Сергеев Дмитрий Витальевич
УЛУЧШЕНИЕ МЕТОДА ОЦЕНКИ МОДЕЛЕЙ
Адрес статьи: www.gramota.net/materials/1/2011/6/29.html

Источник
Альманах современной науки и образования
Тамбов: Грамота, 2011. № 6 (49). С. 85-87. ISSN 1993-5552.
Адрес журнала: www.gramota.net/editions/1.html
Содержание данного номера журнала: www.gramota.net/materials/1/2011/6/

В логиках с дискретным временем моменты времени отделены друг от друга и у каждого момента есть следующий. Утверждение X(P) означает, что утверждение P будет выполнено в следующий момент времени, а G(P) - что P будет выполнено во все будущие моменты времени. Условие «после того, как x станет положительным, y должен оставаться всегда отрицательным» можно следующим образом:

$G(x > 0 \Rightarrow G(y < 0))$

$G(x > 0 \Rightarrow \overline{G(y >= 0)})$

Если же еще использовать логику дерева вычислений (CTL), то появляется возможность делать утверждения не только об отдельных событиях и их порядке во времени, но и о различных возможностях развития событий. Для контроля времени работы системы необходимо явно учитывать величину интервалов времени между событиями. Для этого можно использовать логики явного времени, в утверждениях которых участвуют переменные, обозначающие некоторые моменты времени, и явно указанные величины временных интервалов.

Но все рассмотренные выше логики не позволяют оперировать с процессами. Для этого можно использовать алгебры процессов. Это алгебраические исчисления, объектами которых являются события и процессы, создающие события или реагирующие на них. Обычно для процессов определены операции последовательной (‘;’) и параллельной (‘||’) композиции. Процесс является моделью выполняющейся программы. Последовательная композиция процессов моделирует выполнение сначала первого из них, затем второго. Параллельная композиция моделирует параллельное выполнение обоих процессов. Выбор из двух процессов моделирует выполнение либо первого, либо второго. В большинстве таких исчислений процессы могут взаимодействовать, обмениваясь событиями (один процесс порождает событие, другой или другие его потребляют). Например, цепочку «прием события» - «смена состояния» - «передача события» можно промоделировать следующим образом:

$S = \overline{e}^{'}.S_i^{'}, S_i^{'} = \ .(S_i^{''} \mid !e.nil)$,

где $S_i^{'}$ - начальное состояние процесса;

γ - извещение станции управления;

e - извещение зависимых от объекта-инициатора;

nil - терминальный («мертвый») процесс;

$\overline{e}^{'}$ - событие, получаемое от объекта-инициатора.

Из вышесказанного следует, что формальные модели можно описать логико-алгебраическими исчислениями.

Список литературы

1. **Габай М., Ходкинсон И., Рейнольдс М.** Временная логика: математические основы и вычислительные аспекты. Оксфорд: Clarendon Press, 1994. Т. 1. 653 с.
2. **Кларк Э. М., Грамберг О., Пелед Д.** Верификация моделей программ. М.: МЦНМО, 2002.
3. **Марков А. А.** Элементы математической логики. М.: Издательство МГУ, 1984.
4. **Мендельсон Э.** Введение в математическую логику. М.: Наука, 1984.
5. **Миронов А. М., Жуков Д. Ю.** Математическая модель и методы верификации программных систем // Информационные технологии и вычислительные системы. 2005. 220 с.
6. **Hillston J.** Процесс алгебры для количественного анализа // СЛС: школа информатики. 2005.

УДК 510.6

Дмитрий Витальевич Сергеев
Тульский государственный университет

УЛУЧШЕНИЕ МЕТОДА ОЦЕНКИ МОДЕЛЕЙ©

Тестирование должно показывать, насколько работа данного программного продукта соответствует требованиям. Все требования можно разделить на два вида - рекомендательные и обязательные (критические). Программный продукт должен обязательно соответствовать критическим требованиям, но не обязательно рекомендательным. Разделение требований на два типа позволяет наиболее точно оценить программный продукт с точки зрения соответствия требованиям. При этом происходит оценка каждого состояния, а после прохождения всего пути формируется общая оценка.

В связи с тем, что программные продукты постоянно модифицируются, для каждой версии продукта необходимо писать новую модель тестирования, так как появляются новые требования. При введении

рекомендательных требований и прогнозировании возможных дальнейших модификаций продукта, можно с самого начала написать универсальную модель тестирования для всех версий. А те свойства, которые не включены в начальную версию продукта, обозначить как рекомендательные. И при тестировании версий программного продукта такой моделью каждая последующая версия должна получать оценку выше, чем предыдущая, так как в ней должно быть выполнено больше рекомендательных свойств. Для этого необходимо алгоритм всех программы разделить на более мелкие алгоритмы и состояния выставлять в начале и в конце каждого из алгоритмов. При этом в результате модификации мелкого алгоритма требования к состоянию остаются неизменными. Также в темпоральной логике есть высказывания рекомендательного типа («желательно чтобы переменная a=0, желательно чтобы время между состояниями 1 и 2 было равно 0,1 с»). Без разделения требований на два вида эти утверждения нельзя обработать, и они не будут влиять на результаты тестирования. Для более точной оценки можно дискретизировать степень рекомендательности высказываний для более точной оценки системы на соответствие требованиям к ней. Для этого необходимо ввести весовые критерии. То есть каждое рекомендательное высказывание будет иметь свой вес.

Введем понятие артефакта. Артефакты - создаваемые человеком информационные сущности, документы, участвующие в качестве входных данных и получающиеся в качестве результатов различных деятельностей.

Утверждение 1. Модель программного продукта, включающая оценку его соответствия спецификации, может быть представлена в виде модели Крипке.

Модель Крипке формализует поведение динамической системы за счет определения множества состояний системы и переходов между этими состояниями. Каждое состояние модели помечается набором свойств, характеризующих данное состояние. При проверке каждого состояния продукта, ему должна быть сопоставлена одна из оценок домена меток B={−1, 0, 1}, где -1 означает несоблюдение критичного правила, 0 - несоблюдение рекомендательного, а 1 - успешное выполнение любого правила. Состоянием готовности программного продукта формируется из набора свойств состояния с сопоставленными оценками.

Таким образом, модель тестирования программного продукта имеет следующий вид, основанный модели Крипке:

$$M = (S, S0, R, T, T2, T3), \quad (1)$$

где S - множество состояний; $S0 \subseteq S$ - начальное состояние; $R \subseteq S \times S$ - отношение переходов для всевозможных состояний; T - функция интерпретации состояния системы как набора требований состояний для обязательных требований; T2 - функция интерпретации состояния системы как набора требований состояний для рекомендательных требований; T3 - массив весов рекомендательных требований.

Для оценки состояний программного продукта нужно оценить путь достижения этого состояния. Процедура верификации заключается в проверке правильности состояний, таким образом, осуществляется оценка корректности пути тестирования.

Утверждение 2. Результатом верификации программного продукта является оценка темпорального отношения на модели Крипке.

Семантика темпоральной формулы на модели соответствует следующему отношению:

$$\models: (M \times S \times \text{Formula}) \rightarrow \{\text{true, false}\}, \quad (2)$$

где ***Formula*** определяет корректность пути за счет оценки пути. Таким образом, результатом проверки формулы является однозначная идентификация допустимости пути.

Процедура верификации представляет проверку темпоральной формулы посредством последовательной проверки корректности выполнения каждого артефакта. Реализация процедуры требует формирования модели Крипке для тестируемой системы, дальнейшее ее преобразование в конечный автомат Мура, позволяющий сформировать отчет результатов верификации, а также конечный автомат, реализующий проверку темпоральной формулы.

Формирование возможных путей из текущего состояния позволяет построить модель Крипке.

Для построения модели Крипке для процесса разработки необходимо:

1) определить перечень верифицируемых атомарных высказываний, характеризующих состояния (критические и рекомендательные);

2) сгенерировать все возможные состояния программного продукта и определить их свойства;

3) установить переходы между каждой парой состояний, а также дуги для каждого состояния, указывающие на это состояние.

Следующим этапом выполнения метода верификации выступает преобразование полученной модели Крипке в конечный автомат Мура, регистрирующий каждое изменение состояния автомата в выходной последовательности. Алгоритм преобразования состоит из следующих этапов:

1) каждое состояние модели Крипке однозначно преобразуется в состояние автомата, причем каждое состояние автомата помечается как допускающее; каждая дуга $s, s' \in R$ преобразуется в переход автомата из состояния s в s';

2) формулируется выходной алфавит конечного автомата;

3) определяется функция выходов, отображающая каждое состояние автомата в символ выходного алфавита.

Конечный автомат Мура, полученный посредством преобразования модели Крипке имеет вид:

$$M = (\Sigma, S, \Delta, S0, S, G, T0, T1), \quad (3)$$

где S - конечное множество состояний; $S0 \in S$ - начальное состояние; Σ - конечный входной алфавит; $\Delta \subseteq S \times \Sigma \times S$ - отношение переходов; $G: S \rightarrow S$ - функция выходов; T0 - множество рекомендательных требований; T1 - множество весов рекомендательных требований.

В работе Э. М. Кларка, О. Грамберга и Д. Пеледа показано, что модель Крипке однозначно представима конечным автоматом.

Заключительным этапом процедуры верификации является проверка выполнения темпоральной формулы, задающей условия допустимого выполнения программного продукта. Математическая модель устройства, осуществляющего верификацию модели программного продукта за конечное число шагов, представляет собой конечный автомат, выполняющий последовательные переходы между состояниями и оценивающий выполнение темпоральной формулы.

Алгоритм работы автомата представляет следующую последовательность:

1) определение текущего состояния, т.е. выбор начального состояния конечного автомата Мура, соответствующего текущему набору артефактов;

2) осуществление перехода к следующему состоянию автомата Мура посредством проверки правила корректности артефакта, при этом регистрируется новый набор артефактов текущего состояния, общая оценка состояния заносится в стек автомата;

3) проверка допустимости оценки состояния: если оценка допустима, то она суммируется с суммой предыдущих оценок. Если есть рекомендательное требования для текущего состояния, то проверяется их верность и счетчик количества рекомендательных требования увеличивается на 1. Если рекомендательное требование выполнено, то в стек заносится 1, если нет, то 0. Данный этап повторяется для всех допустимых оценок последовательности состояний. Если же оценка недопустима (-1), то выдается сообщение о некорректной работы системы и формируется контрпример;

4) проверка суммы оценок. Если сумма = 0, то выполняются только критические требования. Если сумма = n, где $n \in (1..k)$, то n показывает, сколько рекомендательных требований выполнилось в данном пути;

5) расчет результатов. Если в стеке нет -1 (то есть все критические требования выполнились), то процентное соотношение всех рекомендательных требований к выполненным рекомендательным требованиям рассчитывается по формуле:

$Q=J/Jmax*100$,

где J - количество выполненных рекомендательных требований;

Jmax - количество всех рекомендательных требований.

Для представления в модели Крипке необходимо установить состояния на входе и на выходе из каждого элемента. Это позволяет контролировать входные и выходные значения более мелких алгоритмов и при модификации этих алгоритмов результаты не меняются. В состояния помимо критических требований записываются рекомендательные. Они могут описывать желательный диапазон изменения переменной или желаемое время перехода между состояниями. Таким образом, тестирование программного продукта происходит по одной и той же модели для различных версий и выдается более полная оценка соответствия программного продукта документации (выводится процент выполненных рекомендательных требований). Это позволяет оценить и сравнить две программы, имеющие одинаковые входные и выходные значения и одинаковые состояния.

Список литературы

1. **Габай М., Ходкинсон И., Рейнольдс М.** Временная логика: математические основы и вычислительные аспекты. Оксфорд: Clarendon Press, 1994. Т. 1. 653 с.
2. **Кларк Э. М., Грамберг О., Пелед Д.** Верификация моделей программ. М.: МЦНМО, 2002.
3. **Марков А. А.** Элементы математической логики. М.: Издательство МГУ, 1984.
4. **Мендельсон Э.** Введение в математическую логику. М.: Наука, 1984.
5. **Миронов А. М., Жуков Д. Ю.** Математическая модель и методы верификации программных систем // Информационные технологии и вычислительные системы. 2005. 220 с.
6. **Hillston J.** Процесс алгебры для количественного анализа // СЛС: школа информатики. 2005.