МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ОПТИКИ АТМОСФЕРЫ СО РАН им. В.Е. ЗУЕВА

НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ИССЛЕДОВАНИИ СЛОЖНЫХ СТРУКТУР

МАТЕРИАЛЫ
ДВЕНАДЦАТОЙ КОНФЕРЕНЦИИ С МЕЖДУНАРОДНЫМ УЧАСТИЕМ
4–8 июня 2018 г.

Мероприятие проведено при финансовой поддержке
Российского фонда фундаментальных исследований (проект № 18-07-20033)

Томск
Издательский Дом Томского государственного университета
2018

линии, которые получаются дублированием линий связи, идущих в элемент, из которого исходит «подозрительная» линия. Опишем способ программирования LUT. Пусть элемент, в который ведёт «подозрительная» линия, реализует функцию $f(x_1, x_2, \ldots, x_n)$, а элемент, из которого исходит «подозрительная» линия, реализует функцию $g(y_1, y_2, \ldots, y_l)$. Пусть «подозрительная» линия соединена с входом x_i элемента, реализующего функцию f, тогда в LUT программируется следующая функция: $f(x_1, x_2, \ldots, x_{i-1}, g(y_1, y_2, \ldots, y_l), x_{i+1}, \ldots, x_n)$. Новая функция зависит от $(l + n - 1)$ переменной. Если m – максимальное число входов вентилей в схеме, то для обеспечения покрытия элемента программируемым блоком должно выполняться следующее условие:

$$l + n - 1 \le m + 1.$$

Рассмотрим пример, иллюстрирующий работу алгоритма (рис. 1).

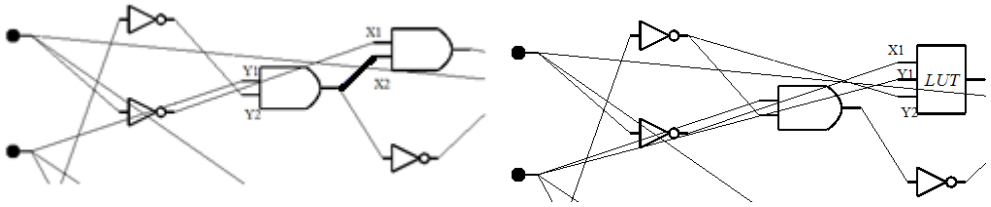

Рис 1. Покрытие LUT'ом двухвходового элемента

Здесь «подозрительная» линия исходит из элемента И. элемент, к которому ведёт эта линия, заменяется программируемым блоком, к нему подводятся все линии (кроме исключаемой), который вели в замещаемый элемент, а также линии, которые вели в элемент И (здесь новые линии получаются путём дублирования). В блоке памяти для рассматриваемого примера программируется функция:

$$f = f(x_1, x_2) = x_1 \wedge x_2 = x_1 \wedge g(y_1, y_2) = x_1 \wedge (y_1 \wedge y_2).$$

Литература

1. *Матросова А.Ю., Останин С.А., Николаева Е.А.* Синтез частично программируемых схем, ориентированный на маскирование вредоносных подсхем (Trojan Circuits) // Труды Института системного программирования РАН. 2017. Т. 29, № 5. С. 61–74.
2. *Jo S., Matsumoto T., Fujita M.* SAT-based automatic rectification and debugging of combinational circuits with LUT insertions // Proc. Of IEEE Asian Test Symposium. 2012. P. 19–24.

К ПОСТРОЕНИЮ ПАРАЛЛЕЛЬНОЙ КОМПОЗИЦИИ РАСШИРЕННЫХ АВТОМАТОВ*

Е.В. Дарусенкова, С.А. Прокопенко, Н.В. Шабалдина

Национальный исследовательский Томский государственный университет, Томск, Россия
darusenkova@gmail.com, s.prokopenko@sibmail.com, nataliamailbox@mail.ru

В работе рассматривается задача описания взаимодействия компонент веб-сервисов, которые, помимо общения друг с другом, также взаимодействуют и с внешней средой (рисунок 1). Адекватной математической моделью для описания веб-сервисов является расширенный автомат [1], и взаимодействие компонент также хотелось бы описать расширенным автоматом. В случаях, когда области значений входных/выходных параметров и контекстных переменных конечны, мы можем полностью промоделировать поведение расширенного автомата и перейти к хорошо изученной модели конечного автомата [2]. Однако, если данные области значений слишком велики или бесконечны, то построить конечный автомат не представляется возможным. В этом случае можно промоделировать поведение расширенного автомата на входных последовательностях длины не больше заданного числа l (так называемый l-эквивалент, который также является конечным автоматом) [3].

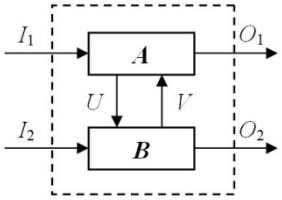

Рис. 1. Параллельная композиция расширенных автоматов

* Работа выполнена при поддержке Российского научного фонда, проект № 16-49-03012.

Методы построения композиции конечных автоматов известны [4, 5], однако вопрос, каким образом вернуться от композиции конечных автоматов к расширенному автомату, описывающему поведение композиции, остается открытым. Кроме того, непросто установить обратную связь между переходами в конечном автомате (в частности, *l*-эквиваленте) и соответствующими им переходами в расширенном автомате. Поэтому хотелось бы обсудить возможность описания взаимодействия компонент без перехода к эквивалентным конечным автоматам.

В работе [6] композиция в алфавитах I_1, U, O_2, являющаяся частным случаем композиции на рис. 1, описана расширенным автоматом и сформулированы ограничения, при которых такое описание возможно. Аналогичным образом хотелось бы построить расширенный автомат для композиции в алфавитах I_1, O_1, U, V и обозначить условия существования такой композиции. В отличие от предыдущего вида композиции, где компоненты следуют одна за другой, и отсутствует внутренний диалог между компонентами, во втором типе композиции возможен внутренний диалог между компонентами, и они взаимодействуют в режиме так называемой «медленной внешней среды» [4]. Данный режим подразумевает, что следующий внешний входной символ может быть подан на композицию только после того, как она выдала внешний выходной символ на предыдущий входной символ.

Прежде всего, для построения параллельной композиции расширенных автоматов мы ограничиваемся случаем, когда предикаты переходов компоненты A не зависят от выходных параметров компоненты B, предикаты переходов компоненты B не зависят от входных параметров компоненты A и выходные параметры компоненты A разделены на внешние (для алфавита O_1) и внутренние (для алфавита U). В дальнейшем планируется ослабить ограничения на класс расширенных автоматов, допускающих композицию без перехода к эквивалентным конечным автоматам.

Литература

1. *Petrenko A.*, *Boroday S.*, *Groz R.* Confirming configurations in EFSM testing // IEEE Trans. Software Eng. 2004. № 30(1). P. 29–42.
2. *Гилл А.* Введение в теорию конечных автоматов. М. : Наука, 1966. 272 с.
3. *Карибский В.В.*, *Пархоменко П.П.*, *Согомонян Е.С.*, *Халчев В.Ф.* Основы технической диагностики. М. : Энергия, 1976. 464 с.
4. *Евтушенко Н.В.*, *Рекун М.В.*, *Тихомирова С.В.* Недетерминированные автоматы: анализ и синтез. Ч. 2: Решение автоматных уравнений : учеб. пособие. Томск : Томский государственный университет, 2009. 111 с.
5. *Villa T.*, *Yevtushenko N.*, *Mishenko A.*, *Brayton R. K.*, *Petrenko A.*, *Sangiovanni-Vincentelli A.* The unknown component problem: theory and applications. Berlin : Springer, 2012. 311 p.
6. *Prokopenko S.* Locating a faulty component of an EFSM composition // The Proceedings of ISP RAS. 2014. Vol. 26, is. 6. P. 47–55.

АВТОМАТИЧЕСКАЯ ГЕНЕРАЦИЯ ТЕСТОВЫХ ШАБЛОНОВ НА ОСНОВЕ СПЕЦИФИКАЦИЙ СИСТЕМЫ КОМАНД

А.С. Проценко, А.Д. Татарников

Институт системного программирования им. В.П. Иванникова РАН, Москва, Россия
protsenko@ispras.ru, andrewt@ispras.ru

Современный микропроцессор является устройством с высокой сложностью, что в итоге приводит к наличию большого количества ошибок на этапе его разработки. Поэтому немалое количество ресурсов выделяется для верификации создаваемого продукта. В процессе верификации микропроцессора использование спецификации модели микропроцессора, описанной на формальном языке, позволяет сократить трудоемкость и повысить качество верификации. Используя описанные таким образом спецификации микропроцессора, можно получить метаданные модели микропроцессора, такие как набор инструкций, их сигнатуру и диапазон значений входных данных. На основе таких данных становится возможно автоматически сгенерировать тестовые шаблоны, описывающие сценарий верификации микропроцессора и отвечающие особым, нужным верификатору, требованиям.

Рассмотрим публикации по описанной нами тематике. В работе [2] рассмотрен инструмент AVS (Architecture Validation Suite), который реализует проверку покрытия путей исполнения команд. TVM (Test Virtual Machines) [3] для архитектуры RISC-V отделяет общие элементы тестов от блоков для команд или версий спецификаций. В статье [4] представлен подход, где в качестве тестов используют загрузку операционной системы (ОС) и встроенные в нее проверки. На сайте компании ARM [5] описан инструмент RIS (Random Instruction Sequence), он осуществляет проверку работы механизмов управления памятью и многоядерности.

Предлагаемый в нашей работе подход реализован в инструменте MicroTESK[6], рис. 1.

В работе для генерации тестовых шаблонов используются несколько параметров, и каждый из шаблонов направлен на решение определенной задачи: а) тестовый шаблон для проверки соответствия версии специфи-