УДК 004.451.23

Ил. С. Свирин, П. А. Силин,
В. В. Сюзев

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ МНОГОПОТОЧНОЙ ПРОГРАММЫ И ПРАВИЛА БЕЗОПАСНОГО МНОГОПОТОЧНОГО ПРОГРАММИРОВАНИЯ

Одной из основных проблем разработки многопоточного программного обеспечения являются взаимные блокировки потоков. Взаимные блокировки потоков чрезвычайно трудно выявить, поскольку их возникновение напрямую связано с относительной динамикой выполнения потоков в программном обеспечении, которая зависит от множества трудно учитываемых факторов. Представлена система правил разработки многопоточной структуры программного обеспечения, направленная на уменьшение числа вносимых на этапе разработки ситуаций взаимной блокировки. Преимущество данной системы перед эмпирически выведенными аналогами заключается в том, что она выведена в процессе разработки математической модели взаимных блокировок.

E-mail: silinp@yandex.ru

Ключевые слова: *многопоточное программное обеспечение, взаимные блокировки, верификация.*

Одна из основных проблем разработки многопоточного программного обеспечения (ПО) — это обеспечение доступа различных потоков к разделяемым ресурсам. Для решения данной проблемы современные системы и средства программирования предоставляют средства синхронизации, которые позволяют решать проблему доступа потоков к разделяемым ресурсам за счет временного перевода некоторых потоков, обращающихся к ресурсам в состояние ожидания. Однако использование средств синхронизации привело к проблеме возникновения взаимных блокировок — ситуаций, когда потоки, переведенные в состояние ожидания, ожидали события, которое никогда не произойдет. Ошибки, связанные с взаимной блокировкой потоков, чрезвычайно трудно выявить, поскольку возникновение взаимных блокировок напрямую связано с относительной динамикой выполнения потоков в ПО, которая зависит от множества трудно учитываемых факторов, часть которых может проявиться только в будущем, например: при переходе на новую платформу или добавлении новой подсистемы. Это свойство делает принципиально невозможным создание алгоритма выявления ошибок синхронизации на этапе тестирования ПО.

Существует несколько подходов к решению данной проблемы.

Динамический анализ основан на мониторинге обращения выполняемой программы к ресурсам и различных вызовов [1, 2]. Этот подход характеризуется низкими затратами вычислительных ресурсов, однако

имеет большее число ложных срабатываний и не гарантирует нахождения всех потенциальных ситуаций блокировки.

Статический анализ использует исходные коды или объектные файлы ПО для построения моделей, которые проверяются на наличие блокировок [3–6]. Этот подход является достаточно эффективным, хотя порождает большое число ложных срабатываний и плохо применим к ПО со сложной объектной структурой.

Верификация моделей по методу Model Checking основана на построении формальной модели ПО [7, 8] с последующей верификацией данной модели с помощью специализированных средств [9, 10]. Этот подход принципиально не дает ложных срабатываний и исключает возможность пропуска блокировок, но чрезвычайно требователен к вычислительным ресурсам.

Подход, представленный в настоящей статье, относится к верификации моделей. Его отличие заключается в наглядности построенной формальной модели, достаточной для того, чтобы на ее основе могла быть построена система правил корректного использования средств синхронизации.

В рамках проблемы выявления потенциальных ситуаций взаимной блокировки наибольший практический интерес представляет разработка на основе формальной модели взаимных блокировок системы правил корректного использования средств синхронизации. Использование данной системы правил позволяло бы разработчикам ПО избегать создания конструкций на основе средств синхронизации, приводящих к появлению потенциальных ситуаций взаимной блокировки. При попытке создания системы правил на основе существующих формальных моделей взаимных блокировок возникает следующая проблема. Формальные модели, включающие в себя алгоритмы выявления потенциальных ситуаций взаимной блокировки, как правило, основаны на использовании математического аппарата (топологии, теории графов), в терминах которого затруднительно сформулировать набор четких и наглядных правил, понятных конечному разработчику, не знакомому с тонкостями данных моделей. Результаты, полученные на основе верификации, вовсе не ориентированы на выявление причин возникновения взаимных блокировок.

Для решения данной проблемы авторами была разработана математическая модель взаимных блокировок, сочетающая необходимый уровень формализма и наглядность получаемых на основе этой модели результатов. Настоящая статья содержит краткое описание этой модели, включая достаточные условия отсутствия потенциальных ситуаций взаимных блокировок в наиболее общем случае и систему правил корректного использования средств синхронизации на основе этого достаточного условия.

Математическая модель взаимных блокировок. Модель взаимных блокировок в многопоточном ПО опирается на четыре класса объектов предметной области:

• разделяемый ресурс — информационный или функциональный объект, к которому возможен доступ из разных потоков;

• субъект доступа — поток, выполняющий доступ к разделяемому ресурсу;

• средство синхронизации — средство, ограничивающее доступ субъектов к разделяемым ресурсам посредством перевода субъекта в состояние ожидания доступности разделяемого ресурса;

• взаимная блокировка — ситуация, характеризующаяся тем, что группа субъектов находится в состоянии ожидания и не может быть выведена из него, независимо от действий других субъектов системы.

Субъект доступа моделируется на основе системы переходов, т.е. субъект отождествляется с совокупностью своих цепочек выполнения. Данная совокупность описывает всевозможные пути выполнения субъекта с точки зрения взаимодействия со средствами синхронизации от состояния покоя до завершающего состояния, отождествляемого с состоянием покоя.

Отождествление состояния покоя и завершающего состояния характеризует одно из фундаментальных свойств модели. В ней субъекты доступа зациклены, т.е., если субъект завершил свое выполнение по некоторому пути выполнения, он не обязательно будет ожидать завершения выполнения остальных субъектов доступа, а может снова начать выполнение по одному из путей.

Все цепочки выполнения субъекта линейны. Однако для отображения всевозможных цепочек выполнения, субъект может включать в себя точки ветвления и точки зацикливания (согласно структурной теореме Э. Дейкстры [11]). Отметим следующие особенности структур ветвления и циклических структур:

• условие, по которому в точке ветвления осуществляется выбор той или иной ветви выполнения, опускается в данной модели, поскольку является несущественным с точки зрения взаимодействия субъекта со средствами синхронизации. Предполагается, что при выполнении субъекта может быть выбрана произвольная из ветвей;

• число итераций цикла, соответствующего точке зацикливания, не специфицируется, т.е. считается, что субъект может выполнить произвольное конечное число итераций цикла (в том числе нулевое). Данное условие объясняется тем, что число итераций цикла может варьироваться при разных выполнениях субъекта (например, число итераций цикла соответствует числу необработанных на данный момент запросов). Данное же условие позволяет учитывать эту особенность, что дает модели необходимую общность.

Средства синхронизации. Каждый акт взаимодействия субъекта доступа со средством синхронизации моделируется как выполнение субъектом некоторого оператора, относящегося к средству синхронизации. Средства синхронизации моделируются как совокупность таких операторов. В общем случае в данной модели выделяются четыре примитива синхронизации: рекурсивный исключающий семафор, нерекурсивный (обычный) исключающий семафор, сигнальная переменная с памятью и сигнальная переменная без памяти.

Исключающий семафор — это средство синхронизации, которое в конкретный момент времени может быть захвачено только одним потоком. Если исключающий семафор захвачен каким-то потоком, то другой поток, обращающийся к семафору, переводится в состояние ожидания. Как только семафор освобождается, выполнение потока продолжается.

Нерекурсивный (*обычный*) *исключающий семафор* описывается двумя операторами взаимодействия:

• оператором захвата нерекурсивного семафора. Он обозначается окружностью, внутри которой изображен символ L и номер нерекурсивного семафора. Номер назначается с целью различать разные экземпляры нерекурсивного семафора;

• оператором освобождения нерекурсивного семафора. Он обозначается окружностью, внутри которой изображен символ U и номер нерекурсивного семафора.

Рекурсивный исключающий семафор также описывается двумя операциями — захвата и освобождения. Отличие от нерекурсивного семафора заключается в том, что данный семафор может быть захвачен несколько раз уже захватившим его субъектом. Когда захвативший его субъект освободит его столько раз, сколько его захватил, рекурсивный семафор может быть захвачен другими субъектами. Отметим, что рекурсивный семафор легко моделируется на основе нерекурсивного. Таким образом, будем рассматривать только нерекурсивные исключающие семафоры (далее – исключающие семафоры).

Сигнальная переменная без памяти — это средство синхронизации, при взаимодействии с которым поток попадает в состояние ожидания до тех пор, пока другим потоком не будет отправлен сигнал об освобождении. Сигнал может быть широковещательным, в таком случае он освобождает все ожидающие потоки, в противном случае — лишь один.

Сигнальная переменная без памяти описывается тремя операторами взаимодействия:

• оператором ожидания на сигнальной переменной без памяти. Его обозначают окружностью, внутри которой изображен символ W и номер сигнальной переменной без памяти;

• оператором отправки (сигнала) для сигнальной переменной без памяти. Его обозначают окружностью, внутри которой изображен символ E и номер сигнальной переменной без памяти;

• оператором широковещания для сигнальной переменной без памяти. Его обозначают окружностью, внутри которой изображен символ B и номер сигнальной переменной без памяти.

Сигнальная переменная с памятью — это средство синхронизации, имеющее целый неотрицательный счетчик. Взаимодействие потока с сигнальной переменной с памятью приводит к уменьшению счетчика, если счетчик в момент взаимодействия уже равен нулю, то поток переходит в состояние ожидание до тех пор, пока счетчик не будет увеличен другим потоком.

Сигнальная переменная с памятью описывается двумя операторами взаимодействия:

• оператором ожидания на сигнальной переменной с памятью. Он обозначается окружностью, внутри которой изображен символ A и номер сигнальной переменной с памятью;

• оператором установки значения для сигнальной переменной с памятью. Он обозначается окружностью, внутри которой изображен символ P и номер сигнальной переменной без памяти.

Время выполнения субъектом оператора взаимодействия полагается равным нулю, поскольку не является существенным параметром с точки зрения модели.

Взаимные блокировки бывают двух типов:

• полные — когда во взаимную блокировку вовлечены все субъекты данной системы;

• частичные — когда во взаимную блокировку вовлечена только часть субъектов системы.

Тем не менее частичные блокировки являются не менее опасными по сравнению с полными, поскольку, как отмечалось ранее, субъекты, попавшие в ситуацию взаимной блокировки, не могут быть выведены из нее независимо от действий других субъектов системы. В дальнейшем будем использовать обобщенное понятие взаимной блокировки, включающее в себя ситуации полной и частичной блокировок.

Разделяемые ресурсы присутствуют в данной модели неявно. Они определяются совокупностью средств синхронизации, обеспечивающих синхронный доступ субъектов к разделяемым ресурсам.

Теперь перейдем непосредственно к модели. Формализуем естественное отношение порядка (“до”–“после”), возникающее между двумя операторами вдоль одной цепочки выполнения субъекта. Отметим, что i-й и j-й исключающие семафоры сравнимы по k-му субъекту, причем i-й исключающий семафор локально меньше j-го, и запишем

$\tau(\mathrm{S}_k, \mathrm{L}_i) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_k, \mathrm{L}_j)$, если в цепочке выполнения k-го субъекта j-й семафор захватывается до того, как i-й семафор отпущен после захвата.

Расширим введенную операцию. Будем сравнивать исключающие семафоры, принадлежащие различным субъектам. Отметим, что i-й исключающий семафор k-го субъекта локально меньше j-го семафора m-го субъекта, и запишем $\tau(\mathrm{S}_k, \mathrm{L}_i) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_m, \mathrm{L}_j)$, если существует такое натуральное $n \geqslant 1$ и такие субъекты $\mathrm{S}_{x(1)}, \ldots, \mathrm{S}_{x(n-1)}$ и исключающие семафоры $y(1), \ldots, y(n-1)$, что

$$\begin{aligned}
&\tau(\mathrm{S}_k, \mathrm{L}_i) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_k, \mathrm{L}_{y(1)}),\\
&\tau(\mathrm{S}_{x(1)}, \mathrm{L}_{y(1)}) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_{x(1)}, \mathrm{L}_{y(2)}),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
&\tau(\mathrm{S}_{x(n-1)}, \mathrm{L}_{y(n-1)}) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_{x(n-1)}, \mathrm{L}_{y(n)}),\\
&\tau(\mathrm{S}_m, \mathrm{L}_{y(n)}) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_m, \mathrm{L}_j),
\end{aligned}$$

где для каждого из субъектов операция $\blacktriangleleft_{\mathrm{L}}$ понимается в точности так, как была введена ранее.

Отметим, что r-й исключающий семафор m-го субъекта локально меньше j-й сигнальной переменной m-го субъекта, и запишем $\tau(\mathrm{S}_m, \mathrm{L}_r) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_m, \mathrm{A}_j(\mathrm{W}_j))$, если взаимодействие с оператором ожидания j-й сигнальной переменной происходит после захвата r-го исключающего семафора и до его отпускания.

Считаем, что в системе субъектов $\mathrm{S} = \{\mathrm{S}_1, \ldots, \mathrm{S}_n\}$ i-я сигнальная переменная с памятью (без памяти) находится под локальным влиянием j-й переменной с памятью (без памяти), если либо существует субъект S_k из S, цепочка выполнения которого содержит оператор установки i-й сигнальной переменной с памятью (оператор отправки или широковещания для переменной без памяти) и оператор ожидания j-й переменной с памятью (без памяти), либо существует субъект S_k из S, цепочка выполнения которого содержит оператор отправки i-й сигнальной переменной с памятью (без памяти) и оператор захвата p-го исключающего семафора, причем выполнены следующие условия:

$$\tau(\mathrm{S}_k, \mathrm{L}_p) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_m, \mathrm{L}_r) \ \text{ и } \ \tau(\mathrm{S}_m, \mathrm{L}_r) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_m, \mathrm{A}_j(\mathrm{W}_j)).$$

В этом случае запишем $\tau(\mathrm{S}_k, \mathrm{P}_i(\mathrm{E}_i, \mathrm{B}_i)) \blacktriangledown_{\mathrm{L}} \tau(\mathrm{S}_m, \mathrm{A}_j(\mathrm{W}_j))$, где возможно $m = k$.

Предположим, что есть система субъектов $\mathrm{S} = \{\mathrm{S}_1, \ldots, \mathrm{S}_n\}$, взаимодействующих с сигнальными переменными (с памятью и без памяти) $\{1, \ldots, k\}$. Система субъектов S называется локально слабо упорядоченной по сигнальным переменным, если для любого подмножества $i_1, \ldots, i_j$ из $\{1, \ldots, k\}$ сигнальных переменных не существует такого

состояния системы, при котором каждый оператор отправки, широковещания и установки E_u $(\mathrm{B}_u, \mathrm{P}_u)$, где u из $\{i_1, \ldots, i_j\}$, находился бы в отношении локальной зависимости $\tau(\mathrm{S}_p, \mathrm{E}_u(B_u))\blacktriangledown_{\mathrm{L}}\tau(\mathrm{S}_q, \mathrm{W}_v(\mathrm{A}_v))$, где v из $\{i_1, \ldots, i_j\}$, т.е. все операторы широковещания, отправки и установки из данного множества одновременно находятся под локальным влиянием операторов ожидания из данного множества. Кроме того, в системе не существует сигнальных переменных, для которых присутствуют только операторы ожидания, но нет операторов отправки, широковещания или установки.

Введены все необходимые определения для формулировки основного результата математической модели взаимных блокировок, который является логическим основанием всех правил корректного использования средств синхронизации на основе данной модели.

Теорема. *Пусть имеется система субъектов* $S = \{S_1, \ldots, S_n\}$ *с точками ветвления и точками зацикливания. Проделаем для субъектов данной системы линеаризацию по точкам зацикливания. Получившуюся систему обозначим* S^0. *Рассмотрим все различные системы субъектов, составленные по следующему принципу*: *на* i*-м месте в системе стоит субъект из декомпозиции* S_i^0. *Предположим, что для каждой системы* S^{00} *из этого множества выполнены два условия*:

1) *для любого исключающего семафора* (*например, j-го*) *не выполняются соотношения вида* $\tau(S_k^{00}, L_j) \blacktriangleleft_L \tau(S_m^{00}, L_j))$;

2) *система* S^{00} *слабо локально упорядочена относительно сигнальных переменных. Тогда в основной системе S нет потенциальных ситуаций взаимной блокировки.*

Под линеаризацией понимается замена цикла точкой ветвления, где одна ветвь содержит цикл точки ветвления, а другая не содержит операторов взаимодействия.

Под декомпозицией субъекта с точками ветвления понимается множество его цепочек выполнения, каждая из которых рассматривается как линейный субъект.

Перейдем к правилам корректного использования средств синхронизации. Правила получены теоретическим путем на основе математической модели взаимных блокировок. Фактически правила представляют собой набор относительно простых конструкций, применение которых в процессе разработки ПО позволяет получать системы, обладающие достаточными (как было показано в модели, описанной выше) для отсутствия блокировок свойствами.

Базовые правила, которые необходимо соблюдать независимо от степени мастерства разработчика:

Правило 1. *Все субъекты системы должны быть однородными с точки зрения логики использования исключающих семафоров, т.е.*

должны удовлетворять правилу "первый захвачен — последний отпущен".

Рассмотрим область цепочки выполнения субъекта от захвата некоторого исключающего семафора до его освобождения. Если две области владения исключающим семафором пересекаются, то одна из них строго вложена в другую. Вложенность областей владения нужна для упрощения анализа взаиморасположения данных областей, что чрезвычайно важно для анализа системы на наличие потенциальных ситуаций взаимной блокировки.

Правило 2. *После прохождения тела цикла все исключающие семафоры, которые были захвачены до его прохождения, должны остаться захваченными, все исключающие семафоры, которые были свободны до его прохождения, должны остаться свободными.*

Иными словами, прохождение тела цикла не меняет состояния субъекта с точки зрения захваченных и освобожденных исключающих семафоров.

Рассмотрим субъект S_1, модель которого показана на рис. 1. Тело цикла точки зацикливания T1 субъекта S_1 содержит только оператор захвата первого исключающего семафора. При однократном выполнении тела цикла субъект S_1 захватывает первый исключающий семафор, освобождая его при дальнейшем выполнении. Рассмотрим теперь случай, когда субъект S_1 выполняет тело цикла 2 раза, при первом выполнении захватывается первый исключающий семафор, затем, при попытке повторного выполнения тела цикла, субъект попадает в ситуацию блокировки, пытаясь захватить не отпущенный нерекурсивный исключающий семафор. Причиной возникновения подобной ситуации является изменение состояния субъекта S_1 относительно захваченных исключающих семафоров после однократного прохода тела цикла.

С другой стороны, в теле цикла точки зацикливания T2 происходит сначала захват второго исключающего семафора, затем его освобождение. Таким образом, субъект возвращается в точку ветвления после прохода тела цикла в точно таком же состоянии. Следовательно, субъект может выполнять тело цикла произвольное конечное число раз,

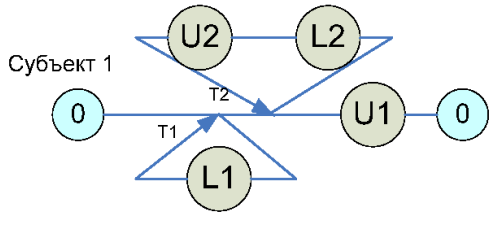

Рис. 1. Различные варианты взаимодействия с исключающими семафорами в теле цикла

при этом блокировка, описанная для случая точки зацикливания, T1 не может возникнуть.

Правило 3. *В системе субъектов нельзя использовать сигнальные переменные, взаимодействие с которыми сводится только к операторам ожидания.*

Если в системе S взаимодействие с сигнальной переменной сводится только к операторам ожидания, то в случае, если некоторый субъект S_k будет переведен данным оператором в состояние ожидания, он окажется в состоянии блокировки. Поскольку ни один из остальных субъектов системы не сможет вывести субъект S_k из состояния ожидания из-за отсутствия в этих субъектах операторов отправки (широковещания или установки) для данной сигнальной переменной.

В этом разделе были перечислены основные правила добротного использования средств синхронизации.

Императивные правила. (Рассчитаны на разработчиков начального уровня.)

Правила ориентированы на накладывание достаточно сильных условий на систему. Основным преимуществом данных правил является их относительно простая проверяемость.

Правило 1. *В системе субъектов S должна существовать перенумерация исключающих семафоров, обладающая следующим свойством, если для некоторого субъекта системы S_r и i-го и j-го исключающих семафоров выполнено соотношение: $\tau(S_k, L_i) \blacktriangleleft_L \tau(S_k, L_j)$, то $i < j$.*

Данное правило выработано с целью избегать соотношений вида $\tau(\mathrm{S}_k, \mathrm{L}_i) \blacktriangleleft_\mathrm{L} \tau(\mathrm{S}_m, \mathrm{L}_i)$, поскольку показано, что отсутствие в системе таких соотношений влечет отсутствие потенциальных ситуаций взаимной блокировки, где все активные участники ожидают на операторах захвата исключающих семафоров. Рассмотрим пример на рис. 2.

Для выполнения соотношения $\tau(\mathrm{S}_1, \mathrm{L}_1) \blacktriangleleft_L \tau(\mathrm{S}_3, \mathrm{L}_1)$ в данной системе неизбежно требуется нарушение правила для субъекта S_3: $\tau(\mathrm{S}_3, \mathrm{L}_3) \blacktriangleleft_\mathrm{L} \tau(\mathrm{S}_3, \mathrm{L}_1)$.

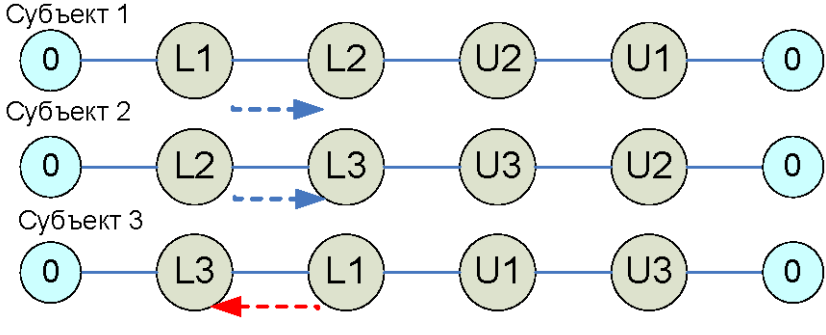

Рис. 2. Порядок захвата исключающих семафоров

Правило 2. *Система субъектов S должна быть сильно упорядоченной. Более того, все сигнальные переменные должны иметь порядок* 0.

Для любой сигнальной переменной системы должен существовать субъект, содержащий хотя бы один оператор отправки (широковещания или установки) для данной переменной. Кроме того, он не должен содержать операторов ожидания для других сигнальных переменных, а также операторов захвата исключающих семафоров L_i, для которых выполнено $\tau(\mathrm{S}_m, \mathrm{L}_i) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_k, \mathrm{L}_s) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_k, \mathrm{W}_j)$.

Данное свойство является самым сильным из требований упорядоченности, но в то же время его проще всего проверить. Фактически, данное свойство гарантирует отсутствие ситуаций блокировки в системе, в которой нет ситуаций блокировки на одних исключающих семафоров.

Правило 3. *Логика использования в системе операторов отправки, широковещания и установки не должна зависеть от выбора конкретной ветви при проходе точек ветвления.*

С точки зрения наличия в системе потенциальных ситуаций взаимной блокировки стоит рассматривать всевозможные варианты поведения системы, даже если выбор той или иной ветви представляется маловероятным. Поясним сказанное на примере системы субъектов, изображенной на рис. 3.

Пусть регулярному выполнению системы соответствует выбор верхней ветви в точке ветвления T1 субъекта S_2. В этом случае оба субъекта беспрепятственно выполняются и в системе нет блокировок. Предположим, что система стала вести себя нерегулярно и после первого прохождения в точке T1 по нижней ветви все время проходит по ней. На данной ветви нет оператора отправки для первой сигнальной переменной без памяти, следовательно, субъект S_1 при данной динамике после взаимодействия с оператором ожидания первой сигнальной переменной оказывается в ситуации блокировки.

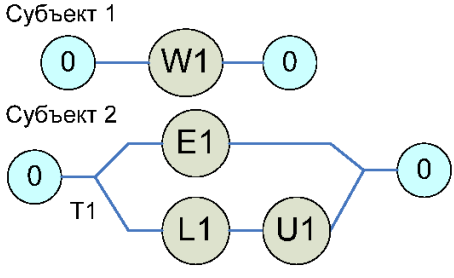

Рис. 3. Пример системы, выполнение которой зависит от выбора конкретной ветви

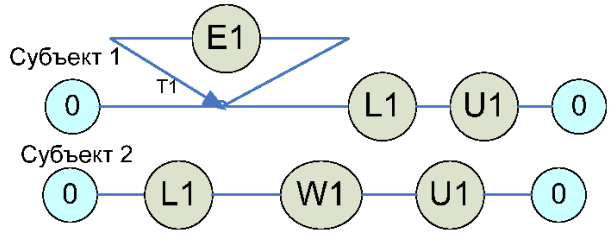

Рис. 4. Пример системы, выполнение которой зависит от входа в тело цикла

Правило 4. *Логика использования в системе операторов отправки, широковещания и установки не должна зависеть от гарантированного выполнения тела цикла.*

Нельзя полагаться на операторы отправки, широковещания и установки, располагающиеся в теле цикла. Рассмотрим в качестве примера систему субъектов, изображенную на рис. 4.

Данный пример иллюстрирует ситуацию, когда при нерегулярном поведении, приводящем субъект S_1 к невыполнению тела цикла, субъект S_2 может перейти в состояние ожидания на операторе ожидания первой сигнальной переменной без памяти, в это время субъект S_1 перейдет в состояние ожидания на операции захвата исключающего семафора.

Правило 5. *Нельзя полагаться на начальное значение счетчика сигнальных переменных с памятью.*

Рассмотрим это правило на примере субъекта на рис. 5.

Представим несколько вариантов. Пусть число выполнений цикла равно единице и начальное значение счетчика первой сигнальной переменной без памяти равно единице, в этом случае субъект S_1 попадет в состояние блокировки уже при первом проходе, поскольку ему необходимо преодолеть три оператора ожидания. Пусть число выполнений цикла равно трем, тогда субъект S_1 будет выполняться при любом значении начального счетчика, поскольку при проходе его значение не будет падать ниже начального. Представим еще более сложный вариант — значение начального счетчика равно 20, а тело цикла выполняется 2 раза. В этом случае субъект S_1 сможет выполниться 20 раз прежде, чем попасть в ситуацию взаимной блокировки.

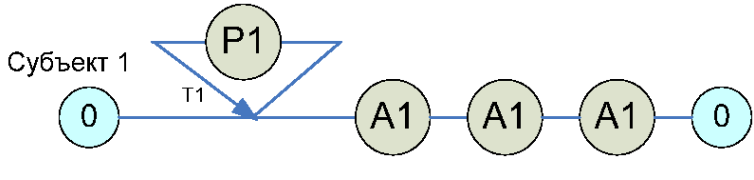

Рис. 5. Пример субъекта, зависящего от начального значения сигнальной переменной с памятью

Правило 6. *В ситуации, когда возможен выбор, использовать сигнальную переменную с памятью или без памяти, следует всегда использовать переменные без памяти.*

Данное правило объясняется тем, что переменные без памяти проще по своей природе, поскольку не имеют счетчика, следовательно, для них легче выявлять потенциальные ситуации взаимной блокировки.

Рассмотрим пример на рис. 6.

Данная система субъектов допускает ситуации взаимной блокировки в зависимости от начальных значений счетчика. Однако остановимся на случае, когда начальные значения счетчиков равны нулю и система не допускает взаимной блокировки. Теперь немного модернизируем систему (рис. 7).

Отметим, что хотя в систему был добавлен оператор установки, на котором субъект не может перейти в состояние ожидания, система теперь допускает взаимные блокировки, например при выполнении сначала субъекта S_2 по нижней ветви. Теперь значение счетчика второй сигнальной переменной без памяти больше нуля. Это означает, что теперь субъект S_2 может попасть на участок взаимодействия с исключающими семафорами одновременно с субъектом S_1, что будет являться причиной блокировки.

Данный пример иллюстрирует сложные для анализа эффекты, возникающие при использовании в системе сигнальных переменных с памятью. Эти эффекты вызваны наличием у сигнальных переменных

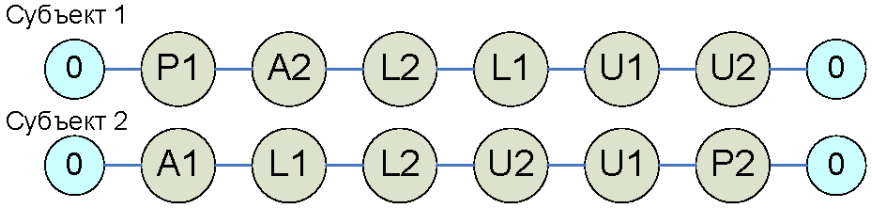

Рис. 6. Пример системы субъектов

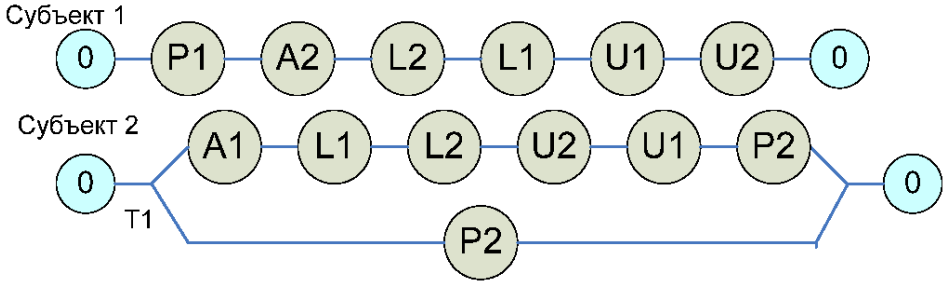

Рис. 7. Модернизированная система субъектов

с памятью счетчика. Во избежание данных эффектов необходимо как можно чаще использовать вместо сигнальных переменных с памятью сигнальные переменные без памяти.

Рекомендательные правила. Набор данных правил рассчитан на опытных разработчиков и задает основные направления анализа систем на наличие в них потенциальных блокировок.

Правило 1. *Если при использовании исключающих семафоров возникает соотношение* $\tau(S_k, L_i \blacktriangleleft_L \tau(S_m, L_i)$, *то необходимо гарантировать, что состояние, инициированное данным соотношением, никогда не реализуется.*

Поскольку соотношение $\tau(\mathrm{S}_k, \mathrm{L}_i) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_m, \mathrm{L}_i)$ означает, что в системе существует набор субъектов и состояний каждого из субъектов, при объединении которых возникает взаимная блокировка. Задача заключается в том, чтобы обеспечить принципиальную невозможность одновременной реализации всех этих состояний. Рассмотрим пример системы, изображенной на рис. 8.

Сначала рассмотрим систему без пятого исключающего семафора. В системе имеется соотношение $\tau(\mathrm{S}_1, \mathrm{L}_1) \blacktriangleleft_L \tau(\mathrm{S}_4, \mathrm{L}_1)$, которое реализуется при последовательном захвате каждым из субъектов одного семафора: сначала субъект S_1 захватывает первый исключающий семафор и переходит в состояние ожидания, затем субъект S_2 захватывает второй исключающий семафор и переходит в состояние ожидания, далее субъект S_3 захватывает третий исключающий семафор и переходит в состояние ожидания, затем субъект S_4 захватывает четвертый исключающий семафор и переходит в состояние ожидания.

Теперь система находится в состоянии взаимной блокировки. Добавим в рассмотрение пятый исключающий семафор. Как и прежде, выполняется соотношение $\tau(\mathrm{S}_1, \mathrm{L}_1) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_4, \mathrm{L}_1)$, однако теперь для

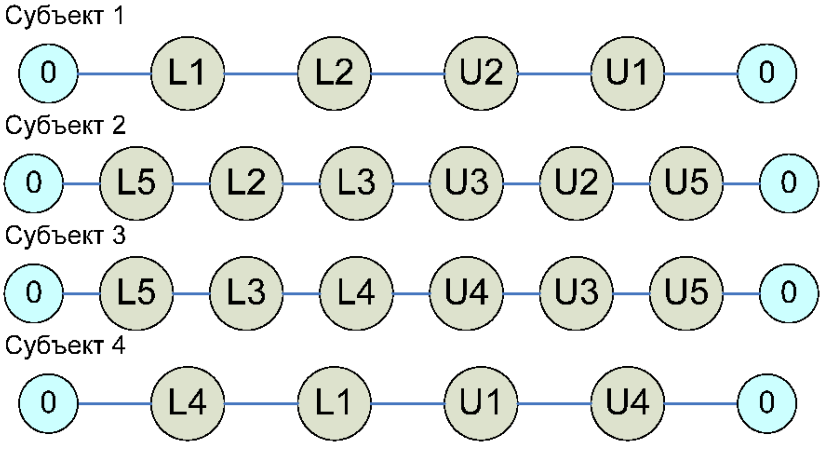

Рис. 8. Пример системы субъектов с нереализующимся состоянием

его реализации необходимо, чтобы субъекты S_2 и S_3 одновременно удерживали пятый исключающий семафор. Таким образом, введение дополнительного семафора избавляет систему от потенциальных ситуаций взаимной блокировки. Примененная методика (и эквивалентные ей) составляют суть вышеописанного правила.

Правило 2. *Если систему S не удается сделать сильно упорядоченной относительно сигнальных переменных, то следует сделать ее локально слабо упорядоченной.*

Поясним понятие сильной упорядоченности (можно показать, что оно влечет свойство слабой упорядоченности). Пусть есть система субъектов $S = \{S_1, \ldots, S_n\}$, пусть субъекты данной системы взаимодействуют с i-й сигнальной переменной (с памятью или без памяти). Если у этой переменной есть хотя бы один оператор отправки (широковещания или установки), характеризующийся тем, что субъект, которому принадлежит данный оператор, не содержит операторов ожидания (для всех сигнальных переменных, с которыми оперирует система), кроме того, он не содержит операторов захвата тех исключающих семафоров, которые локально меньше любой из сигнальных переменных системы. Фактически, это означает, что на данный оператор локально не влияют никакие сигнальные переменные. В этом случае присвоим сигнальной переменной 0-й порядок. Дальше будем определять по индукции. Если у переменной есть хотя бы один оператор отправки (широковещания или установки), характеризующийся тем, что на него влияют только сигнальные переменные 0-го порядка, то присвоим данной переменной 1-й порядок. Если у переменной есть хотя бы один оператор отправки (широковещания или установки), характеризующийся тем, что на него влияют только сигнальные переменные 0-го и 1-го порядка, то присвоим данной переменной 2-й порядок, и т.д. Если каждой сигнальной переменной (с памятью или без памяти) в системе субъектов S можно присвоить некоторый порядок, назовем эту систему сильно (локально) упорядоченной.

Если зависимости переменных устроены так, что систему не получается сделать сильно упорядоченной, то надо эти зависимости разделить таким образом, чтобы они не реализовывались в одном состоянии, тогда система окажется локально слабо упорядоченной. В качестве примера рассмотрим систему на рис. 9.

Данная система не является сильно упорядоченной, поскольку субъект S_1, содержащий единственный оператор отправки для второй сигнальной переменной, содержит оператор захвата четвертого исключающего семафора, кроме того, выполнены следующие соотношения:

$$\tau(S_3, L_4) \blacktriangleleft_L \tau(S_3, L_3),$$
$$\tau(S_2, L_3) \blacktriangleleft_L \tau(S_2, W_1).$$

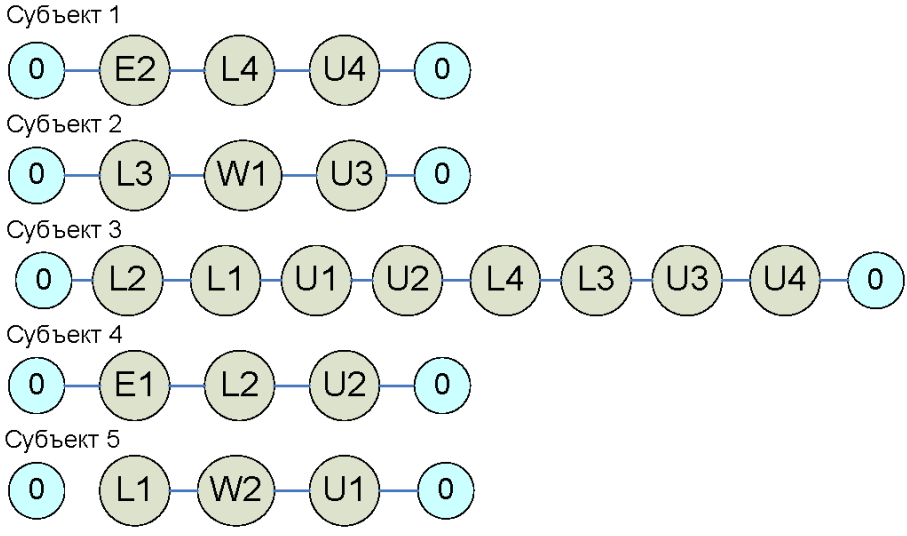

Рис. 9. Пример системы с разделенными зависимостями

Это в свою очередь означает, что верно $\tau(\mathrm{S}_1, \mathrm{E}_2)\blacktriangledown_{\mathrm{L}}\tau(\mathrm{S}_2, \mathrm{W}_1)$. Вместе с тем субъект S_4, содержащий единственный оператор отправки для первой сигнальной переменной, содержит оператор захвата второго исключающего семафора, кроме того, выполнены следующие соотношения:

$\tau(\mathrm{S}_3, \mathrm{L}_2) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_3, \mathrm{L}_1)$,

$\tau(\mathrm{S}_5, \mathrm{L}_1) \blacktriangleleft_{\mathrm{L}} \tau(\mathrm{S}_5, \mathrm{W}_2)$,

что означает справедливость отношения $\tau(\mathrm{S}_4, \mathrm{E}_1)\blacktriangledown_{\mathrm{L}}\tau(\mathrm{S}_5, \mathrm{W}_2)$.

Из этих соотношений следует неупорядоченность системы. Однако система является слабо локально упорядоченной, поскольку оба эти соотношения не реализуются в одном состоянии, а именно они требуют нахождения субъекта S_3 в двух несовместимых состояниях.

Напомним, что свойство локальной слабой упорядоченности очень важно, поскольку оно сохраняется в процессе декомпозиции. В данном описании представлена техника разнесения состояний вдоль одного субъекта, позволяющая сохранять свойство слабой локальной упорядоченности системы.

Правило 3. *При наличии в системе субъектов с точками ветвления или зацикливаний необходимо учитывать все возможные варианты выполнения таких субъектов.*

В качестве примера рассмотрим два субъекта на рис. 10.

Система, состоящая из субъекта S_1, неизбежно попадет в состояние блокировки, если будет выполняться по нижней ветви точки ветвления, причем это произойдет независимо от начального значения счетчика. В то же время система, состоящая из субъекта S_2, модернизирована так, чтобы не допускать состояний блокировок, поскольку

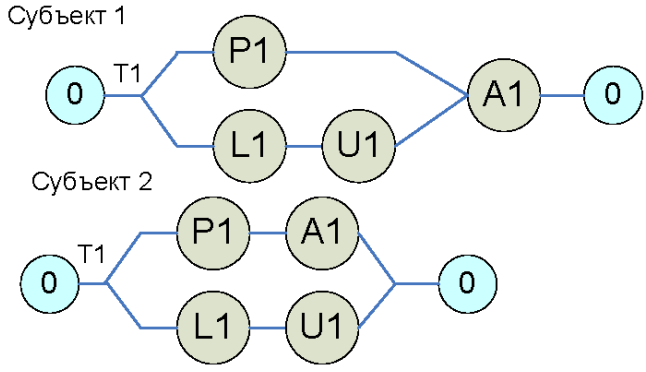

Рис. 10. Пример системы субъектов с точками ветвления

операторы установки и ожидания находятся на одной цепочке выполнения. Данный пример показывает, что необходимо таким образом использовать средства синхронизации, чтобы при любом варианте развития системы избегать ситуаций взаимной блокировки.

Правило 4. *Если стратегия избегания ситуаций взаимной блокировки в системе строится, опираясь на значения счетчика некоторой сигнальной переменной с памятью, то необходимо анализировать систему на наличие потенциальных ситуаций взаимной блокировки при добавлении любого оператора, связанного с данной переменной (не обязательно только оператора ожидания).*

Пример на рис. 7 представляет собой систему, зависящую от конкретных значений счетчиков первой и второй сигнальных переменных. Если оба значения в начале равны нулю, то система не допускает ситуаций взаимной блокировки. Однако, как показывает пример, при добавлении в систему оператора установки данная система уже допускает ситуации взаимной блокировки.

В случае, когда стратегия избегания ситуаций взаимной блокировки основана на подборе конкретных значений для счетчиков сигнальных переменных, необходимо подвергать систему анализу и при добавлении новых операторов установки, поскольку они нарушают динамику изменения значений счетчиков.

Правило 5. *Для разделения состояний эффективнее всего использовать дополнительные исключающие семафоры и разделения состояний вдоль субъекта.*

При обеспечении упорядоченности часто возникает потребность в том, чтобы некоторые соотношения не выполнялись в одном состоянии. Примером на рис. 8 показано, как этого добиться с помощью добавления дополнительного исключающего семафора. Пример на рис. 9 предлагает альтернативный подход — разделение состояний вдоль одного субъекта.

Выводы. Представлены правила корректного использования средств синхронизации. Система правил построена таким образом, что применяющий их разработчик избегает появления в ПО конструкций, которые приводят к невыполнению достаточного условия отсутствия потенциальных ситуаций взаимной блокировки.

Достаточные условия сформулированы на основе математической модели взаимных блокировок, разработанной авторами. В рамках модели доказано, что из достаточного условия следует отсутствие потенциальных ситуаций взаимной блокировки в системе.

Различные группы правил предназначены для разработчиков различного уровня. Императивные правила адресованы разработчикам начального уровня. Рекомендательные правила — опытным разработчикам. Показано, на какие аспекты средств синхронизации следует обратить особое внимание.

СПИСОК ЛИТЕРАТУРЫ

1. Lamport L. Specifying systems: The TLA+ language and tools for hardware and software engineers // Addison-Wesley, 2002.
2. Bensalem S. and Havelund K. Dynamic Deadlock Analysis of Multi-threaded Programs. In Shmuel Ur, Eyal Bin, and Yaron Wolfsthal, editors, Haifa Verification Conference. – 2005. – Vol. 3875. – 208 p.
3. Detlefs D. L., Rustan K., Leino M., Nelson G. and Saxe J. B. Extended static checking. Technical Report 159, Compaq Systems Research Center, Palo Alto, California, USA, 1998.
4. Дал У., Дейкстра Э., Хоор К. Структурное программирование. Structured programming. – М.: Мир, 1975.
5. Engler D. and Ashcraft K. RacerX: Effective, Static Detection of Race Conditions and Deadlocks // In Proc. of the 19th ACM Symposium on Operating Systems Principles. – 2003. – P. 237–252.
6. Artho C. and Biere A. Applying static analysis to large-scale, multi-threaded Java programs // D. Grant, editor, 13th Australien Software Engineering Conference, pages 68–75 // IEEE Computer Society, August 2001.
7. Кларк Э., Грамберг О., Пелед Д. Верификация моделей программ: Model checking. – М.: МНЦМО, 2002.
8. Карпов Ю. MODEL CHECKING. Верификация параллельных и распределенных программных систем. – СПб.: БХВ-Петербург, 2010.
9. Havelund K. and Pressburger T. Model Checking Java Programs using Java PathFinder // International J. on Software Tools for Technology Transfer, 2(4): 366–381, April 2000. Special issue of STTT containing selected submissions to the 4th SPIN workshop, Paris, France, 1998.
10. Holzmann G. Design and validation of computer protocols. – Prentice Hall, 1991.
11. Savage S., Burrows M., Nelson G., Sobalvarro P. and Anderson T. Eraser: A dynamic data race detector for multi-threaded programs // Proceedings of the 16th ACM Symposium on Operating Systems Principles, pp 27–37, Oct. 1997.

Статья поступила в редакцию 15.12.2011