Тестирование на основе моделей

В. В. Кулямин

Лекция 6. Автоматные методы построения тестов. Основные понятия

Автоматные методы построения тестов основаны на предположении, что реальное поведение тестируемой системы может быть полностью описано некоторым автоматом. В виде автомата описывается также требуемое ее поведение, после чего ставится ряд экспериментов, состоящих в выполнении определенных последовательностей воздействий, с целью выяснить, отличается ли реальное поведение от требуемого.

В рамках автоматных методов предлагаются различные способы построения таких наборов тестовых последовательностей, что корректная работа тестируемой системы на них позволяет при некоторых ограничениях уверенно утверждать, что она корректно работает всегда. Основная проблема здесь — построить достаточно небольшой такой набор.

В целом автоматные методы хорошо автоматизируются, обеспечивают хорошую полноту тестирования и позволяют находить весьма сложные ошибки. Однако, они требуют наличия точного и полного описания требований к поведению тестируемой системы, определенных навыков работы с автоматными моделями ПО от разработчиков тестов и некоторых затрат на выполнение тестов. При росте сложности используемых моделей затраты на разработку тестов при помощи таких методов возрастают нелинейно.

Самой простой разновидностью автоматов являются конечные автоматы. Большинство автоматных методов построения тестов основано именно на них. На всякий случай повторим определение конечного автомата из Лекции 4.

Конечный автомат — это набор (S, s_0, I, O, T), где

S — конечное множество, элементы которого называются *состояниями* автомата;

s_0 — элемент S, называемый *начальным состоянием*;

I — конечное множество, элементы которого называются *входными символами*, *входами* или *стимулами*, само I называют *входным алфавитом* автомата;

O — конечное множество, элементы которого называются *выходными символами*, *выходами* или *реакциями*, само O называют *выходным алфавитом* автомата;

$T \subseteq S \times I \times O \times S$ — множество *переходов* автомата. Каждый переход — четверка (s_1, i, o, s_2) — имеет *начальное состояние* s_1, *конечное состояние* s_2, *стимул* i и *реакцию* o. Говорят, что он *выходит из* s_1 и *ведет в* s_2, помечен стимулом i и реакцией o. Этот переход изображают стрелкой, ведущей из s_1 и в s_2 и помеченной i/o.

Автомат может выполнять некоторую последовательность стимулов следующим образом. Сначала он считается находящимся в начальном состоянии. При получении очередного стимула в некотором состоянии недетерминированным образом выбирается один из переходов, выходящих из текущего состояния и помеченных принятым стимулом. Следующим состоянием автомата становится конечное состояние выбранного перехода, а вовне выдается реакция, которой помечен выбранный переход. Выполнение автомата не определено, если в текущем состоянии нет переходов, помеченных получаемым стимулом.

Автомат называется *полностью определенным*, если в каждом его состоянии для каждого стимула есть выходящий из этого состояния переход, помеченный этим стимулом.

Автомат *детерминирован*, если в каждом его состоянии для каждого стимула есть не более одного выходящего из этого состояния перехода, помеченного этим стимулом. Автомат *наблюдаемо детерминирован*, если в каждом его состоянии для каждого стимула и каждой реакции есть не более одного выходящего из этого состояния перехода, помеченного этим стимулом и этой реакцией.

В детерминированном автомате текущее состояние и принятый стимул однозначно определяют новое состояние автомата и выдаваемую реакцию. В наблюдаемо детерминированном новое состояние можно однозначно определить, зная предыдущее, принятый стимул и выданную реакцию.

Для детерминированного автомата (S, s_0, I, O, T) пусть δ и λ обозначают функции переходов и вывода, т.е. $(s_1, i, o, s_2) \in T$ тогда и только тогда, когда $\delta(s_1, i) = s_2$, и $\lambda(s_1, i) = o$. Оба этих отображения можно расширить до отображений состояния и принятой последовательности стимулов в последовательность реакций и итоговое состояние. Именно, для состояния автомата $s \in S$ и последовательности входных символов $\alpha \in I^*$ определим $\delta(s, \alpha) \in S$ и $\lambda(s, \alpha) \in O^*$ рекурсивно следующим образом.

Если α — пустая последовательность ε, то $\delta(s, \alpha) = s$ и $\lambda(s, \alpha) = \varepsilon$.

Если α — непустая и $\alpha = \alpha'a$, где $\alpha' \in I^*$ и $a \in I$, то $\delta(s, \alpha) = \delta(\delta(s, \alpha'), a)$ и $\lambda(s, \alpha) = \lambda(s, \alpha')\lambda(\delta(s, \alpha'), a)$.

Конечный автомат реализует некоторое соответствие последовательностей символов из I и последовательностей символов из O, $\varphi \subseteq I^* \times O^*$. Это соответствие называется его *поведением*. Для детерминированных автоматов $\varphi = \lambda(s_0, \cdot)$, то есть поведение является отображением входных последовательностей в выходные для начального состояния.

С точки зрения внешнего наблюдателя, не имеющего возможности «увидеть» текущее состояние автомата, два автомата с одинаковым поведением неотличимы. Точно также для него неотличимы состояния, в которых на одну и те же последовательности входных символов всегда может быть выдана одна и та же последовательность реакций. Такие состояния (даже в разных автоматах) называются *эквивалентными* или *неотличимыми*. Автоматы называются *эквивалентными*, если эквивалентны их начальные состояния.

Для любого автомата можно построить эквивалентный ему наблюдаемо детерминированный автомат. Этот факт доказывается примерно так же, как возможность использовать детерминированный автомат в качестве распознавателя регулярного языка.

Понятно, что внешнему наблюдателю отличить друг от друга эквивалентные автоматы невозможно. Чаще всего, это и не нужно, поскольку автомат обычно предназначен как раз для того, чтобы реализовывать некоторое соответствие между входными и выходными последовательностями, и любой автомат, делающий это правильно, считается подходящим. Поэтому с точки зрения тестирования автоматов важен только их класс эквивалентности.

В каждом классе эквивалентности автоматов есть единственный (с точностью до изоморфизма, т.е. взаимно-однозначного отображения состояний, сохраняющего начальное состояние и сопоставляющего эквивалентные) автомат с минимальным числом состояний. Такой автомат называют *минимальным* или *приведенным*. При тестировании обычно пытаются проверить соответствие как раз минимальному автомату, описывающему требуемое поведение тестируемой системы.

Специальные типы входных последовательностей автоматов

Чтобы уметь проверять соответствие поведение неизвестного автомата заданному, нужно уметь различать автоматы, у которых различия в поведении есть. Для этого используется несколько техник, основанных на специфических входных последовательностях.

Далее все определения даются для детерминированных автоматов. Для некоторых из этих понятий есть аналоги для автоматов общего вида, но их определения более сложны.

Идентификация состояний

Различающая последовательность (distinguishing sequence) конечного автомата — это такая конечная последовательность стимулов $d \in I^*$, что соответствующие ей

последовательности реакций в разных состояниях автомата разные. То есть, $\forall s_1, s_2 \in S\ s_1 \neq s_2 \Rightarrow \lambda(s_1, d) \neq \lambda(s_2, d)$.

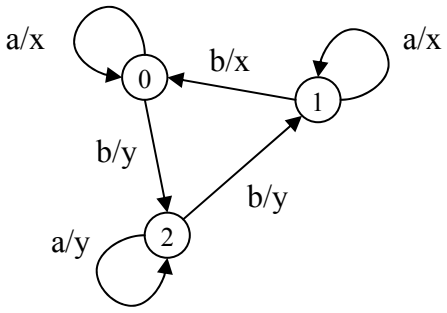


Рисунок 1. Пример конечного автомата, имеющего различающую последовательность.

Например, в автомате, изображенном на рисунке выше различающей последовательностью является последовательность ab, поскольку выполнено $\lambda(0, ab) = xy$, $\lambda(1, ab) = xx$, $\lambda(2, ab) = yy$.

Адаптивной последовательностью в алфавитах I и O называется дерево с корнем, чьи вершины помечены элементами I, а ребра помечены элементами O. Длиной адаптивной последовательности называется максимальное число вершин в цепочках ее вершин от корня до листовой.

К конечному автомату в определенном состоянии можно применять не только обычные последовательности, но и адаптивные. Результатом такого применения можно считать отображение цепочки вершин адаптивной последовательности, начинающейся от корня и продолжающейся вдоль его ребер, в выходной алфавит автомата. На это отображение накладывается следующие два ограничения.

- Если оно определено для некоторой вершины и его значение совпадает с меткой одного из ведущих из нее ребер, оно должно быть определено и для конца этого ребра.
- Если оно определено для одного из потомков некоторой вершины, оно должны быть определено и для самой вершины и значение его на этой вершине должно совпадать с пометкой ребра, ведущего к указанному потомку.

Интуитивно применение адаптивной последовательности можно описать так: мы применяем к автомату последовательно, начиная от корня, стимул, стоящий в очередной вершине, смотрим на полученную реакцию, и если она совпадает с пометкой одного из ребер, ведущих из текущей вершины, идем по нему и применяем стимул, стоящий в его конце, и т.д.

Обычная различающая последовательность называется также статической различающей последовательностью. *Адаптивной различающей последовательностью* автомата называется такая адаптивная последовательность в его алфавитах стимулов и реакций, что результаты ее применения во всех состояниях различны.

Например, для представленного выше автомата в качестве адаптивной различающей последовательности можно взять

$$a \xrightarrow{x} b$$

Длина этой адаптивной последовательности такая же, как у статической, но она позволяет быстрее определить состояние 2 — для этого достаточно получить y в ответ на a.

Различающие последовательности, как статические, так и адаптивные, позволяют однозначно определять состояние автомата, в котором они были применены. Однако не у всякого автомата они есть. Ясно, что из существования статической различающей последовательности следует существование адаптивной. Более того, есть примеры

автоматов, у которых нет статической различающей последовательности, но есть адаптивная. Ниже показан пример автомата, у которого нет даже адаптивной различающей последовательности.

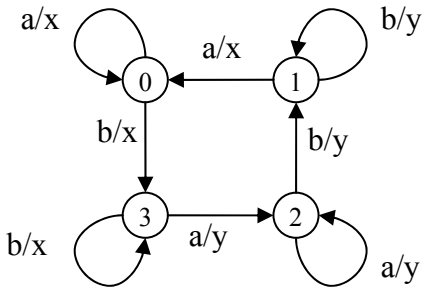


Рисунок 2. Пример автомата без различающей последовательности.

Чтобы убедиться в этом, достаточно проанализировать результаты применения a и b (по отдельности) к этому автомату. Различающая последовательность не может начинаться с a, поскольку после этого стимула перестают быть различимы пары состояний 0 и 1, 2 и 3 — после принятия a в любом из состояний пары автомат оказывается в одном и том же состоянии, выдав при этом одну и ту же реакцию. Точно так же b «слепляет» пары состояний 0 и 3, 1 и 2, поэтому различающая последовательность не может начинаться с b.

Чтобы различать состояния при отсутствии различающей последовательности, необходимы другие техники.

Последовательностью однозначных входов/выходов или *UIO-последовательностью* (unique input-output sequence) для данного состояния s называется такая последовательность стимулов u, что результат ее применения в состоянии s отличается от результатов ее применения во всех других состояниях. То есть, $\forall s_1 \in S\ s_1 \neq s \Rightarrow \lambda(s, u) \neq \lambda(s_1, u)$.

Различающая последовательность является UIO-последовательностью сразу для всех состояний автомата. В автомате, изображенном на Рис. 1, UIO-последовательностью для состояния 1 является также b, а для состояния 2 — a.

Есть автоматы, в которых нет различающих последовательностей, ни статической, ни адаптивной, но для каждого состояния есть UIO-последовательность. К сожалению, бывают и автоматы, ни одно состояние которых не имеет UIO-последовательности. Пример изображен на Рис. 2.

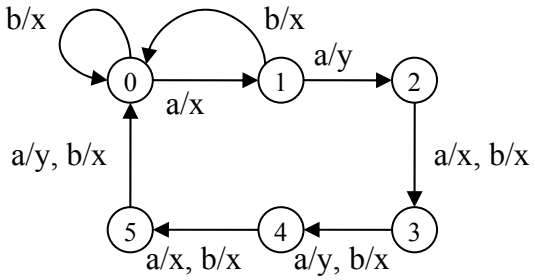
В таких случаях можно использовать *характеризующее* или *диагностическое множество последовательностей* (characterizing set) W автомата. Это такое множество входных последовательностей, что для каждых двух разных состояний автомата в нем найдется последовательность, результаты применения которой в этих состояниях различны. То есть, $\forall s_1, s_2 \in S\ s_1 \neq s_2 \Rightarrow \exists \alpha \in W\ \lambda(s_1, \alpha) \neq \lambda(s_2, \alpha)$.

Диагностическое множество всегда существует для минимального автомата, поскольку в нем поведение в разных состояниях отличается. Для автомата, имеющего различающую последовательность d, можно в качестве W взять {d}. Для автомата, в котором все состояния имеют UIO-последовательности любое множество, содержащее по UIO-последовательности для каждого состояния, является диагностическим. Для автомата, изображенного на Рис. 2, диагностическим является множество {a, b} — легко проверяется, что возможные реакции на это множество во всех его состояниях различны.

Для вычисления статической и адаптивной различающих последовательностей, UIO-последовательностей и диагностического множества можно использовать следующий алгоритм.

Упорядочим каким-либо образом множество I. Перебираем входные последовательности в лексикографическом порядке. Для каждой из них определяем возможные выходные

последовательности при ее применении в разных состояниях, разбиение множества состояний на группы состояний, для которых выходные последовательности одинаковы, а также покрытие множества состояний группами состояний, в которых можно оказаться после получения определенной выходной последовательности. Делать это можно итеративно, основываясь на результатах таких же вычислений для префиксов данной последовательности.



Рассмотрим автомат, изображенный на рисунке выше. Попробуем применить к нему этот алгоритм.

	a	b	aa	ab	ba	bb
0	x/1	x/0	xy/2	xx/0	xx/1	xx/0
1	y/2	x/0	yx/3	yx/3	xx/1	xx/0
2	x/3	x/3	xy/4	xx/4	xy/4	xx/4
3	y/4	x/4	yx/5	yx/5	xx/5	xx/5
4	x/5	x/5	xy/0	xx/0	xy/0	xx/0
5	y/0	x/0	yx/1	yx/0	xx/1	xx/0
	{0,2,4}x {1,3,5}y		{0,2,4}xy {1,3,5}yx	{0,2,4}xx {1,3,5}yx	{0,1,3,5}xx {2,4}xy	
	x{1,3,5} y{0,2,4}		yx{1,3,5} xy{0,2,4}	xx{0,4} yx{0,3,5}	xx{1,5} xy{0,4}	

После построения разбиений для всех последовательностей длины 2 видно, что многократное применение a не будет давать новой информации, просто сдвигая текущее состояние по циклу. Если же использовать b много раз, все состояния просто «слипнутся». Поэтому дальше имеет смысл пробовать только какие-то сочетания a и b.

	aab	aba	abb	baa	bab	aaba
0	xyx/3	xxx/1	xxx/0	xxy/2	xxx/0	xyxy/4
1	yxx/4	yxy/4	yxx/4	xxy/2	xxx/0	yxxx/5
2	xyx/5	xxx/5	xxx/5	xyx/5	xyx/5	xyxy/0
3	yxx/0	yxy/0	yxx/0	xxy/0	xxx/0	yxxx/1
4	xyx/0	xxx/1	xxx/0	xyx/1	xyx/0	xyxx/1
5	yxx/0	yxx/1	xxx/0	xxy/2	xxx/0	yxxx/1
	{0,2,4}xyx {1,3,5}yxx	{0,2,4}xxx {1,3}yxy {5}yxx	{0,2,4,5}xxx {1,3}yxx	{0,1,3,5}xxy {2,4}xyx	{0,1,3,5}xxx {2,4}xyx	{0,2}xyxy {1,3,5}yxxx {4}xyxx
	xyx{0,3,5} yxx{0,4}	xxx{1,5} yxy{0,4} yxx{1}	xxx{0,5} yxx{0,4}	xxy{0,2} xyx{1,5}	xxx{0} xyx{0,5}	xyxy{0,4} yxxx{1,5} xyxx{1}

Теперь мы обнаружили UIO-последовательности у двух состояний — aba для 5 и aaba для 4. Кроме того, уже видно, что baba является UIO-последовательностью для 2, а добавив ba к последовательностям aba и aaba, мы сможем разделить группу состояний {0,4}.

Далее, уже ясно, что разделяющая последовательность не может состоять только из стимулов a, поскольку такая последовательность всегда оставляет группы {0,2,4} и {1,3,5} неразличимыми, но применив b в любом месте, мы «слепим» 3 состояния, два из которых принадлежат одной из таких групп. Поэтому у этого автомата нет статической различающей последовательности.

Суммируем получаемые UIO-последовательности:

0 — aababa/xyxyxy

1 — ababa/yxyxy

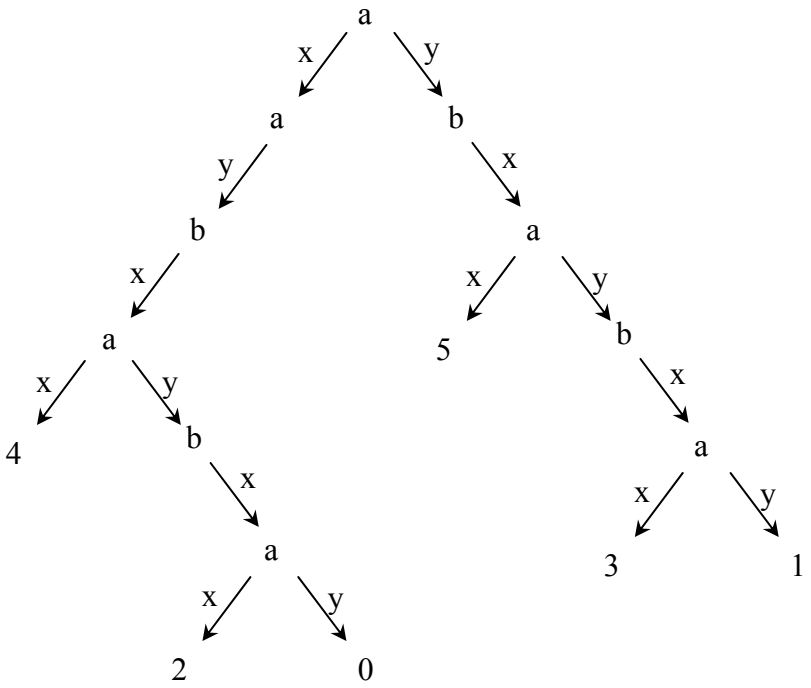
2 — baba/xyxy, aababa/xyxyx

3 — ababa/yxyxx

4 — aaba/xyxx

5 — aba/yxx

Из них можно построить адаптивную различающую последовательность. В качестве листовых вершин добавлены исходные состояния после получения соответствующих последовательностей реакций.



Если число состояний автомата равно n, а число стимулов — p, то его адаптивную различающую последовательность можно вычислить или показать, что ее не существует (при помощи несколько модифицированного алгоритма), за $O(pn^2)$ действий. Если адаптивная последовательность получается, она имеет длину не более $n(n–1)/2$

Если различающая последовательность автомата существует, она может иметь экспоненциальную от числа его состояний длину.

UIO-последовательности, если существуют, тоже могут иметь экспоненциальную длину.

Диагностическое множество всегда может быть построено за время $O(pn^2)$, при этом в нем содержится не более (n–1)-й последовательности длины не более n.

Установочные последовательности

Построение тестов

Для построения тестов на соответствие некоторому автомату нужно его знать. Такой автомат при тестировании называется *спецификацией*.

Мы предполагаем, что реальное поведение тестируемой системы, включающее все ошибки, если они есть, может быть полностью адекватно представлено некоторым конечным автоматом. Этот автомат называется *реализацией*. Мы не знаем, как он устроен, но в ходе тестирования хотим проверить, эквивалентен он спецификации или нет.

Чтобы тестирование стало возможно и реализации, и спецификация должны удовлетворять ряду требований.

В этом курсе рассматриваются методы тестирования только для детерминированных автоматов, поэтому далее будем предполагать, что спецификация и реализация детерминированы. Методы построения тестов для недетерминированных автоматов тоже есть, но формулируются существенно сложнее.

Поскольку мы хотим только проверить эквивалентность поведения, нам все равно, какой из эквивалентных автоматов брать в качестве спецификации. Поэтому можно считать, что спецификация минимальна.

Кроме того, можно предполагать, что в начале работы тестов реализация находится в начальном состоянии и алфавиты стимулов и реакций у спецификации и реализации совпадают. Если последнее не выполнено, мы любую реакцию реализации, не принадлежащую алфавиту реакций спецификации, можем рассматривать как ошибку, а стимул, не являющийся стимулом спецификации, мы просто не будем подавать.

Чтобы не возникало неопределенности при применении стимулов в неизвестных состояниях, будем считать, что и спецификация, и реализация полностью определены, то есть всегда можно применять все входные символы.

Наконец, чтобы суметь вернуться в исходное состояние, попробовав один из тестов, нужно считать, что либо спецификация и реализация сильно связны, то есть из любого их состояния можно, двигаясь по переходам, попасть в любое другое, либо что к обеим применимо специальное действие reset (далее обозначаемое R), которое переводит автомат из любого состояния в начальное и не содержит ошибок.

При наложенных ограничениях, однако, любой конечный набор тестов будет недостаточен, если не ограничить размер реализации, например, число состояний в ней. Если спецификация конечна и задан конечный набор тестов для проверки соответствия ей, мы всегда можем построить цепочку состояний, уводящую от начального, и длинную настолько, что применение всех заданных тестов закончится, не дойдя до какого-то ее состояния. Вот в этом состоянии можно сделать переход, дающий различие в поведении спецификации и реализации. Рассматриваемый тест не сможет обнаружить такую ошибку. Поэтому нужно считать, что количество состояний в реализации не превосходит некоторого заданного числа N.

Итак, от спецификации требуется

- детерминизм;
- минимальность;
- полная определенность;
- сильная связность или наличие reset.

От реализации требуется

- детерминизм;

- полная определенность;
- сильная связность или наличие reset;
- согласованность стимулов и реакций со спецификацией;
- согласованность начального состояния;
- ограниченность.

Кроме этих общих требований можно предполагать наличие вспомогательных действий, облегчающих тестирование.

- Одно из таких действий — reset (R), переводящий автоматы в начальное состояние.
- Другая возможность — наличие специального действия status (S), которое возвращает правильный идентификатор текущего состояния автомата и не изменяет состояние автомата.
- Наконец, очень полезная возможность — наличие действия set (T), которое имеет параметр-идентификатор состояния и корректно переводит автомат в это состояние.

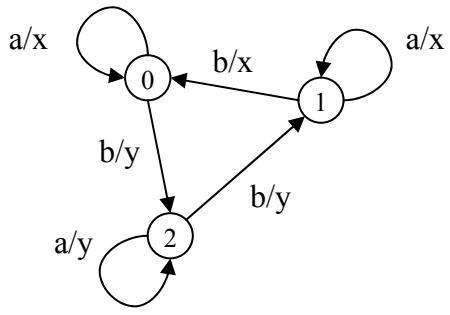
При наличии всех этих возможностей возможно *прямое тестирование*, самый простой метод тестирования автоматов. При тестировании мы хотим проверить эквивалентность автоматов, значит нужно убедиться, что любой переход реализации (определяемый своим начальным состоянием и стимулом) ведет себя так же, как и соответствующий переход в спецификации, то есть ведет в такое же состояние и возвращает ту же реакцию.

- При прямом тестировании для каждого перехода построим тест, который начинается с установки автомата в начальное состояние этого перехода с помощью set, затем выполняет сам переход и проверяет итоговое состояние с помощью status. Получаем набор тестов $\{T(s)aS\}$ для всех $s \in S$, $a \in I$. В каждом таком тесте нужно проверять, что реализация ведет себя так же, как спецификация, то есть выдает ту же реакцию и тот же идентификатор состояния в ответ на S. Если для каждого перехода это так, реализация эквивалентна спецификации.

Откажемся теперь от возможности по установке состояния, редко встречающейся на практике. В этом случае полный тест можно построить при помощи *обхода автомата*, то есть пути, проходящего по каждому его переходам по крайней мере один раз. Если спецификация сильно связна, ее обход существует.

- При тестировании на основе обхода берем одну из входных последовательностей, при применении которой выполняется обходов спецификации. В начале и после каждого стимула вставим действие S. Полученный тест проверяет каждый переход спецификации, и если реализация возвращает всегда те же реакции и те же идентификаторы состояний, она эквивалентна спецификации.

При использовании методов прямого тестирования и обхода гипотеза об ограниченности числа состояний в реализации не используется, поскольку есть мощный инструмент наблюдения реальных состояний — действие status.

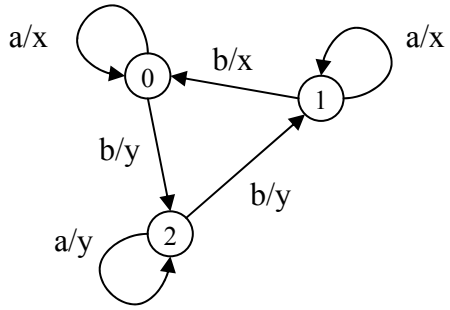


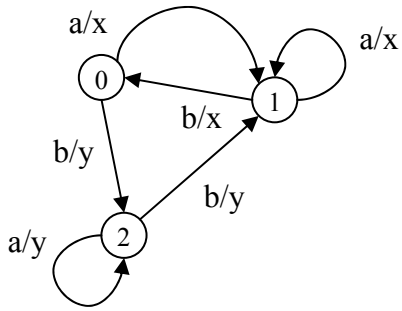
Построим полные тесты по двум описанным методам для изображенного выше автомата.

По методу прямого тестирования получаем последовательность T(0)aST(0)bST(1)aST(1)b ST(2)aST(2)bS. Выходная последовательность при этом должна быть равна x0y2x1x0y2y1.

Обход изображенного автомата выполняется, например, при применении входной последовательности ababab. По методу обхода получаем тест из входной последовательности SaSbSaSbSaSbS и корректной выходной последовательности 0x0y2y2y1x1x0.

Заметим, что при тестировании с помощью обхода действие status дает существенную информацию, без которой этот метод тестирования не может доказать эквивалентность автоматов.





Рассмотрим два автомата, изображенных выше. Автомат слева (тот же, что в разобранном выше примере) будет спецификацией, автомат справа — реализацией. Возьмем другой обход спецификации — bababa. Получаем тест SbSaSbSaSbSaS/0y2y2y1x1x0x0. На предложенной реализации этот тест дает выходную последовательность 0y2y2y1x1x0x1, которая отличается от корректной только в одном месте — последнем идентификаторе состояния.

Легко видеть, что сложность тестирования автомата с n состояниями и p стимулами методом прямого тестирования равна 3pn. Для метода обхода она не превосходит $2pn^2$ и существуют автоматы, для которых она имеет порядок $O(pn^2)$.

В следующей лекции рассматриваются более сложные методы тестирования автоматов, предназначенные для тех случаев, когда нет надежно работающих действий status и reset.

Литература

[1] M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, A. Pretschner (eds.). Model Based Testing of Reactive Systems. LNCS 3472, Springer, 2005.

[2] В. Б. Кудрявцев, С. В. Алешин, А. С. Подколзин. Введение в теорию автоматов. М.: Наука, 1985.