И.Б.Бурдонов, А.С.Косачев.
Верификация композиции распределенной системы.
Труды Всероссийской научной конференции "Научный сервис в сети ИНТЕРНЕТ", Изд-во МГУ, 2005, стр.67-69.
2 стр.

ВЕРИФИКАЦИЯ КОМПОЗИЦИИ РАСПРЕДЕЛЁННОЙ СИСТЕМЫ

И.Б. Бурдонов, А.С. Косачев

Компоненты распределённой системы моделируются асинхронными автоматами с блокировками стимулов и разрушением, а правильность – соответствием $\mathrm{ioco}_{\beta\gamma\delta}$ – отношением между моделью реализации и моделью спецификации. Это отношение основано на безопасных трассах спецификации, которые не приводят к разрушению. Спецификации должны быть безопасными (не «саморазрушающимися») и безопасно-конвергентными (нет дивергенции на безопасных трассах), а реализация удовлетворять гипотезе об отсутствии разрушения и дивергенции на безопасных трассах спецификации. (См. наш доклад «Тестирование компонентов распределённой системы»).

Основная проблема верификации распределённой системы звучит так: если компоненты работают правильно, то почему система работает неправильно? Одна из причин, очевидно, заключается в ложности посылки: на самом деле компоненты работают неправильно. Тестирование компонентов было не полным, и остались ошибки: реализации компонентов (РК) не конформны спецификациям компонентов (СК). Такое часто встречается на практике, однако этим проблема не исчерпывается: может оказаться, что даже при конформных РК реализация системы (РС) всё равно работает неправильно, то есть не конформна системной спецификации (СС). В этом случае претензий к разработчикам компонентов быть не может. Тогда кто виноват и что делать? Очевидно, причина лежит в иной плоскости: СК не согласованы с СС. А это уже ошибка архитектора, который неправильно декомпозировал требования к системе на требования к компонентам и схему их компоновки. По вполне понятным причинам такие ошибки гораздо труднее искать, и они приводят к более печальным последствиям.

Какое соотношение СК и СС считать правильным? При заданных СК и схеме компоновки *корректной системной спецификацией* (КСС) будем считать такую СС, которая удовлетворяет *условию монотонности*: любая РС, составленная из конформных РК, конформна этой СС. Естественно определить *самую сильную корректную системную спецификацию* (ССКСС) как КСС, предъявляющую к системе самые сильные требования.

Первой неожиданностью стало то, что СС, полученная из СК по тем же правилам компоновки, по которым РС получается из РК, может оказаться некорректной. Такая СС предъявляет к системе избыточные требования, которым система может не удовлетворять, хотя все РК конформны своим СК. Причиной этого являются разные уровни абстракции, используемые для определения соответствия (трассы наблюдений) и компоновки компонентов (асинхронные автоматы). В асинхронном тестировании (взаимодействие теста и реализации опосредуется средой передачи) эта проблема известна как *проблема несохранения соответствия* [1]: асинхронный тест обнаруживает ложные ошибки, которые не могут быть обнаружены при синхронном тестировании. Заметим, что из конформности РС не обязательно следует конформность РК: система может работать правильно, хотя в компонентах есть ошибки, но они не проявляются в работе системы в целом. В асинхронном тестировании это называется *проблемой вседозволенности* [1]: асинхронные тесты не могут обнаружить некоторые ошибки, обнаруживаемые при синхронном тестировании. Эта ситуация неизбежна и не так уж плоха, если нас интересует работа системы в целом.

Для формализма асинхронных автоматов мы используем нотацию алгебры процессов CCS (Calculus of Communicating Systems [2]): стимулы снабжаются префиксом “?”, а реакции – префиксом “!”. Для каждого наблюдаемого действия определяется противоположное ему с помощью биекции «подчёркивание»: $\underline{?x} = !x$, $\underline{!y} = ?y$. Внутренняя активность моделируется переходом по символу τ, а разрушение – переходом по символу γ. Алфавит автомата A, то есть множество стимулов и реакций, которые автомат в принципе может принимать и выдавать, обозначим LA. Композиция системы моделируется *оператором параллельной композиции*, который по двум взаимодействующим автоматам A и B строит третий автомат $C=A\uparrow\downarrow B$ в алфавите $LC=(LA\setminus\underline{LB})\cup(LB\setminus\underline{LA})$, моделирующий их совместную работу. Состояния композиции C – это пары состояний компонентов A и B, а переходы делятся на синхронные и асинхронные. Синхронный переход – это τ-переход, порождённый парой противоположных переходов в компонентах: один посылает символ !z, а другой принимает этот же символ ?z; оба компонента меняют свои состояния. Асинхронный переход композиции C порождается одним переходом в одном из компонентов A или B: переход по τ, γ или внешнему символу $z\in LC$; другой компонент сохраняет своё состояние.

$z\in(A\cup\{\gamma,\tau\})\setminus\underline{B}$, в A переход $a \text{—}z\rightarrow a`$ $\Rightarrow$ в C переход $ab\text{—}z\rightarrow a`\,b$;
$z\in(B\cup\{\gamma,\tau\})\setminus\underline{A}$, в B переход $b\text{—}z\rightarrow b`$ $\Rightarrow$ в C переход $ab\text{—}z\rightarrow a\,b`$;
$z\in A\cap\underline{B}$, в A переход $a\text{—}z\rightarrow a`$, в B переход $b\text{—}\underline{z}\rightarrow b`$ $\Rightarrow$ в C переход $ab\text{—}\tau\rightarrow a`\,b`$.

И.Б.Бурдонов, А.С.Косачев.
Верификация композиции распределенной системы.
Труды Всероссийской научной конференции "Научный сервис в сети ИНТЕРНЕТ", Изд-во МГУ, 2005, стр.67-69.
2 стр.

В общем случае оператор ↑↓ коммутативен, но не ассоциативен. Схема компоновки системы – это последовательность применения оператора ↑↓. Например, (A↑↓B)↑↓(C↑↓(D↑↓E)).

В отличие от «прямой» композиции РК (оператор ↑↓) предлагается «косая» композиция СК (оператор ↗↙), определяемая неявно как построение автомата эквивалентного (по $ioco_{\beta\gamma\delta}$) ССКСС. Косая композиция используется для трёх целей. 1) *Проверка компонуемости*: проверка безопасности и безопасно-конвергентности ССКСС гарантирует выполнение реализационной гипотезы для РС как прямой композиции конформных РК. 2) *Верификация корректности СС*: проверка того, что ССКСС конформна заданной СС. 3) *Генерация системных тестов*: при отсутствии заданной СС системные тесты можно генерировать непосредственно по ССКСС.

Утверждается: существует такое преобразование спецификаций F, что косая композиция СК равна прямой композиции преобразованных СК: A↗↙B=F(A)↑↓F(B). Будем говорить, что соответствие $ioco_{\beta\gamma\delta}$ *монотонно относительно* F. Предлагается алгоритм выполнения такого преобразования. Тем самым, мы имеем алгоритм вычисления ССКСС по заданным СК и схеме компоновки.

Заметим, что, если в спецификации нет блокировок и разрушения, то преобразование можно не делать: косая композиция тождественна прямой, $ioco_{\beta\gamma\delta}$ монотонно относительно тождественного преобразования. Именно поэтому стремятся пополнить спецификацию, трактуя неспецифицированные стимулы как принимаемые с дальнейшим поведением по умолчанию [3]. Возникающую здесь проблему потери информации о неспецифицированности стимула после трассы предлагается решать трактовкой такого стимула как разрушающего. Так пополненная спецификация – частный случай *γ-нормального* автомата: если после безопасной трассы σ некоторая реакция !a вызывает разрушение (есть трасса σ·!a·γ), то и любая другая реакция !b, продолжающая трассу (есть трасса σ·!b), вызывает разрушает (есть трасса σ·!b·γ). После нашего пополнения все реакции безопасны и спецификация автоматически γ-нормальна. Если в спецификации допускается разрушение, то важны только безопасные трассы. Поэтому, если они не содержат блокировок, то преобразование F сводится к γ-нормализации, то есть преобразованию спецификации в $ioco_{\beta\gamma\delta}$-эквивалентную ей γ-нормальную спецификацию. Соответствие $ioco_{\beta\gamma\delta}$ монотонно относительно γ-нормализации.

В общем случае, когда блокировки допускаются в безопасных трассах, для доказательства существования преобразования F достаточно рассмотреть объединение всех реализаций, конформных данной спецификации A: U(A)=∪{RA|RA $ioco_{\beta\gamma\delta}$ A}. Для объединения автоматов вводится новое начальное состояние, из которого проводятся τ-переходы в начальные состояния этих автоматов. Прямая композиция R=RA↑↓RB любых реализаций, конформных своим спецификациям, RA $ioco_{\beta\gamma\delta}$ A и RB $ioco_{\beta\gamma\delta}$ B, – это подавтомат прямой композиции таких объединений S=U(A)↑↓U(B). Автомат R удовлетворяет реализационной гипотезе для спецификации S и, как подавтомат, конформен ей: R $ioco_{\beta\gamma\delta}$ S.

Определение преобразования через конформные реализации неконструктивно: нужен алгоритм, который строил бы преобразованную спецификацию F(S) непосредственно по исходной спецификации S. Идея алгоритма заключается в следующем.

Подмножество U состояний спецификации S *τ-замкнуто*, если любой τ-переход u—τ→u`, начинающийся в U, u∈U, заканчивается также в U, u`∈U. В качестве нестабильных состояний F(S) выбираются все τ-замкнутые подмножества состояний S. Начальное состояние нестабильно – это τ-замыкание начального состояния S (состояния, достижимые из него по τ-переходам). Из состояния U проводим τ-переход в каждое стабильное состояние (I(U),I), где I – это множество стимулов и не более одной реакции, а I(U) определяется в зависимости от I (не для всяких I и U существует I(U)).

1) I содержит все стимулы и реакцию !y, а из U есть переход по каждому стимулу и реакции !y. Тогда I(U)=U.

2) I не содержит реакций, а I(U) – наибольшее подмножество U такое, что стимул ?x∈I тогда и только тогда, когда из некоторого состояния U есть ?x-переход.

3) I содержит не все стимулы и содержит реакцию !y, а I(U) – наибольшее подмножество U такое, что стимул ?x∈I тогда и только тогда, когда из некоторого состояния U есть ?x-переход, и, кроме того, из некоторого состояния U есть !y-переход.

Из состояния (I(U),I) проводим переход по каждому символу z∈I в τ-замыкание множества состояний, достижимых из I(U) по z-переходам. Из состояния U проводим аналогичные переходы по каждому z, но только в том случае, если при этом можно попасть в состояние S, недостижимое по z из стабильных состояний (I(U),I) для всех возможных I.

Исключение из этого правила – переход по стимулу z=?x, за которым в спецификации может следовать разрушение; в этом случае переход по ?x заканчивается в специальном γ-состоянии, в котором определяется γ-петля.

И.Б.Бурдонов, А.С.Косачев.

Верификация композиции распределенной системы.

Труды Всероссийской научной конференции "Научный сервис в сети ИНТЕРНЕТ", Изд-во МГУ, 2005, стр.67-69.

2 стр.

ЛИТЕРАТУРА:

1. C. Jard, T. Jéron, L. Tanguy, C. Viho "Remote testing can be as powerful as local testing", FORTE XII/ PSTV XIX' 99, China, pp. 25-40.
2. R. Milner "Communication and Concurrency". Prentice-Hall, 1989.
3. M. van der Bijl, A. Rensink, J. Tretmans "Compositional testing with ioco" // Formal Approaches to Software Testing: Third International Workshop, FATES 2003, Montreal, Quebec, Canada. LNCS v. 2931, Springer, pp. 86-100.