УДК 519.712

АЛГОРИТМЫ РАСЩЕПЛЕНИЯ ДЛЯ ЗАДАЧИ О ПРОПОЗИЦИОНАЛЬНОЙ ВЫПОЛНИМОСТИ

Исхаков Р.Р.
научный руководитель д-р физ.-мат. наук Быкова В.В.
Сибирский федеральный университет

Задача о пропозициональной выполнимости (англ. вариант – SATISFIABILITY, или коротко SAT) формулируется следующим образом. Пусть X – конечное множество переменных, принимающих значения false и true. Формула F в конъюнктивной нормальной форме (КНФ) представляет собой конъюнкцию конечного числа клозов, где клоз – дизъюнкция литералов, не содержащая ни одной переменной одновременно с ее отрицанием, а всякий литерал – некоторая переменная $x \in X$ или ее отрицание $\neg x$. Требуется установить, существует ли для входной формулы F, заданной в КНФ над множеством переменных X, набор значений переменных, при котором каждый клоз формулы F принимает значение true (такой набор принято называть выполняющим). Формула считается выполнимой, если для нее существует хотя бы один выполняющий набор, и невыполнимой, если выполняющего набора не существует.

Входной формуле F обычно приписывают числовую величину, которая принимает неотрицательные целые значения и характеризует размерность этой формулы. Такие величины называют мерами сложности формулы логики высказываний. В качестве основных мер сложности рассматривают: $N = |X|$ – число переменных; K – число клозов; $L = |F|$ – длину формулы F, как количество входящих в нее литералов. Относительно этих мер сложности традиционно определяют вычислительную сложность (время работы) исследуемого алгоритма решения задачи SAT.

Задача SAT служит объектом исследований и приложений в таких областях как искусственный интеллект, временная и пространственная логика, теория реляционных баз данных, распознавание образов, автоматическое доказательство теорем, техническое проектирование сложных систем. В 1971 году в статье Стивена Кука был впервые введен термин «NP-полная задача», и задача SAT была первой задачей, для которой было доказано это свойство. К полиномиально разрешимым случаям для SAT приводят следующие классы формул: пустые формулы (не содержат клозов); формулы в 2-КНФ (каждый их клоз включает не более двух литералов); хорновские формулы.

Задача SAT всегда может быть решена алгоритмом полного перебора, который последовательно анализирует все 2^N различных наборов значений N переменных. Такой алгоритм устанавливает тривиальную верхнюю оценку вычислительной сложности задачи SAT в худшем случае. Поиск алгоритмов, работающих быстрее полного перебора, активно ведется с начала с 60-х годов прошлого века и до настоящего времени. Среди них наиболее известными являются алгоритмы, основанные на локальном поиске, и алгоритмы расщепления (DPLL-алгоритмы и PPSZ-алгоритмы).

Типичный алгоритм, основанный на локальном поиске, вначале выбирает начальный набор и затем изменяет его шаг за шагом, пытаясь приблизиться к выполняющему набору. Если после определенного количества шагов выполняющий набор не найден, алгоритм порождает другой начальный набор и опять изменяет его шаг за шагом. Число таких попыток ограничено; если ни в одной из них не удается найти выполняющий набор, алгоритм заканчивает работу, с ответом «Невыполнима».

Алгоритмы расщепления – это декомпозиционные алгоритмы. Они сводят задачу SAT для входной формулы F к конечному числу подзадач, каждая из которых –

это SAT для формул меньшей сложности, чем сложность F. Причем это сведение может быть детерминированным (алгоритм производит рекурсивные вызовы самого себя для формул меньшей сложности, полученных при расщеплении F) или вероятностным (случайный выбор одной из образованных после расщепления формул). Детерминированные алгоритмы расщепления принято назвать DPLL-алгоритмами по первым буквам фамилий их авторов: M. Davis, H. Putman, G. Logemann, D. Loveland. Такой алгоритм заменяет входную формулу двумя формулами, полученными присваиванием некоторой переменной x значений `true` и `false` соответственно. Затем алгоритм упрощает каждую из полученных формул и рекурсивно вызывает процедуру для каждой из упрощенных формул. Основное отличие PPSZ-алгоритмов (их авторы R. Paturi, P. Pudlak, M.E. Saks, F. Zane) от DPLL-алгоритмов заключается в выборе переменной для присваивания значений `true` и `false`: осуществляется случайный выбор переменной вместо некоторого детерминированного правила. Подавляющее большинство алгоритмов, разработанных за последние пятьдесят лет для задачи SAT, базируется на идеях, заложенных в DPLL-алгоритмах.

Всякий DPLL-алгоритм работает по следующей схеме.

Вход: формула F в КНФ над X.

Ответ: «Выполнима» / «Невыполнима».

1. Редуцировать F, т. е. преобразовать формулу F в формулу F_0 меньшей сложности с помощью конечного набора правил редукции. Эти правила применять к F до тех пор, пока хотя бы одно из них применимо.
2. Если задача SAT тривиально решается для F_0, то выдать соответствующий ответ.
3. Выбрать переменную $x \in X$ по определенному правилу. Таких переменных может быть несколько. Их число определяется правилом расщепления, употребляемом на следующем шаге.
4. Выполнить расщепление формулы F_0 на конечное число формул меньшей сложности по заданному закону – правилу расщепления, используя различные значения переменных, выбранных на шаге 2.
5. Осуществить рекурсивные вызовы данного алгоритма применительно к полученным формулам. Выдать ответ, основываясь на результатах, возвращенных рекурсивными вызовами: если хотя бы один из рекурсивных вызовов вернул выполняющий набор, то ответ выдать ответ «Выполнима», и в противном случае – «Невыполнима».

Различные расщепляющие алгоритмы отличаются друг от друга в основном правилами редукции на шаге 1, стратегией выбора переменных на шаге 3, правилами расщепления на шаге 4.

Разработана программа DPLL-Solver, реализующая DPLL-алгоритм для задачи SAT. В программе редуцирование входной формулы выполняется по правилам: «чистый литерал», «единичный клоз», «поглощение», «резолюция». Применяемое на шаге 4 правило расщепления заменяет формулу F_0 на две формулы $F_1 = F_0[x]$ и $F_2 = F_0[\neg x]$, где x на шаге 3 определяется согласно эвристике: «выбрать переменную, входящую в наибольшее количество клозов». В качестве тривиальных случаев на шаге 2 выступают: пустой клоз (интерпретируется как `false`); пустая формула (интерпретируется как `true`). Для создания программы использована среда разработки Microsoft Visual Studio 2010 Express. Входные данные задаются в виде текстового файла в формате DIMACS, где указываются: число переменных, число клозов и сами клозы. Выходные данные содержат сведения об использованных правилах редукции и найденный выполняющий набор, если он существует. Разработанная программа может быть использована в многочисленных приложениях задачи SAT.