностей выходных воздействий, тогда для $a \in \Sigma$ обозначим $Z^*(a) \in Z^*$ последовательность выходных воздействий, которая происходит при срабатывании перехода с пометкой a.

Также для произвольного состояния q автомата A введем обозначение $Z^*(q) \in Z^*$ для последовательности выходных воздействий, которая должна выполняться при попадании (переходе) автомата в состояние q.

И, наконец, пусть $Y(a)$ – предикат, зависящий от состояний вложенных автоматов, истинность которого необходима для срабатывания перехода с пометкой a.

Правило перехода из состояния q в состояние q' по метке a имеет вид:

$q, a:$ **if** $e = E(a)$ **and** $(\forall x \in X(a) : x = true)$ **and** $Y(a) = true$
then $Z^*(a); Z^*(q');$ **goto** q'.

Из изложенного следует, что правило перехода в новое состояние в общем случае можно описать следующим образом. После получения события автомат, в зависимости от своего текущего состояния, реагирует (или никак не реагирует) на событие, опрашивает параметры объекта управления (входные переменные), учитывает состояния вложенных автоматов, затем производит последовательность выходных воздействий, включая и те выходные воздействия, которые необходимо совершить при попадании в новое состояние, и только после этого переводится в новое состояние (которое в случае петли может быть тем же самым).

Выходное воздействие первого типа, направленное на объект управления, считается сразу же осуществленным после его применения. Выходное воздействие второго типа, представляющее собой передачу управления с событием вложенному автомату, считается выполненным только лишь после реакции вложенного автомата на это событие, которая заключается в том, что либо автомат переходит в новое состояние (срабатывает один из переходов вложенного автомата), либо событие игнорируется вложенным автоматом (ни один из переходов сработать не может). До тех пор пока выходное воздействие второго типа не выполнится, работа (процесс перехода в новое состояние) главного автомата приостанавливается.

Важно отметить, что правила переходов, а точнее условия срабатывания переходов, должны удовлетворять условию детерминированности (или ортогональности), т. е. при поступлении некоторого события может быть готов к срабатыванию не более чем один переход. Если ни один переход в текущем состоянии при поступлении некоторого события сработать не может (условия переходов не выполняются), то событие игнорируется (переменная событий e для этого автомата обнуляется).

Далее рассмотрим иерархическую модель автоматных программ на примере системы управления кофеваркой [12].

3. АВТОМАТНАЯ МОДЕЛЬ СИСТЕМЫ УПРАВЛЕНИЯ КОФЕВАРКОЙ

Рассмотрим системно независимую часть автоматной программы, которая отвечает за *логику* управления кофеваркой.

Модель кофеварки представлена на рис. 2. Кофеварка позволяет выбирать количество порций кофе для варки с помощью кнопок "+" (увеличить количество порций на единицу) и "–" (уменьшить количество порций на единицу).

В кофеварке предусмотрена возможность индикации отсутствия воды и основных неисправностей (например, не работает нагреватель или неисправен один из клапанов). Количество пор-

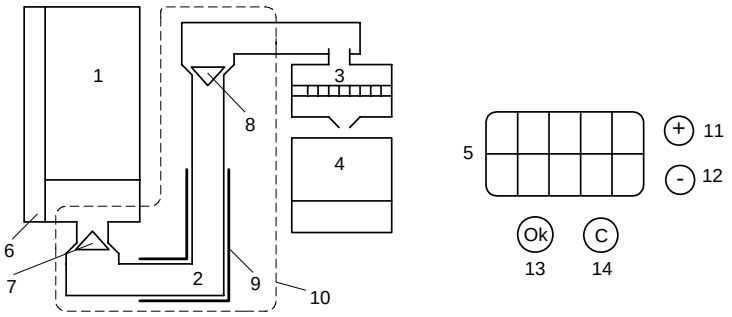

Рис. 2. Кофеварка: 1 – емкость для воды, 2 – емкость для кипячения, 3 – фильтр с кофе, 4 – колба для готового напитка, 5 – дисплей, 6 – датчик воды, 7 – входной клапан (клапан 1), 8 – выходной клапан (клапан 2), 9 – нагреватель кипячения воды, 10 – бойлер, 11 – кнопка увеличения "+", 12 – кнопка уменьшения "–", 13 – кнопка "Ok", 14 – кнопка "C" (отмена).

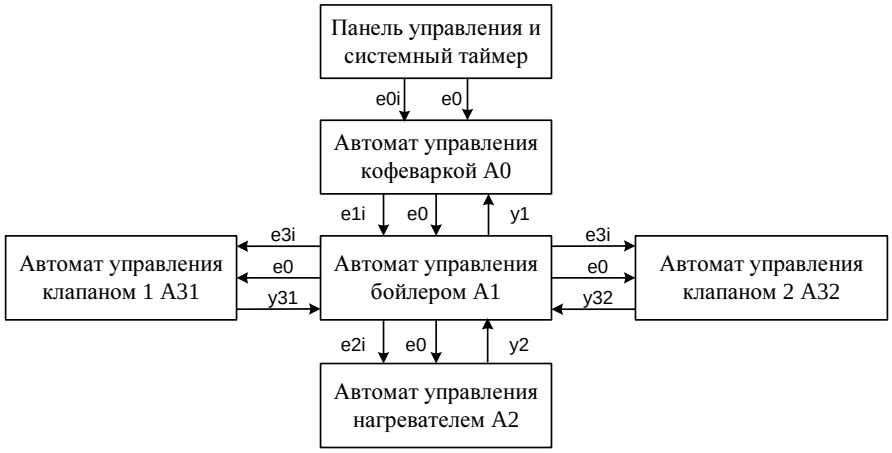

Рис. 3. Диаграмма взаимодействия автоматов управления.

ций изменяется от 1 до 5. При попытке увеличить максимальное значение порций нажатием кнопки "+" ничего не происходит (аналогично для значения порций 1 нажатие "–" игнорируется). Отдельно выделяется объект "бойлер", который состоит из клапанов 1 и 2, нагревателя и емкости для кипячения. Дисплей кофеварки разбит на части, использующиеся для индикации состояний клапанов, нагревателя и собственно кофеварки.

В кофеварке выделяются четко выраженные подобъекты управления (бойлер, клапаны и нагреватель), которые имеют свои собственные подсистемы управления. Каждая подсистема управления представлена в виде конечных автоматов Мура-Мили, выстроенных в иерархию. В результате логическая часть системы управления кофеваркой имеет вид иерархической системы взаимодействующих автоматов. Автомат, находящийся выше по иерархии, управляет вложенными в него автоматами путем генерации событий и передачи автоматам управления с этими событиями. Кроме того, автомат следит за состояниями вложенных автоматов, так как от них могут зависеть его собственные переходы по состояниям.

Диаграмма взаимодействия автоматов представлена на рис. 3. Автомат $A0$, называемый основным, получает от панели управления (панели кнопок) события $e0i$, на которые он реагирует. Специальное событие $e0$, генерируемое постоянно системным таймером, используется для проверки условий переходов (условий на дугах автоматов), которые не предусматривают реакцию ни на одно из событий $e0i$. Автомат $A0$ взаимодействует с автоматом $A1$, передавая ему управление с событиями $e1i$ и $e0$. Автомат

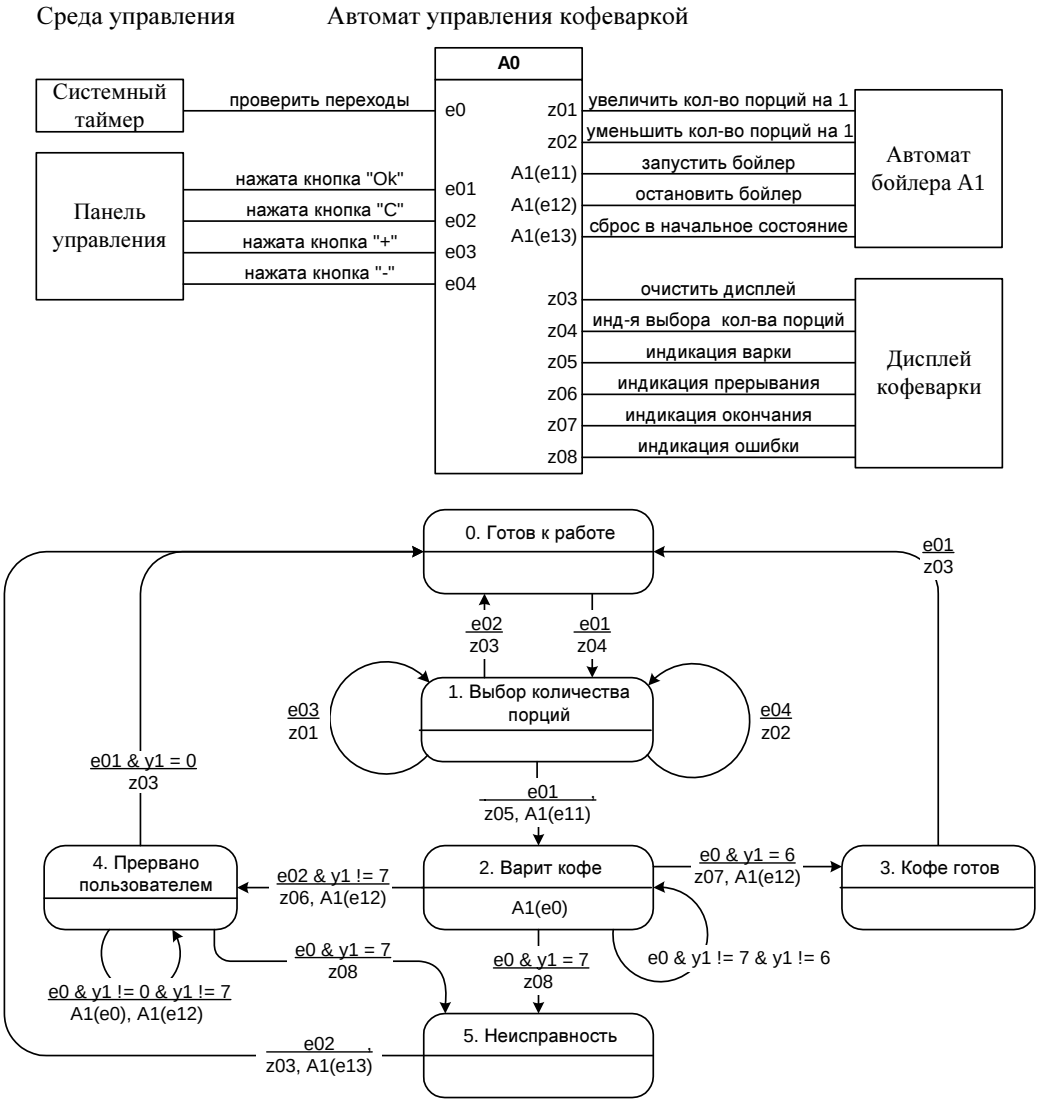

Рис. 4. Схема связей и граф переходов автомата управления кофеваркой $A0$.

$A0$ отслеживает состояния $A1$ через переменную $y1$. При таком взаимодействии автомат $A0$ считается главным, а $A1$ – вложенным. Аналогичным образом происходит взаимодействие автомата $A1$ с вложенными автоматами $A2$, $A31$ и $A32$.

Автомат $A0$ реализует логику управления кофеваркой. Он реагирует на нажатие кнопок на панели управления (при задании количества порций, запуске процесса варки, отмене варки, сбросе ошибки и т. д.), передавая управление на время варки кофе взаимодействующему с ним автомату управления бойлером $A1$. Информация о текущих состояниях кофеварки отображается на дисплее.

Схема связей и граф переходов автомата управления кофеваркой $A0$ представлены на рис. 4.

Автомат $A1$ реализует логику управления бойлером, который непосредственно отвечает за процесс варки кофе.

В процессе варки контролируются работоспособность клапанов, нагревателя, а также проверяется наличие воды. В случае неисправности одного из клапанов или нагревателя варка прерывается и высвечивается на дисплее соответствующее предупреждение. В случае отсутствия воды в емкости варка приостанавливается, пока вода не будет долита либо пользователь не прервет варку.

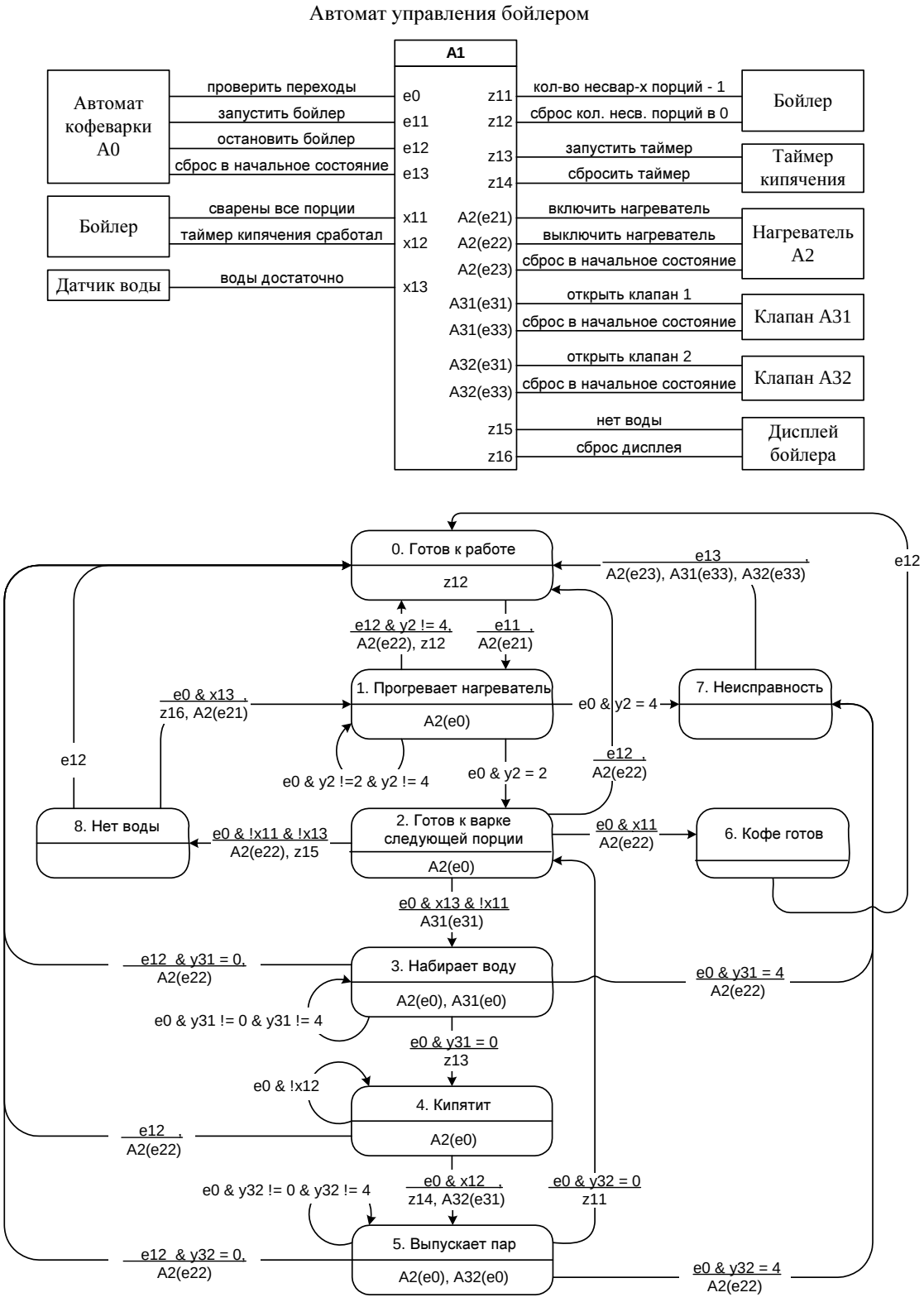

Рис. 5. Схема связей и граф переходов автомата управления бойлером $A1$.

Если не произошло поломок, процесс варки повторяется до тех пор, пока не будет сварено установленное количество порций кофе. Один цикл варки соответствует одной порции.

Автомат $A1$ взаимодействует с автоматом управления $A0$, передавая последнему текущее состояние и получая от него события. Автомат $A1$ также взаимодействует с автоматами $A31$, $A32$ (отвечающими за управление клапанами 1 и 2) и $A2$ (управляющим нагревателем), получая их состояния и посылая им события.

Схема связей и граф переходов автомата управления бойлером $A1$ представлены на рис. 5.

Автомат $A2$ реализует логику управления нагревательным элементом. Он поддерживает температуру нагревателя в заранее заданном диапазоне (для температуры устанавливаются допустимые минимум и максимум). Предусмотрена возможность установления неисправности вида "нагреватель не работает".

После включения нагревателя происходит его прогрев до нижней границы рабочего диапазона. Если он не прогрелся до необходимой температуры за заданное время, то нагреватель считается неисправным. Нагреватель включается при достижении температурой нижней границы рабочего интервала и выключается при достижении верхней границы.

Автомат $A2$ взаимодействует только лишь с автоматом управления бойлером $A1$ и не имеет вложенных автоматов.

Автоматы $A31$ и $A32$ полностью идентичны. Под автоматом $A3$ понимается как автомат $A31$, так и автомат $A32$. Автомат $A3$ служит для реализации логики управления работой клапана в режиме "открыть-пауза-закрыть". Автомат имеет четыре состояния и, не имея вложенных автоматов, взаимодействует только с автоматом управления бойлером $A1$, передавая последнему свое состояние и получая от него события. Автомат $A3$ начинает цикл своей работы по событию $e31$ ("открыть клапан"). Сначала клапан открывается, затем делается пауза для набора воды или выпуска пара, а после этого клапан закрывается. В случае если за определенное время клапан не откроется (либо не закроется), он считается неисправным.

Схемы связей и графы переходов автомата управления нагревателем $A2$ и автомата управления клапаном $A3$ представлены на рис. 6.

4. СПЕЦИФИКАЦИЯ И ВЕРИФИКАЦИЯ "АВТОМАТНЫХ" МОДЕЛЕЙ

При построении автоматной программы в рамках иерархической модели логика программы сосредотачивается в основном автомате, который распределяет управление вложенным автоматам в зависимости от поведения управляемого объекта. Каждый автомат программы взаимодействует только со своими главным и вложенными автоматами, что облегчает понимание программы. Во время проектирования или верификации такой автоматной программы возможно рассмотрение части или некоторого поддерева системы автоматов в зависимости от той функции, которая реализуется выделенной подсистемой автоматов. Любая подсистема взаимодействующих автоматов в иерархической модели представляет собой дерево автоматов, которое можно рассматривать как отдельную систему (как отдельную автоматную программу). Это позволяет менять масштаб всей системы, относя не интересующие проектировщика в данный момент времени автоматы к внешней среде, т. е. к внешнему объекту управления, и сосредоточивать внимание только на анализируемой подсистеме. И, наконец, при верификации спецификация и анализ свойств автоматной программы проводятся проще при понятной и несложной по структуре верифицируемой модели.

Автоматная программная модель является идеальным объектом для автоматической верификации методом проверки модели (model checking) [7], смысл которого состоит в следующем.

Поведение программной модели описывается конечной системой переходов, которая называется структурой Крипке. Конечность системы переходов означает, что система имеет конечное число состояний. При некотором упрощении конечную систему переходов можно представить как конечный ориентированный граф с явно выделенной начальной вершиной.

Далее в рамках структуры Крипке с использованием языка темпоральной логики специфицируются свойства программной модели, истинность которых для этой модели затем и проверяется. Используя такие простые объекты, как конечная система переходов и формулы темпоральной логики, возможно автоматическим спо-

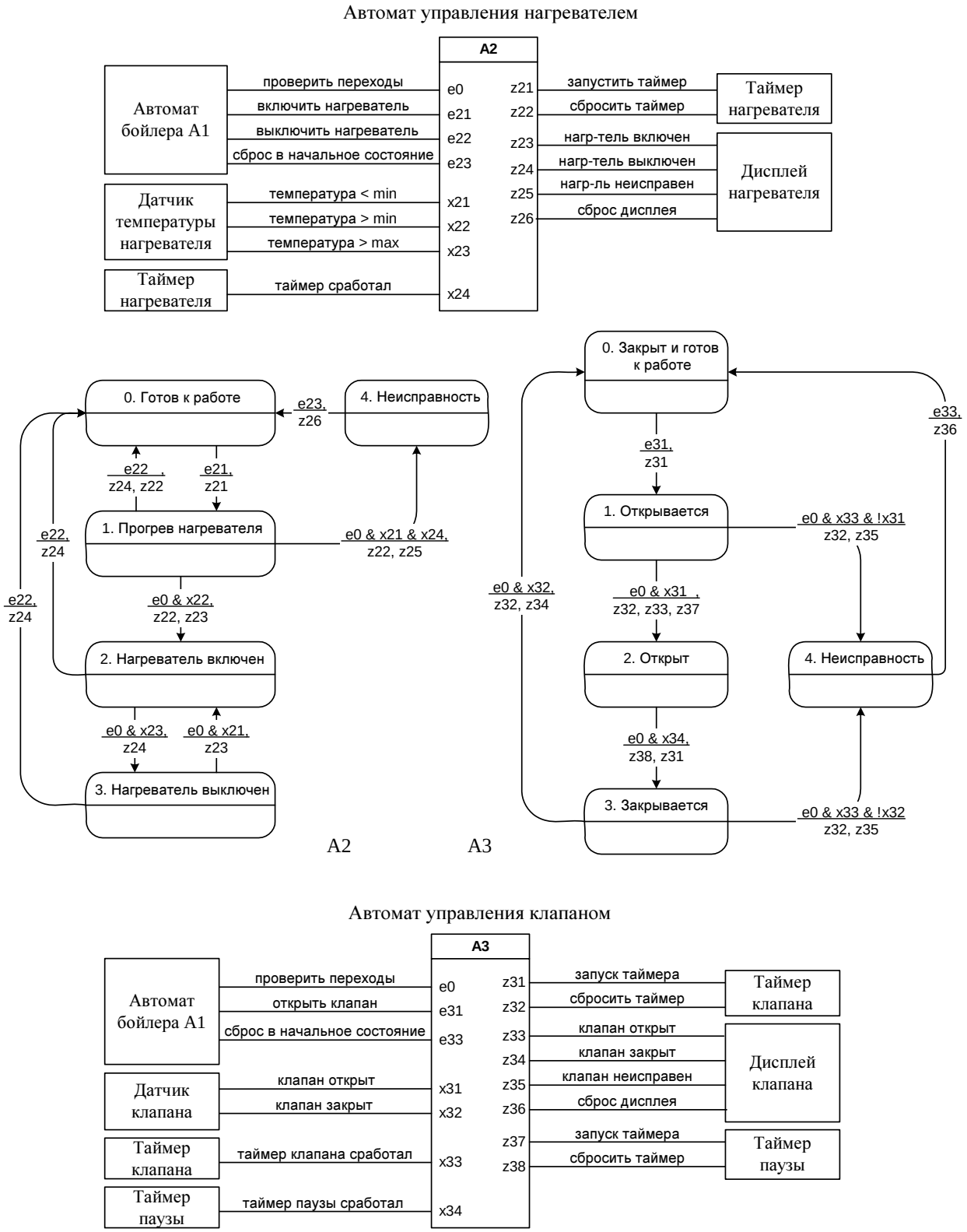

Рис. 6. Схема связей и граф переходов автоматов нагревателя $A2$ и клапана $A3$.

собом выполнить проверку свойств модели, заданных в виде формул.

Темпоральные логики играют важную роль в формальной верификации. Они используют-

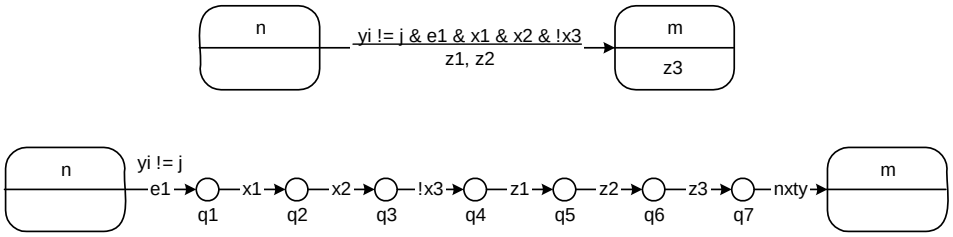

Рис. 7. Выделение промежуточных состояний для автоматного перехода.

ся для выражения свойств процессов. Такими свойствами, например, являются следующие: "*процесс никогда не достигнет тупикового состояния*" или "*в любом бесконечном исполнении процесса действие b происходит бесконечно часто*".

Задача проверки модели – это задача, состоящая в том, чтобы определить выполнимость для процесса, который задается системой переходов (структурой Крипке), свойства, выраженного формулой темпоральной логики. Сформулируем задачу проверки модели.

Задача проверки модели

Дано: конечная система переходов (структура Крипке), начальное состояние s_0 этой системы и формула φ темпоральной логики.

Вопрос: удовлетворяет ли состояние s_0 системы переходов формуле темпоральной логики φ?

5. СТРУКТУРА КРИПКЕ АВТОМАТНОЙ МОДЕЛИ

Структурой Крипке над множеством элементарных высказываний P называется система переходов $\boldsymbol{S} = (S, s_0, \to, L)$, где

- S – конечное множество состояний;
- $s_0 \in S$ – начальное состояние;
- $\to \subseteq S \times S$ – тотальное отношение переходов (тотальность означает, что для каждого состояния $s \in S$ должно существовать состояние $s' \in S$, для которого имеет место $(s, s') \in \to$, т. е. $s \to s'$);
- $L : S \to 2^P$ – функция, помечающая каждое состояние множеством элементарных высказываний, истинных в этом состоянии.

Путь в структуре Крипке из состояния s_0 – это бесконечная последовательность состояний $\pi = s_0 s_1 s_2 \ldots$ такая, что для всех $i \geq 0$ выполняется $s_i \to s_{i+1}$.

Для некоторого пути $\pi = s_0 s_1 s_2 s_3 \ldots$ обозначим π^i суффикс π, который получается удалением из π первых i состояний – например, $\pi^1 = s_1 s_2 s_3 \ldots$, а $\pi(i)$ будет обозначать i-ое состояние пути, $\pi(0) = s_0$, $\pi(1) = s_1$, и т. д.

Рассмотрим для произвольной иерархической программы, построенной на принципах автоматного программирования, ее автоматную модель A, представляющую собой набор взаимодействующих автоматов $(A_0, A_1, \ldots, A_n)$.

Для этой автоматной модели A построим структуру Крипке, которая, с одной стороны, описывает все возможные состояния системы A, а с другой стороны, задает семантику элементарных высказываний, истинных в этих состояниях.

Рассмотрим произвольный автомат A_k из системы автоматов A. Выделим в автомате A_k, помимо основных состояний, множество его промежуточных состояний, в которых автомат пребывает во время перехода из одного основного состояния в другое. Промежуточное состояние перехода автомата будем фиксировать каждый раз, когда автомат совершит одно из элементарных действий, т. е. автомат отреагирует на некоторое событие e_k, обратится к объекту управления с запросом значений входных переменных x_k ($!x_k$) или произведет некоторое выходное воздействие на объект управления или вложенный автомат z_k.

Продемонстрируем идею выделения промежуточных состояний для перехода произвольного автомата A_k (рис. 7).

На рис. 7 промежуточный переход с пометкой $nxty$ вводится для того, чтобы иметь возможность непосредственно отслеживать момент перехода вложенного автомата в следующее основ-

ное состояние с одновременной передачей управления главному автомату (в случае промежуточного перехода $nxty$ основного автомата, активным остается по-прежнему основной автомат). Внутренний переход с пометкой $e1$ может произойти при наступлении события $e1$, если выполняется условие $yi! = j$. Все остальные переходы не имеют условий и являются активными при попадании автомата в соответствующие промежуточные состояния. В указанной на рисунке последовательности переходов по внутренним состояниям обязательно сначала идет переход, соответствующий входному событию, далее переходы с пометками входных запросов, а затем переходы, помеченные выходными воздействиями. Таким образом, можно разделить последовательность внутренних состояний на три части, которые идут друг за другом строго в соответствии с указанным порядком. В каждой части метки внутренних переходов располагаются в порядке следования соответствующих элементов в выражении на дуге основного (исходного) перехода.

Далее для произвольного автомата под переходом в следующее состояние будем предполагать *внутренний* переход в основное или промежуточное состояние.

Каждому автомату A_k из набора A сопоставим переменные y_k, xm_k, zd_k, ev_k и st_k ($0 \le k \le n$). А для всей системы автоматов A в целом введем переменные ev, xm, zd, act, $auto$ и $evnt$.

Каждая из введенных переменных имеет следующий смысл.

1. Переменная y_k хранит последнее основное состояние автомата A_k.

2. Переменная xm_k содержит последний выполненный запрос параметров объекта управления. Значениями xm_k являются имена входных воздействий в прямой или инверсной форме, т. е. xm_k может содержать либо x, либо $!x$, где $x \in X_{Ak}$.

3. Переменная zd_k используется для хранения имени последнего выполненного выходного воздействия $z \in Z_{Ak}$.

4. Переменная ev_k содержит имя последнего обработанного автоматом A_k входного события $e \in E_{Ak}$, т. е. имя такого события, на которое автомат A_k отреагировал.

5. Переменная st_k хранит текущее состояние автомата A_k. Значениями st_k являются как основные, так и промежуточные состояния автомата.

6. Переменная $auto$ содержит номер активного в данный момент времени автомата из набора A, т. е. номер последнего автомата, получившего управление.

7. Переменная $evnt$ хранит имя поступившего на обработку входного события для вложенного автомата, которому было передано управление. Значениями переменной являются имена входных событий всех вложенных автоматов из A и пустое значение 0. Переменная $evnt$ принимает значение 0 тогда, когда входное событие было обработано или же проигнорировано автоматом (если в текущем состоянии автомат не может отреагировать на данное входное событие). В переменную $evnt$ записывается имя некоторого события, если произошла передача управления с этим событием от главного автомата к вложенному.

8. Переменная act используется для хранения имени последнего элементарного действия, произошедшего в системе автоматов A. Значениями переменной act могут быть имена входных событий $a \in E_{\mathrm{A}}$, входных запросов x и $!x$, где $x \in X_{\mathrm{A}}$, выходных воздействий $z \in Z_{\mathrm{A}}$, а также имена специальных действий 0, $nxty$ и end. После перехода активного автомата в новое промежуточное или основное состояние переменная act содержит метку этого промежуточного перехода. Переменная act принимает значение 0, когда вложенный автомат, находясь в основном состоянии, не может обработать входное событие и игнорирует его. При этом автомат совершает пустое действие 0 и переходит в то же самое основное состояние, в котором находился, т. е. автомат остается в прежнем состоянии. Если $act = nxty$, то это означает, что некоторый вложенный автомат перешел в свое следующее основное состояние с одновременной передачей

управления главному автомату. Значение end используется для идентификации тупикового состояния всей системы автоматов. Для того чтобы выполнить условие тотальности отношения переходов для структуры Крипке, устанавливается, что из тупикового состояния (структуры Крипке) есть всего лишь один переход в то же состояние, и при этом совершается действие $act = end$.

9. Переменные ev, xm и zd используются для хранения имен последних выполненных входных событий, входных запросов и выходных воздействий соответственно в рамках всей системы A в целом.

Тогда состоянием s структуры Крипке $\boldsymbol{S}_{\mathrm{A}}$ автоматной модели A является вектор значений переменных

$$(act, auto, evnt, ev, xm, zd, ev_0, xm_0, zd_0, y_0, st_0, \ldots, ev_n, xm_n, zd_n, y_n, st_n).$$

В этом случае состояние s можно рассматривать как отображение, которое задается над множеством переменных и используется для определения значения этих переменных в состоянии s. Например, запись $s(act)$ представляет собой значение переменной act в состоянии s. В начальном состоянии s_0 структуры Крипке $\boldsymbol{S}$ все переменные y_i и st_i содержат начальные состояния соответствующих автоматов A_i, переменная $auto = 0$, т. е. активным является основной автомат A_0, $evnt$ не содержит событий и установлена в 0, а всем остальным переменным присваивается начальное значение $intl$, которое вводится специально для инициализации и далее не используется.

Отношение переходов структуры Крипке $\boldsymbol{S}_{\mathrm{A}}$ автоматной модели A определяется в соответствии с поведением системы автоматов A.

1. Переход системы A соответствует только одному из внутренних переходов некоторого автомата из A.

2. При выполнении условий перехода некоторый автомат A_k может совершить этот переход, если в данном состоянии он является активным (ему передано управление), что отражает значение переменной $auto = k$.

3. Для активного вложенного автомата A_k переход с пометкой e, где e – некоторое входное событие, может произойти, если переменная $evnt = e$ и выполняется условие для этого перехода – истинен предикат над значениями переменных состояний y_i, сопоставленный переходу. После срабатывания этого перехода изменяются значения переменных act, $evnt$, ev, ev_k и st_k. Переменная $evnt$ принимает нулевое значение 0, означающее, что событие было обработано, а переменным act, ev и ev_k присваивается имя события e. В st_k помещается внутреннее состояние, в которое произошел переход. Если активным автоматом является основной автомат A_0, то переход с пометкой e происходит при выполнении условия перехода, а $evnt = 0$.

4. Если вложенный автомат A_k получил от главного автомата управление с событием e, которое находится в переменной $evnt$, но не может на него отреагировать – не существует из данного основного состояния перехода с пометкой e или для него не выполнены условия перехода, то происходит пустое действие $act := 0$, переменная $evnt$ обнуляется и управление передается главному автомату – в переменную $auto$ заносится номер главного автомата, а все остальные переменные не изменяют своих значений.

5. Если произошел переход с пометкой x, где x – некоторый входной запрос в прямой или инверсной форме (x может иметь вид $!x'$), то происходят изменения следующих переменных: $act := x$, $xm := x$, $xm_k := x$, а st_k получает значение внутреннего состояния, в которое был сделан переход.

6. При срабатывании перехода автомата A_k с пометкой z, где z – выходное воздействие, происходят присваивания $act := z$, $zd := z$, $zd_k := z$, и st_k получает значение внутреннего состояния, в которое был сделан переход. Более того, если z принадлежит к выходным воздействиям второго типа, т. е. $z = A_{k'}(e)$, где $A_{k'}$ – вложенный автомат, а e – передаваемое вместе с управлением входное событие для автомата $A_{k'}$, то $evnt$

получает значение e, и активным становится вложенный автомат $A_{k'}$ за счет присваивания $auto := k'$.

7. Если произошел переход с пометкой $nxty$ в основное состояние j во вложенном автомате $A_{k'}$, то происходят присваивания $act := nxty$, $y_{k'} := j$, $st_{k'} := j$ и передача управления главному автомату A_k через $auto := k$. Если переход $nxty$ произошел в основном автомате A_0, то переменная $auto = 0$ остается без изменения.

8. Если из некоторого состояния автомата A_k существует переход, который помечен переменными z, x, $!x$ или $nxty$, то для его срабатывания не требуется дополнительных условий, кроме активности этого автомата: $auto = k$.

9. Если система взаимодействующих автоматов A попала в тупиковое состояние, что равносильно невозможности совершить переход из основного состояния основного автомата A_0 из-за нарушения условий переходов, то происходит специально введенный переход $act := end$, который ведет в то же самое состояние, а значения всех остальных переменных остаются без изменений.

Благодаря построенной структуре Крипке при спецификации и верификации имеется возможность в качестве элементарных высказываний использовать предикаты над значениями введенных переменных, что позволяет выражать следующие свойства состояний автоматных моделей.

1. В каждом состоянии автоматной модели существует возможность отслеживать, какое последнее действие произошло перед попаданием в текущее состояние с помощью переменной act.

2. Имеется возможность определения типа последнего действия, т. е. возможность определить, произошло ли входное событие, входной запрос или выходное воздействие, посредством выражений $act = ev$, $act = xm$ и $act = zd$. Например, если в текущем состоянии автоматной модели выполняется $act = zd$, это означает, что последним действием, которое произошло при переходе в данное состояние, является некоторое выходное воздействие. Уточнить, к какому автомату системы A принадлежит выходное воздействие, можно с помощью выражений вида $act = zd_i$. Наконец, истинность выражения $act = z$ означает, что последним действием при переходе в текущее состояние было выходное воздействие z.

3. Через переменную $auto$ в каждом состоянии автоматной модели определяется активный в данный момент времени автомат. Например, если в состоянии системы A выполняется $auto = 0$, то это означает, что активным является основной автомат A_0.

4. Для каждого автомата A_i в текущем состоянии системы A можно узнать через переменную y_i последнее основное состояние, в котором он пребывал. Более того, выражение $y_i = st_i$ означает, что автомат A_i в данном состоянии системы A находится в своем основном состоянии. Вместо выражения $y_i = st_i$ будем также использовать выражение $y_i == y_i$ с целью минимизации количества переменных, используемых при задании элементарных высказываний. Тогда, если $y_0 = 0$ означает, что последним основным состоянием автомата A_0 было состояние 0, то выражение $y_0 == 0$ показывает, что автомат A_0 в данный момент времени находится в своем основном состоянии 0 – последнее выражение равноценно $y_0 = 0 \& y_0 = st_0$.

5. С помощью выражения $act = end$ можно отслеживать тупиковые состояния системы автоматов A, а с помощью выражения $act = nxty$ отслеживается переход одного из автоматов системы в свое новое основное состояние с одновременной передачей управления главному автомату.

6. ТЕМПОРАЛЬНАЯ ЛОГИКА CTL ДЛЯ АВТОМАТНОЙ МОДЕЛИ

Одними из наиболее популярных темпоральных логик для спецификации и верификации свойств программных систем являются логика CTL (branching-time logic, или computation tree

logic) и логика линейного времени LTL (linear-time logic). Для целей верификации автоматных программ логика LTL заслуживает особого внимания, поскольку любая формула в рамках этой логики по сути представляет собой автомат Бюхи, описывающий (принимающий) бесконечные допустимые пути структуры Крипке, которая, в свою очередь, задает поведение (все возможные исполнения) проверяемой на корректность автоматной программы. Это позволяет при верификации и спецификации автоматных программ оперировать в основном таким простым понятием, как "автомат". Таким образом, можно в некотором смысле говорить, что LTL является более естественным средством спецификации свойств автоматных программ. С другой стороны, логика CTL также широко используется в формальной верификации, и спецификация на языке CTL происходит во многом аналогичным образом. Далее в работе делается попытка оценить удобство использования ("менее естественной" для автоматных моделей) логики CTL для задания темпоральных свойств автоматных программ.

Формулы логики CTL для структуры Крипке $\boldsymbol{S}_{\mathrm{A}}$ автоматной модели A строятся по следующей грамматике:

$$\varphi ::= true|p|\neg\varphi|\varphi\wedge\varphi|\varphi\vee\varphi|$$
$$AX\,\varphi|EX\,\varphi|AYi\,\varphi|EYi\,\varphi|AF\,\varphi|$$
$$EF\,\varphi|AG\,\varphi|EG\,\varphi|A(\varphi U\varphi)|$$
$$E(\varphi U\varphi)|A(\varphi\tilde{U}\varphi)|E(\varphi\tilde{U}\varphi),$$

где $p\in P$ – элементарное высказывание, определенное над множеством состояний автоматной модели A, а i в операторах AYi и EYi представляет собой номер некоторого автомата из системы A.

Отношение выполнимости $\models$ для состояния s структуры Крипке $\boldsymbol{S}_{\mathrm{A}}=(S,s_0,\to,L)$ и формулы φ логики CTL индуктивно определяется следующим образом:

- $s\models true$ и $s\not\models false$;
- $s\models p$ для $p\in P\Leftrightarrow p\in L(s)$;
- $s\models\neg\varphi\Leftrightarrow s\not\models\varphi$;
- $s\models\varphi\wedge\psi\Leftrightarrow s\models\varphi$ и $s\models\psi$;
- $s\models\varphi\vee\psi\Leftrightarrow s\models\varphi$ или $s\models\psi$;
- $s\models EX\,\varphi\Leftrightarrow\exists s'\ s\to s'$ и $s'\models\varphi$ – существует такое следующее состояние структуры Крипке $\boldsymbol{S}_{\mathrm{A}}$, в котором будет выполняться формула φ;
- $s\models AX\,\varphi\Leftrightarrow\forall s'$ из $s\to s'$ следует $s'\models\varphi$ – во всех следующих состояниях структуры $\boldsymbol{S}_{\mathrm{A}}$ должна выполняться формула φ;
- $s\models E(\varphi U\psi)\Leftrightarrow$ для некоторого пути π, выходящего из состояния $s=\pi(0)$, существует такое $j\geq 0$, что $\pi(j)\models\psi$, и при этом для всех i, $0\leq i<j$, $\pi(i)\models\varphi$;
- $s\models A(\varphi U\psi)\Leftrightarrow$ для каждого пути π, выходящего из состояния $s=\pi(0)$, существует такое $j\geq 0$, что $\pi(j)\models\psi$, и при этом для всех i, $0\leq i<j$, $\pi(i)\models\varphi$;
- $s\models EF\,\varphi\Leftrightarrow E(true\,U\,\varphi)$ – из s в структуре Крипке $\boldsymbol{S}_{\mathrm{A}}$ существует путь, проходящий через состояние, в котором выполняется φ;
- $s\models AF\,\varphi\Leftrightarrow A(true\,U\,\varphi)$ – любой путь из состояния s в структуре Крипке $\boldsymbol{S}_{\mathrm{A}}$ обязательно проходит через состояние, в котором выполняется φ;
- $s\models E(\varphi\tilde{U}\psi)\Leftrightarrow$ для некоторого пути π, выходящего из состояния $s=\pi(0)$, для любого $j\geq 1$ такого, что $\pi(j)\not\models\psi$, существует i, $0\leq i<j$, такое, что $\pi(i)\models\varphi$ (для некоторого пути формула ψ истинна и должна оставаться истинной до тех пор, пока φ не станет истинной);
- $s\models A(\varphi\tilde{U}\psi)\Leftrightarrow$ для каждого пути π, выходящего из состояния $s=\pi(0)$, для любого $j\geq 1$ такого, что $\pi(j)\not\models\psi$, существует i, $0\leq i<j$, такое, что $\pi(i)\models\varphi$ (для каждого пути формула ψ истинна и должна оставаться истинной до тех пор, пока φ не станет истинной);
- $s\models EG\varphi\Leftrightarrow E(false\,\tilde{U}\,\varphi)$ – существует путь из состояния s в структуре Крипке $\boldsymbol{S}_{\mathrm{A}}$, на протяжении которого (в каждом состоянии пути) выполняется φ;
- $s\models AG\,\varphi\Leftrightarrow A(false\,\tilde{U}\,\varphi)$ – на протяжении любого пути (в каждом состоянии каждого

пути) из состояния s в структуре Крипке $\boldsymbol{S}_{\mathrm{A}}$ выполняется φ;

- $s \models EY\,i\,\varphi \Leftrightarrow EX\,E(y_i\,!=\,y_i U y_i == y_i \& \varphi)$ – среди непосредственно следующих основных состояний автомата A_i из системы A существует такое, что при попадании в него выполняется формула φ;
- $s \models AY\,i\,\varphi \Leftrightarrow AX\,A(y_i\,!=\,y_i U y_i == y_i \& \varphi)$ – для каждого непосредственно следующего основного состояния автомата A_i из системы A должно выполняться условие, что при попадании в него формула φ является истинной.

Последние два оператора вводятся для того, чтобы более понятно и естественно выражать свойства, относящиеся в большей степени к основным состояниям автоматов A_i из системы A.

Кроме введенных логических связок $\wedge$ и $\vee$, традиционно используют связки $\to$ и $\leftrightarrow$:

- $\varphi \to \psi \equiv \neg\varphi \vee \psi$;
- $\varphi \leftrightarrow \psi \equiv (\varphi \to \psi) \wedge (\psi \to \varphi) \equiv (\neg\varphi \vee \psi) \wedge$ $\wedge(\neg\psi \vee \varphi)$.

Для темпоральных операторов справедливы (вытекают непосредственно из определений) следующие соотношения:

- $AX\,\varphi \equiv \neg EX\neg\varphi$;
- $AF\,\varphi \equiv \neg EG\neg\varphi$;
- $AG\,\varphi \equiv \neg EF\neg\varphi$;
- $A(\varphi \tilde{U} \psi) \equiv \neg E(\neg\varphi U \neg\psi)$;
- $E(\varphi \tilde{U} \psi) \equiv \neg A(\neg\varphi U \neg\psi)$.

Применяя последние два соотношения, можно отказаться от использования в темпоральных формулах операторов $E\tilde{U}$ и $A\tilde{U}$, поскольку их определения довольно сложны, что может привести к трудностям при словесной интерпретации (понимании) формул.

Далее наряду с символами $\wedge$ и $\vee$ будут также использоваться символы & и $|$.

Темпоральное свойство, заданное формулой φ темпоральной логики CTL, считается истинным для структуры Крипке $\boldsymbol{S}_{\mathrm{A}} = (S, s_0, \to, L)$ автоматной модели A тогда и только тогда, когда выполняется $s_0 \models \varphi$.

7. ТЕМПОРАЛЬНОЕ СВОЙСТВА АВТОМАТНЫХ МОДЕЛЕЙ

Рассмотрим несколько примеров темпоральных свойств, которые являются общими для всех конкретных иерархических автоматных моделей (для любой иерархической системы взаимодействующих автоматов).

“Тупиковое состояние”. Свойство, описывающее возможность попадания автоматной программы в тупиковое состояние, из которого нельзя выйти, применимо к любой автоматной программе (к модели любой автоматной программы рассматриваемого типа). В рамках описанной структуры Крипке это свойство может быть описано на языке темпоральной логики CTL следующим образом:

$$EF\ act = end.$$

Эта формула означает, что существует путь из начального состояния s_0 структуры Крипке в такое состояние, из которого произошел переход с пометкой end. По построению структуры Крипке $\boldsymbol{S}_{\mathrm{A}}$ такое состояние является тупиковым, так как переход end добавляется в случае, когда других переходов из данного состояния больше нет. В связи с иерархической структурой автоматной модели тупиковая ситуация происходит лишь тогда, когда нет возможности выйти из основного состояния основного автомата A_0 из-за нарушения условий переходов или вообще при полном отсутствии переходов из данного основного состояния.

“Тупиковое состояние вложенного автомата”. В предыдущем примере тупиковое состояние всей системы было связано с попаданием автомата A_0 в свое основное состояние, из которого нельзя было выйти. В данном случае рассматривается свойство, выражающее наличие тупика во вложенных автоматах. Предположим, что структура Крипке $\boldsymbol{S}_{\mathrm{A}}$ перешла в новое состояние посредством перехода некоторого вложенного автомата A_k из A в свое следующее основное состояние с одновременной передачей управления главному автомату. Далее, пусть на протяжении всех возможных путей из этого состояния структуры Крипке любое входное событие, которое передается вместе с управлением автомату A_k, будет игнорироваться им либо из-за

нарушений условий переходов, либо из-за отсутствия переходов, помеченных этим входным событием. Получается, что вложенный автомат попал в некоторое основное состояние, из которого он больше никогда не выйдет. И это основное состояние в данный момент времени можно назвать тупиковым для автомата A_k, несмотря на то, что в другой ситуации оно может и не быть таковым. Свойство, выражающее, что некоторое основное состояние автомата A_k может стать тупиковым, задается следующим образом:

$EF\ AG(auto = k \to act! = ev_k)$ или
$EF\ AG(auto = k \& evnt! = 0 \to AX\ act = 0)$ или
$EF\ AG(auto = k \& y_k == y_k \to AX\ act = 0)$.

"Забытый автомат". Рассмотрим еще одну разновидность тупикового состояния вложенного автомата. Начиная с некоторого момента времени (с некоторого состояния пути структуры Крипке $\boldsymbol{S}_{\mathrm{A}}$) вложенному автомату A_k никогда не будет передано управление. Таким образом, автомат A_k с некоторого момента времени навсегда останется в одном из своих основных состояний. Это свойство выражается так:

$$EF\ AG\ auto! = k.$$

Далее рассмотрим ряд примеров темпоральных свойств, относящихся к конкретной автоматной модели системы управления кофеваркой (раздел 3).

8. СПЕЦИФИКАЦИЯ ТЕМПОРАЛЬНЫХ СВОЙСТВ АВТОМАТНОЙ МОДЕЛИ СИСТЕМЫ УПРАВЛЕНИЯ КОФЕВАРКОЙ

Удобство и целесообразность применения введенной структуры Крипке и темпоральной логики CTL для спецификации и верификации свойств автоматных моделей продемонстрируем на примере автоматной модели системы управления кофеваркой.

Рассмотрим следующее свойство системы управления кофеваркой, заданное на естественном языке.

"Корректное прерывание работы кофеварки пользователем". Если не сломаны клапаны и нагреватель и кофеварка варит кофе в обычном режиме, то после нажатия кнопки "С" основной автомат управления кофеваркой A_0 переходит в состояние "Прервано пользователем", а все вложенные автоматы обязательно должны вернуться в свои начальные состояния до того, как кофеварка будет запущена вновь.

Перефразируем это свойство, используя элементы автоматной модели.

Если при $y_{31}! = 4$, $y_{32}! = 4$, $y_2! = 4$ и $y_0 == 2$ произойдет событие e_{02}, то автомат A_0 обязательно его обработает (не проигнорирует) и перейдет в состояние $y_0 = 4$, и, когда произойдет событие e_{01} при $y_0 = 0$, система уже будет находиться в состоянии $y_{31} = 0$, $y_{32} = 0$, $y_2 = 0$ и $y_1 = 0$.

Рассмотрим структуру Крипке, которая порождается анализируемой системой взаимодействующих автоматов. Перепишем свойство с применением понятия пути в структуре Крипке автоматной модели.

Для всех путей модели выполнено следующее. Если в состоянии пути истинно выражение $y_{31}! = 4 \& y_{32}! = 4 \& y_2! = 4 \& y_0 == 2$ и в следующий момент времени произойдет событие e_{02}, то это событие обязательно будет обработано и следующим основным состоянием автомата A_0 будет $y_0 == 4$. И когда после этого в будущем автомат A_0 окажется в состоянии $y_0 == 0$ и обработает событие e_{01}, то это будет означать, что система автоматов перед обработкой e_{01} уже находилась в состоянии $y_{31} == 0 \& y_{32} == 0 \& y_2 == == 0 \& y_1 == 0 \& y_0 == 0$.

Заменим все высказывания над путями темпоральными операторами. Получим следующую темпоральную формулу логики CTL для описываемого свойства:

$AG\,(y_{31}! = 4 \& y_{32}! = 4 \& y_2! = 4 \& y_0 == 2 \to$
$\to EX\ act = e_{02} \&$
$\$AX\,(act = e_{02} \to$
$\to AY_0\,(y_0 == 4 \&$
$\&AG\,(y_0 = 0 \& act = e_{01} \to$
$\to y_{31} == 0 \& y_{32} == 0 \& y_2 == 0 \& y_1 == 0))))$.

Рассмотрим и специфицируем аналогично еще ряд свойств для системы управления кофеваркой.

"Определение неисправности". Если произошла поломка нагревателя или одного из клапанов, то кофеварка (основной автомат A_0) обязательно перейдет в состояние "Неисправность".

1. Для каждого пути (или исполнения) структуры Крипке рассматриваемой системы взаимодействующих автоматов должно быть выполнено следующее требование. Если система перейдет в состояние $(y_{31} = 4|y_{32} = = 4|y_2 = 4)\&y_0 = 2$, то следующим отличным от состояния $y_0 = 2$ ("Варит кофе") у автомата A_0 обязательно – рано или поздно, будет состояние $y_0 = 5$.

2. $AG((y_{31} = 4|y_{32} = 4|y_2 = 4)\&y_0 = 2 \rightarrow \rightarrow A(y_0 = 2Uy_0 = 5))$.

"Индикация неисправности". Если основной автомат управления кофеваркой A_0 перешел в состояние "Неисправность", на дисплее обязательно высвечивается сообщение о типе неисправности (не допускается ситуация, при которой неисправность произошла, но пользователя кофеварка не информирует о типе неисправности).

1. Это свойство целесообразно переписать в негативной форме следующим образом. Существует путь в структуре Крипке, который ведет в состояние $y_0 = 5$, и на протяжении этого пути не будет выводиться на дисплей ни одно из сообщений о поломке клапана или нагревателя – не произойдет ни одного из действий $act = z_{35}$ или $act = z_{25}$.

2. $!E((act! = z_{35}\&act! = z_{25})Uy_0 = 5)$.

"Сброс неисправности". Если автомат A_0 находится в состоянии "Неисправность", то после нажатия кнопки "C" все автоматы, включая сам A_0, сбрасываются в начальные состояния, до того как кофеварка будет запущена вновь.

1. На протяжении всех путей структуры Крипке, если в состоянии пути выполняется $y_0 = 5\&act = e_{02}$, то следующим состоянием A_0 в любом случае будет $y_0 = 0$, и обработка события e_{01} при $y_0 = 0$ может произойти лишь тогда, когда выполняется $y_1 = 0\&y_2 = 0\&y_{31} = 0\&y_{32} = 0$.

2. $AG(y_0 = 5\&act = e_{02} \rightarrow AY_0(y_0 = 0\& \&AX(act = e_{01} \rightarrow y_1 = 0\&y_2 = 0\&y_{31} = 0\& \&y_{32} = 0)))$.

"Сброс дисплея после неисправности". Если автомат A_0 после неисправности находится в состоянии "Готов к работе", то на дисплее кофеварки (включая дисплеи бойлера, нагревателя и клапанов) никакие сообщения о поломке не отображаются.

1. Если на дисплей кофеварки (включая дисплеи бойлера, нагревателя и клапанов) было выдано сообщение о неисправности, то до того как автомат A_0 перейдет в начальное состояние $y_0 = 0$, это сообщение должно быть сброшено. В данном случае также целесообразно записать это свойство в негативной форме, ожидая, что указанное условие на некотором пути структуры Крипке нарушается.

2. $!EF((act = z_{35}\&E(act! = z_{36}Uy_0 = 0))|$ $(act = z_{25}\&E(act! = z_{26}Uy_0 = 0))|$ $(act = z_{08}\&E(act! = z_{06}Uy_0 = 0)))$.

"Необходимость наличия воды". Кофеварка никогда не будет варить кофе без воды (соответственно автомат управления бойлером A_1 никогда не перейдет из состояния "Готов к варке следующей порции" в состояние "Кипятит", если воды недостаточно).

1. Перепишем высказывание в негативном виде. Существует путь в структуре Крипке из состояния $y_1 == 2$ в состояние $y_1 = 4$, на протяжении которого не был опрошен датчик наличия воды или результат опроса датчика был отрицательным.

2. $!EF(y_1 == 2\&E(act! = x_{13}Uy_1 = 4))$.

"Недопустимость перегрева нагревателя". Когда нагреватель достигает максимальной температуры, он всегда выключается – не существует такой ситуации (такого бесконечного процесса), при которой во время эксплуатации кофеварки нагревательный элемент будет нагреваться бесконечно долго после преодоления максимального температурного порога.

1. Если кофеварка была запущена, то до ее возврата в начальное состояние не произойдет перегрева нагревательного элемента. Другими словами, во время работы кофеварки не может быть такой ситуации,

при которой автомат A_2, находясь в состоянии $y_2 == 2$ ("Нагреватель включен"), бесконечно долго не имеет возможности проверить свои условия переходов – не может обратиться с запросом к датчику температуры или не получает команды выключения. Такая ситуация возможна в двух случаях. В первом случае основной автомат A_0 попадает в состояние, из которого можно выйти лишь по нажатию кнопки или (по системному таймеру) обратившись с запросом параметров к объекту управления, в то время как автомат A_2 находится в состоянии $y_2 == 2$. Тогда автомат A_2 может потенциально бесконечно долго ожидать сигналов e_0 и e_{22} от автомата A_0 и так их и не дождаться, что приводит к перегреву нагревательного элемента. Во втором случае существует бесконечный путь в структуре Крипке автоматной модели, на протяжении которого автомат A_2 в состоянии $y_2 == 2$ также не получает событий e_0 и e_{22} от основного автомата A_0. Этот случай включает в себя и возможность попадания системой автоматов в тупиковое состояние при условии $y_2 == 2$, так как будет учитываться бесконечный путь, ведущий в тупиковое состояние, и далее проходящий из-за петли $act = end$ только по нему. Важно заметить, что автомат A_0 не содержит каких-либо запросов x.

2. $!EF(y_2 == 2 \& auto = 0 \& y_0 == y_0 \& AX(act != != e_0 \& act! = end)) \& !EF\ EG(y_2 == 2 \& \&(auto = 2 \rightarrow AX(act! = e_0 \& act! = e_{22})))$.

"Начальное состояние". Если автомат A_0 перешел в начальное состояние "Готов к работе", то все вложенные автоматы уже находятся в своих начальных состояниях.

1. Для всех путей структуры Крипке автоматной модели, если автомат A_0 попал в состояние $y_0 == 0$, то все вложенные автоматы находятся в начальных состояниях: $y_1 == 0 \& y_2 == 0 \& y_{31} == 0 \& y_{32} == 0$.

2. $AG(y_0 == 0 \rightarrow y_1 == 0 \& y_2 == 0 \& y_{31} == == 0 \& y_{32} == 0)$.

"Условие реактивности". Система управления кофеваркой никогда не попадет в такое состояние, в котором она никак не реагирует ни на события системного таймера, ни на нажатие кнопок "Ok" и "C".

1. Не существует пути в такое состояние, из которого система не может выйти. Другими словами, система никогда не попадет в дедлок (deadlock) – в тупиковое состояние. Поскольку структура Крипке автоматной модели имеет тотальное отношение переходов, то каждое тупиковое состояние имеет единственный переход само в себя. При этом совершается действие $act = end$.

2. $!(EF\ act = end)$.

Итак, приведенные примеры показывают, что понятные, но в то же время довольно сложно формулируемые на естественном языке свойства системы могут быть успешно заданы в виде формулы логики CTL, что позволяет проверять их автоматическим образом для рассмотренной модели. Отметим, что аналогичные свойства, записанные на языке логики LTL, имеют более простой вид.

9. РЕДУКЦИЯ МОДЕЛИ

Поскольку при автоматической верификации методом проверки модели (методом model checking) применяются переборные алгоритмы, очень важно порождать структуру Крипке автоматной модели таким образом, чтобы она содержала как можно меньше состояний, но попрежнему позволяла бы верифицировать темпоральные свойства автоматной модели.

В данном случае возможно осуществление редукции (уменьшения) введенной структуры Крипке автоматной модели относительно проверяемых темпоральных свойств, заданных формулами логики CTL.

Опишем несколько примеров, когда целесообразно проводить редукцию модели относительно формулы.

Если в темпоральной формуле элементарные высказывания представляют собой предикаты, определенные только с использованием выражений над основными состояниям вида $y_i == k$, то в модели можно не рассматривать промежуточные состояния, которые не имеют отношения

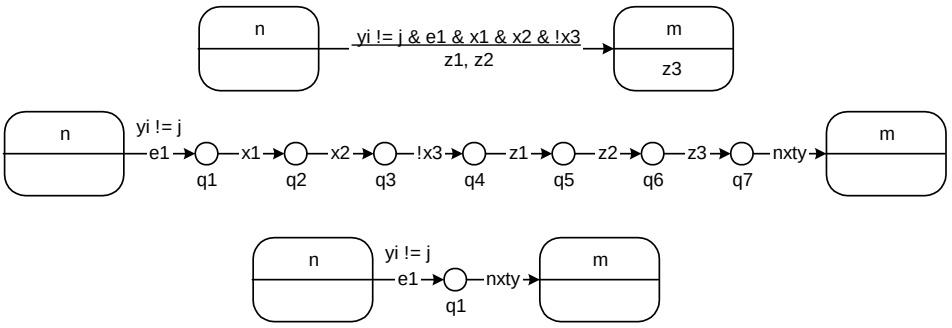

Рис. 8. Пример редукции числа промежуточных состояний в структуре Крипке для автоматных переходов, не содержащих выходных воздействий второго типа.

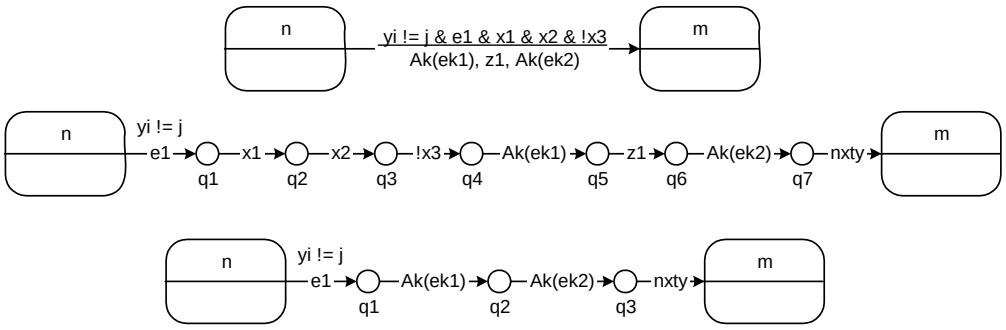

Рис. 9. Пример редукции числа промежуточных состояний в структуре Крипке для автоматных переходов, которые имеют выходные воздействия второго типа.

к выходным воздействиям второго типа (раздел 2) – можно удалить из модели все промежуточные состояния, не связанные с передачей управления между автоматами.

Например, рассмотрим формулу

$$AG(y_0 == 0 \to y_1 == 0 \& y_2 == 0 \& \\ \& y_{31} == 0 \& y_{32} == 0).$$

Если эта формула является истинной (или ложной) для некоторой структуры Крипке автоматной модели, то эта формула останется истинной (или ложной) для редуцированной структуры Крипке, которая получена из исходной с помощью преобразования (редукции), показанного на рис. 8. На рисунке исходный переход автомата не содержит выходных воздействий второго типа, что позволило сократить до одного число промежуточных состояний при переходе между основными состояниями автомата.

На рис. 9 показан пример редукции структуры Крипке в случае, когда во время автоматного перехода происходит передача управления вложенному автомату.

Рассмотрим еще один пример формулы, для которой возможно уменьшение размерности структуры Крипке автоматной модели.

$$EF(act = z_2 \& EX\ E(act != z_2 U y_0 = 0)).$$

Эта формула выражает следующее свойство. Существует путь структуры Крипке, в котором встречается переход, помеченный выходным воздействием z_2, и после этого перехода на протяжении оставшейся части пути до состояния $y_0 = 0$ переход с меткой z_2 больше не встречается. Для этого свойства переходы с пометками, отличными от z_2, не представляют интереса, и их можно обезличить, а идущие подряд несколько обезличенных переходов заменить одним или вообще исключить из структуры Крипке. На рис. 10 приведен пример редукции структуры Крипке в соответствии с указанной формулой для автоматного перехода, не содержащего выходных воздействий второго типа.

Таким образом, после анализа темпоральной формулы возможно построение структуры Крипке автоматной модели с меньшим числом

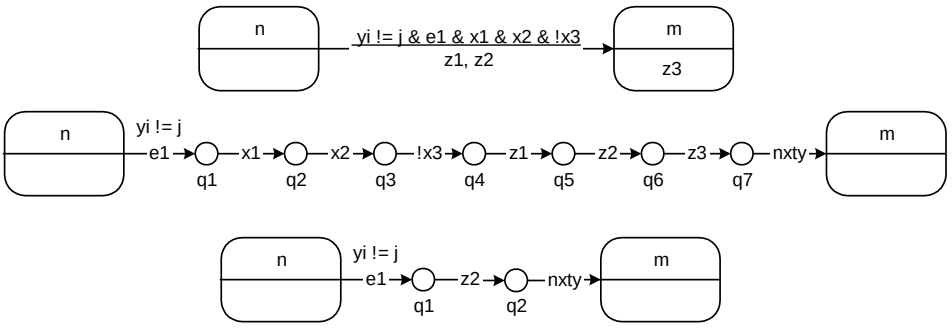

Рис. 10. Редукция числа промежуточных состояний в структуре Крипке относительно формулы $EF(act = z_2 \& EX\ E(act! = z_2 U y_0 = 0))$ для автоматных переходов, не содержащих выходных воздействий второго типа.

состояний, которая будет инвариантной относительно формулы, что позволяет осуществлять проверку свойства автоматной модели, заданного этой формулой, за меньшее время.

В связи с этим представляют интерес классы темпоральных свойств, для которых редукция структуры Крипке автоматной модели является наиболее эффективной.

10. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

Определение структуры Крипке иерархической автоматной модели и описанная спецификация свойств позволяют применить для верификации автоматных программ метод проверки модели. Для этого в дальнейшем целесообразно использовать уже существующие пакеты прикладных программ-верификаторов, которые разрабатываются и поддерживаются ведущими научными лабораториями на протяжении довольно длительного времени (более десяти лет).

Однако проблема состоит в том, что каждый верификатор имеет свой формализм для задания модели и свой способ порождения структуры Крипке для этой модели. Более того, верификаторы имеют и свою модификацию (реализацию) темпоральной логики, которая в ряде случаев может оказаться менее выразительной, чем, например, рассмотренная темпоральная логика CTL.

Возникает серьезная задача адекватного задания структуры Крипке автоматной модели средствами уже существующих верификаторов. Под адекватностью понимается порождение верификатором такой структуры Крипке для заданной формальной модели, которая не допускает потери каких-либо свойств исходной автоматной модели, а также исключает появление не существующих свойств. Должно быть гарантировано, что после проверки свойств для модели, заданной в рамках программы-верификатора, результат этой проверки будет однозначно применим и к исходным свойствам исходной автоматной модели. Кроме того, при выборе верификатора необходимо руководствоваться и выразительной способностью реализованной в нем темпоральной логики, чтобы иметь возможность выражать описанные выше типы свойств автоматных моделей.

Таким образом, после выбора адекватного и выразительного верификатора задача практической реализации автоматической проверки свойств автоматных программ может быть представлена в виде, показанном на рис. 11.

На рис. 11 важной деталью является утилита адекватной трансляции автоматной модели и ее свойства в такие формальную модель и спецификацию, которые допускаются интерфейсом программы-верификатора. В соответствии с требованиями утилита трансляции должна всегда обеспечивать корректное однозначное соответствие между результатом проверки свойства верификатором и истинностью этого свойства автоматной модели.

11. ЗАКЛЮЧЕНИЕ

Автоматная программа является исключительно удобным объектом для верификации методом проверки модели. Свойства программ-

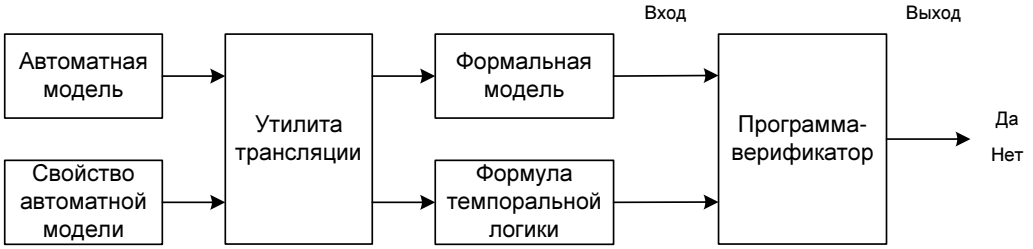

Рис. 11. Применение метода проверки модели для верификации автоматных программ.

ной системы в виде автоматов формулируются и специфицируются естественным и понятным образом, легко соотносятся со взаимодействующими автоматами, которые задают логику "автоматной" программы, так как элементами управляющих автоматов являются либо четко выраженные состояния объекта управления, либо понятные действия над ним. Проверка свойств осуществляется в терминах, которые естественно вытекают из автоматной модели программы. Элементарные высказывания в рамках свойств определяются над элементами модели – событиями, входными и выходными воздействиями и состояниями. Если после верификации методом проверки модели тестирование выявит ошибку, то вид этой ошибки, скорее всего, будет относиться к некорректной программной реализации выходных воздействий, а не к нарушению логики программы, что при исправлении не потребует глобальной перестройки "автоматной" программы (все сведется к локальным исправлениям внутри одной или нескольких отдельных, обычно небольших процедур, корректность которых затем может быть доказана методом дедуктивного анализа).

Разработка формальных методов и технологий моделирования, спецификации и верификации автоматных программ и построения на этой базе интегрированного программного комплекса позволит проектировать и реализовывать надежные программные системы управления ответственными объектами. Для этого целесообразно использовать уже существующие пакеты прикладных программ-верификаторов, которые разрабатываются и поддерживаются ведущими зарубежными и российскими научными лабораториями и центрами. Большинство подобных программных средств верификации предоставляются бесплатно для проведения научно-исследовательских работ. Однако существуют и коммерческие версии программ-верификаторов, позволяющие использовать их для выполнения промышленных заказов как непосредственно, так и в комбинации с другими средствами верификации и анализа программ. Исследования в этом направлении [13–15] привели к созданию прототипа программы верификации под названием "Система моделирования и верификации автоматных программ" (рег. номер свидетельства: 2007611856; авторы: Виноградов Р.А., Кузьмин Е.В., Соколов В.А.).

Важно отметить, что при построении реактивных систем реального времени с использованием автоматного подхода к программированию классического метода проверки модели недостаточно для анализа сложных временных темпоральных свойств (свойств с ограничениями по времени). В этом случае необходимо применять существующие методы и средства верификации систем реального времени, представляющие собой развитие метода проверки модели. Основной акцент при построении и моделировании временных систем ставится на интерпретации времени. При автоматном программировании синхронных систем реального времени целесообразно строить и верифицировать модели с дискретным временем. Спецификация в этом случае осуществляется на языке темпоральной логики реального времени. Расширенная с учетом дискретного реального времени темпоральная логика позволяет выражать такие свойства, как "всегда верно, что за p последует q не позднее чем через 3 единицы времени". Примером такой темпоральной логики является логика RTCTL (real-time CTL), которая используется для спецификации свойств в верификато-

ре VERUS. Непрерывное время, с другой стороны, является естественной моделью для асинхронных систем, поскольку промежуток времени, разделяющий события, может быть сколь угодно мал. Стандартным формализмом для моделирования и анализа асинхронных систем реального времени являются временные автоматы.

Для верификации автоматных программ с использованием моделей на основе временных автоматов могут быть, например, применимы такие программные средства, как UPPAAL [16] и KRONOS [17].

И, наконец, для моделирования автоматных программ могут быть также использованы линейные гибридные автоматы (обобщение временных автоматов). Но в этом случае для решения задач верификации используются лишь частичные алгоритмы. Однако, как указывается в руководстве к системе верификации линейных гибридных автоматов HyTech [18], на практике в большинстве случаев при проверке реальных примеров частичные алгоритмы анализа, применяемые в HyTech, останавливались. Более того, существует широкий класс линейных гибридных систем, для которых итерационные методы и процедуры анализа разрешимы (всегда сходятся). Таким образом, по всей видимости, линейные гибридные автоматы представляют собой граничный "сверху" по выразительности формализм, который еще может быть применим для моделирования и анализа автоматных программ.

Авторы признательны профессору А.А. Шалыто за то, что он привлек их внимание к данной проблематике автоматного программирования.

СПИСОК ЛИТЕРАТУРЫ

1. *Шалыто А.А.* Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998. 628 с. http://is.ifmo.ru/books/switch/1/.
2. *Шалыто А.А.* Автоматное проектирование программ. Алгоритмизация и программирование задач логического управления // Известия академии наук. Теория и системы управления. 2000. №6. С. 63–81. http://is.ifmo.ru.
3. *Шалыто А.А.* Алгоритмизация и программирование для задач логического управления и "реактивных" систем // Автоматика и телемеханика. Обзоры. 2001. №1. С. 3–39. http://is.ifmo.ru.
4. *Шалыто А.А., Туккель Н.И.* Программирование с явным выделением состояний // Мир ПК. 2001. №8. С. 116–121. №9. С. 132–138. http://is.ifmo.ru.
5. *Шалыто А.А., Туккель Н.И.* SWITCH-технология – автоматный подход к созданию программного обеспечения "реактивных" систем // Программирование. 2001. №5. С. 45–62.
6. *Грис Д.* Наука программирования. Пер. с англ. М.: Мир, 1984.
7. *Кларк Э.М., Грамберг О., Пелед Д.* Верификация моделей программ: Model Checking. М.: МЦНМО, 2002.
8. CPNTools. http://www.daimi.au.dk/CPNTools/.
9. SPIN. http://spinroot.com/spin/whatispin.html.
10. SMV. Symbolic Model Verificator. Carnegie Mellon University. http://www.cs.cmu.edu/ modelcheck/smv.html.
11. CADP. Construction and Analysis of Distributed Processes. http://www.inrialpes.fr/vasy/cadp/.
12. *Кессель С.В.* Разработка системы управления кофеваркой на основе автоматного подхода // СПбГУ ИТМО, 2003. http://is.ifmo.ru/projects/coffee2/.
13. *Виноградов Р.А., Кузьмин Е.В., Соколов В.А.* Верификация автоматных программ средствами CPN/Tools // Моделирование и анализ информационных систем. Ярославль: ЯрГУ, 2006. Т. 13. №2. С. 4–15.
14. *Кузьмин Е.В., Соколов В.А.* О некоторых подходах к верификации автоматных программ // Сборник докладов семинара Go4IT – шаг к новым технологиям Интернета. Москва. Институт системного программирования РАН. 2007. С. 43–48.
15. *Кузьмин Е.В., Васильева К.А.* Верификация автоматных программ с использованием LTL // Моделирование и анализ информационных систем. Ярославль: ЯрГУ, 2007. Т. 14. №1. С. 3–14.
16. UPPAAL. http://www.uppaal.com.
17. KRONOS. http://www-verimag.imag.fr/TEMPORISE/kronos.
18. HyTech. http://embedded.eecs.berkeley.edu/research/hytech/.