УДК 004.04 DOI 10.23683/2311-3103-2018-7-133-143

И.А. Ляпунова, М.Н. Левченко, В.В. Гривцов

РАЗРАБОТКА ALL-SAT-РЕШАТЕЛЯ ДЛЯ ПОИСКА ВЕРИФИКАЦИОННЫХ НАБОРОВ В ТЕСТИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Существует множество методов обнаружения ошибок и контроля качества программ, совокупность которых называется верификацией программного обеспечения. На сегодняшний день активно развиваются системы, предполагающие автоматическое доказательство соответствия программного обеспечения так называемой спецификации, т.е. предъявляемым к нему требованиям. Тем не менее, в большинстве проектов в основе процесса верификации до сих пор стоит именно тестирование. Зачастую этим занимаются люди, задачей которых является разработка тестов для поиска ошибок, допущенных в коде программистами. Упрощённо тесты представляют собой набор «входных» параметров, подающихся на вход в программу, а по завершении ее работы «выходные» параметры сравниваются с «ожидаемыми» по логике, описанной в спецификации. Тестирование является наиболее распространенным методом выявления ошибок в работе программного обеспечения и уже в 60-х годах использовалось для проверки программ, применяющихся в научных исследованиях, в авиакосмической отрасли, энергетике, медицине и прочих сложных системах. Процесс тестирования подобных продуктов проводился строго формализованно и в основе этого процесса стояла разработка тестовых процедур, которая проводилась на низком уровне автоматизации, и за более чем 50 лет развитие в этой области было несущественным. В начале 70-х годов тестирование определялось как «процесс, направленный на демонстрацию корректности работы программы», но вскоре стало ясно, что оно не способно выявить все критические ошибки проверяемого софта, но актуальности применения не потеряло. Такой вывод был подтвержден рядом происшествий, из-за которых крупные компании и государственные учреждения понесли большие финансовые потери. Ярким примером может служить крушение космического аппарата Mariner-1, произошедшее 22 июля 1962 года из-за ошибки в программе бортового компьютера. Таким образом, тестирование можно определить, как многократное выполнение программы с намерением найти в ней ошибки, но это в свою очередь не доказывает её корректность. В настоящей работе рассматривается применение SAT-решателей для решения проблемы поиска верификационных наборов для тестирования программного обеспечения и разработка модификация ALL-SAT-решателя.

Тестовые наборы; автоматическая генерация; решатель; булевы ограничения.

I.A. Lyapunova, M.N. Levchenko, V.V. Grivtsov

DEVELOPING AN ALL-SAT SOLVER TO SEARCH FOR VERIFICATION KITS IN TESTING THE SOFTWARE

There are many methods of error detection and quality control programs, the combination of which is called software verification. To date, systems are being actively developed that imply automatic proof of software compliance with the so-called specification, i.e. requirements for it. Nevertheless, in most projects, testing is still the basis of the verification process. Often, this is done by people whose task is to develop tests for finding errors made by programmers in the code. Typically, tests are a set of "input" parameters supplied to the entrance to the program, and upon completion of its work, the "output" parameters are compared with "expected" according to the logic described in the specification. Testing is the most common method of detecting errors in the software and in the 1960s was used to test programs used in scientific research in the aerospace industry, energy, medicine and other complex systems. Testing of such products was carried out strictly formalized and the basis of this process was the development of test procedures. It was conducted at a low level of automation, and for more than 50 years, development in this area has been insignificant. In the early 70s, testing was defined as “a process aimed at demonstrating the

correctness of the program's work". But it soon became clear that it was not able to identify all the critical errors of the software being tested. This conclusion was confirmed by a number of incidents, due to which large companies and government agencies suffered heavy financial losses. An example is the crash of the Mariner-1 spacecraft, which occurred on July 22, 1962 due to an error in the onboard computer program. Thus, testing can be defined as the repeated execution of a program with the intention of finding errors in it, but this in turn does not prove its correctness. This paper discusses the use of SAT solvers to solve the problem of finding verification kits for software testing and the development of an ALL-SAT solver modification.

Test suites; automatic generation; solver; boolean constraints.

Введение. Верификация – это исследование и обоснование того, что программа соответствует своей спецификации. Другими словами, это процесс поиска ошибок в программе или доказательство того, что их нет.

Основные методы верификации тестов. Рассмотрим основные этапы разработки ПО и приближенное распределение внесенных и обнаруженных ошибок (рис. 1). Первый этап включает в себя анализ предметной области, требований пользователя и завершается разработкой спецификации программного обеспечения. В процессе проектирования происходит разработка архитектуры ПО и выполняется его детализация. Реализация – написание кода программы. Далее проводится тестирование (компонентное и системное), и если оно не дало информацию об ошибках кода, то программа может быть передана для эксплуатации [1–4].

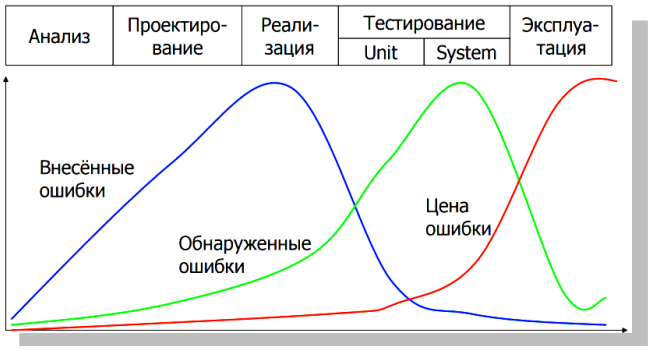

Рис. 1. Этапы разработки ПО и информация об ошибках

Методы верификации ПО можно разделить на тестирование и формальные методы, основанные на использовании математического аппарата, реализованного в языках, методах и средствах спецификации и верификации программ.

В задаче тестирования важным пунктом является обоснование полноты тестового покрытия. Как правило, в той или иной форме это обоснование сводится к подтверждению выполнения всех функций спецификации. Оно может быть выполнено методом «черного ящика» (black box testing) с полным покрытием входных данных и методом «белого ящика» (white box testing) с полным покрытием кода программы.

Автоматическая проверка корректности ПО представляет значительную проблему в условиях отсутствия строгих стандартов разработки спецификации и написания кода. К тому же данные методы имеют следующие существенные недостатки: медленная скорость работы, может потребоваться вмешательство человека и в общем случае нельзя построить полную систему аксиом и правил вывода. Таким образом, тестирование софта остается наиболее востребованной и эффективной процедурой верификации ПО для большинства крупных проектов, в которых внедрение формальных методов представляется невозможным.

Процесс тестирования условно можно разделить на 3 этапа (рис. 2):

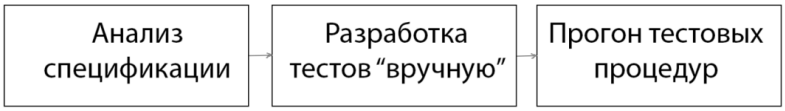

Рис. 2. Схема процесса тестирования

Анализ спецификации представляет собой процесс ознакомления тестировщика с требованиями к ПО. Наиболее трудоемким и долгим процессом является разработка тестов, которая может составлять от 30 до 60 % от общей трудоемкости создания программ. Это связано с тем, что тесты разрабатываются «вручную», на низком уровне автоматизации и перед тестировщиком стоит задача разработать такое количество тестов, которое позволило бы отыскать в коде программы все возможные ошибки. Зачастую особенности процесса разработки требуют проверки программного комплекса методом «черного ящика», когда приходится работать только с входными и выходными параметрами программы, составляя тесты, соответствующие требованиям, что в свою очередь приводит к задаче нахождения достаточного количества различных комбинаций входных условий, полный перебор которых либо не эффективен и приводит к существенному замедлению прогона тестов, либо не возможен в силу бесконечного множества таких комбинаций [4–8].

При тестировании «чёрного ящика» главной задачей является покрытие логики, представленной в требованиях к ПО. Тестирование всевозможных путей алгоритма в общем случае невозможно, т.к. единственным теоретически обоснованным способом осуществления этого является перебор всех комбинаций выходных параметров. Даже если тестирование будет проводиться на модели, состоящей исключительно из параметров булевского типа, то необходимо будет составить тестовый набор, содержащий все комбинации входов. Очевидно, что не только прогон тестов, но и их генерация при большом количестве входных параметров N, будет производиться за астрономически большое время $T(N)=O(2^N)$. Несмотря на кажущуюся простоту большинства требований к программному обеспечению, в которых может быть всего два входа и один выход, этот выход может являться внутренним параметром для всей системы и подаваться в качестве входного параметра в другом требовании. В таких случаях в формировании одного выходного параметра может участвовать большое количество входов.

Ввиду того факта, что тестирование всех возможных путей алгоритма в общем случае невозможно, используются критерии комбинаторного покрытия условий. Он требует такого достаточного набора тестов, который обладает следующими свойствами:

1) вызывает все возможные комбинации результатов условия в каждом решении;
2) передает управление каждой точке входа по крайней мере один раз.

В процессе разработки тестов косвенно осуществляется разбиение выходной области программы на конечное число классов эквивалентности. Оно происходит такими образом, чтобы можно было предположить, что каждый тест, принадлежащий некоторому классу, эквивалентен всем остальным представителям данного класса. Таким образом мы можем предполагать наличие такого тестового набора, который будет состоять из тестов, попарно не принадлежащих одному и тому же классу. Назовем такой набор минимальным тестовым набором.

Как видно, комбинированное покрытие условий – непростая задача, поскольку даже при построенном разбиении входных условий число комбинаций обычно очень велико. Если нет систематического выбора подмножества комбинаций входных условий, то вероятность нахождения всех ошибок программы очень мала.

Для обеспечения хорошего тестового покрытия в сложных алгоритмах зачастую приходится использовать метод функциональных диаграмм. Он помогает систематизировано выбирать хорошие тесты, а также косвенно позволяет обнаруживать недостатки в спецификации.

Функциональная диаграмма (рис. 3) представляет собой формальный язык, на который транслируется спецификация. Ей можно сопоставить комбинаторную логическую сеть, которая принята в электронике. Она отображает логические связи входных, внутренних и выходных параметров. Путем методического прослеживания состояний условий диаграммы она преобразуется в таблицу решений с ограниченными входами. Каждый столбец таблицы решений соответствует тесту.

Данный метод позволяет найти оптимальный тестовый набор, но пользуются им нечасто. Это связано с тем, что:

a. бóльшая часть требований может быть покрыта тестами, составленными «в уме», в силу своей простоты;

b. опытным тестировщикам зачастую важнее (например, из-за ограниченности во времени) разработать такой набор тестов, который не обеспечивает полного покрытия, но соответствует его представлениям о возможных локализациях ошибок ПО.

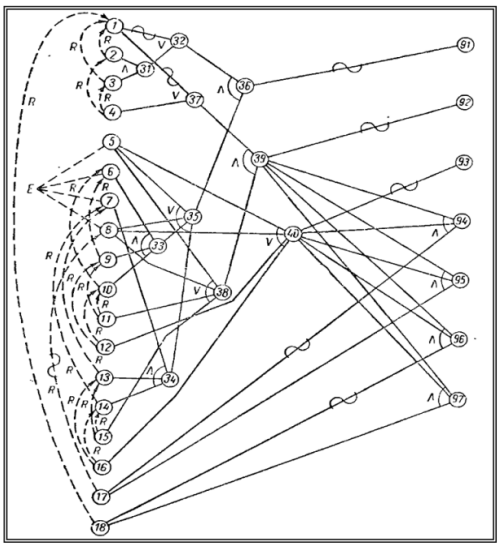

Рис. 2. Пример функциональной диаграммы

Метод функциональных диаграмм (рисунок 3) примечателен тем, что является некоторым подобием логических формул, в которые можно преобразовать алгоритмы, представленные в спецификации.

Таким образом, имеет смысл рассматривать возможность генерации тестовых наборов, основываясь на логической формуле, в которую транслируется спецификация. Логическая семантика является простейшим способом определения логики программы.

Постановка задачи. Существует большое множество алгоритмов, которым можно сопоставить модель, легко транслируемую в логическую формулу. В данном случае все переменные не булевского типа и операции над ними следует опустить или заменить на булевы, если это возможно. Широко известны методы решения таких комбинаторных проблем как генерация автоматических шаблонных тес-

тов для интегральных схем или проверка эквивалентности логических цепей. Они предполагают использование SAT-решателей для доказательства эквивалентности комбинационных схем. Применение SAT и SMT-решателей все чаще встречается в формальных методах верификации, но они всегда требуют не только строгого описания спецификации, но еще и строго стандартизированного кода.

Средства автономного тестирования (CAT, SAT) - программа или некоторый скрипт, которая имитирует окружение для отдельного модуля тестируемого ПО и позволяет тестировать этот модуль в отрыве от системы.

Обычно каждый модуль помимо входных и выходных данных имеет несколько зависимостей в виде опять же дополнительных входных данных или данных от системы. Для того чтобы модуль можно было протестировать необходимо имитировать или заглушить эти зависимости.

Тестирование модулей – процедура тестирования отдельных подпрограмм или некоторый комплекс функций программы. Здесь подразумевается, что, перед тем как начинать тестирование программы в целом, следует протестировать отдельные небольшие модули, образующие эту программу. Такой подход позволяет управлять комбинаторикой тестирования, так как первоначально внимание концентрируется на небольших модулях программы. Так же не стоит забывать о более легкой отладке, скорости тестирования и возможности распараллелить тестирование модулей.

Тестирование, имея очевидные преимущества, также сталкивается с рядом проблем: для больших последовательных или параллельных программ это очень сложно перебрать все входные данные, а для динамических структур – это невозможно; огромные размеры требуемого покрытия; полное покрытие не гарантирует отсутствия ошибок. При этом если тестирование позволяет выявлять частые ошибки, то формальные методы нацелены на поиск критических ошибок. Это связано с тем, что применение формальных методов зачастую сопряжено с гораздо большими затратами ресурсов и времени. Здесь нужно отметить, что тестирование и формальные методы не являются взаимоисключающими способами верификации ПО и могут выполняться совместно, повышая эффективность поиска ошибок.

Разработка ALL-SAT-решателя. Поиск всех выполняющих наборов булевских формул получил широкое применение в таких областях как проверка неограниченных символьных моделей, QBF-задачах, задачах булевой оптимизации и многих других [8–11, 16–19].

Для реализации ALL-SAT-решателя наиболее очевидна возможность взять за основу алгоритм существующего SAT-решателя. Этот путь одновременно является и одним из наиболее эффективных.

Задача о выполнимости булевых формул (the Boolean Satisfiability Problem или SAT) – задача, которая заключается в определении, существуют ли такие значения переменных, при которых данная булева формул (набор переменных, скобок и операций конъюнкции, дизъюнкции и логического отрицания) получает значение «истина». SAT является NP-полной задачей, поэтому она крайне важна для теории алгоритмов, и ее решение способно дать ответ на вопрос равенства классов P и NP. В настоящее время по задаче SAT проводится огромное количество исследований – как практических, так и теоретических. Каждый год проводятся конкурс программ, так называемых SAT-решателей. Такие программы активно используются в прикладных областях: в автоматизации разработки микросхем, криптоанализе, искусственном интеллекте и во многом другом. «NP-полнота» SAT заключается в том, что к ней сводится большое количество задач из класса NP за полиномиальное время (теорема Кука-Левина). Поэтому использование SAT-решателя (рис. 4, 5) может оказаться полезным для решения таких распространенных про-

блем, как составление расписания в учебных заведениях или построение рационального маршрута (задача коммивояжера). Для реализации мультиплатформенности было решено написать программу на основе использования веб-технологий (HTML, CSS, JavaScript). Таким образом, данный SAT-решатель представляет собой веб-страницу с полем ввода и кнопкой, запускающей алгоритм решения задачи.

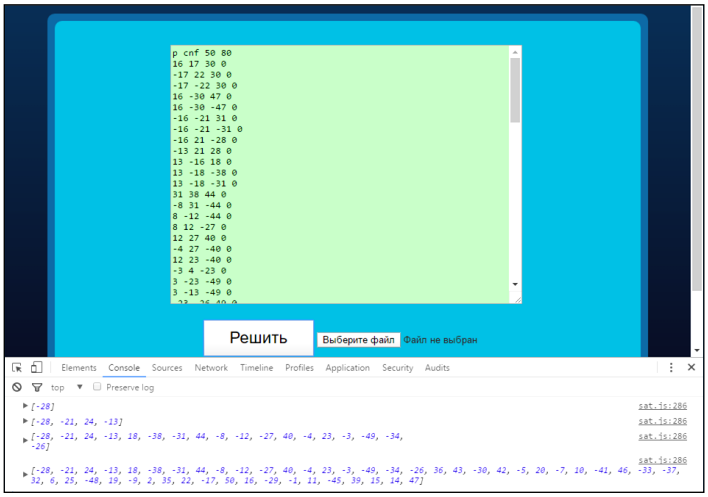

Рис. 4. Пример скрипта работы SAT-решателя когда формула выполнима

Самым простым способом нахождения всех решающих наборов является изменение алгоритма таким образом, чтобы после каждого найденного решения l к исходной формуле дописывался дизъюнкт вида $\bar{l}$, накладывающий ограничение на нахождение такого же решающего набора [18, 19]. И так дописываем отрицание каждого решающего набора к исходной КНФ до тех пор, пока формула не станет неразрешима.

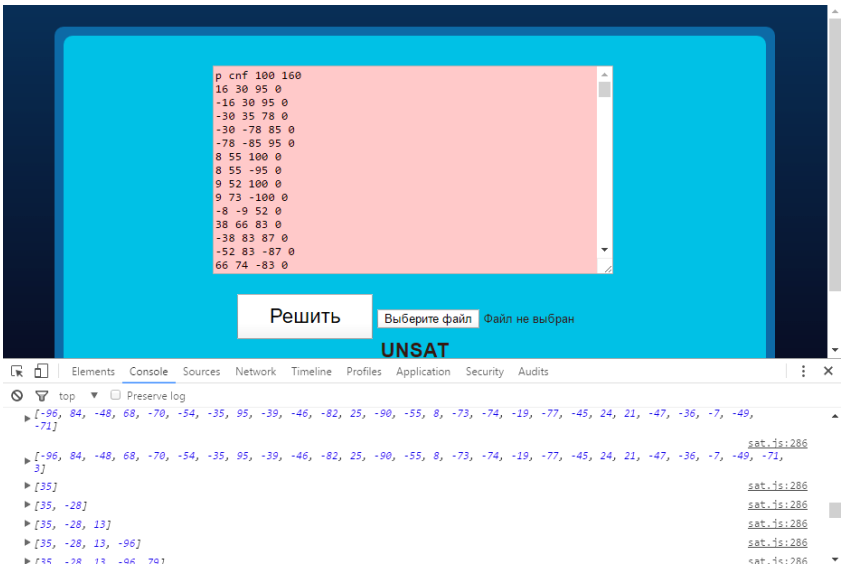

Рис. 5. Пример скрипта работы SAT-решателя когда формула не выполнима

Можно также изменить алгоритм DPLL таким образом, чтобы в случае нахождения решения внутренний механизм анализа конфликтов возвращал вычисления на последний неконфликтный уровень в дереве приписаний.

В силу того, что количество решений формулы априори достаточно велико, было решено использовать второй подход к нахождению всех выполняющих наборов булевой формулы.

Несмотря на то, что задача SAT является NP-полной и неизвестны алгоритмы ее решения за полиномиальное время, существует ее частный случай – 2-SAT, который решается за линейное время. Она так же заключается в поиске такого набора булевых переменных, чтобы формула обращалась в единицу, но на вход подается формула в 2-КНФ, т.е. каждый дизъюнкт состоит из двух литералов.

Алгоритм основывается на построении графа импликаций, вершинами которого являются все литералы формулы и их отрицания, а ребра соответствуют импликативным связям.

Заметим здесь, что для того, чтобы задача 2-SAT была выполнима, необходимо и достаточно, чтобы для любого литерала x_i вершины находились в разных компонентах сильной связности графа импликаций. Этот критерий можно проверить за время $0(N+M)$ с помощью алгоритма поиска сильно связных компонент.

Теперь постоим алгоритм нахождения решения задачи 2-SAT. Стоит отметить, что несмотря на то, что решение существует, для некоторых переменных может выполняться, что из x_i достижимо x_i, или (но не одновременно), из x_i^{-} достижимо x_i. В таком случае выбор одного из значений переменной x_i будет приводить к противоречию, в то время как выбор другого не будет. Будем выбирать из двух значений то, которое не приводит к возникновению противоречий. Выбрав какое-либо значение, мы должны запустить из него обход в глубину/ширину и пометить все значения, которые следуют из него, т.е. достижимы в графе импликаций. Соответственно, для уже помеченных вершин никакого выбора делать не нужно, для них значение уже выбрано и зафиксировано. Нижеописанное правило применяется только к непомеченным ещё вершинам.

Пусть $comp[v]$ обозначает номер компоненты сильной связности, которой принадлежит вершина v, причём номера упорядочены в порядке топологической сортировки компонент сильной связности в графе компонентов (т.е. более ранним в порядке топологической сортировки соответствуют большие номера: если есть путь из v в w, то $comp[v] \leq comp[w]$). Тогда, если $comp[x_i] < comp[xi^{-}]$, то выбираем второе значение, иначе выбираем x_i.

Докажем, что при таком выборе значений мы не придём к противоречию. Пусть, для определённости, выбрана вершина x (случай, когда выбрана вершина x_i^{-}, доказывается симметрично).

Во-первых, докажем, что из x_i недостижимо x_i^{-}. Действительно, так как номер компоненты сильной связности $comp[x_i]$ больше номера компоненты $comp[x_i^{-}]$, то это означает, что компонента связности, содержащая x, расположена левее компоненты связности, содержащей !x, и из первой никак не может быть достижима последняя.

Во-вторых, докажем, что никакая вершина x_j , достижимая из x_i , не является "плохой", т.е. неверно, что из x_j достижимо x_j^{-}. Докажем это от противного. Пусть из x достижимо x_j, а из y достижимо x_j^{-}. Так как из x_i достижимо x_j , то, по свойству графа импликаций, из y^{-} будет достижимо x_i^{-}. Но, по предположению, из x_j достижимо x_j^{-}. Тогда мы получаем, что из x_i достижимо x_i^{-}, что противоречит условию, что и требовалось доказать.

Данный алгоритм находит искомые значения переменных в предположении, что для любой переменной x_i вершины находятся в разных компонентах сильной связности.

Теперь мы можем собрать весь алгоритм воедино и построим граф импликаций. Найдём в этом графе компоненты сильной связности за время $O(N+M)$. Пусть $comp[v]$ - это номер компоненты сильной связности, которой принадлежит вершина v. Проверим, что для каждой переменной x_i вершины лежат в разных компонентах, т.е. $comp[x_i] \neq comp[\neg x_i]$. Если это условие не выполняется, то вернуть "решение не существует". Если $comp[x_i] > comp[\neg x_i]$, то переменной x_i выбираем значение *true*, иначе - *false*.

Выводы. Получившийся решатель может быть запущен на большинстве устройств, имеющих браузер. Особый интерес вызывает возможность реализации подобных алгоритмов на ГРИД-системе [10]. В ходе отладки и тестирования программы было установлено, что ALL-SAT -решатель работает корректно и справляется с поставленными перед ним задачами [4, 5]. Можно также сделать вывод, что полученный решатель заметно уступает в производительности реализованному на C++ MiniSAT [4, 5], но в ходе отладки и тестирования программы было установлено, что решатель работает корректно и справляется с поставленными перед ним задачами. Также были проведены тесты на мобильных устройствах и различных версиях браузеров.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Gu J., Purdom P.W., Franco J., Wah B.W.* Algorithms for satisfiability (SAT) problem: A survey. DIMACS Volume Series on Discrete Mathematics and Theoretical Computer Science: The Satisfiability (SAT) Problem. Vol. 35. American Mathematical Society, Providence, RI. – P. 19-151.
2. *Een N., Sorensson N.* An Extensible SAT-solver // in SAT. – 2003. – P. 502-508.
3. *Levin L.A.* Universal sequential search problems // Problems of Information Transmission. – 1973. – No. 9:3. – P. 265-266.
4. *Богданов Д.С.* Автоматизированная генерация верификационных наборов тестовых процедур // ВКР. Южный Федеральный Университет, 2017.
5. *Богданов Д.С., Ляпунова И.А., Тетруашвили Е.В.* Задача разработки SAT-решателя для поиска верификационных наборов в тестирования программного обеспечения // Инженерный вестник Дона. – 2017. – № 4. – С. 77.
6. *Кажаров Х.А., Ляпунова И.А., Чистяков А.Е.* Программная реализация численного решения обратной задачи транспорта веществ // Инженерный вестник Дона. – 2015. – № 4 (38). – URL: ivdon.ru/ru/magazine/archive/n4y2015/3437.
7. *Дегтярева Е.Е., Проценко Е.А., Чистяков А.Е.* Программная реализация трехмерной математической модели транспорта взвеси в мелководных акваториях // Инженерный вестник Дона. – 2012. – № 4-2 (23). – URL: ivdon.ru/ru/magazine/archive/n4p2y2012/1283.
8. *Семёнов А.А.* О преобразованиях Цейтина в логических уравнениях // Теоретические основы прикладной дискретной математики. – 2009. – С. 28-50.
9. *Mitchell D.* Linear Time: Unit Propagation and Horn-SAT // Notes on Satisfiability-Based Problem Solving. – 2015. – P. 1-5.
10. *Цейтин Г.С.* О сложности вывода в исчислении высказываний // Записки научных семинаров ЛОМИ АН СССР. – 1968. – Т. 8. – С. 234-259.
11. *Semenov A., Zaikin O.* Using Monte Carlo method for searching partitionings of hard variants of Boolean satisfiability problem. In: Malyshkin, V. (ed.) // Parallel Computing Technologies - 13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31 - September 4, 2015: Proceedings. Lecture Notes in Computer Science. – Springer, 2015. – Vol. 9251. – P. 222-230.
12. *Turing A.M.* On Computable Numbers, with an Application to the Entscheidungsproblem // Proceedings of the London Mathematical Society, 1937.

13. *Abramovici M., Breuer M.A. and Friedman A.D.* Digital Systems Testing and Testable Design. – Computer Science Press, 1990.
14. *Hutter M.* Convergence and Error Bounds for Universal Prediction of Nonbinary Sequences // Lecture Notes in Computer Science. – 2001. – Vol. 2167. – P. 239.
15. *Курейчик В.М., Таран А.Е., Ляпунова И.А.* Реализация муравьиного алгоритма на ГРИД-системе // Вестник Ростовского государственного университета путей сообщения. – 2015. – № 4 (60). – С. 48-52.
16. *Степанченко И.В.* Методы тестирования программного обеспечения // РПК «Политехник». – Волгоград, 2006. – С. 75.
17. *Сухинов А.И., Проценко Е.А., Чистяков А.Е., Шретер С.А.* Сравнение вычислительных эффективностей явной и неявной схем для задачи транспорта наносов в прибрежных водных системах // Вычислительные методы и программирование: новые вычислительные технологии. – 2015. – Т. 16, № 3. – С. 328-338.
18. *Малинин С.Н.* Тестирование объектно-ориентированного программного обеспечения на основе моделирования конечными автоматами: дисс. … канд. техн. наук: 05.13.01. – Нижний Новгород, 2010. – 156 с.
19. *Отпущенников И.В.* Методы и средства преобразования процедурных описаний дискретных функций в булевы уравнения: дисс. … канд. техн. наук: 05.13.11. – Иркутск, 2011. – 128 с.
20. *Cegielski P., Richard D.* On arithmetical first-order theories allowing encoding and decoding of lists // Theoretical Computer Science. – 1999. – Vol. 222, No. 1-2. – P. 55-75.
21. *Emese Balogh, Anton Deguet, Robert C. Susi, Axel Krieger, Anand Viswanathan, Cynthia M´enard, Jonathan A.* Coleman and Gabor Fichtinger. Visualization, Planning, and Monitoring Software for MRI-Guided Prostate Intervention Robot // Proceedings Lecture Notes in Computer Science. Springer. – Vol. 3217. – P. 73-80.
22. *Beltran M., Castillo G., Kreinovich V.* Algorithms That Still Produce a Solution (Maybe Not Optimal) Even When Interrupted: Shary's Idea Justified // Reliable Computing. – 1998. – Vol. 4, No. 1. – P. 39-53.

REFERENCES

1. *Gu J., Purdom P.W., Franco J., Wah B.W.* Algorithms for satisfiability (SAT) problem: A survey. DIMACS Volume Series on Discrete Mathematics and Theoretical Computer Science: The Satisfiability (SAT) Problem. Vol. 35. American Mathematical Society, Providence, RI, pp. 19-151.
2. *Een N., Sorensson N*. An Extensible SAT-solver, *in SAT*, 2003, pp. 502-508.
3. *Levin L.A.* Universal sequential search problems // Problems of Information Transmission, 1973, No. 9:3, pp. 265-266.
4. *Bogdanov D.S.* Avtomatizirovannaya generatsiya verifikatsionnykh naborov testovykh protsedur [Automated generation of verification sets of test procedures], VKR, [Final qualifying work]. Southern Federal University, 2017.
5. *Bogdanov D.S., Lyapunova I.A., Tetruashvili E.V.* Zadacha razrabotki SAT-reshatelya dlya poiska verifikatsionnykh naborov v testirovaniya programmnogo obespecheniya [Problem of development of SAT-solver for search of verification sets in software testing], *Inzhenernyy vestnik Dona* [Engineering Bulletin of the Don], 2017, No. 4, pp. 77.
6. *Kazharov Kh.A., Lyapunova I.A., Chistyakov A.E.* Programmnaya realizatsiya chislennogo resheniya obratnoy zadachi transporta veshchestv [Software implementation of the numerical solution of the inverse problem of transport of substances], *Inzhenernyy vestnik Dona* [Engineering Bulletin of the Don], 2015, No. 4 (38). Available at: ivdon.ru/ru/magazine/archive/n4y2015/3437.
7. *Degtyareva E.E., Protsenko E.A., Chistyakov A.E.* Programmnaya realizatsiya trekhmernoy matematicheskoy modeli transporta vzvesi v melkovodnykh akvatoriyakh [Software implementation of a three-dimensional mathematical model of suspension transport in shallow waters], *Inzhenernyy vestnik Dona* [Engineering Bulletin of the Don], 2012, No. 4-2 (23). Available at: ivdon.ru/ru/magazine/archive/n4p2y2012/1283.
8. *Semenov A.A.* O preobrazovaniyakh TSeytina v logicheskikh uravneniyakh [On Zeitin transformations in logical equations], *Teoreticheskie osnovy prikladnoy diskretnoy matematiki* [Theoretical foundations of applied discrete mathematics], 2009, pp. 28-50.

9. *Mitchell D.* Linear Time: Unit Propagation and Horn-SAT, *Notes on Satisfiability-Based Problem Solving*, 2015, pp. 1-5.
10. *Tseytin G.S.* O slozhnosti vyvoda v ischislenii vyskazyvaniy [On the complexity of the derivation in the calculus of statements], *Zapiski nauchnykh seminarov LOMI AN SSSR* [Notes of scientific seminars of LOMI an SSSR], 1968, Vol. 8, pp. 234-259.
11. *Semenov A., Zaikin O.* Using Monte Carlo method for searching partitionings of hard variants of Boolean satisfiability problem. In: Malyshkin, V. (ed.),/ *Parallel Computing Technologies - 13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31 - September 4, 2015: Proceedings. Lecture Notes in Computer Science*. Springer, 2015, Vol. 9251, pp. 222-230.
12. *Turing A.M.* On Computable Numbers, with an Application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society, 1937*.
13. *Abramovici M., Breuer M.A. and Friedman A.D.* Digital Systems Testing and Testable Design. Computer Science Press, 1990.
14. *Hutter M.* Convergence and Error Bounds for Universal Prediction of Nonbinary Sequences, *Lecture Notes in Computer Science*, 2001, pp. 2167, pp. 239.
15. *Kureychik V.M., Taran A.E., Lyapunova I.A.* Realizatsiya murav'inogo algoritma na GRID-sisteme [Implementation of ant algorithm on GRID system], Vestnik Rostovskogo gosudarstvennogo universiteta putey soobshcheniya [Vestnik of Rostov state University of railway engineering], 2015, No. 4 (60), pp. 48-52.
16. *Stepanchenko I.V.* Metody testirovaniya programmnogo obespecheniya [Methods of software testing], *RPK «Politekhnik»* [RPK "Polytechnic"]. Volgograd, 2006, pp. 75.
17. *Sukhinov A.I., Protsenko E.A., Chistyakov A.E., Shreter S.A.* Sravnenie vychislitel'nykh effektivnostey yavnoy i neyavnoy skhem dlya zadachi transporta nanosov v pribrezhnykh vodnykh sistemakh [Comparison of computational efficiency of explicit and implicit schemes for the problem of sediment transport in coastal water systems], *Vychislitel'nye metody i programmirovanie: novye vychislitel'nye tekhnologii* [Computational methods and programming: new computational technologies], 2015, Vol. 16, No. 3, pp. 328-338.
18. *Malinin S.N.* Testirovanie ob''ektno-orientirovannogo programmnogo obespecheniya na osnove modelirovaniya konechnymi avtomatami: diss. … kand. tekhn. nauk [Testing of object-oriented software based on finite state machine modeling: cand. of eng, sc. diss.]: 05.13.01. Nizhniy Novgorod, 2010, 156 p.
19. Otpushchennikov I.V. Metody i sredstva preobrazovaniya protsedurnykh opisaniy diskretnykh funktsiy v bulevy uravneniya: diss. … kand. tekhn. nauk [Methods and means of converting procedural descriptions of discrete functions into Boolean equations: cand. of eng. sc. diss]: 05.13.11. Irkutsk, 2011, 128 p.
20. *Cegielski P., Richard D.* On arithmetical first-order theories allowing encoding and decoding of lists, *Theoretical Computer Science*, 1999, Vol. 222, No. 1-2, pp. 55-75.
21. *Emese Balogh, Anton Deguet, Robert C. Susi, Axel Krieger, Anand Viswanathan, Cynthia M´enard, Jonathan A.* Coleman and Gabor Fichtinger. Visualization, Planning, and Monitoring Software for MRI-Guided Prostate Intervention Robot, *Proceedings Lecture Notes in Computer Science. Springer*, Vol. 3217, pp. 73-80.
22. *Beltran M., Castillo G., Kreinovich V.* Algorithms That Still Produce a Solution (Maybe Not Optimal) Even When Interrupted: Shary's Idea Justified, *Reliable Computing*, 1998, Vol. 4, No. 1, pp. 39-53.

Статью рекомендовал к опубликованию д.ф.-м.н., профессор А.И. Жорник.

Ляпунова Ирина Артуровна – Южный федеральный университет; e-mail: ialyapunova@sfedu.ru; 347900, г. Таганрог, пер. Некрасовский, 44; тел.: 88634371705; к.т.н.; доцент.

Левченко Марина Николаевна – e-mail: mnlevchenko@sfedu.ru; к.ф.-м.н.; доцент.

Гривцов Владимир Владиславович – e-mail: gvv@sfedu.ru; 347900, г. Таганрог, ул. Александровская, 68, кв. 35; тел.: +79185949373; к.т.н.; доцент.

Lyapunova Irina Arturovna – Southern Federal University; 44, Nekrasovskiy, Taganrog, 347928, Russia; e-mail: ialyapunova@sfedu.ru; phone: +78634371705; cand. of eng.; sc.; associate professor.

Levchenko Marina Nikolaevna – e-mail: mnlevchenko@sfedu.ru; cand. of phis.-math. sc.; associate professor.

Grivtsov Vladimir Vladislavovich – e-mail: gvv@sfedu.ru; 68, Alexandrovskaya street, ap. 35, Taganrog, 347900, Russia; phone: +79185949373; cand. of eng.; sc.; associate professor.

УДК 681.51 DOI 10.23683/2311-3103-2018-7-143-154

А.Н. Попов

СИНЕРГЕТИЧЕСКИЙ СИНТЕЗ СЛЕДЯЩИХ РЕГУЛЯТОРОВ

Рассматривается применение принципов и методов синергетической теории управления для синтеза автоматических регуляторов, обеспечивающих решение задачи слежения. Следящий регулятор должен обеспечивать изменение одной из переменных объекта управления в соответствии с некоторым временным сигналом, поступающим на вход системы и являющимся априори неизвестной функцией времени. При проектировании большинства следящих систем используются частотные методы синтеза, предполагающие линейное математическое описание управляемого объекта и заключающиеся в определении структуры и параметров корректирующего звена, которое обеспечивает устойчивость системы при заданном порядке астатизма по входному воздействию. Такой подход становится малоэффективен, если динамика объекта управления является существенно нелинейной. Целью исследования является разработка аналитических процедур синтеза следящих регуляторов для общего класса нелинейных систем. В статье представлено два подхода к проведению процедуры синергетического синтеза следящих регуляторов. Первый из них предполагает вычисление производных входного сигнала по времени. Второй подход основан на идее текущей кусочно-линейной аппроксимацией входного сигнала и синтезе асимптотического наблюдателя коэффициента наклона аппроксимирующей прямой. В этом случае следящая система оперирует только с входным сигналом и его дифференцирование не требуется. Предлагаемые подходы иллюстрируются простыми примерами. Для подтверждения теоретических исследований проведено компьютерное моделирование синтезированных следящих систем при различных входных сигналах. По их результатам можно сделать вывод, что предложенные подходы позволяют синтезировать регуляторы, способные отрабатывать задающие воздействия общего класса непрерывных функций времени, и могут послужить теоретической основой для проектирования следящих систем управления нелинейными динамическими объектами.

Следящие системы автоматического управления; синергетический синтез регуляторов; синтез асимптотических наблюдателей.

A.N. Popov

SYNERGETIC SYNTHESIS OF TRACING CONTROL SYSTEMS

Paper is devoted to the usage of synergetic control theory for synthesis of automatic tracking control systems. The tracking regulator should provide change of one of control object variables according to some temporary signal arriving on an input of a system and being a priori unknown function of time. At design of the majority of the tracking control systems the frequency methods of synthesis assuming the linear mathematical description of the control object are used. These methods allow to define structure and parameters of the linear compensating element which provides stability of a system and the set astaticism number on entrance influence. Such approach becomes ineffective if dynamics of a control object is significantly nonlinear. Research objective is development of analytical procedures of synthesis of the tracking regulators for the general class