Тестирование на основе моделей

В. В. Кулямин

Лекция 7. Автоматные методы построения тестов. Продолжение

Напомним общий контекст различных методов тестирования на основе конечных автоматов.

Рассматривается описание требований к поведению тестируемой системы, представленное в виде конечного автомата, называемого спецификацией. Реальное поведение тестируемой системы, со всеми имеющимися в ней ошибками, также может быть полностью смоделировано конечным автоматом, называемым реализацией. Реализация неизвестна, известно только, что это конечный автомат, удовлетворяющий ряду условий.

Задача состоит в построении как можно более компактного набора тестов — входных последовательностей (и соответствующих им в спецификации выходных), — позволяющих отличить реализацию от спецификации всякий раз, когда они не эквивалентны. Соответственно, если поведение реализации на этом наборе тестов не будет отличаться от поведения спецификации, можно быть уверенным, что они эквивалентны.

На спецификацию и реализацию накладываются дополнительные ограничения.

Спецификация

- детерминирована;
- минимальна;
- полностью определена;
- сильно связна или имеет действие reset(R), достоверно приводящее из любого состояния в начальное.

От реализации требуется

- детерминизм;
- полная определенность;
- сильная связность или наличие reset;
- согласованность стимулов и реакций со спецификацией — входной и выходной алфавиты реализации те же;
- согласованность начального состояния — в начале работы реализация находится в начальном состоянии;
- ограниченность — число состояний в реализации не превосходит некоторого числа N.

Методы, использующие reset

Методы построения тестов, рассматриваемые ниже, условно можно разделить на одношаговые или однофазные и многошаговые. В первых все тесты строятся по одной и той же общей процедуре, а вторые используют несколько разных процедур или шагов.

Одношаговые методы

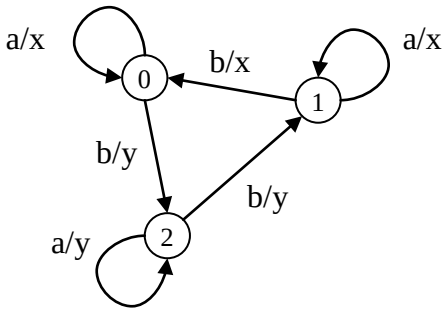
Первый метод построения тестов для автоматов без специальных действий set или status был разработан в 1973 Василевским [3]. Его более понятное изложение на английском языке приведено в статье Chow [4], поэтому большинство англоязычных авторов дают ссылки на эту статью. В статье Chow этот метод назван *W-методом* в честь Василевского.

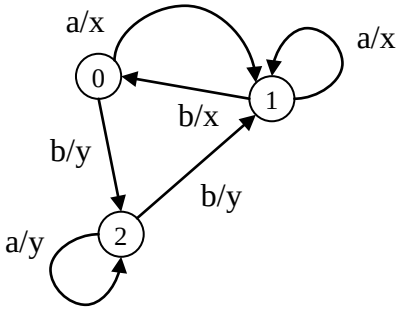
Без действия status проверить, что некоторый переход в реализации ведет в состояние, эквивалентное конечному состоянию соответствующего перехода в спецификации, становится не так просто. W-метод использует для этого диагностическое множество W, которое имеется для каждого минимального детерминированного полностью определенного автомата (см. Лекцию 6).

Как и в ранее описанных методах, необходимо для каждого перехода, однозначно определяемого своими начальным состоянием и стимулом, выполнить его, проверить корректность реакции, а затем проверить корректность его конечного состояния.

Чтобы выполнить все переходы, используется reset и покрывающее множество.

Покрывающее множество (или *базис достижимости*) C для конечного автомата — это минимальное такое множество входных последовательностей, что вместе с каждой последовательностью оно содержит все ее начала и позволят из начального состояния попасть в любое состояние автомата. То есть, #C = #S & ∀αa ∈ C α ∈ C & ∀s ∈ S ∃α ∈ C s = δ(s0, α). (Здесь # обозначает число элементов в множестве).





Для автомата, изображенного слева, покрывающим множеством является {ε, b, bb}.

Если число состояний в реализации не превосходит числа состояний в спецификации, т.е. N = n, W-метод действует так.

- В каждом тесте первым действием является reset, затем идет какая-нибудь из последовательностей из покрывающего множества (так мы оказываемся в произвольном состоянии), затем идет любой стимул (так выполняется произвольный переход), затем выполняется одна из последовательностей диагностического множества W. Чтобы проверить все переходы, нужно после каждого из них выполнить все последовательности из W.
  Другими словами этот метод можно записать так: строятся все входные последовательности из конкатенации множеств CIW, перед каждой из них вставляется R и все полученные последовательности конкатенируются.

Построим набор тестов для изображенного выше слева автомата с помощью W-метода.

В рассматриваемом автомате C = {ε, b, bb}, I = {a, b}, W = {a, b}. Значит, IW = {aa, ab, ba, bb}, и поэтому {R}CIW = RaaRabRbaRbbRbaaRbabRbbaRbbbRbbaaRbbabRbbbaRbbbb.

Корректная последовательность реакций:

xx.xy.yy.yy.yyy.yyy.yyx.yyx.yyxx.yyxx.yyxx.yyxy.

Если с помощью этого теста проверять реализацию, изображенную выше справа, будет получен следующий результат — xx.xx.yy.yy.yyy.yyy.yyx.yyx.yyxx.yyxx.yyxx.yyxy. Полученная ошибка выделена красным цветом.

- Для произвольного N >= n W-метод строит набор тестов $\{R\}CI^{N-n+1}W$. То есть, вместо однократных стимулов при построении всех возможных переходов используются все возможные последовательности стимулов длины N–n+1.

W-метод может быть применен для произвольной детерминированной полностью определенной спецификации, однако он дает достаточно большой набор тестов. Для

сокращения размеров тестов можно использовать другие способы идентификации состояний (см. Лекцию 6), например, UIO-последовательности и различающие последовательности.

D-метод применяется для спецификаций, имеющих различающую последовательность d. Он полностью аналогичен W-методу, только вместо диагностического множества W используется различающая последовательность d.

- D-метод строит набор тестов $\{R\}CI^{N-n+1}d$.

Для уже рассматриваемой ранее спецификации существует различающая последовательность d = ab. Поэтому построенный с помощью D-метода тест для N = n выглядит так: RaabRbabRbaabRbbabRbbaabRbbbab/xxy.yyy.yyyy.yyxx.yyxxx.yyxxy. Для приведенной выше ошибочной реализации результат его выполнения выглядит так: xxx.yyy.yyyy.yyxx.yyxxx.yyxxx.

В D-методе наряду со статической может также использоваться адаптивная различающая последовательность.

UIO-метод использует после каждого перехода вместо диагностического множества UIO-последовательность конечного состояния этого перехода. Однако, UIO-метод не гарантирует обнаружения всех ошибок — UIO-последовательности изменяются из-за ошибок, и поэтому может существовать ошибочная реализация, в которой для ошибочного конечного состояния одного из переходов UIO-последовательность дает корректные результаты.

Многошаговые методы

Другим методом, позволяющим сократить размер тестов, является *частичный W-метод* или *Wp-метод*. Он, в отличие от ранее представленных методов, выполняется в несколько шагов и использует идентифицирующие множества.

Идентифицирующее множество (identification set) W_s для состояния s — такое множество входных последовательностей, что для любого другого состояния автомата одна из этих последовательностей дает результат, отличающийся от результата ее применения в s. То есть, $\forall s_1 \in S\ s_1 \neq s \Rightarrow \exists \alpha \in W_s\ \lambda(s, \alpha) \neq \lambda(s_1, \alpha)$.

В качестве идентифицирующего множества состояния всегда можно выбрать некоторое подмножество диагностического множества автомата.

Шаги Wp-метода для N = n следующие.

- Сначала выполняются тесты $\{R\}CW$. Их успешное выполнение гарантирует, что все состояния реализации подобны состояниям спецификации, т.е. выдают те же результаты на все последовательности из W.
- Каждый переход, не проверенный на предыдущем шаге, т.е. не покрытый при выполнении множества C, выполняется и проверяется с помощью идентифицирующего множества своего конечного состояния (являющегося подмножеством использованного ранее W).

Построим набор тестов по Wp-методу для той же спецификации, изображенной выше.

$C = \{\varepsilon, b, bb\}$, $I = \{a, b\}$, $W = \{a, b\}$, $W_0 = W$, $W_1 = \{b\}$, $W_2 = \{a\}$.

Первый шаг дает тест RaRbRbaRbbRbbaRbbb/x.y.yy.yy.yyx.yyx.

На втором шаге требуется проверить только переходы по стимулу a и переход по стимулу b в состоянии 1. Получаем RaaRabRbaaRbbabRbbbaRbbbb/xx.xy.yyy.yyxx.yyxx.yyxy.

Полный тест выглядит следующим образом: RaRbRbaRbbRbbaRbbbRaaRabRbaaRbbab RbbbaRbbbb/x.y.yy.yy.yyx.yyx.xx.xy.yyy.yyxx.yyxx.yyxy.

Результат его выполнения для приведенной выше ошибочной реализации: x.y.yy.yy.yyx.yyx.xx.xx.yyy.yyxx.yyxx.yyxy.

Если у каждого состояния автомата есть UIO-последовательность, можно использовать *UIOv-метод*, гарантирующий, в отличие от UIO-метода, обнаружение всех ошибок.

UIOv-метод получается из Wp-метода заменой W на множество, содержащее UIO-последовательности всех состояний, а W_s — на UIO-последовательность состояния s.

В нашем примере полный тест по UIOv-методу выглядит так:

RaRbRabRbaRbbRbabRbbaRbbbRbbabRaabRbaaRbbabRbbbab/x.y.xy.yy.yy.yyy.yyx.yyx.yyxx.xxy.yyy.yyxx.yyxxy.

Сложность тестов в общем случае и их минимизация

Несмотря на то, что D-метод и Wp-метод обычно строят тесты, меньшие по размерам, чем W-метод, их сложность в общем случае описывается иначе.

В общем случае размер и сложность вычисления тестов по W-методу или Wp-методу одинакова и равна по порядку $O(p^{N-n+1}n^3)$ ($O(pn^3)$ для N = n). Эта оценка не может быть улучшена — существуют спецификации, которые нельзя отличить от всех ошибочных реализаций с N состояниями с помощью набора тестов, имеющего размер меньше, чем $O(p^{N-n+1}n^3)$.

Поскольку длина различающей последовательности может быть экспоненциальной от числа состояний, сложность D-метода в общем случае тоже экспоненциальная.

Для каждого конкретного случая можно, однако, сокращать размер тестов, построенных этими методами. Основное правило, которое используется при сокращении — если начало одно из элементарных тестов (последовательностей, заключенных между двумя reset'-ами) совпадает с другим элементарным тестом, то второй элементарный тест можно выбросить. Ясно, что в этом случае ошибка, обнаруживаемая выбрасываемым тестом всегда обнаруживается более длинным тестом.

Рассмотрим все полученные для нашего примера тесты.

- W-метод.
  RaaRabRbaRbbRbaaRbabRbbaRbbbRbbaaRbbabRbbbaRbbbb.
  Сокращение дает
  RaaRabRbaaRbabRbbaaRbbabRbbbaRbbbb.
- D-метод.
  RaabRbabRbaabRbbabRbbaabRbbbab.
  Сократить нельзя.
- Wp-метод.
  RaRbRbaRbbRbbaRbbbRaaRabRbaaRbbabRbbbaRbbbb.
  Сокращение дает
  RaaRabRbaaRbbabRbbbaRbbbb.
- UIOv-метод.
  RaRbRabRbaRbbRbabRbbaRbbbRbbabRaabRbaaRbbabRbbbab.
  Сокращение дает
  RbabRaabRbaaRbbabRbbbab.

Таким образом, в этом примере эквивалентность любой реализации с не более чем 3-мя состояниями может быть проверена с помощью теста

RaabRbaaRbbabRbbbab/xxy.yyy.yyxx.yyxxy.

Методы, не использующие reset

При отсутствии надежно работающей операции reset тестирование автоматов становится несколько сложнее. Существуют методы, позволяющие построить тесты для произвольной

детерминированной полностью определенной сильно связной спецификации, но размер получаемых тестов может быть достаточно велик. Такие методы используют установочные последовательности (homing sequences).

Рассмотрим здесь только один метод, работающий без reset, и предполагающий? Что спецификация обладает различающей последовательностью d.

Поскольку спецификационный автомат сильно связен, для каждой пары его состояний s_1 и s_2 существует *переводящая последовательность* стимулов $t(s_1, s_2)$, выполнение которой в s_1 переводит автомат в состояние s_2.

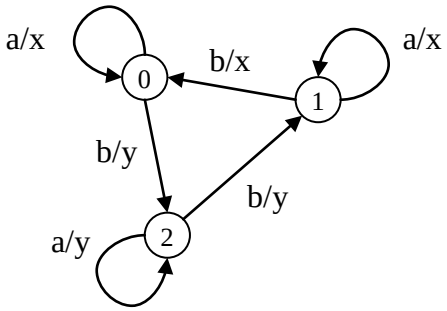
Обозначим для каждого $i >= 0$ через s_i' итоговое состоянием после выполнения d в s_i.

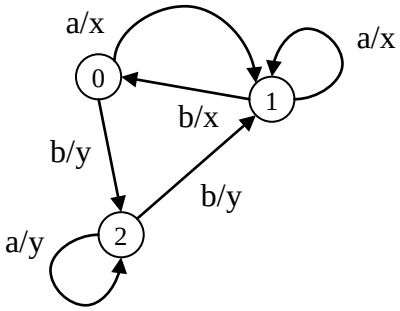
Тогда тест $d\ t(s_0', s_1)\ d\ t(s_1',s_2)\ d \ldots d\ t(s_{n-2}',s_{n-1})\ d$ проверяет, что в реализации для каждого состояния спецификации есть подобное, в котором d дает ту же последовательность реакций.

Чтобы после этого проверить, что некоторый переход работает правильно, нужно перейти в начало перехода s_i, выполнить его и выполнить d. Ошибки в реализации могут привести не в начало этого перехода, в другое место. Однако, мы уже знаем, что последовательность $dt(s_{i-1}', s_i)$, будучи применена в состоянии s_{i-1}, во-первых, проверит, что это действительно такое состояние с помощью d, а во-вторых, приведет после этого в s_i уже проверенным способом. Поэтому, попав в некоторое состояние s, для еще не проверенного перехода s_i –a-> s' выполним $t(s, s_{i-1})\ d\ t(s_{i-1}', s_i)\ a\ d$.

Получаемая таким образом последовательность обеспечит проверку всех переходов.

Построим такой тест для нашего примера спецификации.





Имеем $d = ab$, $s_0' = s_2$, $s_1' = s_0$, $s_2' = s_1$. Если мы будем обходить состояния в порядке s_0-s_2-s_1, то переводящие последовательности пусты. Поэтому первый этап дает abababab, и в его конце мы оказываемся в состоянии s_2.

Далее будем проверять переходы в следующем порядке: 1-a->1, 2-a->2, 0-a->0, 1-b->0, 0-b->2, 2-b->1. В этом случае промежуточные переводящие последовательности пусты, за исключением двух последних случаев, поэтому на втором этапе получаем такую входную последовательность: abaab.abaab.abaab.abbab.babbab.babbab.

Итоговый тест:

abababab.abaab.abaab.abaab.abbab.babbab.babbab/xyyyxxxy.yyxxx.xyyyy.xxxxy.yyxxy.yxxyy y.xxyyxx.

Результат, возвращаемый ошибочной реализацией: xxxxxxxx…, после этого тестирование можно не продолжать.

В этом примере многие ошибки могут быть найдены уже на первом этапе, поскольку в полученную на нем последовательность входят все переходы. Однако, если бы в спецификации имелись переходы по другим символам, отличным от a и b, ошибки в них обнаруживались бы только на втором этапе.

Использование других автоматных моделей

Другие автоматные модели — системы размеченных переходов, расширенные или взаимодействующие автоматы — при их использовании для тестирования чаще всего приводятся к конечным автоматам.

После этого становится можно использовать большое количество методов тестирования, разработанных для конечных автоматов.

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.

[3] М. П. Василевский. О распознавании неисправностей автоматов. Кибернетика, 9(4):93-108, 1973.

[4] T. S. Chow. Testing Software Design Modeled by Finite-State Machines. IEEE Transactions on Software Engineering, 4(3):178-187, 1978.