ОБХОД НЕИЗВЕСТНОГО ГРАФА КОЛЛЕКТИВОМ АВТОМАТОВ

И.Б. Бурдонов, А.С. Косачев

Введение.

Задача обхода неизвестного ориентированного графа автоматом используется во многих приложениях. В данной статье подразумевается тестирование детерминированных систем: граф — это граф автомата тестируемой системы, автомат на графе — тестирующая система, а проход по дуге — это тестовое воздействие и наблюдение результата [1]. В качестве практического примера можно привести работу [6], где выполнялось функциональное тестирование различных подсистем модели процессора: кэш третьего уровня, управление прерываниями и пр. Модельные графы содержали от нескольких тысяч до нескольких миллионов узлов и несколько миллионов дуг. Тест выполнялся максимально на 150 компьютерах.

Важно отметить, что автомат системы — это, вообще говоря, расширенный автомат, который обычно существенно меньше эквивалентного ему обычного автомата. Расширенный автомат мы будем называть E-графом, а просто графом — граф эквивалентного ему обычного автомата, который строится в процессе обхода.

Автомат-обходчик выполняется на одной машине (процессор с памятью), а наличие нескольких автоматов на разных машинах позволяет существенно распараллелить работу. При тестировании клонирование тестируемой системы обычно возможно только в начальном состоянии. Поэтому автомат начинает работу с начальной вершины графа. Будем считать, что за один тик создаётся не более одного клона тестируемой системы.

Нижняя оценка времени обхода для одного или ограниченного числа автоматов равна $O(nm)$, где n — число вершин графа, а m — число дуг [2]. Если число автоматов не ограничено и автоматы «знают», куда им идти, можно считать, что каждый автомат двигается к «своей» дуге, по которой проходит первым. Тогда число автоматов равно $O(m)$, последний из них до «своей» дуги проходит путь длиной не более $O(n)$, и суммарная оценка равна $O(m)+O(n)=O(m)$.

Для того, чтобы автомат мог обходить любой конечный граф, требуется доступ по чтению/записи к неограниченной рабочей памяти, в которой накапливается информация о пройденной части графа. Если эта память — часть памяти автомата (машины), автомат не конечен на классе всех графов. Если все вычисления автоматов и передачи сообщений выполняются не дольше (по порядку), чем проход дуги графа, то существуют алгоритмы с верхней оценкой, совпадающей с нижней. Если число автоматов больше одного, но ограничено, то распараллеливание ускоряет обход, не меняя порядка оценки в наихудшем случае [3].

Проблема возникает, когда граф не помещается (хотя E-граф помещается) в память машины, что эквивалентно конечности автомата. Есть два подхода.

Первый подход применим, когда E-граф не расширенный, т.е. совпадает с графом. Рабочая память реализуется на вершинах графа: автомат может писать/читать из текущей вершины символы конечного алфавита. Для одного автомата известен алгоритм с верхней оценкой $O(nm+n^2 loglogn)$ [4], а при повторном обходе $O(nm+n^2 l(n))$, где $l(n)$ — число логарифмирований, при котором достигается соотношение $1 \le log(log...(n)...) < 2$ [5]. Если автоматов несколько, каждый из них может читать пометки в вершинах, оставленные другими автоматами, без обмена сообщениями. Для двух автоматов верхняя оценка совпадает с нижней — $O(nm)$. Такой подход может применяться, например, для сети интернета, когда вершина — это узел сети, а проход по дуге — передача сообщения между узлами. В этом случае E-граф не клонируется, а каждый автомат может создавать новые автоматы в той вершине, где он находится. Это меняет оценки времени обхода, но в данной статье мы этот подход не рассматриваем.

При тестировании вершина графа — это состояние тестируемой системы, и автомат ничего не может в неё писать. Применяется второй подход: рабочая память — это суммарная память коллектива конечных автоматов, обменивающихся сообщениями. Для k машин можно обходить графы в k раз большие, чем для одной машины. Если размер графа не ограничен, число автоматов в коллективе также не ограничено.

Формализация автоматов на графе.

Автомат на графе, начиная с начальной вершины графа, двигается по дугам, проходя некоторый маршрут. Обход выполнен, если по каждой дуге прошёл хотя бы один автомат. Когда автомат находится в вершине и хочет пройти по выходящей из неё дуге, он эту дугу должен указать. Для этого используется нумерация дуг, выходящих из вершины. Для детерминированных систем разные дуги, выходящие из одной вершины, имеют разные номера так, что все выходящие дуги нумеруются от 1 до числа выходящих дуг. Номер дуги является одним из выходных символов автомата, поэтому для конечности автомата выходящая полустепень вершины должна быть ограничена числом t. Это ограничение можно обойти преобразованием графа: вершина с k выходящими дугами заменяется на $]k/t[$ новых вершин, между которыми распределяются выходящие дуги, максимально по $t-1$ дуг в каждой, новые вершины связываются в список дополнительными дугами; дуги, входившие в исходную вершину, теперь заканчиваются в первой из новых вершин.

Для того, чтобы автомат понял, в какой вершине оказался, он может узнать уникальный идентификатор текущей вершины и число выходящих из неё дуг.

Если автоматов несколько, для каждого из них создаётся своя копия Е-графа (тестируемой системы), которая помнит текущую вершину.

Мы будем рассматривать расширенный конечный автомат, в котором, кроме управляющего состояния из конечного алфавита, имеется конечное число ячеек, в которых хранится либо идентификатор (вершины), либо адрес (автомата). Сообщение содержит тега из конечного алфавита и конечное число параметров: идентификаторов и адресов. Заметим, что размер ячейки и параметра сообщения не ограничен на классе всех графов. Удобно считать, что управляющее состояние и тег сообщения — это тоже набор значений конечного числа ячеек, но ограниченного размера. Над идентификаторами и адресами определены только операции присваивания и сравнения на равенство, левой и правой частями которых могут быть только ячейки и параметры принимаемого или посылаемого сообщения. Для обмена сообщениями есть два примитива: послать сообщение по указанному адресу, принять любое сообщение от любого автомата. Предполагается, что сообщения не теряются и не искажаются средой передачи, и сохраняется порядок передачи сообщений с одними и теми же отправителем и получателем. Для создания и уничтожения автоматов используются сообщения, посылаемые вовне коллектива автоматов. Взаимодействие с Е-графом также понимается как обмен сообщениями с Е-графом. Тем самым, входной и выходной символ автомата — это сообщение.

Мы дадим описание алгоритма обхода и оценку его сложности в терминах расширенных автоматов. Однако заметим, что такой расширенный автомат эквивалентен набору конечных взаимодействующих через сообщения автоматов, число которых зависит от размера графа. Ячейка расширенного автомата разбивается на ячейки ограниченного размера, по одной на конечный автомат. Сообщение разбивается на последовательность сообщений-пакетов ограниченного размера.

Алгоритм работы коллектива автоматов.

Рассматривается обход ориентированного графа с выделенной начальной вершиной. После инициализации в каждый момент времени имеются:

- *Пройденный граф* — подграф графа, определяемый всеми пройденными дугами.
- *Остов* — остов пройденного графа, ориентированный от корня — начальной вершины графа;
- *Дерево*, являющееся поддеревом остова с тем же корнем и содержащим все пройденные начальные вершины непройденных дуг.

Если пройденная вершина принадлежит остову, но не принадлежит дереву, работа со всеми выходящими из неё дугами закончена – по ним больше не будет проходов.

Автомат работает в трёх режимах: генератор, регулятор, движок. Вначале имеется только генератор.

Генератор создаётся извне и получает сообщение ***ты генератор***. При инициализации генератор создаёт Е-граф (посылает сообщение вовне ***создай Е-граф*** и получает сообщение ***Е-граф создан***) и становится регулятором начальной вершины, которая в этот момент времени является единственной пройденной вершиной и единственной вершиной остова и дерева, а также создаёт первый движок в начальной вершине графа (посылает сообщение вовне ***создай автомат*** и получает сообщение ***автомат создан***) и посылает ему сообщение ***ты движок***. После инициализации генератор генерирует новые Е-графы и связанные с ними движки так же, как это делается для первого движка.

Регулятор связан с пройденной вершиной графа так, что с каждой пройденной вершиной связан ровно один регулятор. Исключение составляют терминальные вершины, которые не нужно отличать друг от друга и для которых регуляторы не создаются. Регулятор не двигается по графу и предназначен для регулирования движения движков.

Только в режиме движка автомат двигается по графу, начиная с его корня, сначала по пройденным выходящим дугам. Для прохода по дуге движок посылает Е-графу сообщение ***проход по дуге*** и получает сообщение ***ответ на проход***, текущей вершиной становится конец пройденной им дуги. Далее движок посылает регулятору текущей вершины сообщение ***куда идти*** и получает ответное сообщение ***иди по дуге***. Если в ответ сообщается нулевой номер выходящей дуги, то это означает, что идти некуда и движок останавливается. В противном случае движок проверяет сообщенный ему адрес регулятора конца выходящей дуги. Если он не пуст, эта дуга пройдена, и цикл повторяется. Если пуст, движок проходит по непройденной дуге $a{-}i{\to}b$, ведущей из вершины a, имеющей в ней номер i и ведущей в вершину b. Если вершина b терминальная (число дуг равно нулю), движок посылает регулятору вершины a сообщение ***конец*** с указанием номера дуги i и пустого адреса в качестве адреса регулятора вершины b, а сам останавливается. В противном случае движок должен узнать по идентификатору вершины b, пройдена она или ещё нет, т.е. имеется ли регулятор этой вершины. Для этого выполняется опрос регуляторов, который мы опишем ниже.

- Если вершина b пройдена (получено сообщение ответ на опрос), движок посылает регулятору вершины a сообщение конец с указанием номера дуги i и адреса регулятора вершины b.
    - Если вершина b не принадлежит дереву или все выходящие из неё дуги либо законченные, либо проходятся (в ответе на опрос указан нулевой номер выходящей дуги), движок останавливается.

- Если вершина b принадлежит дереву, движок продолжает двигаться по дереву.

- Если вершина b ещё не пройдена (получено сообщение опрос), движок становится регулятором вершины b, а дуга a─i→b добавляется к дереву. Для этого движок посылает регулятору вершины a сообщение новая вершина с указанием номера дуги i и свой собственный адрес в качестве адреса регулятора вершины b. Став регулятором, движок создаёт новый движок (посылает сообщение создай автомат и получает сообщение автомат создан) в текущей вершине, посылает ему сообщение ты движок, передавая ему E-граф, а сам перестаёт двигаться по графу.

Движение движков управляется регуляторами вершин. Каждая дуга *a─i→b*, выходящая из вершины *a*, имеет в регуляторе этой вершины одно из пяти состояний:

- не пройдена,
- проходится, т.е. по дуге прошёл движок, но ещё не известил регулятор вершины a о том, является ли вершина b не пройденной и не терминальной или нет,
- на дереве, т.е. дуга пройдена и принадлежит дереву,
- законченная дуга остова, т.е. дуга остова пройдена, но не принадлежит дереву,
- законченная хорда остова, т.е. пройденная хорда остова.

Регулятор хранит номер *i* текущей выходящей дуги, состояние которой *не пройдена* или *на дереве*. Если таких дуг нет, то *i*=0. Когда движок спрашивает у регулятора вершины *a*, по какой дуге ему идти, регулятор в ответ сообщает номер *i*.

Далее регулятор корректирует состояние текущей выходящей дуги: *не пройдена → проходится*, *на дереве → на дереве*. Новым значением номера текущей выходящей дуги *i* становится номер дуги, следующей после дуги *i* в циклическом списке выходящих дуг, которая либо *не пройдена*, либо является дугой *на дереве*. Если таких дуг больше нет, то *i:=0*.

Если регулятор получает сообщение ***новая вершина*** с указанием номера *j* выходящей дуги, то состояние *j*-ой дуги меняется: *проходится → на дереве*, запоминается адрес регулятора конца этой дуги. Если *i=0*, то *i:=j*.

Если регулятор получает сообщение ***конец*** с указанием номера *j* выходящей дуги, то это либо дуга, которая *проходится*, либо дуга *на дереве*. В первом случае новое состояние дуги — *законченная хорда остова*, запоминается адрес регулятора конца этой дуги. Во втором случае новое состояние дуги — *законченная дуга остова*, если *j=i*, то *i* корректируется, как описано выше.

Если все выходящие дуги *законченные*, регулятор вершины посылает сообщение ***конец*** регулятору предшествующей вершины дерева, а регулятор начальной вершины дерева посылает сообщение ***конец*** вовне, что означает завершение обхода.

В конце обхода пройденный граф совпадает со всем графом, дерево отсутствует. Регуляторы в совокупности описывают весь граф с выделенным в нём остовом.

Опрос регуляторов выполняется после прохода движка по непройденной дуге. Движку нужно найти регулятор по идентификатору вершины, в которой он оказался. Для этого используется односторонний список регуляторов, регулятор начальной вершины (он же генератор) находится в голове списка. Движок посылает генератору сообщение ***опрос***, содержащее идентификатор вершины и адрес движка. Это сообщение передаётся по списку регуляторов. Каждый регулятор, получив сообщение ***опрос***, сравнивает идентификатор вершины из параметра со своим идентификатором вершины. Если идентификаторы совпадают, регулятор посылает движку сообщение ***ответ на опрос***. Если идентификаторы не совпадают и регулятор не последний в списке, он посылает сообщение ***опрос*** дальше по списку, не меняя параметров. Последний регулятор пересылает сообщение ***опрос*** движку, и запоминает адрес движка как адрес следующего в списке регуляторов, чтобы движок стал регулятором в хвосте списка.

Ячейки автомата:

- собственный адрес автомата;
- внешний адрес;
- адрес E-графа;
- адрес генератора;
- адрес следующего в списке регуляторов;
- идентификатор текущей вершины;
- регулятор текущей вершины;
- входящая дуга остова;
- адрес регулятора начала входящей дуги остова;
- число дуг;
- номер текущей выходящей дуги;
- список — состояние выходящей дуги (i), i=1..t;
- список — адрес регулятора конца выходящей дуги (i), i=1..t;

- конечное число рабочих ячеек, используемых только до приёма следующего сообщения.

Сообщения и их параметры:

- ***ты генератор***: внешний адрес отправителя, адрес генератора;
- ***создай автомат***: адрес отправителя;
- ***автомат создан***: адрес созданного автомата;
- ***создай E-граф***: адрес отправителя;
- ***E-граф создан***: адрес E-графа, идентификатор начальной вершины, число выходящих дуг;
- ***проход по дуге***: адрес отправителя-движка, номер выходящей дуги;
- ***ответ на проход***: идентификатор вершины, число выходящих дуг;
- ***ты движок***: адрес движка, адрес генератора, адрес E-графа, номер выходящей дуги, адрес регулятора конца выходящей дуги, адрес регулятора текущей вершины;
- ***конец***: номер выходящей дуги, адрес регулятора конца выходящей дуги;
- ***куда идти***: адрес отправителя-движка;
- ***иди по дуге***: номер выходящей дуги, адрес регулятора конца выходящей дуги;
- ***новая вершина***: номер выходящей дуги, адрес регулятора конца выходящей дуги;
- ***опрос***: адрес отправителя-движка, идентификатор вершины;
- ***ответ на опрос***: адрес регулятора вершины – отправителя, номер выходящей дуги, адрес регулятора конца выходящей дуги.

Оценка сложности алгоритма.

Движки могут быть двух типов в зависимости от того, 1) проходит ли движок до остановки непройденную ранее дугу или 2) останавливается, не пройдя ни одной новой дуги. Число движков 1-го типа равно $O(m)$. Движок 2-го типа останавливается, получив нулевой номер дуги в сообщении ***иди по дуге*** или ***ответ на опрос***. Нулевой номер дуги регулятор сообщает в том случае, когда либо A) имеются дуги в состоянии *проходится*, а все остальные дуги *законченные*, либо B) все дуги *законченные*. Ситуация A наступает тогда, когда какой-то движок пошёл по последней непройденной дуге, а остальные дуги либо *законченные*, либо *проходятся*, а заканчивается после завершения опроса регуляторов. Поскольку число регуляторов не превышает $O(n)$, опрос регуляторов заканчивается за время $O(n)$. Тем самым, ситуация A закончится через $O(n)$ тиков. Ситуация B наступает тогда, когда регулятор вершины b получил сообщение ***конец*** для некоторой выходящей дуги, а все остальные дуги уже *законченные*. Если вершина b начальная, через $O(1)$ тиков обход закончится. В противном случае в вершину b входит одна дуга дерева $a{-}i{\to}b$. Через $O(1)$ тиков регулятор вершины b пошлёт сообщение ***конец*** регулятору вершины a, тот получит это сообщение и дуга $a{-}i{\to}b$ перестанет быть дугой дерева. Следовательно, ситуация B закончится через $O(1)$ тиков. Поскольку регулятор вершины только один раз оказывается в каждой из ситуаций A и B, число движков 2-го типа, останавливающихся в данной вершине, не превышает $O(n)$. Суммарно число движков 2-го типа равно $O(n^2)$. Итак, число всех движков равно $O(m+n^2)$. Поскольку каждый движок генерируется через $O(1)$ тиков после генерации предыдущего движка, время обхода равно $O(m+n^2)$.

Кластеризация по машинам.

Автомат работает в одной машине, но в одной машине может быть несколько автоматов. Передача сообщения между автоматами в одной и в разных машинах занимает существенно разное время, поэтому имеет смысл оптимизировать распределение автоматов по машинам. Сравнивая нижнюю оценку $O(m)$ и верхнюю оценку $O(m+n^2)$, видим, что оптимизировать можно только опрос регуляторов, требующий $O(n)$ тиков и суммарно по $O(n)$ опросам дающий $O(n^2)$ тиков. Если список регуляторов кластеризован, т.е. все регуляторы из одной машины расположены подряд, то время опроса равно $O(p)$, где p — число машин. Поскольку объём памяти машины $O(1)$, это не влияет на порядок оценки: $O(n)=O(p)$, что означает наличие такого коэффициента k, что $n=kp+o(p)$. Но на практике значение k очень важно, так как означает ускорение обхода в k раз.

Для кластеризации списка регуляторов нужно выделять память под регуляторы отдельно от памяти для движков и связанных с ними E-графов. Но в описанном выше алгоритме регулятор — это бывший движок. Поэтому предлагается следующая модификация алгоритма. Движок перед опросом создаёт новый регулятор, который и проводит опрос. Если искомый регулятор не найден, новый регулятор ставится в конец списка, а если найден, движок уничтожает новый регулятор.

Память под регуляторы выделяется сначала в первой машине, а когда в ней нет места, то во второй машине, и так далее. Но что делать, когда машины закончились? Можно закончить работу, считая что граф слишком большой. Но можно продолжить обход с понижением скорости. Для этого будем уничтожать движки вместе с привязанными к ним E-графам сначала в первой машине, затем во второй и так далее, оставляя только один движок в последней машине. Скорость уменьшится из-за уменьшения параллелизма (числа движков) и из-за фрагментации по машинам списка регуляторов. Но мы всё же сможем обходить графы больших размеров, если учесть, что E-граф занимает значительно больше памяти, чем регулятор.

ЛИТЕРАТУРА:

1. И.Б. Бурдонов, А.С. Косачев, В.В. Кулямин, А.К. Петренко "Подход UniTesK к разработке тестов" // Программирование, 2003 г., №6, с. 25-43
2. И.Б. Бурдонов, А.С. Косачев, В.В. Кулямин "Неизбыточные алгоритмы обхода ориентированных графов. Детерминированный случай" // Программирование, 2003 г., №5, с. 59-69.
3. И.Б. Бурдонов, С.Г. Грошев, А.В. Демаков, А.С. Камкин, А.С. Косачев, А.А. Сортов "Параллельное тестирование больших автоматных моделей" // Вестник ННГУ, 2011 г., №3, с. 187-193
4. И.Б. Бурдонов "Обход неизвестного ориентированного графа конечным роботом" // Программирование, 2004 г., № 4, с. 11-34
5. И.Б. Бурдонов "Проблема отката по дереву при обходе неизвестного ориентированного графа конечным роботом" // Программирование, 2004 г., № 6, с. 6-29
6. A. Demakov, A. Kamkin, A. Sortov "High-Performance Testing: Parallelizing Functional Tests for Computer Systems Using Distributed Graph Exploration". Open Cirrus Summit 2011, Moscow.