МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АНГАРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И МАТЕМАТИЧЕСКОЙ ГЕОФИЗИКИ СО РАН

НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ИССЛЕДОВАНИИ СЛОЖНЫХ СТРУКТУР

МАТЕРИАЛЫ ЧЕТЫРНАДЦАТОЙ МЕЖДУНАРОДНОЙ КОНФЕРЕНЦИИ 19–24 сентября 2022 г.

Томск
Издательский Дом Томского государственного университета
2022

Экспериментальные результаты показали, что рассматриваемая клиентская реализация не накапливает информации о предыдущем сетевом взаимодействии, что, соответственно, не позволяет избежать избыточного сетевого взаимодействия клиентской и серверной частей. В дальнейшем планируется формализовать выявление лишнего сетевого взаимодействия путем анализа формальных моделей.

Литература

1. *J. Klensin.* Simple Mail Transfer Protocol [Электронный ресурс] // RFC Editor. – URL: https://www.rfc-editor.org/rfc/rfc5321.html (дата обращения: 06.07.2022).
2. enough_mail 2.1.1. IMAP, POP3 and SMTP clients for Dart and Flutter email developers [Электронный ресурс] // pub.dev. The official package repository for Dart and Flutter apps. – URL: https://pub.dev/packages/enough_mail (дата обращения: 06.07.2022).

ПРОВЕРКА СВОЙСТВА РЕАКТИВНОСТИ В DART-РЕАЛИЗАЦИИ SMTP-КЛИЕНТА

А.П. Сотников, М.Л. Громов, С.А. Прокопенко, Н.В. Шабалдина

Томский государственный университет, Томск, Россия
sotnikhtc@gmail.com

CHECKING INPUT-OUTPUT BEHAVIOUR OF THE SMTP-CLIENT DART IMPLEMENTATION

A.P. Sotnikov, M.L. Gromov, S.A. Prokopenko, N.V. Shabaldina

Tomsk State University, Tomsk, Russia

При работе клиент-серверных приложений в некоторых случаях важным является сохранение свойства реактивности, которое подразумевает, что следующий запрос от клиента к серверу отправляется после того, как клиент получил ответ на предыдущий запрос. Например, в SMTP протоколе [1] необходимо дождаться ответа на команду (запрос) EHLO перед тем, как продолжать взаимодействие с сервером. Это связано с тем, что сервер в ответе на данную команду указывает поддерживаемые им механизмы аутентификации и прочую информацию, которая нужна для организации дальнейшего взаимодействия. Кроме того, сохранение взаимодействующими реализациями свойства реактивности позволяет описать их поведение (в том числе, совместное) при помощи модели с конечным числом переходов, например, конечного автомата [2], что может быть полезно для анализа системы и синтеза тестов.

В данной работе мы проверяем сохранение свойства реактивности в Dart-реализации клиентской части SMTP. В языке Dart есть механизм асинхронного выполнения функций, необходимый, например, для работы графического интерфейса без блокирования элементов интерфейса. Асинхронные функции присутствуют практически в любой реализации клиентской части, и при некорректном их использовании могут привести к нарушению свойства реактивности при взаимодействии с сервером. Поэтому актуальным является наличие в клиентской реализации проверки, что свойство реактивности не нарушается. Убедиться в том, что такая проверка осуществляется, можно с помощью юнит-теста, в котором команды вызываются без ключевого слова await или использования метода .then. Отсутствие await и .then как раз позволяет смоделировать ситуацию нарушения свойства реактивности за счет того, что следующая команда юнит-теста отправляется без ожидания ответа на предыдущую команду.

В качестве исследуемой программы для эксперимента по проверке свойства реактивности была выбрана Dart-реализация клиентской части SMTP из работы [3]. Написана программа-сервер, отвечающая на запросы клиента с фиксированной задержкой, что имитирует работу реальной сети. План эксперимента следующий. Используем экземпляр класса клиента SMTP. С помощью вызовов методов этого класса добьёмся того, чтобы на сервер была отправлена последовательность команд, в которой очередная команда должна по-хорошему подаваться в зависимости от ответа сервера на предыдущую команду. Методы будем вызывать без использования ключевого слова `await` или использования метода `.then`. Если на очередной вызов метода до прихода ответа на предыдущие команды клиент ответит отказом (выбросит исключение или вернёт ошибку), то рассматриваемая реализация клиентской части следит за сохранением реактивности, иначе нет.

В эксперименте была выбрана следующая последовательность. Клиент сначала посылает на сервер команду EHLO. Напомним, что ответ на эту команду содержит, в том числе, поддерживаемые сервером механизмы аутентификации. Далее клиент пытается аутентифицироваться на сервере, указывая неподдерживаемый сервером механизм аутентификации, после чего посылает запрос на отправку письма. Подчеркнем, что в юнит-тесте все команды клиента намеренно отправлялись одна за другой (без анализа ответов сервера). Сервер ответил ошибкой на попытку аутентификации с неподдерживаемым механизмом, но к

этому времени уже была отправлена следующая команда; то есть все команды успешно отправились на сервер, но на самом деле клиент не прошёл аутентификацию, и письмо не было отправлено. Таким образом, Dart-реализация [3] не содержит проверки сохранения свойства реактивности. В дальнейшем планируется формализовать проверку свойства реактивности с использованием моделей с конечным числом переходов.

Литература

1. *J. Klensin.* Simple Mail Transfer Protocol [Электронный ресурс] // RFC Editor. – URL: https://www.rfc-editor.org/rfc/rfc5321.html (дата обращения: 06.07.2022).
2. *Евтушенко Н.В., Петренко А.Ф., Ветрова М.В.* Недетерминированные автоматы: анализ и синтез. Ч. 1. Отношения и операции: учеб. пособие. – Томск: Томский государственный университет, 2006. – 142 с.
3. enough_mail 2.1.1. IMAP, POP3 and SMTP clients for Dart and Flutter email developers [Электронный ресурс] // pub.dev. The official package repository for Dart and Flutter apps. – URL: https://pub.dev/packages/enough_mail (дата обращения: 06.07.2022).

АЛГОРИТМ ПОСТРОЕНИЯ ROBDD-ГРАФА, ПРЕДСТАВЛЯЮЩЕГО МНОЖЕСТВО ВСЕХ ДОСТИЖИМЫХ РЕАКЦИЙ КОМБИНАЦИОННОЙ ЛОГИЧЕСКОЙ СХЕМЫ

В.А. Провкин, А.Ю. Матросова

Томский государственный университет, Томск, Россия
prowkan@mail.ru, mau11@yandex.ru

ALGORITHM OF BUILDING A ROBDD GRAPH REPRESENTING THE SET OF ALL REACHEABLE REACTIONS OF A COMBINATIONAL CIRCUIT

V.A. Provkin, A.Yu. Matrosova

Tomsk State University, Tomsk, Russia

В ряде задач диагностики дискретных устройств требуется определить множество всех двоичных наборов, которые могут появиться на выходных полюсах заданной комбинационной схемы. В частности, знание множества всех достижимых наборов может использоваться для построения частично определённых функций внутренних полюсов, зависящих от предшествующих внутренних полюсов [1]. Такие функции, в свою очередь, находят применение в различных задачах: маскирование неисправностей и вредоносных подсхем, проверка корректности неполностью реализованных схем и др.

Пусть задана комбинационная логическая схема C, реализующая систему булевых функций $f_1(x_1, x_2, \ldots, x_n), f_2(x_1, x_2, \ldots, x_n), \ldots, f_m(x_1, x_2, \ldots, x_n)$. Рассмотрим следующую функцию, которая зависит от входных и выходных полюсов схемы C:

$$f(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m) = (y_1 \sim f_1(x_1, x_2, \ldots, x_n)) \wedge (y_2 \sim f_2(x_1, x_2, \ldots, x_n)) \wedge \ldots \wedge (y_m \sim f_m(x_1, x_2, \ldots, x_n)).$$

Каждый единичный набор этой функции представляет собой достижимую реакцию схемы после отбрасывания значений входных переменных. Представим эту функцию в виде ROBDD графа, причём порядок переменных для разложения Шеннона выберем следующим: $y_1 \prec y_2 \prec \ldots \prec y_m \prec x_1 \prec x_2 \prec \ldots \prec x_n$.

Таблица 1

Таблица истинности для некоторой схемы

x_1	x_2	x_3	x_4	y_1	y_2	y_3
0	0	0	0	0	1	1
0	0	0	1	1	1	0
0	0	1	0	1	0	1
0	0	1	1	0	1	1
0	1	0	0	1	1	0
0	1	0	1	1	0	1
0	1	1	0	0	1	1
0	1	1	1	1	1	0
1	0	0	0	1	0	1
1	0	0	1	0	1	1
1	0	1	0	1	1	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	1	0
1	1	1	0	1	0	1
1	1	1	1	0	1	1