**Опубликовано в материалах XV Международной научно-методической конференции «Высокие интеллектуальные технологии и инновации в образовании и науке». СПбГ ПУ. 2008, с. 285–288.**

**С. Э. Вельдер, А. А. Шалыто**

# ВЕРИФИКАЦИЯ ПРОСТЫХ АВТОМАТНЫХ ПРОГРАММ НА ОСНОВЕ МЕТОДА *MODEL CHECKING*

*Верификация моделей* – это набор идей и методов для построения моделей работающих программ, математической формулировки требований к ним, отражающих правильность их работы (называемых также *спецификацией*), и создания алгоритмов для формальной проверки (доказательства или опровержения) этих требований.

*Model checking* – это автоматизированный подход, позволяющий для заданной *модели поведения* системы с конечным (возможно, очень большим) числом состояний и *логического свойства* (*требования*) проверить, выполняется ли это свойство в рассматриваемых состояниях данной модели.

В проблематике верификации сформировалось два направления: аксиоматическое и алгоритмическое. При первом из них разрабатывается набор аксиом, с помощью которого может быть описана как сама система, так и ее свойства (эти свойства записываются в виде пред- и постусловий). *Model checking* составляет основу второго направления.

Качество программ может быть обеспечено за счет формальной верификации, но для программы произвольного вида процесс верификации трудно автоматизировать. В то же время, для автоматных программ [1] этот вопрос в значительной мере может быть решен.

Класс автоматных программ является наиболее удобным для верификации таким подходом [2], так как в этом случае модель программы может быть формально построена по спецификации ее поведения, задаваемой в общем случае

системой взаимодействующих конечных автоматов, в то время как для программ других классов модель приходится строить эвристически.

Структура автоматных программ, в которых функции входных и выходных воздействий почти полностью отделены от логики программ, делает практичным верификацию этих функций на основе формальных доказательств с использованием пред- и постусловий.

Первым шагом в процессе верификации автоматной программы является преобразование графа переходов исходного автомата в *модель Крипке* (структуру Крипке), для которой удобно формулировать проверяемые свойства программ.

Использование структуры Крипке предполагает применение темпоральной логики для записи требований. В настоящей работе требования описываются на языке темпоральной логики *CTL*.

Собственно верификация модели выполняется по структуре Крипке и требованиям, записанным в виде формулы на языке темпоральной логики. Она осуществляется в два этапа. Первый из них предназначен для определения набора состояний в структуре Крипке, в которых выполняется заданное формулой требование, а второй – по заданному исходному состоянию и подформуле заданного требования на основе построенного набора состояний формирует сценарий (путь, представляющий собой последовательность состояний), который подтверждает или опровергает эту подформулу.

После этого сценарий, построенный для модели Крипке, должен быть преобразован в сценарий для исходного автомата.

Исследования в области моделирования автомата (преобразования его в структуру Крипке) проводились в работах [3–6]. Основной проблемой при таком преобразовании являлась неоднозначность семантической интерпретации формулы языка *CTL*, записанной для модели Крипке, в исходном автомате. Для ее решения применялась модификация семантики языка *CTL*.

В данной работе основное внимание уделяется моделированию автомата и преобразованию сценария в модели Крипке, полученного в результате вери-

фикации, в сценарий в исходном автомате. Под сценарием в исходном автомате подразумевается уже не просто последовательность состояний, а еще и события, которые произошли при переходах между этими состояниями, значения булевских входных переменных, которые имели место на каждом переходе, и последовательность выходных управляющих воздействий, которые были вызваны на этом переходе.

Методы моделирования, рассматриваемые в настоящей работе, позволяют однозначно преобразовывать пути, построенные в качестве сценариев для *CTL*-формул, из модели Крипке в автомат. Это особенно полезно в случаях, когда моделирование проводится совместно с исполнением автомата, его визуализацией и отладкой в инструментальных средствах для поддержки автоматного программирования.

Модель Крипке представляет собой набор состояний и переходов между ними, причем каждому состоянию соответствует некоторое множество *атомарных предложений*. В работе построены несколько различных методов генерации множества атомарных предложений автоматной программы и преобразования автомата с булевыми входными переменными в модель Крипке.

Выделим несколько основных методов такого преобразования:

1. Метод атомарных переходов.
2. Установка состояний на событиях и выходных воздействиях.
3. Создание полного графа переходов.
4. Редукция полного графа переходов с внесением тесных отрицаний внутрь атомарной формулы.

Кратко опишем преобразование автомата Мили в модель Крипке по каждому из этих методов.

В *первом методе* у автомата «стираются» все события и переменные и оставляются 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. Кроме того, комбинаторные свойства у модели,

построенной по данному методу, не отличаются от таковых у исходного автомата. Следовательно, структура Крипке для композиции автоматов не отличается от композиции структур Крипке отдельных автоматов.

Во *втором методе* у автомата стираются только входные воздействия. Каждый переход разбивается на блоки, соответствующие событиям и выходным воздействиям, расположенным друг за другом. Данный метод уже позволяет упоминать события и выходные воздействия в формулах темпоральной логики.

В *третьем методе* для автомата строится модель, в которой для каждого события сформирована полная система переходов. Полученная модель имеет большой размер, но в ней можно проверять любые темпоральные свойства.

В *четвертом методе* переходы раскладываются на атомарные блоки из событий и выходных воздействий, а входные воздействия специальным образом в них сохранены. Можно сказать, что *второй метод* является его упрощением (абстракцией). Метод редукции строит маленькие модели (линейного размера по отношению к размеру исходного автомата) и позволяет проверять практически все свойства, которые можно сформулировать относительно состояний, событий и входных и выходных воздействий. Он является наилучшим по соотношению эффективность-выразительность. При этом заметим, что модели, построенные по одному методу, отличаются от моделей, построенных по другому методу, только тем, что в них отождествляются некоторые характеристики. Такая аналогия позволяет *абстрагировать* различные модели.

Для удобства и эффективности в построенных методах в состояния модели Крипке добавлены «управляющие» атомарные предложения для состояний модели, построенных, соответственно, из состояний, событий и выходных воздействий исходного автомата. Это сделано для того, чтобы при записи формулы в темпоральной логике можно было различать тип исследуемого состояния.

Идеи этих методов (для более узкого класса задач) были описаны в работе [7]. В настоящей работе строго выделяется набор методов моделирования для соответствующих типов автоматов. В частности, *четвертый метод* (редукция графа переходов) теперь позволяет автоматизировать не только моделирование

автомата, но и преобразование требований для автомата в требования, сформулированные для модели Крипке.

**Источники**


1. *Шалыто А. А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998. http://is.ifmo.ru/books/switch/1
2. *Кузьмин Е. В., Соколов В. А.* О дисциплине специализации «Верификация программ» /Доклады II научно-методической конференции преподавателей математического факультета и факультета ИВТ Ярославского государственного университета им. П. Г. Демидова «Преподавание математики и компьютерных наук в классическом университете». Ярославль: ЯрГУ. 2007, с. 91–101. http://is.ifmo.ru/verification/_ver_prog.pdf
3. *Sebastiani R.* Introduction to Formal Methods, 2005–2006. http://dit.unitn.it/~rseba/DIDATTICA/fm2005/02_transition_systems.pdf
4. *Margaria T.* Model Structures. Service Engineering – SS 06. https://www.cs.uni-potsdam.de/sse/teaching/ss06/sveg/ps/2-ServEng-Model-Structures.pdf
5. *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
6. *Hull R.* Web Services Composition: A Story of Models, Automata and Logics. Bell Labs, Lucent Technologies, 2004. http://edbtss04.dia.uniroma3.it/Hull.pdf
7. *Вельдер С. Э., Шалыто А. А.* О верификации простых автоматных программ на основе метода "Model Checking" //Информационно-управляющие системы. 2007. № 3, с. 27–38. http://is.ifmo.ru/download/27-38.pdf