РАСПРЕДЕЛЕННАЯ МАШИНА ВЫВОДА ОБЪЕКТНО-СОБЫТИЙНЫХ МОДЕЛЕЙ

В.В. Пекунов

ОАО «Информатика», Иваново
E-mail: pekunov@mail.ru

Данная работа является развитием работы [1]. Ее целью является ввод понятия об инкрементальном логическом выводе объектно-событийных моделей (ОСМ), который может быть настолько трудоемок, что требует распределенных вычислений. Логический вывод применяется, в частности, для автоматизированного построения моделей, порождающих решающие программы, по формальной постановке проблемы.

Теорема об обратном логическом выводе (ТЛ1). Задача обратного логического вывода для произвольного целевого выражения, включающего единичные высказывания (предикаты), а также связки «И» и «ИЛИ», может быть реализована ОСМ, структура и параметры которой отражают форму такого целевого утверждения, если

а) единичному высказыванию сопоставить единственный узел ОСМ;

б) между узлами-высказываниями, подписанными на одно событие – связка «И»;

в) наличие связки «ИЛИ» может быть определено параметром некоего узла А, например, для исходящих из него параллельных ветвей;

г) существует предельная (атомарная) логическая ОСМ, способная для единичного предиката решить задачу унификации с единственным фактом.

Доказательство.

1. Докажем, что целевое высказывание может быть описано структурой и параметрами ОСМ. Пункт (а) формулировки доказуем, если узлы-объекты ОСМ относятся к соответствующим классам-предикатам. Пункт (б) следует из идеологии сетевого графика работ (одного из основных принципов ОСМ). Узлы-высказывания, между которыми существует отношение следования, в ходе одного события доказываются последовательно, при его отсутствии *возможно параллельное доказательство*. Рассмотрим пункт (в). В этом случае узел А подписывает на текущее событие одну произвольную ветвь, а в случае необходимости пересогласования – какую-либо из иных ветвей, в соответствии с алгоритмом обратного логического вывода (ОЛВ).

2. Процесс ОЛВ имеет процедурную трактовку, алгоритм которой может быть реализован ОСМ. Это подтверждается существованием трассы у машины ОЛВ, для которой легко проиллюстрировать возможность построения эквивалентной ОСМ. При этом задача доказательства сложных выражений сводится к сериям задач доказательства более простых выражений на вложенных ОСМ-методах с соответствующим управлением процессом доказательства, реализуемым по событийной схеме. Заключительными элементарными этапами такого алгоритма будут вызовы процедур (методов классов объектов-узлов), решающих задачи унификации единичного высказывания с единичным фактом и сводимые к ним задачи унификации термов. В роли таких методов будут выступать предельные логические ОСМ. Таким образом, условия, введенного пунктом (г) формулировки, необходимо и достаточно для завершения доказательства.

Следствие. Задача отката с генерацией следующих вариантов доказательства в предыдущих, включенных в цель предикатах, может быть тривиально реализована путем условного планирования события – следующей попытки доказательства целевого

утверждения, на которое будут подписаны подлежащие пересогласованию элементы цели. Текущая позиция вывода может храниться в полях объектов-узлов ОСМ.

Теорема о предельной логической ОСМ (ТП5). Предельная (атомарная) логическая объектно-событийная модель, решающая задачу унификации единственного предиката с одним фактом, тождественна абстрактной предельной ОСМ.

Доказательство. Абстрактная предельная ОСМ однозначно отображается в машину Тьюринга (МТ) и включает единственный узел. Набор возможных значений V_i каждого *i*-го аргумента предиката ($i = \overline{1,N}$) в реальной вычислительной системе в сочетании со значением $\varnothing$, указывающим на несвязанность аргумента, является конечным множеством $P_i = V_i \cup \{\varnothing\}$.

Декартово произведение множеств P_i в сочетании с символом nil («ложь») представляет собой алфавит МТ: $Q = (P_1 \times P_2 \times \ldots \times P_N) \cup \{nil\}$. Один из символов $p \in Q$ соответствует комплексу значений аргументов, заданных фактом. На ленту помещается символ, соответствующий значениям входных аргументов предиката, в ту же ячейку помещается символ p при успешной унификации или nil, если таковая невозможна.

Пусть в начале доказательства календарь содержит начальное событие q_1. Пусть также возможно конечное событие q_2.

Легко составить программу для единственного метода-обработчика события q_1, включающего команды условного планирования (переходы МТ) события q_2 с помещением на ленту символа p, соответствующего успешно унифицируемым значениям аргументов (символам входного алфавита МТ), или символа nil для таких значений аргументов, которые не могут быть унифицированы. Программа вычислима по Тьюрингу.

Следствие. Также доказано, что выполняется условие, содержащееся в ТЛ1.

Теорема об унификации ОСМ (ТМ7). Отобразим ОСМ в совокупность фактов о ее структуре и параметрах. Для унификации ОСМ знаниям об отображении возможных вариантов решения задач в структуру и параметры ОСМ, необходимо и достаточно инкрементально доказать серию логических утверждений, следующих из текущих структуры и параметров ОСМ, до полного согласования последних с фактами, изъятыми и введенными в базу знаний о модели в процессе доказательства.

Доказательство. Согласно ТЛ1, если отдельным объектам модели соответствуют некоторые логические утверждения, то их совокупность представляет некоторое целевое утверждение, которое может быть доказано или опровергнуто путем обратного логического вывода. Этот процесс может сопровождаться включением/изъятием фактов из базы знаний, в том числе фактов о структуре/параметрах модели. Если база знаний не изменилась, то она полностью согласована с текущей ОСМ. Если база знаний изменилась, то это может означать возникновение нового целевого утверждения, следующего из модифицированной модели, которое также нуждается в доказательстве. Даже если целевое утверждение не изменилось, оно нуждается в повторном рассмотрении во избежание противоречия с изменившимся содержанием базы фактов. Таким образом процесс согласования подразумевает инкрементальное (многоэтапное) доказательство динамически генерируемого целевого утверждения, следующего из структуры/параметров модели. Теорема доказана.

Теорема о когнитивном характере ОСМ (ТК1). В процессе построения ОСМ функционирует как когнитивная система.

Доказательство. По определению система когнитивной архитектуры подразумевает многоэтапное рассуждение на основе фактов, находящихся в данный момент в рабочей области, в процессе которого в такую область рассуждения включаются новые факты и/или изымаются уже имеющиеся. Такой процесс может продолжаться до полного согласования системы рассуждений с базой имеющихся и выведенных фактов.

Утверждение об обратной связи (ТМ8). Результат исполнения как конечной ОСМ, так и ОСМ, являющихся промежуточными этапами ее унификации, может содержать новые факты, нуждающиеся в согласовании со структурой/параметрами ОСМ.

Доказательство. Утверждение согласуется с теоремой о когнитивном характере процессов в ОСМ (ТК1).

Следствие. Необходима формальная машина вывода ОСМ, позволяющая поэтапно унифицировать ОСМ, планировать исполнение конечных и промежуточных ОСМ, исполнять их, обобщать и *при необходимости* согласовывать результаты.

Теорема о распределенной машине вывода ОСМ (ТМ9). Условиям, указанным следствием из ТМ8, удовлетворяет система T = (S, Q, I, A, M, P), включающая

а) стек S троек (C, B, F), где C – ОСМ или машина вывода, B – локальная база фактов ОСМ или пустое множество для машины, $F \in \{0,1\}$ – логический признак готовности к исполнению,

б) очередь Q асинхронно или синхронно запущенных машин,

в) очередь команд I, для которой операции помещения и извлечения являются атомарными,

г) рабочую область A,

д) активную систему M = (C, B, F),

е) ссылку P на машину-родителя.

Машина должна обладать следующими свойствами:

А. Способность выполнить следующие команды:

mov – копировать активную систему M в A.

push – переместить систему из A на вершину стека S со статусом «не готова к исполнению». Выполнение данной операции не запускает процесс унификации для указанной системы.

epush – выполняется аналогично **push**, системе присваивается статус «готова к исполнению».

pop – взять систему с вершины стека S в A.

fork – создать новую машину вывода и поместить ее в A. При этом с вершины стека S текущей машины снимается система и перемещается в стек созданной машины. В отличие от обычной модели, машина вывода может быть отправлена на исполнение на иное вычислительное устройство, что обеспечивает возможность работы в распределенном режиме.

assert(факт) – поместить факт в рабочую область A машины.

retract(факт) – удалить факт из рабочей области A машины.

consult – пополнить базу фактов ОСМ, находящейся в M, базой фактов из рабочей области A, которая при этом очищается. Если A изначально пуста, база фактов не изменяется.

Б. Алгоритм вывода:

1. Если M пуста, то

1.1. Если стек S пуст, то ожидаем очистки очереди Q (то есть завершения работы запущенных машин), в процессе ожидания обрабатываем команды, поступающие в очередь I. Затем машина заканчивает работу.

1.2. Если стек S не пуст, то система с вершины стека перемещается в M.

2. Если в M присутствуют данные, требующие обработки, то переходим к пункту 3. Иначе очищаем M и переходим к пункту 1.

К данным, требующим обработки, относятся: а) ОСМ, требующая унификации, б) иная машина вывода.

3. Если содержимое M относится к классу (б), то это иная машина вывода. Тогда:

3.1. Машина вывода из M помещается в очередь Q, где либо *асинхронно или синхронно* запускается на исполнение на свободном вычислительном устройстве, либо синхронно запускается на текущем, если свободных устройств нет. В процессе исполнения модели, обрабатываемые запущенной машиной, имеют возможность, помещая **assert** в P.I, включать факты в рабочую область A текущей машины. Машина удаляется из очереди тогда и только тогда, когда закончит работу.

3.2. Очищаем M и переходим к пункту 1.

4. Ожидаем очистки очереди Q, обрабатывая команды, поступающие в очередь I.

5. Если содержимое M относится к классу (a), то это тройка (ОСМ, БФ, ГОТ). Тогда выполняются следующие действия:

5.1. Унификация ОСМ и далее, если ГОТ=1, исполнение. В процессе работы ОСМ может помещать в очередь I текущей машины синхронно исполняемые команды.

5.2. Если в результате исполнения изменилась БФ, переходим к пункту 5.1. Иначе очищаем M и переходим к пункту 1.

Доказательство. Алгоритм содержит цикл 5.1-5.2, обеспечивающий поэтапную унификацию ОСМ, ее исполнение с последующим согласованием структуры и параметров ОСМ с фактами, полученными в ходе исполнения. Для реализации данного такта машине вывода необходим компонент M, а также возможность выполнить команды **assert, consult** и, возможно, **retract**, требующие наличия аккумулятора A. Команда **consult** не изменяет базу фактов при пустом A, это необходимое условие, предотвращающее зацикливание системы вывода. Таким образом, в части, относящейся к начальной ОСМ и результату ее унификации, условия следствия из ТМ8 выполняются.

Цикл 1-5 обеспечивает обработку моделей, непосредственно поступивших в стек текущей машины вывода, а также запуск динамически порожденных машин для обработки переданных им моделей. Этим обосновывается наличие стека S. Таким образом, поскольку существуют команды **mov** и **push/epush**, в стек могут быть помещены ОСМ, являющиеся промежуточными этапами унификации. В отличие от **push**, которая помещает в стек модель, ориентируя ее исключительно на унификацию, команда **epush** помимо унификации предполагает исполнение модели, что позволяет учесть соответствующие требования ТМ8. Команда **pop** позволяет при необходимости избежать исполнения некоторых ОСМ и машин вывода и учета соответствующих результатов. Команды **mov**, **push/epush** и **fork** позволяют поместить промежуточные ОСМ в динамически порождаемые машины вывода для запуска и обработки в своем физическом/логическом контексте (в том числе на ином вычислительном устройстве). Для хранения порождаемых машин была введена очередь Q, причем механизм очереди позволяет установить приоритеты их исполнения. Для реализации доступа из дочерних машин к данным родительской в систему вывода введена ссылка P. Задача удовлетворения условиям следствия из ТМ8 для моделей, переданных в дочерние машины, рассматриваемых независимо от текущей машины, сводится к исходной. Команды **assert/retract**, **consult** обеспечивают (*в случае такой необходимости*) взаимодействие с порожденными в ходе их работы базами фактов, что окончательно удовлетворяет условиям следствия из ТМ8 для текущей машины, поскольку в необходимых случаях обеспечивает согласование с результатами исполнения любых промежуточных ОСМ, сгенерированных самой машиной или любой порожденной ею машиной.

Команды, поступающие в машину, ставятся в очередь I и исполняются синхронно, в соответствии с принципом FIFO (First In First Out). Необходимость в наличии очереди объясняется требованием отсутствия коллизий при одновременном поступлении в машину нескольких команд от дочерних машин. Режим ожидания, реализованный пунктом 4 алгоритма, исключает одновременную работу родительской и дочерних машин,

предотвращая потенциальные коллизии поступления с их сторон команд работы с одними и теми же областями данных (в первую очередь – с аккумулятором).

Следствие. При асинхронном запуске машин из очереди Q имеем параллельный логический вывод, при синхронном – последовательный.

Рассмотрим реализацию машины вывода ОСМ в системе порождения программ PGEN++. В соответствии с ТМ9 машина вывода поддерживает порождение новых машин вывода для асинхронного решения подзадач. Эти машины содержатся в очереди Q родительской машины и могут либо последовательно исполняться на одном компьютере, либо передаваться на параллельную обработку на иные компьютеры. Таким образом, архитектура системы должна быть распределенной. Была выбрана схема, в которой любой из экземпляров системы порождения программ может выполнять как функции сервера, обеспечивающего взаимодействие с пользователем и распределение вычислительных ресурсов, так и функции клиента (исполнителя порожденных заданий).

Роль конкретного экземпляра системы определяется динамически, по факту очередности запуска: первый экземпляр берет на себя функции сервера, остальные – клиентов. Основная часть заданий обычно генерируется сервером, однако возможны и ситуации, когда в процессе исполнения заданий на клиенте генерируется новая их серия (имеется в виду случай, когда дочерняя машина вывода породила еще одну или несколько машин). В таком случае клиентские экземпляры системы порождения берут на себя часть серверных функций (засылка задания со всеми необходимыми файлами и ожидание результата его обработки) по отношению к порожденным заданиям, за исключением распределения вычислительных ресурсов – им занимается лишь серверный экземпляр системы. Он и предоставляет (по поступающим запросам) клиентские экземпляры системы в качестве исполнителя порожденных машин вывода.

Данная идеология работы определила и механизмы реализации управляющей выводом части системы порождения программ. Каждый экземпляр системы является многопоточным приложением, в котором *постоянно работает не менее трех потоков*: основного, управляющего и очереди заданий.

Основной поток содержит машину вывода и генерирует для сервера сообщение об окончании работы машины (об освобождении узла системы) и об окончании исполнения задания.

Управляющий поток реализует обработку запросов, поступающих от других экземпляров системы. В начале работы он посылает групповой запрос (IP Multicast) узлам локальной сети, чтобы определить, является ли текущий экземпляр первым из запущенных, то есть серверным, или же клиентским. Далее поток переходит в режим генерации и обработки запросов по сети, преимущественно касающихся занятости узлов и их выделения. Он также принимает задания от серверного экземпляра системы.

Поток, реализующий очередь заданий, также запускается при старте системы. Он отслеживает занятость текущего экземпляра системы заданием, сообщая серверу об его освобождении, ведет список генерируемых в системе заданий, запрашивает свободные узлы (экземпляры системы) для отсылки задания и ведет их список (это относится в первую очередь к серверу), при появлении свободного узла отсылает ему задание.

При отсылке каждого задания для него также порождается *отдельный поток*, который устанавливает связь с удаленным компьютером, на котором обрабатывается задание, и переходит в режим ожидания результатов.

Итак, в данной работе дано понятие о логическом выводе ОСМ, предложена распределенная машина вывода ОСМ, определена технология ее реализации на комплексе ЭВМ, особенно эффективная в случае применения многоядерных процессоров.

Литература

1. Пекунов В.В. Теория объектно-событийного моделирования последовательных и параллельных процессов. Следствия и приложения в программировании, моделировании и порождении программ // Сб. матер. XI Всеросс. конф. «Высокопроизводительные параллельные вычисления на кластерных системах». – Нижний Новгород: Изд-во ННГУ, 2011. – С.238-243.