Актуальные проблемы математики и информатики. Сборник статей к 20-летию факультета ИВТ ЯрГУ им. П.Г. Демидова, Ярославль: ЯрГУ, 2006. С. 27-32.

О верификации «автоматных» программ

Е. В. Кузьмин, В. А. Соколов

Ярославский государственный университет им. П.Г. Демидова

В последние годы в Западной Европе при построении промышленных систем широко применяется синхронное программирование. Несмотря на популярность и повсеместное использование универсальных языков программирования, таких как C++ и Java, активно проводятся работы по созданию специализированных языков и технологий, предназначенных, главным образом, для реализации программных систем управления ответственными объектами. Ряд языков и технологий, таких как Lustre, Esterel, SyncCharts, Argos, тесно связанных с понятием конечного автомата и оперирующих им в декларативной или графической форме, объединяются под общим названием «синхронное программирование». Эта технология программирования находит применение и активно внедряется в области точного приборостроения, при производстве авиатехники, в судостроении и т. д.

С 1991 года в России под руководством профессора А. А. Шалыто (заведующего кафедрой технологии программирования СПГУ ИТМО) для указанного класса систем развивается технология программирования с явным выделением состояний (SWITCH-технология), которая в качестве языка спецификации использует графы переходов (взаимодействующие событийные автоматы Мили–Мура). SWITCH-технология, или «автоматное» программирование, представляет собой разновидность синхронного программирования, особенностью которой является использование универсальных языков. Технология автоматного программирования является достаточно эффективной при построении программного обеспечения для «реактивных» систем и систем логического управления. Эта технология, не исключая других методов построения программного обеспечения «без ошибок», существенно более конструктивна, так как позволяет начинать «борьбу с ошибками» еще на стадии алгоритмизации [1, 2, 3, 4, 7].

В Ярославском госуниверситете им. П. Г. Демидова на кафедре теоретической информатики «автоматный» подход к программированию (как и само синхронное программирование) был выбран как наиболее подходящий с точки зрения анализа программной корректности. И в последнее время в рамках общего направления «Моделирование и анализ

информационных систем» проводятся исследования по моделированию, спецификации и верификации «автоматных» программ. Несмотря на то, что сама идея синхронного и «автоматного» программирования направлена на построение надёжных программ, задача проверки их правильности по-прежнему остаётся актуальной. Причём, если в промышленных языках синхронного программирования эта проблема решается в «жёсткой» привязке к компилятору, то в «автоматном» программировании, как языково-неориентированной технологии, в большей степени вопрос остаётся открытым.

При автоматном подходе к проектированию и построению программ выделяются две части: системно независимая и системно зависимая. Первая часть реализует логику программы и задаётся 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 (и затем реализована системно зависимая часть), соответствующий автомату.

Интересным является тот факт, что к «автоматным» программам могут быть успешно применимы (в совокупности) все существующие методы анализа корректности: тестирование, метод доказательства теорем и метод проверки модели. *Тестирование* применяется после окончательного написания программы. Но, как известно (Э. Дейкстра), если при тестировании ошибки найдены не были, это ещё не означает, что их нет вовсе. *Метод доказательства теорем* [5] представляется очень трудоёмким методом с сильной привязкой к семантике языка программирования. Но при желании, вполне может быть применим для «автоматных» программ после их полного построения, непосредственно для проверки корректности процедур, соответствующих выходным воздействиям или входным воздействиям типа «запрос». 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 доказательства теорем. Но этот метод оказывается бесполезным при проверке логики программы. С другой стороны для проверки логики «автоматных» программ идеально подходит *метод проверки модели* (model checking) [6, 8]. При этом методе для программы строится формальная конечная модель (т. е. с конечным числом состояний), а проверяемые свойства задаются с помощью формул темпоральной логики. Проверка выполнимости темпоральных формул, задающих свойства модели, происходит автоматическим образом.

Автоматный подход к программированию с точки зрения моделирования и анализа программных систем имеет ряд преимуществ по сравнению с традиционным подходом. При построении модели для программы, написанной традиционным способом, возникает серьёзная проблема адекватности этой программной модели исходной программе. Модель может не учитывать ряд программных свойств или порождать несуществующие свойства. При автоматном программировании такая проблема исключена, поскольку набор взаимодействующих автоматов, описывающий логику программы, уже является адекватной моделью, по которой формально и изоморфно 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 model checking представляет собой переборный метод. К тому же свойства программной системы в виде автоматов формулируются и специфицируются естественным и понятным образом, легко соотносятся со взаимодействующими автоматами, которые задают логику «автоматной» программы. Если в качестве примеров рассмотреть системы управления кофеваркой, банкоматом или лифтом, можно привести такие свойства, как «кофеварка не может варить кофе без воды», «нагревательный элемент кофеварки не должен перегреваться», «кабина лифта не должна двигаться при открытых дверях» или «при нормальной работе банкомат вернёт пользователю карту, если она была вставлена». Все эти свойства легко соотносятся с автоматами управления и без особого труда задаются с помощью темпоральных логик (таких, как CTL и LTL), так как элементами управляющих автоматов являются либо чётко 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.

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 задаёт поведение (все возможные исполнения) проверяемой на корректность «автоматной» программы («автоматной» модели). Что позволяет при спецификации и верифи-

кации «автоматных» программ оперировать в основном таким простым понятием, как «автомат».

Таким образом, перечисленное выше позволяет говорить об эффективном применении для верификации «автоматных» программ (для анализа корректности логики «автоматных» программ) метод проверки модели. Для этого целесообразно использовать уже существующие пакеты прикладных программ-верификаторов, которые разрабатываются и поддерживаются ведущими научными лабораториями и центрами 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, темпоральные логики CTL и LTL.

Возникает серьёзная 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 потерю 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 нём темпоральной логики, чтобы иметь возможность выражать различные типы свойств «автоматных» моделей.

Одним из наиболее подходящих средств верификации является система SPIN, разрабатываемая в лаборатории Bell. SPIN – это система верификации моделей для логики LTL на лету с использованием явного перечисления состояний и редукции частичных порядков, представляющее собой инструментальное средство, которое используется главным образом для верификации асинхронных программных систем и, в частности, коммуникационных протоколов. Входной язык системы SPIN для описания моделей и спецификации свойств носит название Promela. В состав этого языка входят синтаксические конструкции нескольких различных языков программирования. Логические и арифметические выражения языка Promela унаследованы от языка C. Синтаксис канального взаимодействия процессов ориентирован на CSP (communicating sequential processes) Хоара. Условные операторы и операторы цикла основаны на охраняемых командах Дейкстры. Язык Promela ориентирован на опи-

сание автоматных конструкций, и его гибкость позволяет адекватным образом задавать поведение «автоматной» модели.

Из всего указанного выше можно заключить, что «автоматная» программа является исключительно удобным объектом для верификации. В частности, если после верификации методом проверки модели тестирование выявляет ошибку, то вид этой ошибки будет относиться к некорректной программной реализации выходных воздействий, а не к нарушению логики программы, что при исправлении не потребует глобальной перестройки «автоматной» программы (всё сведётся к локальным исправлениям внутри некоторой отдельной процедуры).

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, «Статьи»).

[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, раздел «Статьи»).

[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, раздел «Статьи»).

[5] *Грис, Д.* Наука программирования / Д. Грис; пер. с англ. — М.: Мир, 1984. — 416 с.

[6] *Кларк Э. М.*, *Грамберг О.*, *Пелед Д.* Верификация моделей программ: Model Checking. — М.: МЦНМО, 2002. — 416 с.

[7] *Кузьмин Е. В.* Иерархическая модель автоматных программ // Моделирование и анализ информационных систем. Т.13, 1 (2006). — ЯрГУ, 2006. — с. 27–34.

[8] *Кузьмин Е. В., Соколов В. А.* Структурированные системы переходов. — М.: ФИЗМАТЛИТ, 2006. — 178 с.