2010 Математические основы компьютерной безопасности №1(7)

DOI 10.17223/20710410/7/3

УДК 004.942

О ВОЗМОЖНОСТИ ИСПОЛЬЗОВАНИЯ ВРЕМЕННЫХ ЛОГИК И ВЕРИФИКАТОРОВ МОДЕЛЕЙ В ЗАДАЧЕ ОБНАРУЖЕНИЯ ОПАСНЫХ ОТКЛОНЕНИЙ СИСТЕМЫ

С. Е. Прокопьев

г. Москва

E-mail: prszs@mail.ru

В настоящей статье предложен вариант использования аппарата временных логик и верификаторов моделей в задаче контроля отклонений сложной информационной системы от «нормального» поведения, а также алгоритм слепого поиска закономерностей «черного ящика», представимых в рамках формализма временных логик.

Ключевые слова: *временные логики, верификаторы моделей, контроль отклонений системы, слепой поиск закономерностей.*

1. О проблемах использования временных логик и верификаторов моделей в анализе сложных систем

Временные логики являются мощным способом формального специфицирования свойств систем. Для этого исследуемая система представляется в виде одной из разновидностей автомата — т. н. модели Крипке $M{=}(S, S_0, R, L)$, где S — множество состояний системы; S_0 — множество начальных состояний системы; R — отношение переходов; L — функция разметки, сопоставляющая каждому состоянию некторый набор предикатов, истинных в данном состоянии.

Проблема верификации на модели формулируется следующим образом [1]. Пусть анализируемая система задана в виде модели Крипке M и есть формула временной логики f, специфицирующая некоторое свойство системы. Требуется найти в множестве S_0 подмножество всех состояний, в которых выполняется f, т. е. множество $\{s{\in}S_0 \mid M, s{\models}f\}$. Если указанному подмножеству принадлежат все начальные состояния системы, то модель M удовлетворяет свойству f.

В настоящее время существуют эффективные алгоритмы верификации на модели, например, в рамках логики ветвящегося времени (CTL) удается исследовать системы, имеющие число состояний порядка 10^{120}. Естественным является желание использовать этот мощный инструмент анализа последовательностей состояний в системах с бóльшим числом состояний, например, в информационных системах на базе современных операционных систем. Очевидно, что для сложных систем необходимо искать их упрощенные модели (абстракции), однако зачастую для абстракции, интересующей нас с точки зрения специфицирования свойств системы в рамках формализма временной логики, задачи нахождения корректного множества достижимых состояний, не превышающего порога применимости верификатора моделей, и корректного алгоритма переходов между ними оказываются нерешаемыми ввиду сложности анализируемой системы.

Возможным выходом может являться динамическое построение абстракции: в информационную систему (которая рассматривается как «серый ящик»: при необходимости мы можем заглянуть внутрь и посмотреть алгоритм работы системы в интересу-

ющем нас месте) внедряется набор датчиков, снимающих значения некоторых ее параметров в интересующие нас моменты. По полученным последовательностям показаний датчиков можно пытаться восстанавливать модель Крипке, соответствующую интересующей нас абстракции сложной системы, с последующим анализом заданных на этой абстракции свойств с использованием аппарата временных логик и верификаторов моделей. При этом чем сильнее (в допустимых пределах) будет комбинаторный взрыв корректных (т. е. достижимых в реальной системе) траекторий переходов в динамически построенной модели Крипке, тем эффективнее будет анализ свойств системы с использованием данного подхода — по сравнению с простой проверкой наблюдаемых траекторий на соответствие правилам, заданным в рамках отдельных траекторий, — так как в полной мере будут задействованы возможности современных алгоритмов верификации на модели. Тем не менее на практике для сложной информационной системы задача формулирования требуемых свойств последовательностей показаний датчиков, извлекаемых из «серого ящика», также может быть нерешаемой.

С учетом вышесказанного, имеет смысл подход, связанный с контролем отклонений абстракций от некоторого заданного «нормального» поведения. Под «нормальным» понимается поведение системы, наблюдавшееся в течение некоторого предварительного тестового периода — т. н. периода обучения. Если в ходе рабочей эксплуатации системы наблюдаются отклонения от зафиксированного «нормального» поведения, то их можно рассматривать как нарушения свойств безопасности системы. Определение «нормального» поведения системы предполагает обнаружение — желательно автоматическое — набора отдельных «нормальных» закономерностей ее поведения.

В настоящее время основным инструментом поиска закономерностей поведения системы на этапе обучения и контроля сохранения этих закономерностей на этапе ее рабочей эксплуатации являются нейронные сети. Однако с учетом того, что многие важные закономерности функционирования системы могут быть описаны в терминах временных логик, а также с учетом эффективности современных алгоритмов верификации на модели, для решения задачи контроля отклонений предлагается использовать аппарат временных логик и верификаторов моделей. Для этого предлагается осуществлять полуавтоматический подбор (с использованием верификаторов моделей и частичных знаний об алгоритме работы «серого ящика») формул временной логики, выполнимых на модели системы, построенной на этапе обучения по снимаемым последовательностям значений датчиков. Настоящая статья носит постановочный характер, рассматривая некоторые возможные подходы к построению системы обнаружения отклонений на базе временных логик.

2. Общее описание системы обнаружения отклонений

Первичными элементами системы обнаружения отклонений (СОО) являются датчики $(D_0, ..., D_{m-1})$, внедренные в контролируемую систему для регистрации происходящих в системе событий, замера значений интересующих параметров системы и т. д. СОО, в соответствии с некоторым алгоритмом (через заданные интервалы времени, при наступлении определенных событий и т. д.), снимает показания датчиков и вычисляет текущее значение вектора предикатов $\bar{P}=(P_0, ..., P_i, ..., P_{n-1})$, где $P_i=P(D_0^{Pi}, ..., D_{k(Pi)-1}^{Pi})$ и $k(Pi) < m$ — число датчиков, от которых зависит предикат P. В простейшем случае предикат имеет вид «датчик D_j сработал», в более сложном — зависит от нескольких датчиков и накладывает на их значения некоторое условие.

В рамках задачи построения СОО необходимо расставить в системе датчики и определить предикаты так, чтобы по снимаемым обучающим последовательностям

векторов предикатов можно было построить модель Крипке, соответствующую интересующей нас абстракции контролируемой системы. Неформально, под абстракцией системы понимается скоррелированный с ней (через датчики) автомат. Наиболее интересны абстракции с меньшим, чем у реальной системы, числом состояний и выраженными в более компактном виде, чем в реальной системе, правилами переходов.

Исходными данными для СОО являются последовательности упорядоченных по времени векторов значений предикатов $\{(\bar{p}_{t_0}, ..., \bar{p}_{t_i}, ..., \bar{p}_{t_{\text{cur}}})\}$, где $\bar{p}_{t_i}=(p^0_{t_i}, ..., p^{n-1}_{t_i})$, $p^j_{t_i}$ — значение предиката P_j в момент времени t_i, t_0 — время первого, а t_{cur} — последнего замера значений датчиков в данной последовательности. Каждая последовательность соответствует одной траектории вычислений, прошедшей через абстракцию.

На этапе обучения на основе траекторий, извлекаемых из последовательностей векторов предикатов, восстанавливается модель Крипке, соответствующая данной абстракции. Далее перебором проверяется выполнимость на ней различных формул временных логик с использованием современных алгоритмов верификации на модели (например, описанных в [1]), а также частичного знания алгоритма работы «серого ящика». Формулы, выполнимые на построенной модели, являются контролирующими формулами СОО и используются для проверки сохранения выявленных темпорально-логических закономерностей на множестве последовательностей векторов значений предикатов, наблюдаемых в ходе рабочей эксплуатации системы.

3. Алгоритм слепого поиска закономерностей, представимых в рамках формализма временных логик

В качестве одного из вариантов поиска абстракции и построения ее модели Крипке предлагается описанный ниже алгоритм.

В основу данного алгоритма положены следующие интуитивные рассуждения. Как было указано выше, под абстракцией системы понимается скоррелированный с этой системой автомат. Таким образом, для того чтобы найти абстракцию, необходимо из снимаемых с реальной системы последовательностей векторов значений предикатов извлечь информацию о ее состояниях и правилах переходов. Предположим, что часть координат обучающих векторов несут информацию о текущих состояниях (текущих вершинах графа переходов), а часть из оставшихся — о значениях переменных разметки этих состояний. Тогда можно использовать следующий подход к поиску абстракции и построению ее модели Крипке. Сначала разобъем каждый обучающий вектор на два подвектора: *подвектор состояний* (составлен из координат состояний) и *подвектор разметки* (составлен из оставшихся координат). Затем из последовательностей пар подвекторов состояний и разметки строим лес деревьев, состояния которого соответствуют подвекторам состояний и размечены множествами подвекторов разметки. Далее, для каждого множества подвекторов разметки каждого состояния находим координаты, в которых эти подвекторы отличаются, и вычеркиваем их глобально из разметки всего леса (интуитивно, это означает, что эти предикаты не скоррелированы с данной абстракцией). Если после вычеркивания подвекторы разметки не оказались пустыми, то склеиваем совпадающие поддеревья и замыкаем конечные состояния сами в себя. Получаем модель Крипке, в которой множеством начальных состояний будет множество корней каждого дерева в исходном лесе. Описание данного алгоритма в более формальном виде будет представлено ниже.

Определение 1. Пусть выбран критерий C, определяющий, подходит ли множество двоичных векторов для описания состояний абстракции (например, использующий описанные выше стратегии). Будем говорить, что множество двоичных векторов

образует группу состояний относительно критерия C, если критерий C выполняется на этом множестве. *Правилом прореживания* критерия выявления группы состояний C будем называть способ выбора подмножества множества, образующего группу состояний относительно критерия C.

Пример 1. Определим критерий C_{example} следующим образом: множество двоичных векторов удовлетворяет критерию C_{example}, если каждый вектор имеет вес 0 или 1. Определим правило прореживания критерия C_{example} как исключение из этого множества всех нулевых векторов.

Можно предложить следующие стратегии определения критериев группы состояний (могут применяться одновременно): 1) искать множества подвекторов с наибольшим расстоянием Хэмминга (интуитивно, чем в большем числе предикатов отражается состояние абстракции, тем сильнее подвекторы состояний будут отличаться друг от друга); 2) искать множества подвекторов минимальной мощности (интуитивно, чем больше совпадений подвекторов состояний, тем больше вероятность того, что мы нашли абстракцию); 3) по количеству совпадений поддеревьев (аналогично предыдущему).

Пусть задан критерий выявления группы состояний C с правилом прореживания R, а также алгоритм генерации A конечного множества темпорально-временных формул по конечному набору предикатов. Пусть имеется множество $W = \{V(k) : k = 1, \ldots, K\}$ последовательностей разной длины, составленных из двоичных векторов длины n. Предлагается следующий алгоритм слепого поиска абстракций и темпорально-логических закономерностей в них:

1. Для каждого $i = 1, \ldots, n-1$, для каждого подмножества $Pos_j^{(i,n)}$ мощности i множества всех координат $\{1, \ldots, n\}$, $j = 1, \ldots, C_n^i$, выполняем шаги 2–11.
2. Для множества векторов $W_{Pos_j^{(i,n)}}$, полученного ограничением векторов множества W на координатах $Pos_j^{(i,n)}$, проверяем, образует ли оно группу состояний относительно критерия C. Если нет, то переходим на следующую итерацию шага 3.
3. Множество $ST = Pos_j^{(i,n)}$ будем называть *текущим множеством координат состояния*, а множество LB оставшихся координат — *текущим множеством координат разметки*. Построим из множества W множество $WW = \{(V_{ST}(k), V_{LB}(k)) : k = 1, \ldots, K\}$ пар последовательностей подвекторов, полученных ограничением последовательностей векторов $V(k)$ на множествах координат ST и LB.
4. Из множества пар последовательностей WW получаем множество пар последовательностей $WW' = \{(V'_{ST}(k), V'_{LB}(k)) : k = 1, \ldots, K\}$ путем применения для каждого k правила прореживания R к последовательности $V_{ST}(k)$ и удаления из последовательности $V_{LB}(k)$ подвекторов с теми же порядковыми номерами, что и у подвекторов, удаленных из $V_{ST}(k)$.
5. Множество пар последовательностей WW' будем рассматривать как множество последовательностей пар $WW' = \left\{V_{(ST,LB)}(k) = \{(v_{ST}(k,l), v_{LB}(k,l))\}_l : k = 1, \ldots, K\right\}$. Вектор $v_{ST}(k,l)$ будем называть левой половиной элемента l последовательности $V_{(ST,LB)}(k)$, а $v_{LB}(k,l)$ — правой. Из множества WW' индуктивно по $k = 1, \ldots, K$ строим лес F_K, каждое состояние которого размечено парой $(v_{ST}, \{v_{LB}\})$, где v_{ST} — подвектор состояния (назовем его левой половиной разметки вершины леса), а $\{v_{LB}\}$ — множество подвекторов разметки (назовем его правой половиной разметки вершины леса), следующим образом:
   а) Число вершин F_1 равно длине последовательности $V_{(ST,LB)}(1)$. Каждой вершине F_1 взаимно однозначно соответствует элемент последовательности

$V_{(ST,LB)}(1)$, который является ее разметкой. В графе F_1 вершины w' и w'' с разметками (v'_{ST}, v'_{LB}) и (v''_{ST}, v''_{LB}) соединяются направленным ребром от w' к w'', если (v''_{ST}, v''_{LB}) следует сразу за (v'_{ST}, v'_{LB}) в последовательности $V_{(ST,LB)}(1)$. Получаем лес F_1, состоящий из одного дерева, состоящего из одной ветви.

б) Лес F_k строится из F_{k-1} следующим образом. Если в лесе F_{k-1} нет дерева с корнем, левая половина разметки которого совпадает с левой половиной первого элемента $V_{(ST,LB)}(k)$, то F_k будет состоять из F_{k-1} и дерева (из одной ветви), построенного из $V_{(ST,LB)}(k)$ способом, описанным выше для $k = 1$. Если такое дерево есть, то идем из его корня в соответствии с направлениями ребер, последовательно извлекая по одному элементу из $V_{(ST,LB)}(k)$ и сравнивая левые половины разметки преемников текущей вершины дерева с левой половиной текущего элемента $V_{(ST,LB)}(k)$. Пусть w — текущая вершина дерева, а (v_{ST}, v_{LB}) — текущий элемент последовательности $V_{(ST,LB)}(k)$. Если у w есть вершина-преемник, левая половина разметки которой равна v_{ST}, то добавляем v_{LB} в его правую половину (в множество его подвекторов разметки). Если таких преемников у w нет, создаем новую вершину, размеченную парой (v_{ST}, v_{LB}), и направляем в нее из w ребро.

6. Избавляемся от множественной разметки: для каждого состояния леса F_K находим в векторах из его правой половины разметки координаты, в которых эти векторы отличаются, и вычеркиваем эти координаты из множества LB и векторов разметки всех состояний дерева. Если после вычеркивания всех конфликтующих координат LB стало пустым, переходим на следующую итерацию шага 3. Иначе имеем разметку каждого состояния, состоящую ровно из одного вектора.
7. Запоминаем S_0 — текущее множество вершин леса F_K.
8. Преобразуем лес F_K в граф G путем склеивания совпадающих поддеревьев в F_K и замыкания конечных состояний F_K самих в себя.
9. Обозначим через S множество состояний графа G, определим отношение переходов R в соответствии с переходами графа G, определим функцию разметки L как ограничение разметки графа G на подвекторах разметки (т. е. удалим подвекторы состояний). Получим модель Крипке $M = (S, S_0, R, L)$.
10. Используя алгоритм A, генерируем конечный набор формул TF, определенных на множестве предикатов, соответствующих оставшимся координатам текущих подвекторов разметки. Перебираем для модели M все формулы из множества TF. Если никакая формула из TF не выполнима на M, переходим на следующую итерацию шага 3, иначе получаем множество выполнимых формул $TF_{\text{sat}} \subset TF$.
11. Считаем, что темпорально-временная закономерность найдена, если $TF_{\text{sat}} \neq \varnothing$.

В конечном итоге, если мы смогли построить достаточно компактную модель Крипке, в которую «уложилось» множество обучающих последовательностей, и нашли некоторый набор выполнимых на ней формул, то можно предполагать, что мы узнали о существовании и получили (возможно, неполное) описание абстракции, скоррелированной с системой через построенный граф. При этом подбор темпорально-логических формул является способом поиска по этому графу закономерностей абстракции.

Описанный выше алгоритм является крайним вариантом, рассматривающим систему как «черный ящик». Очевидно, что слепой поиск закономерностей, представимых в виде формул временной логики, имеет неподъемную трудоемкость для нетривиальных абстракций и нетривиальных формул даже с учетом эффективности современных алгоритмов на графах и алгоритмов верификации на модели. На практике выбор датчиков, предикатов состояний, предикатов разметки, множества проверяемых формул

должен происходить с учетом частичного знания алгоритма работы системы («серого ящика») и предположений о возможных закономерностях системы. В простейшем виде предикаты состояний могут иметь вид «произошло событие $event$» или «была вызвана функция $func$». Тогда при поиске абстракции можно использовать критерий образования группы состояний и правило прореживания из примера 1.

Алгоритм построения графа можно дополнить эвристическим поиском циклов перед склеиванием совпадающих поддеревьев на шаге 3, а также склеиванием «похожих» поддеревьев согласно некоторому критерию. При этом будет происходить потеря корректности построенного графа относительно абстракции, но зато может сильно сократиться его объем. Заметим, что допустимыми будут некорректные сокращения модели, инвариантные относительно выполнимости на ней формул, описывающих найденные закономерности.

ЛИТЕРАТУРА

1. *Кларк Э.М., Грамберг О., Пелед Д.* Верификация моделей программ: Model Checking: пер. с англ. / под ред. Р. Смелянского. М.: МЦНМО, 2002. 416 с.