Инженерный вестник Дона, №4 (2017)
ivdon.ru/ru/magazine/archive/n4y2017/4479

Задача разработки SAT-решателя для поиска верификационных наборов в тестирования программного обеспечения

Д.С. Богданов, И.А. Ляпунова, Е.В. Тетруашвили

Южный федеральный университет, Ростов-на-Дону

Аннотация: Данная работа посвящена автоматической генерации верификационных наборов тестовых процедур; предложен и разработан алгоритм нахождения тестовых наборов посредством трансляции программ в логические формулы и их преобразования для решения задачи выполнимости булевых формул.
Ключевые слова: тестовые наборы, автоматическая генерация, решатель, булевы ограничения.

Актуальность разработки систем автоматизированного тестирования (SAT)

Верификация программного обеспечения (ПО) является наиболее важным и в то же время трудоемким этапом разработки систем с высокой ценой ошибки. Любое несоответствие таких систем выдвинутым к ним требованиям может привести к большим финансовым потерям, гибели людей и ущербу окружающей среде. Методы верификации ПО активно развиваются с середины XX века и с каждым годом возрастает потребность в их автоматизации. Тестирование является наиболее распространенным и универсальными способом поиска ошибок в программах, но сталкивается с множеством трудностей при увеличении сложности систем.

Существует множество методов обнаружения ошибок и контроля качества программ, совокупность которых называется верификацией программного обеспечения. Сейчас активно развиваются системы, предполагающие автоматическое доказательство соответствия ПО так называемой спецификации, т.е. предъявляемым к нему требованиям. Тем не менее, в большинстве крупных 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. Обычно 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.

В работе показано, что задача тестирования ПО может сводиться к задаче выполнимости булевых формул (SAT) и выполнимости формул в теориях (SMT). Существует очевидная возможность перевода простейшего алгоритма в булеву формулу, для которой будут введены некоторые преобразования, позволяющие существенно ограничить полный перебор возможных комбинаций входных параметров.

Разработка решателя

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 [1, 2].

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, [3]). Поэтому использование SAT-решателя может оказаться полезным для решения таких распространенных проблем, как составление

расписания в учебных заведениях или построение рационального маршрута (задача коммивояжера).

В задаче о выполнимости дана булева формула Ф, которая состоит из N переменных, операторов конъюнкции (∨), дизъюнкции (∧) и отрицания (¬), а также скобок. Задача заключается в поиске ответа на вопрос, можно ли присвоить такие значения переменным, чтобы формула стала истинной.

SAT-решатели фокусируются на задаче в конъюнктивной нормальной форме, так как для формул в дизъюнктивной нормальной форме решение тривиально и производится за линейное время – для разрешимости достаточно, чтобы присутствовала хотя бы одна конъюнкция, не содержащая одновременно противоположные литералы.

В алгоритме мультиплатформенного SAT-решателя было решено использовать самое необходимое: распространение переменной, наблюдение за двумя литералами, а также CDCL (Conflict-Driven Clause Learning). Алгоритм будет основан на алгоритме Дэвиса-Патнема-Логемана-Лавленда (DPLL) – это итеративная процедура, в основе которой стоит подстановка в формулу значений с последующим «распространением булевых ограничений» (Boolean Constraint Propagation, BCP), наблюдение за конфликтами и бэктрекинг.

Распространение булевых ограничений основано на простом правиле единичного дизъюнкта.

Обычно решение SAT задачи заключается лишь в ответе на вопрос «существует ли такой набор значений переменных, при котором формула обращается в 1». Но на практике всегда важно знать этот набор, называемый «сертификатом». Поэтому очевидно, что распространение булевых ограничений должно сопровождаться присвоением единичным дизъюнктам значения 1.

Каждый раз, когда BCP не способно больше сократить формулу и решающий набор не найден, псевдослучайным образом задается значение одной из переменных и снова запускается распространение булевых ограничений. Так продолжается до тех пор, пока либо не будет найден «сертификат», либо не произойдет конфликтная ситуация, заключающаяся в появлении контрарной пары переменных в дереве частичного приписания.

При возникновении конфликтной ситуации запоминается полученное частичное приписание, которое не приведет к решению.

Необходимо учесть, что задачи для CNF-SAT-решателей принято формулировать следующим образом: первая строка текстового документа должна отображать информацию о «проблеме» (через пробел вводится количество переменных и количество дизъюнктов); каждая следующая строка представляет собой дизъюнкт, переменные в котором также разделяются пробелом; каждый дизъюнкт завершается нулем «0»; в качестве переменных выступают числа, соответствующие индексу переменной; если перед переменной стоит знак «-», то это обозначает отрицание данной переменной.

Пример задачи:

p cnf 42 133
-1 -7 0
-1 -13 0
-1 -19 0
-1 -25 0
-1 -31 0
-1 -37 0
-7 -13 0
-7 -19 0
-7 -25 0

…… [119 дизъюнктов]

18 17 16 15 14 13 0

24 23 22 21 20 19 0

30 29 28 27 26 25 0

36 35 34 33 32 31 0

42 41 40 39 38 37 0

Для реализации мультиплатформенности было решено написать программу с использованием веб-технологий (HTML, CSS, JavaScript). Таким образом, данный решатель представляет собой веб-страницу с полем ввода и кнопкой, запускающей алгоритм решения задачи.

Примеры работы решателя можно увидеть на рисунках 1-2.

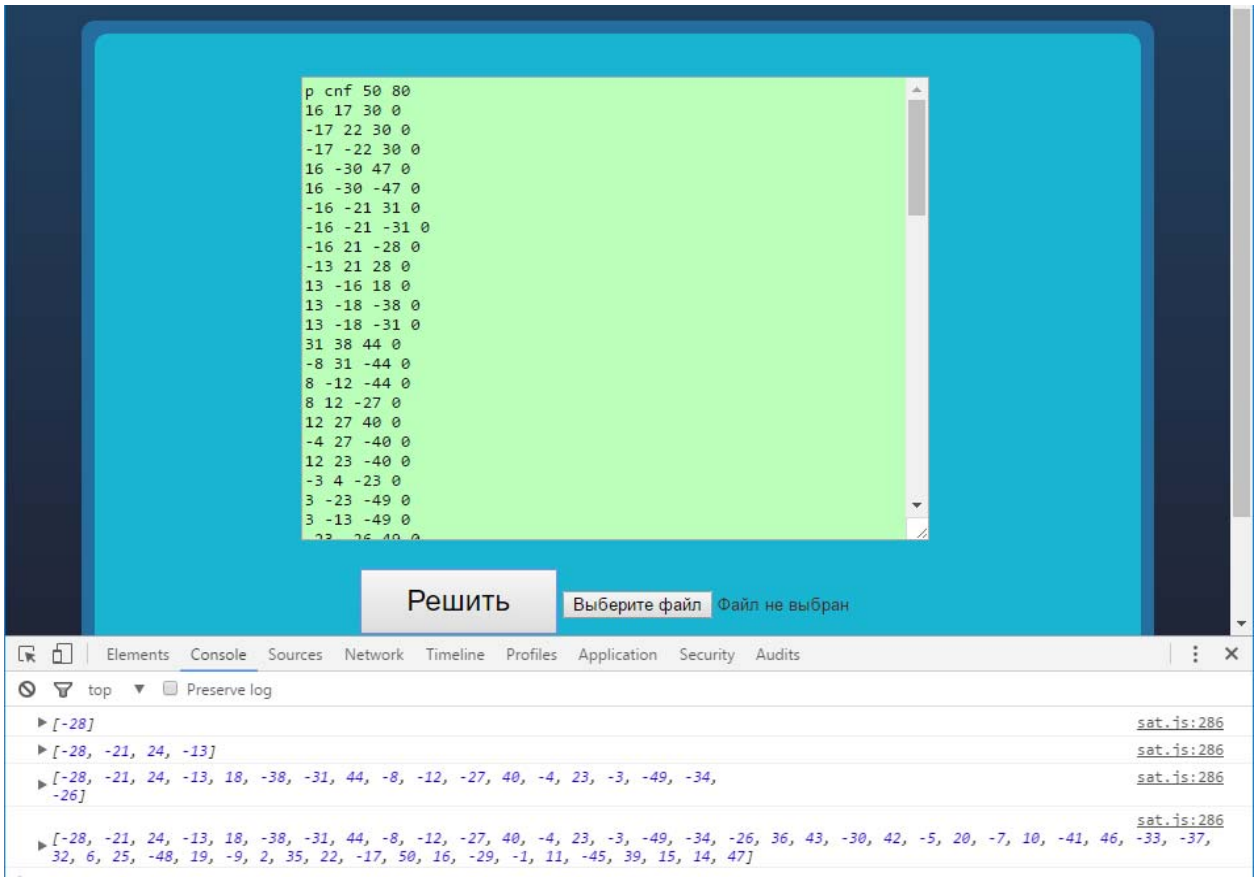

Рис. 1 – Пример работы программы (формула выполнима - SAT)

Получившийся решатель может быть запущен на большинстве устройств, имеющих браузер.

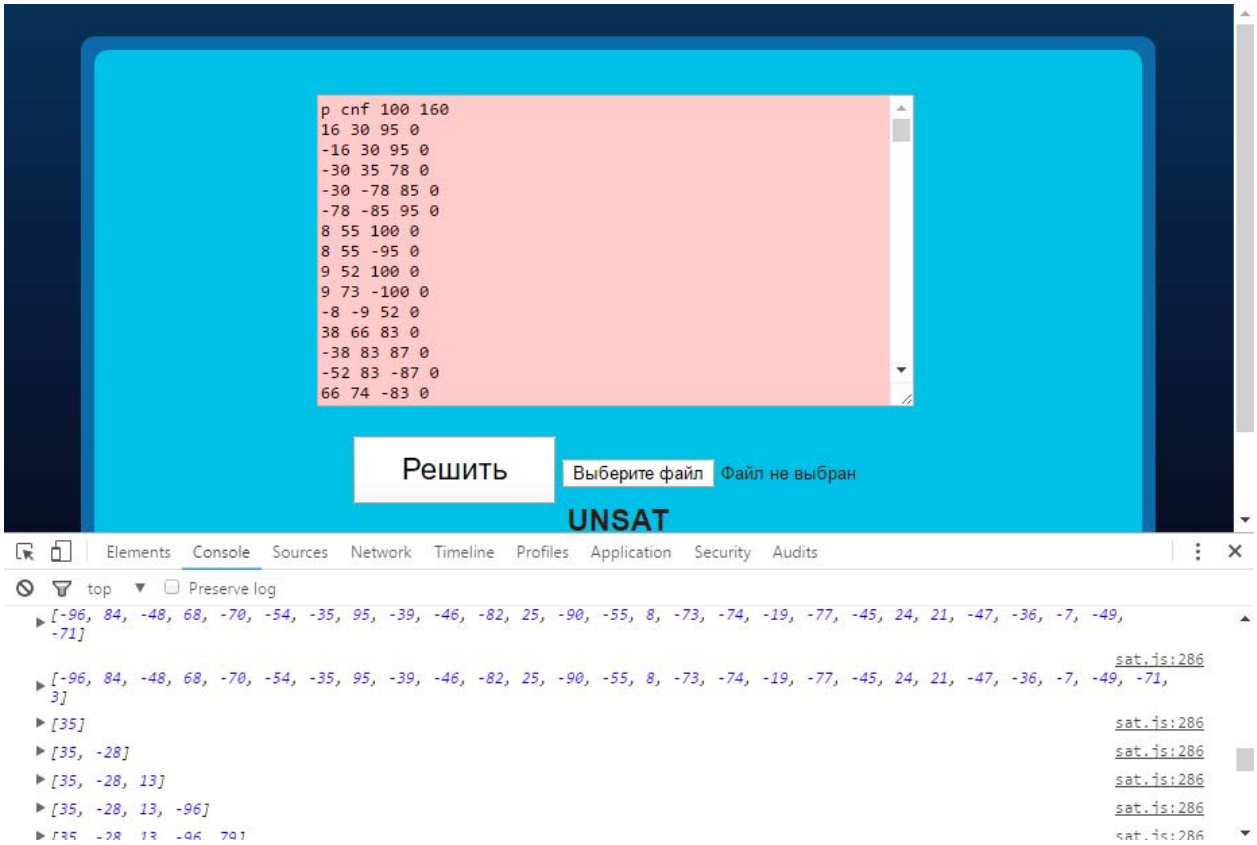

Рис. 2 – Пример работы программы (формула невыполнима – UNSAT)

Результаты решения формул в тестовых наборах представлены в таблице №1. Здесь большое значение имеют как характеристики вычислительных ресурсов, так и время работы программы [4, 5].

Таблица №1

Время решения формул

Число переменных	Число дизъюнкт	Время решения (сек)
5	40	0.00633
10	60	0.0084
20	150	0.01491
50	300	0.04902
100	1000	0.12906
100	1500	0.14763
200	3000	0.38784
1000	8000	1.11069

2000	10000	1.30737

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 [6 - 9]. Особый интерес вызывает возможность реализации подобных алгоритмов на ГРИД-системе [10]. Можно сделать вывод, что полученный решатель заметно уступает в производительности реализованному на C++ MiniSAT, 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.

Литература

1. J. Gu, P.W. Purdom, J. Franco, B.W. Wah, 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. N. Een, N. Sorensson. «An Extensible SAT-solver» in SAT 2003, pp. 502-508.

3. Levin, L. A. (1973). Universal sequential search problems. Problems of Information Transmission, 9:3, pp. 265–266.

4. Кажаров Х.А., Ляпунова И.А., Чистяков А.Е. Программная реализация численного решения обратной задачи транспорта веществ // Инженерный вестник Дона, 2015. №4 URL: ivdon.ru/ru/magazine/archive/n4y2015/3437.

5. 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 URL: ivdon.ru/ru/magazine/archive/n4p2y2012/1283.

6. Семёнов А.А. О преобразованиях Цейтина в логических уравнениях – Теоретические основы прикладной дискретной математики, 2009. – С. 28-50.

7. Mitchell D. Linear Time: Unit Propagation and Horn-SAT // Notes on Satisfiability-Based Problem Solving, 2015. pp.1-5.

8. Цейтин Г.С. О сложности вывода в исчислении высказываний// Записки научных семинаров ЛОМИ АН СССР. 1968. Т.8. С.234–259.

9. 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, vol. 9251, pp. 222–230. Springer (2015).

10. 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.

References

1. J. Gu, P.W. Purdom, J. Franco, B.W. Wah, 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. N. Een, N. Sorensson. «An Extensible SAT-solver» in SAT 2003, pp. 502-508.

3. Levin, L. A. (1973). Universal sequential search problems. Problems of Information Transmission, 9:3, pp. 265–266.

4. Kazharov H.A., Lyapunova I.A., Chistjakov A.E. Inženernyj vestnik Dona (Rus), 2015, №4. URL: ivdon.ru/ru/magazine/archive/n4y2015/3437.

5. Degtyareva E.E., Protsenko E.A., Chistyakov A.E. Inženernyj vestnik Dona (Rus), 2012, №4-2. URL: ivdon.ru/ru/magazine/archive/n4p2y2012/1283.

6. A. A. Semjonov, PDM, 2009, № 4(6), pp. 28–50.

7. Mitchell D. Notes on Satisfiability-Based Problem Solving, 2015, pp.1-5.

8. G. S. Cejtin. Zap. nauchn. sem. LOMI, 1968, tom 8, pp. 234–259.

9. 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, vol. 9251, pp. 222–230. Springer (2015).

10. Kurejchik V.M., Taran A.E., Ljapunova I.A. Vestnik Rostovskogo gosudarstvennogo universiteta putej soobshhenija, 2015, № 4 (60), pp. 48-52.