# ACI-ТРАНЗАКЦИИ В СЕТЯХ ПЕТРИ

**А. А. Демидов**[1]

## Введение

Пусть конкуренция в системе устроена так, что каждый процесс считывает все разделяемые данные в локальную память, вносит необходимые изменения, и затем записывает эти данные целиком обратно в общую память. Если операции чтения и записи атомарны, то есть мгновенны для конкурирующих процессов, то система все время будет оставаться в непротиворечивом состоянии. Но изменение этого состояния будет почти последовательным, из многих совместных историй будет выбран один вариант, остальные результаты будут отброшены по принципу «последний забирает все».

Этот подход лежит в основе ACI-транзакций, не гарантирующих свойство D – долговечность подтвержденных изменений. Оказывается, атомарность можно реализовать с помощью версий без привлечения синхронизации, а распараллеливаемость зависит от самих данных и не является принципиальным ограничением – с этой целью вводятся так называемые «открытые объекты», позволяющие определить области данных, которые могут изменяться параллельно и независимо друг от друга. Из-за отбрасывания части результатов ACI-транзакции не всегда могут использоваться в задачах, связанных с описанием реальных систем.

**Пример 1.** Пусть в интернет-магазине два покупателя A и B одновременно пытаются оплатить последнюю единицу некоторого товара. В реальной ситуации товар достанется кому-то одному из них, а второй абсолютно точно останется с носом. Если же мы только тестируем эту систему, нам неважно, кому конкретно, A или B, достанется этот товар. Один процесс временно может считать, что товар достался покупателю A, а второй – что покупателю B. ACI-транзакции гарантируют, что при объединении в общее состояние часть конфликтующих данных будет отброшена, и останется только один из вариантов, кому достался выбранный товар.

Сети Петри являются естественной сферой применения ACI-транзакций. Любая модель с дискретными событиями состоит из сети взаимосвязанных очередей [1] (§ 18.4.1), а сети Петри [2] являются удобным языком описания таких структур. Не на всех задачах применение ACI-транзакций будет эффективным – проблема связана с укрупнением открытых объектов, которое определяется структурой сети, описывающей конкретную задачу. Впрочем, неудобные задачи есть у любого метода.

## Понятие ACI-транзакций

Теория ACI-транзакций подробно дается в [3], здесь приводятся основные тезисы, необходимые для сохранения логики изложения.

ACI-транзакциями назовем способ работы с данными, обеспечивающий известные свойства ACID: атомарность, непротиворечивость, изолированность, но не гарантирующий сохранности подтвержденных изменений. Отсутствие последнего свойства позволяет ACI-транзакциям работать без откатов (транзакции всегда завершаются), но использование подтвержденных изменений в последующих транзакциях не гарантируется. Все три свойства, включая атомарность, могут быть реализованы без синхронизации.

[1] Институт программных систем им. А. К. Айламазяна РАН, Исследовательский центр искусственного интеллекта, ул. Петра I, д.4а, с. Веськово, 152021, Переславский р-н, Ярославская обл.; м.н.с., e-mail: alex@dem.botik.ru.

Выделим в системе предметный слой, где будем определять объекты; совокупность низлежащих программно-аппаратных уровней назовем ядром. Обозначим $G$ множество элементов данных – таких единиц хранения информации, операции с которыми атомарны на уровне ядра. С точки зрения предметного слоя, эти операции будут элементарными, то есть атомарными и неделимыми на субоперации. Чтобы обеспечить атомарность последовательностей элементарных операций на уровне предметного слоя вводятся средства группировки операций, такие как критические секции или транзакции. Пусть каждая транзакция в отдельности действует рационально, то есть меняет глобальное состояние непротиворечивым образом, не считывает и не записывает данные без необходимости.

**Определение 1.** Будем говорить, что элемент данных $a \in G$ ограничивает значения элемента данных $b \in G$ и записывать $a \to b$, если множество допустимых значений элемента $b$ непосредственно зависит от значения, которое имеет элемент $a$.

Множество пар вида $a \to b$ определяет отношение на множестве $G$, которое назовем отношением (непосредственной) зависимости элементов данных. Это отношение нетранзитивно: $a \to b \to c \not\Rightarrow a \to c$.

Каждая транзакция реализует некоторый алгоритм, поэтому она находится в курсе того, какие элементы данных потенциально могут ограничивать значение изменяемого ею элемента. Чтобы актуализировать ограничения, транзакция должна поинтересоваться текущими значениями этих элементов (прочитать или вычислить их на основе других значений) – эти действия транзакции указывают на наличие зависимостей.

В каждой точке $g \in G$ определим базис $\pi(g)$ фильтра $\tau(g)$ [4](§ 1.3) как систему из единственного множества элементов данных, ограничивающих значения элемента $g$ (при этом полагаем $g \to g$):

$$\tau(g) = \mathrm{fil}\,\pi(g), \quad \text{где} \quad \pi(g) = \{\{x \in G : x \to g\}\}.$$

**Определение 2.** Произвольное подмножество $S \in \tau(g)$ называется предокрестностью точки $g \in G$.

**Определение 3.** Множество называется открытым, если оно является предокрестностью каждой из своих точек.

**Определение 4.** Предокрестность, содержащая открытое множество, называется окрестностью. Окрестность, совпадающая с открытым множеством, называется открытой окрестностью.

Открытые объекты примечательны тем, что совершенно не зависят от состояния остальной части данных. Поэтому достаточно обеспечить атомарность операций с таким объектом, чтобы обеспечить непротиворечивость любых операций с разделяемыми данными, не меняющих топологию. То, что другие данные зависят от открытого объекта, не представляет проблемы – рациональная транзакция сама производит все необходимые изменения связанных объектов. Реализация операций, меняющих топологию, возможна с использованием непрерывности операций и предобъектов.

**Определение 5.** Предокрестность, атомарность операций чтения и записи которой обеспечена, будем называть предобъектом.

**Определение 6.** Чтение данных непрерывно, если сохраняет предокрестности, то есть каждая предокрестность при чтении имеет тот же вид, какой она имела при записи в некоторой транзакции.

**Теорема 1.** *Если чтение данных непрерывно, то предобъекты обеспечивают непротиворечивое изменение произвольного глобального состояния.*

*Доказательство.* Пусть две транзакции меняют элементы данных. Для произвольного множества $S \subset G$ обозначим $U_S = \bigcup_{s \in S} \pi(s)$ минимальный предобъект, включающий все элементы из $S$. Множества $A$ и $B$ элементов, изменяемые соответственно первой и второй транзакцией, могут:

совпадать, если $A = B$;

пересекаться, если $A \cap B \neq \varnothing$;

касаться, если $(U_A \cap B \neq \varnothing) \vee (A \cap U_B \neq \varnothing)$;

быть отделены, если $(U_A \cap B = \varnothing) \wedge (A \cap U_B = \varnothing)$.

Если множества отделены или, с учетом атомарности операций, совпадают, то версии $U_A$ и $U_B$ независимы друг от друга. Если множества пересекаются или касаются, тогда версии $U_A$ и $U_B$ могут содержать зависимые элементы данных, что может привести к противоречию. Необходимо либо отбросить одну из конфликтующих версий без отката транзакции, либо искусственно дополнить первоначальными значениями элементов каждую из зависимых версий до объединения множеств $A \cup B$ – после чего ситуация сводится к случаю, когда множества $A$ и $B$ совпадают. ■

Непрерывность чтения не требует, чтобы пересекающиеся предокрестности были изменены в одной транзакции – она предоставляет гораздо больше свободы, чем кажется на первый взгляд. Чтобы предокрестности не разрывались при чтении, наряду с измененными значениями необходимо сохранять значения всех прочитанных транзакцией элементов, в том числе и измененных (из-за вычисления значений): рисунки 1.a–1.c.

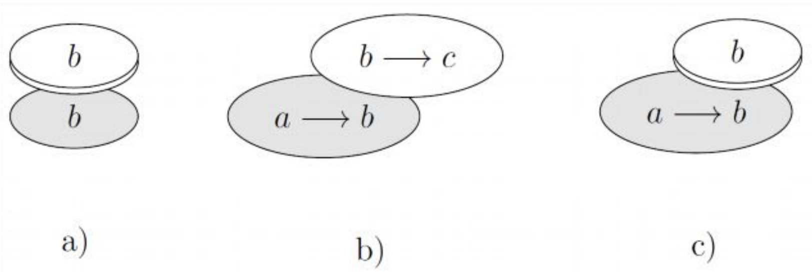


**Рис. 1. Варианты пересечения предокрестностей**

Типичная процедура сборки предобъектов из фрагментов представлена на рисунке 1.b. Нижний предобъект был создан транзакцией, прочитавшей значение элемента $a$ и изменившей значение элемента $b$, а верхний – транзакцией, прочитавшей значение элемента $b$ и изменившей значение элемента $c$. В итоге значение элемента $b$ предобъекта $a \to b$ получается равным значению этого элемента предобъекта $b \to c$, что позволяет непрерывно объединить данные фрагменты в один предобъект $a \to b \to c$. Более подробно вопрос реализации непрерывности разобран в основной работе [3].

**Замечание 1.** Версионность достаточно поддерживать ограниченное время – до завершения транзакций, которые старше самих версий.

**Распараллеливание сетей Петри**

Сети Петри являются естественной сферой применения ACI-транзакций. Договоримся, что не будем рассматривать распараллеливание по типу «что если?», когда отдельные области сети обсчитываются заранее в надежде, что на их входах впоследствии появятся требуемые фишки (если сеть разлагается на подсети, то будем применять метод внутри каждой из них). Это условие мягко ограничивает сверху количество моделирующих процессов средним числом фишек в сети. Также не будем рассматривать эффективность метода на начальных стадиях, когда распределение фишек в сети еще не установилось, и все они находятся вблизи начальных позиций.

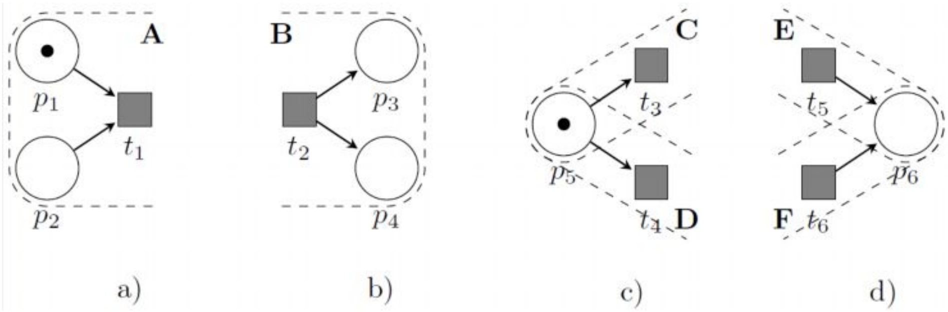


**Рис. 2. Предобъекты A–F на элементах сетей Петри**

Для каждого перехода можно определить набор входных позиций (мест), от нахождения маркера в которых зависит срабатывание данного перехода, а также набор выходных позиций, в которых маркер оказывается после такого срабатывания [2]. Эти зависимости позволяют с каждым переходом ассоциировать свой предобъект, включающий все входные и все выходные позиции данного перехода (рисунки 2.a–2.d).

По правилам ACI-транзакций переходы $t_3$ и $t_4$ конфликтуют за фишки во входной позиции, а $t_5$ и $t_6$ – за выходную позицию для размещения фишек (рисунки 2.c и 2.d). Это не совсем то, что требуется для сетей Петри, поскольку входная позиция может содержать несколько фишек, а емкость выходной позиции вообще не ограничена.

Несоответствие ожиданиям связано с тем, что на самом деле переходу не нужно знать точное количество фишек во входящей позиции, ему достаточно быть уверенным, что фишек хватит для его срабатывания. Формально это означает, что транзакциям следует использовать коммутативные операции инкремента $+N$ и декремента $-N$ позиций, которые не требуют чтения значений в локальную память. Коммутативные операции не конфликтуют, однако декремент может нарушить ограничение количества фишек $n \geq 0$. Поэтому будем предварительно считывать значения входных позиций, но изменять значения будем с помощью операций $\pm N$ (которые коммутируют между собой, но, вообще говоря, не коммутируют с операцией чтения).

Тогда каждая предокрестность на рисунке 1.a (вверху) будет содержать: начальное количество фишек во входных позициях, конечное количество фишек во входных и выходных позициях, а также дельты этих значений (дельты можно не хранить, если для выходных позиций также запомнить начальные значения).

Теперь достаточно ослабить требования теоремы 1 и считать конфликтом только такое пересечение предокрестностей, которое приводит к отрицательному количеству фишек в какой-либо входной позиции. При объединении неконфликтующих пересекающихся предобъектов конечные количества фишек в области пересечения необходимо пересчитать, иначе следующее чтение позиций вернет некорректные значения.

После назначения предобъектов сеть Петри может быть запущена на параллельной ЭВМ без синхронизации процессов.

**Заключение**

На базе ACI-транзакций предложен эффективный метод распараллеливания вычислений в сетях Петри сложной структуры. Метод позволяет принципиально избежать синхронизации конкурирующих процессов, что делает его хорошо масштабируемым. При этом вопрос эффективности на различных конфигурациях сетей составляет предмет дальнейших исследований.

**Литература**

[1] **Таха Х.А.** Введение в исследование операций. 6-е изд.: пер. с англ. – М.: Издательский дом «Вильямс», 2001.

[2] **Котов В.Е.** Сети Петри. – М.: Наука, 1984.

[3] **Демидов А.А.** Ничтожность САР-тезиса Брюера в задачах имитационного моделирования // Тр. V международной конф. «Системный анализ и информ. технологии» САИТ-2013. – Т. 2. – Красноярск: ИВМ СО РАН, 2013. – С. 22–29.

[4] **Кутателадзе С.С.** Основы функционального анализа. Современная математика – студентам и аспирантам. 4-е изд., испр. – Новосибирск: Изд-во Ин-та математики, 2001.